ФЕДЕРАЛЬНОЕ АГЕНТСТВО

ПО ТЕХНИЧЕСКОМУ РЕГУЛИРОВАНИЮ И МЕТРОЛОГИИ

НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

ГОСТ Р 55346— 2012/ ISO/PAS 20542:2006

Системы промышленной автоматизации и интеграция

ПРЕДСТАВЛЕНИЕ И ОБМЕН ПРОИЗВОДСТВЕННЫМИ ДАННЫМИ. БАЗОВАЯ МОДЕЛЬ ИНЖЕНЕРНОГО ПРОЕКТИРОВАНИЯ СИСТЕМ

ISO/PAS 20542:2006
Industrial automation systems and integration — Product data representation and exchange — Reference model for systems engineering
(IDT)

Издание официальное

Москва
Стандартинформ
2015

gost.e-ecolog.ru

Предисловие

1 ПОДГОТОВЛЕН АНО «Международная академия менеджмента и качества бизнеса» на основе собственного аутентичного перевода на русский язык стандарта, указанного в пункте 4

2 ВНЕСЕН Техническим комитетом по стандартизации ТК 100 «Стратегический и инновационный менеджмент»

3 УТВЕРЖДЕН И ВВЕДЕН В ДЕЙСТВИЕ Приказом Федерального агентства по техническому регулированию и метрологии от 29 ноября 2012 г. № 1707-ст

4 Настоящий стандарт идентичен международному документу ISO/PAS 20542:2006 «Системы промышленной автоматизации и интеграция. Представление и обмен производственными данными. Базовая модель инженерного проектирования систем» (ISO/PAS 20542:2006 Industrial automation systems and integration — Product data representation and exchange — Reference model for systems engineering).

При применении настоящего стандарта рекомендуется использовать вместо ссылочных международных стандартов соответствующие им национальные стандарты Российской Федерации, сведения о которых приведены в дополнительном приложении ДА

5 ВВЕДЕН ВПЕРВЫЕ

*Правила применения настоящего стандарта установлены в ГОСТ Р 1.0—2012 (раздел 8). Информация об изменениях к настоящему стандарту публикуются в ежегодном (по состоянию на 1 января текущего года) информационном указателе «Национальные стандарты», а официальный текст изменений и поправок — в ежемесячном указателе «Национальные стандарты». В случае пересмотра (замены) или отмены настоящего стандарта соответствующее уведомление будет опубликовано в ближайшем выпуске ежемесячного информационного указателя «Национальные стандарты». Соответствующая информация, уведомление и тексты размещаются также в информационной системе общего пользования — на официальном сайте Федерального агентства по техническому регулированию и метрологии в сети Интернет (*gost.e-ecolog.ru

© Стандартинформ, 2015

Настоящий стандарт не может быть полностью или частично воспроизведен, тиражирован и распространен в качестве официального издания без разрешения Федерального агентства по техническому регулированию и метрологии

Содержание

1 Область применения . . . 1
2 Нормативные ссылки . . . 2
3 Термины и определения . . . 2
3.1 Термины, определенные в ИСО 10303-1:1994 . . . 2
3.2 Термины, определенные в работах Института прикладной математики (Institut für Instrumentelle Mathematik) IIM-2:1962 . . . 3
3.3 Прочие определения . . . 3
3.4 Аббревиатуры . . . 3
4 Информационные требования . . . 3
4.1 Функциональные модули . . . 3
4.2 Определения типов ARM-модели . . . 13
4.3 Определения прикладных компонентов ARM-модели . . . 23
4.4 Функции в ARM-модели . . . 166
5 Требования соответствия . . . 167
5.1 Класс A: Административная информация . . . 168
5.2 Класс B: Управление работами . . . 168
5.3 Класс C: Управление внесением изменений . . . 169
5.4 Класс D: Ссылки на документацию . . . 169
5.5 Класс E: Классификация элементов системы . . . 169
5.6 Класс F: Задание приоритетности элементов . . . 169
5.7 Класс G: Графическое представление информации . . . 169
5.8 Класс 01: Представление требований к текстам . . . 169
5.9 Класс 02: Представление требований к текстам и соответствующим понятиям . . . 169
5.10 Класс 03: Формы представления структурных требований . . . 170
5.11 Класс 04: Диаграммы потоков данных . . . 170
5.12 Класс 05: Функциональные блок-схемы потоков . . . 170
5.13 Класс 06: Диаграммы поведения систем . . . 170
5.14 Класс 07: Структурный анализ систем . . . 171
5.15 Класс 08: Физическая архитектура систем . . . 171
5.16 Класс 09: Объектно-ориентированный анализ систем . . . 171
5.17 Класс 10: Объектно-ориентированные статические структуры систем . . . 171
5.18 Класс 11: Объектно-ориентированное поведение систем . . . 172
5.19 Класс 12: Объектно-ориентированная практическая реализация систем . . . 172
5.20 Класс 13: Представление систем . . . 172
5.21 Класс 14: Верификация систем . . . 172
5.22 Класс 15: Среда функциональной архитектуры и требования к ней . . . 172
5.23 Класс 16: Среда физической архитектуры и требования к ней . . . 173
5.24 Класс 17: Среда полного инженерного проектирования систем . . . 173
5.25 Класс 18: Среда объектно-ориентированного инженерного проектирования систем . . . 174
5.26 Класс 19: Среда расширенного инженерного проектирования систем . . . 174
Приложение A (обязательное) Требования, зависящие от метода практической реализации . . . 191
Приложение B (обязательное) Формуляр для утверждения протокола соответствия практической реализации системы (PICS) . . . 192

Приложение C (обязательное) Регистрация информационных объектов 194
Приложение D (справочное) Модель функционирования приложения 195
Приложение E (справочное) Ссылочная модель приложения EXPRESS-G 224
Приложение F (обязательное) Листинг для интерпретации на компьютере 265
Приложение ДА (справочное) Сведения о соответствии ссылочных международных стандартов ссылочным национальным стандартам Российской Федерации 392
Библиография 393

Введение

Комплекс международных стандартов ИСО 10303 предназначен для представления информации о продукции и для обмена данными о продукции в компьютерно-интепретируемой электронной форме. Основной целью данного комлекса является создание нейтрального механизма, способного описывать продукцию на протяжении всего жизненного цикла. Этот механизм подходит не только для файлового обмена, но также для использования в качестве базиса с целью архивирования и распространения баз данных с информацией о продукции.

Настоящий стандарт по своему содержанию очень близок к стандартам серии ИСО 10303.

Общедоступные спецификации (PAS) определяют обмен данными и совместное использование протокола данных системного проектирования. Он определяет контекст, цели и требования для различных этапов разработки информационной системы. Настоящий стандарт может применяться для любых систем (воздушные суда, автомобили, морские суда, железнодорожный транспорт и производственные предприятия).

В контексте космической и авиационной техники, например, термин «система» включает в себя различные значения: авиационная электронная система (например, программная, навигационная, с интерфейсом человек — система); система конструкции летательного средства (например, эвакуационная для экипажа и пассажиров, производства и распределения электроэнергии, контроля за окружающей средой, управления топливными ресурсами). Выше перечислены системы, которые входят в состав более крупных информационных систем (формирование систем из подсистем), которые могут быть частью авиационной техники, наземных вспомогательных служб, систем, обеспечивающих логистическую поддержку и обучение. Подобные системы выполнены преимущественно из активных компонентов, сенсоров, дисплеев, приводных механизмов, которые соединены между собой напрямую или с помощью коммуникационных средств. Систему нельзя рассматривать как простую совокупность из отдельных компонентов; требуется обязательно принимать во внимание и контролировать поведение интегрированных компонентов и свойств системы (которые по большей части работают в реальном времени).

Следующие утверждения представляют основные преимущества в процессе системного проектирования продукции:

- высокая стоимость разработки распространяется на небольшое количество производимых единиц;
- риск в равной степени разделяется на все вовлеченные стороны;
- необходимость сокращения времени пребывания на рынке;
- включение в себя различных технологий;
- повышенная реализуемость за счет возможной доработки конечного продукта под нужды конкретного пользователя.

Еще одним важным преимуществом является возможность повторного использования разработок и компонентов из других систем и для связи с другими системами. Повторное использование является ключевым элементом информационной модели.

Общедоступные спецификации (PAS), на которых основан настоящий стандарт, очень тесно связаны с признанными стандартами по системному проектированию [IEEE 1220], [EIA 632] и [ИСО 15288]. Системное проектирование не зависит от технологий, «горизонтальная» интеграционная дисциплина применяется к любому типу системы. Системное проектирование пересекается с техническими и аналитическими дисциплинами в течение всего жизненного цикла существования системы. С точки зрения специалистов по системному проектированию, необходимо осуществлять техническую поддержку на протяжении всего жизненного цикла продукции, а также проводить ее верификацию и валидацию. При системном проектировании необходимо учитывать двунаправленную связь между системами — участниками взаимодействия. В процессе анализа требования предъявляются как к входным, так и выходным данным.

Жизненный цикл системы может быть разделен на несколько этапов:

- оценку применимости;
- системное проектирование;
- реализацию;
- функционирование;
- техническую поддержку;
- вывод из эксплуатации.

Проектирование системы можно разделить на две части: на работу с предметной областью и на разработку самой системы. Несмотря на то что системное проектирование является наиболее важной фазой в процессе проектирования системы, также необходимо уделять пристальное внимание и другим этапам, в частности техническому предложению. Настоящий стандарт содержит требования к этапу системного проектирования для обмена данными, в особенности когда речь идет об интеграции в единый комплекс, мультидисциплинарной, многотехнологичной системе или продукции.

Настоящий стандарт определяет контекст, цель и требования к данным для обмена информацией между заинтересованными сторонами в процессе системного проектирования и конкретизирует набор сущностей и атрибутов для выполнения данных требований.

Настоящий стандарт устанавливает основу для дальнейшего развития и тестовые комплексы для проверки совместимости и возможности дальнейшего развития.

НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

Системы промышленной автоматизации и интеграция

ПРЕДСТАВЛЕНИЕ И ОБМЕН ПРОИЗВОДСТВЕННЫМИ ДАННЫМИ. БАЗОВАЯ МОДЕЛЬ ИНЖЕНЕРНОГО ПРОЕКТИРОВАНИЯ СИСТЕМ

Industrial automation systems and integration. Product data representation and exchange. Reference model for systems engineering

Дата введения — 2014—01—01

1 Область применения

Настоящий стандарт устанавливает область применения и информационные требования к проектированию систем.

Настоящий стандарт распространяется на:

- технические аспекты проектирования систем, определяющие функциональную и физическую архитектуру (что рассматриваемая система делает, как делает, и насколько хорошо она это делает);
- техническое управление проектированием: разработку календарного плана, оформление документов, сопутствующее проектирование, процедуры проверки достоверности, процедуры верификации;
- управление проектом: управление отслеживаемостью и конфигурацией системы, анализ компромиссных решений;
- промышленное управление (управление кооперацией промышленных партнеров);
- определение других систем, с которыми взаимодействует рассматриваемая система;
- контекст системы в каждой фазе ее жизненного цикла;
- поддержку методик иерархического разделения и объектно-ориентированного моделирования;
- функциональные и нефункциональные требования систем в каждой фазе жизненного цикла системы;
- определение статического и динамического поведения систем;
- фазу жизненного цикла проектирования систем;
- данные описания систем;
- данные спецификаций требований систем;
- данные, определяющие статическое поведение систем в терминах функций и потоков между функциями;
- данные, определяющие динамическое поведение систем;
- данные, определяющие физическую архитектуру;
- данные, определяющие разделение систем;
- данные, определяющие верификацию и проверку достоверности систем;
- данные, определяющие проектирование и промышленное управление;
- данные, определяющие управление конфигурацией.

Настоящий стандарт не распространяется на:

- дисциплины специалистов;
- фазы жизненного цикла проектирования;
- фазы жизненного цикла внедрения системы, управления системой, технического обслуживания системы, вывода системы из эксплуатации;

Издание официальное

- особые данные области применения;
- данные, используемые исключительно на фазах оценки целесообразности, внедрения системы, управления системой, технического обслуживания системы, вывода системы из эксплуатации.

2 Нормативные ссылки

В настоящем стандарте использованы нормативные ссылки на следующие стандарты, которые необходимо учитывать при использовании настоящего стандарта. В случае ссылок на документы, у которых указана дата утверждения, необходимо пользоваться только указанной редакцией. В случае когда дата утверждения не приведена, следует пользоваться последней редакцией ссылочных документов, включая любые поправки и изменения к ним:

ИСО/МЭК 8824-1:2002 Информационные технологии. Абстрактные синтаксические обозначения. Версия 1 (ASN.1). Часть 1: Спецификация базовых обозначений (ISO/IEC 8824-1:2002 Information technology — Abstract Syntax Notation One (ASN.1): Specification of basic notation);

ИСО 10303-1:1994 Системы промышленной автоматизации и интеграция. Представление и обмен производственными данными. Часть 1: Обзор и фундаментальные принципы (ISO 10303-1:1994 Industrial automation systems and integration — Product data representation and exchange — Part 1: Overview and fundamental principles);

ИСО 10303-11:2004 Системы промышленной автоматизации и интеграция. Представление и обмен производственными данными. Часть 11: Методы описания: Описание языка EXPRESS (ISO 10303-11:2004 Industrial automation systems and integration — Product data representation and exchange — Part 11: Description methods: The EXPRESS language reference manual);

ИСО 10303-21:2002 Системы промышленной автоматизации и интеграция. Представление и обмен данных продукта. Часть 21: Методы практической реализации: Кодирование открытого текста структуры обмена (ISO 10303-21:2002 Industrial automation systems and integration — Product data representation and exchange — Part 21: Implementation methods: Clear text encoding of the exchange structure);

ИСО 10303-22:1998 Системы промышленной автоматизации и интеграция. Представление и обмен производственными данными. Часть 22: Методы практической реализации: доступ к интерфейсу для стандартных данных (ISO 10303-22:1998 Industrial automation systems and integration — Product data representation and exchange — Part 22: Implementation methods: Standard data access interface).

3 Термины и определения

3.1 Термины, определенные в ИСО 10303-1:1994

В ИСО 10303-1:1994 определены следующие термины, используемые в настоящем стандарте:
- приложение;
- объект приложения;
- модель действия приложения;
- модель, интерпретированная для приложения;
- протокол приложения;
- базовая модель приложения;
- класс соответствия;
- данные;
- обмен данными;
- информация;
- интегрированный ресурс;
- продукция;
- производственные данные;
- функциональный модуль.

3.2 Термины, определенные в работах Института прикладной математики (Institut für Instrumentelle Mathematik) IIM-2:1962

Нижеприведенный термин, используемый в настоящем стандарте, определен в работах Института прикладной математики IIM-2:1962:

- Сети Петри.

3.3 Прочие определения

В настоящем стандарте используются следующие термины с соответствующими определениями:

3.3.1 **система** (system): Упорядоченный набор элементов, людей, продуктов (устройств, программного обеспечения), а также процессов (производственных мощностей, оборудования, материалов и процедур), которые являются связанными между собой, чье поведение удовлетворяет требованиям заказчика или производственным потребностям, а также обеспечивает жизненный цикл продукции.

3.3.2 **инженерное проектирование систем** (systems engineering): Междисциплинарный кооперативный подход к получению, разработке и подтверждению сбалансированного обеспечения жизненного цикла системы, удовлетворяющего ожиданиям заказчика и общественным потребностям.

3.4 Аббревиатуры

Нижеследующие аббревиатуры используются в настоящем стандарте:

AAM — модель действия приложения (Application activity model);

AP — протокол приложения (Application protocol);

ARM — ссылочная модель приложения (Application reference model);

CC — класс соответствия (Conformance class);

UoF — функциональный модуль (Unit of functionality);

URL — унифицированный указатель [информационного] ресурса (Universal resource locator);

ASCII — американский стандартный код обмена информацией (American Standard Code for Information Interchange);

cb — причинное поведение (Causal behaviour);

fsm — конечное состояние машины (Finite state machine);

N/A — неприменимый (Not applicable);

OO — объектно-ориентированный (Object oriented);

OOSE — объектно-ориентированное проектирование системы (Object oriented systems engineering);

PAS — общедоступная спецификация (Publicly Available Specification).

4 Информационные требования

В данном разделе устанавливаются информационные требования к ссылочной модели приложения для общедоступной спецификации проектирования систем.

Информационные требования приведены как набор функциональных блоков и объектов приложения. Информационные требования определены с помощью терминологии, приведенной в настоящем стандарте.

Примечание 1 — Информационные требования соответствуют действиям, идентифицированным внутри области применения протокола приложения, описанного в приложении D.

Примечание 2 — Графическое представление информационных требований приведено в приложении E.

4.1 Функциональные модули

Данный подраздел устанавливает функциональные модули (UoFs) в рассматриваемой PAS-спецификации, а также элементы поддержки, необходимые для ее определения. Данная спецификация устанавливает следующие функциональные модули и прикладные компоненты.

- Change_Management;
- Element_Classification;
- Element_Prioritization;
- Explicit_Functional_Reference;
- Explicit_Physical_Reference;
- External_Document_Reference_Mechanism;

- Functional_Allocation;
- Functional_Behaviour_Basics;
- Functional_Behaviour_Causal_Chain;
- Functional_Behaviour_Finite_State_Machine;
- Functional_Behaviour_Interaction;
- Functional_Hierarchy;
- Functional_Performance;
- Graphics;
- Measurement_Unit;
- OO_Behaviour;
- OO_Common;
- OO_Implementation;
- OO_Static_Structure;
- OO_Use_Case;
- Person_Organization;
- Physical_Architecture;
- Properties;
- Relationship_Cardinality;
- Requirement_Allocation;
- Requirement_Representation;
- Requirement_Representation_Implied_Functionality;
- Requirement_Representation_Structured_Formats;
- Structured_Text;
- System_Architecture;
- System_Validation;
- System_Verification;
- Version_Management;
- Work_Management.

В настоящем стандарте отсутствуют описания других функциональных модулей.

Функциональные модули и описания поддерживающих их функций приведены ниже.

4.1.1 Функциональный модуль Change_Management

Функциональный модуль Change_Management содержит компоненты для представления процесса внесения изменений, начиная от идентификации объектов и заканчивая созданием протокола внесения изменений.

В функциональном модуле Change_Management определены следующие прикладные компоненты:

- change_order;
- change_order_relationship;
- change_report;
- change_report_element_assignment;
- change_request;
- critical_issue;
- critical_issue_impact;
- critical_issue_relationship;
- issue_source_relationship;
- issue_system_assignment.

4.1.2 Функциональный модуль Elements_Classification

Функциональный модуль Elements_Classification содержит компоненты для представления элементов классификационных деревьев с целью предоставления набора критериев, используемых для повышения пригодности и возможности повторного использования.

В функциональном модуле Element_Classification определены следующие прикладные компоненты:

- package;
- package_classification_assignment;
- package_classification_system;
- package_element_assignment;
- package_hierarchy_relationship;
- selection_package.

4.1.3 Функциональный модуль Element_Prioritization

Функциональный модуль Element_Prioritization содержит компоненты для определения дискретного диапазона приоритетов и присвоения элементам спецификации определенных уровней приоритетности.

В функциональном модуле Element_Prioritization определены следующие прикладные компоненты:

- rank_assignment;
- rank_group;
- rank_relation;
- ranking_element;
- ranking_system.

4.1.4 Функциональный модуль Explicit_Functional_Reference

Функциональный модуль Explicit_Functional_Reference содержит компоненты для выполнения однозначных ссылок на объекты в иерархии функций.

В функциональном модуле Explicit_Functional_Reference определены следующие прикладные компоненты:

- functional_link_reference;
- functional_reference_configuration;
- functionality_instance_reference;
- functionality_reference_composition_relationship;
- functionality_reference_relationship;
- persistent_storage_equivalence_relationship;
- persistent_storage_reference.

4.1.5 Функциональный модуль Explicit_Physical_Reference

Функциональный модуль Explicit_Physical_Reference содержит элементы для выполнения однозначных ссылок на физические иерархии.

В функциональном модуле Explicit_Physical_Reference определены следующие прикладные компоненты:

- physical_instance_reference;
- physical_reference_configuration;
- physical_reference_relationship;
- system_physical_configuration.

4.1.6 Функциональный модуль External_Document_Reference_Mechanism

Функциональный модуль External_Document_Reference_Mechanism содержит объекты для ссылок на внешние документы из системной спецификации. Эти документы могут представляться в любом формате как для обработки на компьютере, так и без него.

В функциональном модуле External_Document_Reference_Mechanism определены следующие прикладные компоненты:

- digital_document;
- document_assignment;
- documentation_reference;
- documentation_relationship;
- non_digital_document;
- partial_document_assignment.

4.1.7 Функциональный модуль Functional_Allocation

Функциональный модуль Functional_Allocation содержит объекты, фиксирующие связи размещения, начиная от элементов функциональной архитектуры и заканчивая элементами физической архитектуры.

В функциональном модуле Functional_Allocation определены следующие прикладные компоненты:

- clock_reference_context_relationship;
- functional_link_allocation_relationship;
- functionality_allocation_relationship.

4.1.8 Функциональный модуль Functional_Behaviour_Basics

Функциональный модуль Functional_Behaviour_Basics содержит объекты, определяющие функциональное поведение семантических модификаторов и механизмов объединения моделей функционального поведения с функциональными моделями.

В функциональном модуле Functional_Behaviour_Basics определены следующие прикладные компоненты:

- functional_behaviour_model;
- functional_behaviour_model_assignment.

4.1.9 Функциональный модуль Functional_Behaviour_Causal_Chain

Функциональный модуль Functional_Behaviour_Causal_Chain содержит объекты, определяющие функциональное поведение, основанное на графе причинности. Данный функциональный модуль дает возможность представлять функциональные характеристики, обнаруживаемые в RDD-диаграммах поведения и ViTECH-CORE-EFFBD-диаграммах. Элементы данного функционального модуля используют расширенную систему обозначений сетей Петри.

В функциональном модуле Functional_Behaviour_Causal_Chain определены следующие прикладные компоненты:

- causal_block_bound;
- cb_firing_condition;
- cb_functional_behaviour_model;
- cb_functional_place;
- cb_functional_transition;
- cb_initial_marking;
- cb_input_relationship;
- cb_output_relationship;
- cb_place;
- cb_place_function_association;
- cb_place_reference;
- cb_transition;
- cb_transition_relationship;
- cb_transition_unbounded_weight.

4.1.10 Функциональный модуль Functional_Behaviour_Finite_State_Machine

Функциональный модуль Functional_Behaviour_Finite_State_Machine (fsm) содержит объекты, определяющие функциональное поведение автоматов с конечным числом состояний. Данный функциональный модуль включает поддержку спецификации, представленной в кодах конечных автоматов (в стиле Мура и Мили) и диаграммами состояний.

В функциональном модуле Functional_Behaviour_Finite_State_Machine определены следующие прикладные компоненты:

- formal_data_interaction_port;
- fsm_and_state;
- fsm_command_interaction_relationship;
- fsm_data_interaction_binding;
- fsm_data_interaction_relationship;
- fsm_generic_state;
- fsm_initial_state_transition;
- fsm_model;
- fsm_or_state;
- fsm_state;
- fsm_state_composition_relationship;
- fsm_state_transition;
- fsm_transient_state;
- function_reference;
- functional_state_context;
- generic_state_context;
- initial_state_transition_specification_assignment;
- name_binding;
- specification_state_assignment;
- state_context_relationship;
- state_function_interaction_port;
- state_machine_functional_behaviour_model;
- state_transition_specification_assignment.

4.1.11 Функциональный модуль Functional_Behaviour_Interaction

Функциональный модуль Functional_Behaviour_Interaction содержит объекты, определяющие взаимодействие между функциями, и порядок использования сигналов взаимодействия для активации и деактивации этих функций. Указанный функциональный модуль также содержит объекты, определяющие тип элемента обмениваемой информации. Данный модуль тесно связан с их функциональной иерархией в том смысле, что элементы, определенные таким образом, являются необходимым условием применения функционального модуля взаимодействия для функционального поведения.

В функциональном модуле Functional_Behaviour_Interaction определены следующие прикладные компоненты:

- abstract_data_type_definition;
- abstract_data_type_member;
- actual_io_port;
- aggregate_data_type_definition;
- bi_directional_port_indicator;
- binary_data_type_definition;
- complex_data_type_definition;
- complex_value;
- compound_value;
- control_io_port;
- data_field;
- data_instance;
- derived_data_type_definition;
- elementary_maths_space;
- event_data_type_definition;
- finite_integer_interval;
- finite_real_interval;
- finite_space;
- formal_io_port;
- functional_link;
- functional_link_group;
- hibound_integer_interval;
- hibound_real_interval;
- integer_data_type_definition;
- integer_interval;
- io_buffer;
- io_composition_port;
- io_port;
- io_port_binding;
- lobound_integer_interval;
- lobound_real_interval;
- logical_data_type_definition;
- maths_space;
- real_data_type_definition;
- real_interval;
- record_data_type_definition;
- record_data_type_member;
- recursive_data_type_definition;
- string_data_type_definition;
- undefined_data_type_definition;
- union_data_type_definition;
- union_data_type_member;
- user_defined_data_type_definition.

4.1.12 Функциональный модуль Functional_Hierarchy

Функциональный модуль Functional_Hierarchy включает сущности, определяющие функциональное разбиение структуры системы.

В функциональном модуле Functional_Hierarchy определены следующие прикладные компоненты:

- composite_function_definition;
- function_instance;
- functional_decomposition_relationship;
- general_function_definition;
- general_functionality_instance;
- io_split_join;
- leaf_function_definition;
- persistent_storage.

4.1.13 Функциональный модуль Functional_Performance

Функциональный модуль Functional_Performance содержит объекты, представляющие временные ограничения на функциональные элементы спецификации.

В функциональном модуле Functional_Performance определены следующие прикладные компоненты:

- clock;
- clock_assignment_relationship;
- execution_time.

4.1.14 Функциональный модуль Graphics

Функциональный модуль Graphics содержит объекты, определяющие простое представление типа «узел — звено» в спецификации системы. Данный функциональный модуль упрощает приближенное формирование схемы спецификации системы.

В функциональном модуле Graphics определены следующие прикладные компоненты:

- actual_port_position;
- coordinate_translation_information;
- formal_port_position;
- graphics_link;
- graphics_node;
- graphics_point;
- graphics_view;
- multi_level_view;
- view_relationship;
- visual_element.

4.1.15 Функциональный модуль Measurement_Unit

Функциональный модуль Measurement_Unit содержит объекты, представляющие единицы измерения в спецификации.

В функциональном модуле Measurement_Unit определен следующий прикладной компонент:

- unit.

4.1.16 Функциональный модуль OO_Behaviour

Функциональный модуль OO_Behaviour содержит объекты, представляющие поведение объектно-ориентированной спецификации.

В функциональном модуле OO_Behaviour определены следующие прикладные компоненты:

- oo_action;
- oo_action_state;
- oo_action_state_transition;
- oo_action_temporal_relationship;
- oo_association_end_role;
- oo_association_role;
- oo_attribute_link_end_association;
- oo_call_action;
- oo_classifier_role;
- oo_collaboration;
- oo_create_action;
- oo_interaction;
- oo_link;
- oo_link_end;
- oo_message;

- oo_message_temporal_relationship;
- oo_reception;
- oo_send_action;
- oo_signal;
- oo_signal_behavioural_feature_relationship;
- oo_stimulus;
- oo_stimulus_argument.

4.1.17 Функциональный модуль OO_Common

Функциональный модуль OO_Common содержит объекты, общие для множества объектно-ориентированных представлений спецификации.

В функциональном модуле OO_Common определены следующие прикладные компоненты:

- oo_constraint;
- oo_constraint_model_element_relationship;
- oo_dependency;
- oo_element_import;
- oo_generalization;
- oo_instance_classifier_relationship;
- oo_model_element_stereotype_relationship;
- oo_model_element_tagged_value_relationship;
- oo_stereotype;
- oo_tagged_value;
- oo_view;
- oo_view_context_element_relationship;
- oo_view_relationship;
- oo_view_system_view_relationship.

4.1.18 Функциональный модуль OO_Implementation

Функциональный модуль OO_Implementation содержит объекты, предназначенные для представления объектно-ориентированной системной спецификации.

В функциональном модуле OO_Implementation определены следующие прикладные компоненты:

- oo_component;
- oo_component_allocation;
- oo_element_residence.

4.1.19 Функциональный модуль OO_Static_Structure

Функциональный модуль OO_Static_Structure содержит объекты, предназначенные для представления диаграмм классов в объектно-ориентированной парадигме.

В функциональном модуле OO_Static_Structure определены следующие прикладные компоненты:

- oo_argument;
- oo_association;
- oo_association_class;
- oo_association_end;
- oo_association_end_classifier_relationship;
- oo_association_end_qualifier_association;
- oo_attribute;
- oo_attribute_instance;
- oo_behavioural_feature;
- oo_class;
- oo_interface;
- oo_method;
- oo_object;
- oo_operation;
- oo_operation_interface_association;
- oo_package;
- oo_parameter.

4.1.20 Функциональный модуль OO_Use_Case

Функциональный модуль OO_Use_Case содержит объекты, предназначенные для представления точек зрения на объектно-ориентированную спецификацию.

В функциональном модуле OO_Use_Case определены следующие прикладные компоненты:
- oo_actor;
- oo_extension;
- oo_extension_point;
- oo_inclusion;
- oo_use_case.

4.1.21 Функциональный модуль Person_Organization

Функциональный модуль Person_Organization содержит объекты, предназначенные для представления информации о персонале и его роли в организации.

В функциональном модуле Person_Organization определены следующие прикладные компоненты:
- address;
- date_and_person_assignment;
- date_and_person_organization;
- date_assignment;
- date_time;
- organization;
- organization_relationship;
- person;
- person_in_organization;
- person_organization_assignment.

4.1.22 Функциональный модуль Physical_Architecture

Функциональный модуль Physical_Architecture содержит объекты, предназначенные для представления физической архитектуры системы. Модель устанавливает абстрактный подход и не выдает никаких предположений относительно технологии или области для элементов физической архитектуры.

В функциональном модуле Physical_Architecture определены следующие прикладные компоненты:
- actual_physical_port;
- formal_physical_port;
- functional_representation_relationship;
- general_physical_definition;
- physical_binding;
- physical_composition_relationship;
- physical_connection;
- physical_instance;
- physical_link_definition;
- physical_node_definition;
- physical_port.

4.1.23 Функциональный модуль Properties

Функциональный модуль Properties содержит объекты, определяющие общий способ указания задаваемых пользователем атрибутов для широкого круга объектов в других прикладных компонентах модели. Его возможности поддерживаются из-за того, что модель не может содержать атрибуты, удовлетворяющие всем областям инженерного проектирования систем.

В функциональном модуле Properties определены следующие прикладные компоненты:
- nominal_value;
- plus_minus_bounds;
- property_assignment;
- property_definition;
- property_relationship;
- property_value;
- property_value_function;
- property_value_relationship;
- requirement_allocation_property_relationship;
- value_limit;
- value_list;
- value_range;
- value_with_unit.

4.1.24 Функциональный модуль Relationship_Cardinality

Функциональный модуль Relationship_Cardinality содержит объекты, определяющие ограниченные и неограниченные выражения для кардинальности.

В функциональном модуле Relationship_Cardinality определены следующие прикладные компоненты:

- cardinality_list;
- cardinality_range;
- infinite_cardinality;
- single_cardinality.

4.1.25 Функциональный модуль Requirement_Allocation

Функциональный модуль Requirement_Allocation содержит объекты, предназначенные для представления связей распределения и подконтрольности между требованиями и элементами в функциональной и физической архитектуре.

В функциональном модуле Requirement_Allocation определены следующие прикладные компоненты:

- requirement_allocation_relationship;
- specific_requirement_allocation_relationship.

4.1.26 Функциональный модуль Requirement_Representation

Функциональный модуль Requirement_Representation содержит объекты, предназначенные для представления требований к фиксации связей между ними и представлению требований различным системам. Требования представляются в текстовой форме.

В функциональном модуле Requirement_Representation определены следующие прикладные компоненты:

- effectiveness_measure;
- effectiveness_measure_assignment;
- effectiveness_measure_relationship;
- requirement_class;
- requirement_class_relationship;
- requirement_composition_relationship;
- requirement_definition;
- requirement_instance;
- requirement_occurence;
- requirement_relationship;
- requirement_relationship_input_assignment;
- requirement_relationship_resulting_relationship;
- requirement_requirement_class_assignment;
- requirement_traces_to_requirement_relationship;
- textual_requirement_definition.

4.1.27 Функциональный модуль Requirement_Representation_Implied_Functionality

Функциональный модуль Requirement_Representation_Implied_Functionality содержит объекты, предназначенные для соединения требований с функциональными характеристиками или физическими элементами, которые предполагаются этими требованиями.

В функциональном модуле Requirement_Representation_Implied_Functionality определены следующие прикладные компоненты:

- data_transfer;
- implied_external_interaction.

4.1.28 Функциональный модуль Requirement_Representation_Structured_Formats

Функциональный модуль Requirement_Representation_Structured_Formats содержит объекты, предназначенные для представления требований, выражаемых в структурированных форматах или моделях.

В функциональном модуле Requirement_Representation_Structured_Formats определены следующие прикладные компоненты:

- model_defined_requirement_definition;
- structured_requirement_definition.

4.1.29 Функциональный модуль Structured_Text

Функциональный модуль Structured_Text содержит объекты, предназначенные для представления текстовых элементов, например отдельных текстовых строк, структурированных текстовых потоков, таблиц и аннотированных на каком-либо языке текстовых элементов.

В функциональном модуле Structured_Text определены следующие прикладные компоненты:
- textual_paragraph;
- textual_section;
- textual_specification;
- textual_table.

4.1.30 Функциональный модуль System_Architecture

Функциональный модуль System_Architecture содержит объекты, предназначенные для представления системы согласно ее спецификации и положению в иерархии систем. Каждая система может описываться с использованием любого числа ее представлений в различных жизненных циклах, сценариях и режимах эксплуатации. Модуль также поддерживает представление интерфейса между системами с помощью системной модели особого вида.

Объекты для представления системы работают как коллекторы, предоставляющие строительные блоки для элементов системной спецификации, например, для представления требований, функциональной и физической архитектуры.

В функциональном модуле System_Architecture определены следующие прикладные компоненты:
- context_function_relationship;
- context_physical_relationship;
- partial_system_view;
- partial_system_view_relationship;
- requirement_relationship_context_assignment;
- requirement_system_view_assignment;
- root_requirement_system_view_assignment;
- system_composition_relationship;
- system_definition;
- system_instance;
- system_instance_relationship;
- system_instance_relationship_port;
- system_instance_replication_relationship;
- system_substitution_relationship;
- system_view;
- system_view_assignment;
- system_view_context;
- triggered_system_view_relationship;
- verification_specification_system_view_relationship.

4.1.31 Функциональный модуль System_Validation

Функциональный модуль System_Validation содержит объекты, предназначенные для представления различных аспектов контрольной информации, формируемой в процессе инженерного проектирования системы, например, о персонале и организации (в виде текстовых элементов и комментариев).

В функциональном модуле System_Validation определены следующие прикладные компоненты:
- approval;
- approval_assignment;
- approval_person_organization;
- approval_relationship;
- assessment;
- assessment_relationship;
- justification;
- justification_relationship.

4.1.32 Функциональный модуль System_Verification

Функциональный модуль System_Verification содержит объекты, предназначенные для представления требований к методам испытаний и валидации. Элементы верификации системы могут присваиваться элементам архитектуры системы, анализа требований, прикладным компонентам функциональной и физической архитектуры.

В функциональном модуле System_Verification определены следующие прикладные компоненты:
- realized_system;
- realized_system_composition_relationship;

- verification_report_for_verification_specification;
- verification_result;
- verification_specification;
- verification_specification_allocation.

4.1.33 Функциональный модуль Version_Management

Функциональный модуль Version_Management содержит объекты, предназначенные для фиксации характера развития элементов спецификации по времени. Этот модуль является основополагающим, поскольку может использоваться во многих элементах большого числа прикладных компонентов. Он отражается в определениях конкретных объектов в каждом прикладном компоненте.

В функциональном модуле Version_Management определены следующие прикладные компоненты:

- configuration_element;
- configuration_element_relationship;
- configuration_element_version;
- configuration_element_version_relationship;
- element_identifier.

4.1.34 Функциональный модуль Work_Management

Функциональный модуль Work_Management содержит объекты, предназначенные для представления работ по инженерному проектированию, а также логики, которая приводит к инициализации работ по проекту.

В функциональном модуле Work_Management определены следующие прикладные компоненты:

- effectivity;
- effectivity_assignment;
- effectivity_relationship;
- engineering_process_activity;
- engineering_process_activity_element_assignment;
- engineering_process_activity_relationship;
- project;
- project_event_reference;
- project_relationship;
- start_order;
- start_request;
- work_order;
- work_request.

4.2 Определения типов ARM-модели

В данном подразделе определены типы прикладных компонентов и приведены их описания.

4.2.1 Прикладной компонент approval_assignment_select

EXPRESS-описание:

```
*)
    TYPE approval_assignment_select = SELECT WITH (change_report, configuration_element, configuration_element_version, critical_issue,critical_issue_impact, document_assignment, element_critical_issue_relationship, engineering_process_activity, instance_definition_select,package_element_assignment, partial_system_view_relationship, project, requirement_allocation_relationship, work_order, work_request);
    END_TYPE;
(*
```

4.2.2 Прикладной компонент assessment_assignment_select

EXPRESS-описание:

```
*)
    TYPE assigned_requirement_relationship_select = SELECT WITH (requirement_allocation_relationship, requirement_relationship);
    END_TYPE;
(*
```

4.2.3 Прикладной компонент assigned_requirement_relationship_select

EXPRESS-описание:

```
*)
    TYPE assigned_requirement_relationship_select = SELECT WITH (requirement_relationship);
    END_TYPE;
(*
```

4.2.4 Прикладной компонент basic_identifier

EXPRESS-описание:

```
*)
    TYPE basic_identifier = STRING;
    END_TYPE;
(*
```

4.2.5 Прикладной компонент boolean_value

EXPRESS-описание:

```
*)
    TYPE boolean_value = BOOLEAN;
    END_TYPE;
(*
```

4.2.6 Прикладной компонент buffer_synchronisation_enumeration

EXPRESS-описание:

```
*)
    TYPE buffer_synchronisation_enumeration = ENUMERATION OF (asynchronous,synchronouos);
    END_TYPE;
(*
```

4.2.7 Прикладной компонент cardinality_association_select

EXPRESS-описание:

```
*)
    TYPE cardinality_association_select = SELECT WITH (cardinality_list, cardinality_range, single_
    cardinality);
    END_TYPE;
(*
```

4.2.8 Прикладной компонент causal_weight_select

EXPRESS-описание:

```
*)
    TYPE causal_weight_select = SELECT WITH (cb_transition_unbounded_weight, natural_number);
    END_TYPE;
(*
```

4.2.9 Прикладной компонент change_element_select

EXPRESS-описание:

```
*)
    TYPE change_element_select = SELECT WITH (instance_definition_select);
    END_TYPE;
(*
```

4.2.10 Прикладной компонент change_report_element_select

EXPRESS-описание:

```
*)
    TYPE change_report_element_select = SELECT WITH (instance_definition_select);
    END_TYPE;
(*
```

4.2.11 Прикладной компонент control_characters
EXPRESS-описание:
```
*)
    TYPE control_characters = ENUMERATION OF (cr,tab);
    END_TYPE;
(*
```

4.2.12 Прикладной компонент control_type_enumeration
EXPRESS-описание:
```
*)
    TYPE control_type_enumeration = ENUMERATION OF
    (activate,activate_deactivate);
    END_TYPE;
(*
```

4.2.13 Прикладной компонент data_composition_select
EXPRESS-описание:
```
*)
    TYPE data_composition_select = SELECT WITH (actual_io_port, formal_io_port, io_composition_port);
    END_TYPE;
(*
```

4.2.14 Прикладной компонент data_direction
EXPRESS-описание:
```
*)
    TYPE data_direction = ENUMERATION OF
    (from_system, to_system);
    END_TYPE;
(*
```

4.2.15 Прикладной компонент data_type_definition_select
EXPRESS-описание:
```
*)
    TYPE data_type_definition_select = SELECT WITH (maths_space, user_defined_data_type_definition);
    END_TYPE;
(*
```

4.2.16 Прикладной компонент data_type_value_select
EXPRESS-описание:
```
*)
    TYPE data_type_value_select = SELECT WITH (boolean_value, complex_value, compound_
    value, integer_value, logical_value, real_value, text);
    END_TYPE;
(*
```

4.2.17 Прикладной компонент date_assignment_select
EXPRESS-описание:
```
*)
    TYPE date_assignment_select = SELECT WITH (approval, approval_assignment, assessment, con-
    figuration_element, configuration_element_version,configuration_element_version_relationship, con-
    text_physical_relationship, critical_issue, critical_issue_impact, document_assignment,documen-
    tation_relationship, element_critical_issue_relationship, engineering_process_activity, engineering_
    process_activity_element_assignment, instance_definition_select, justification, justification_re-
    lationship, package_element_assignment, partial_system_view_relationship, project, require-
    ment_allocation_relationship, requirement_system_view_assignment, work_order, work_request);
    END_TYPE;
(*
```

4.2.18 Прикладной компонент default_context_select

EXPRESS-описание:

```
*)
    TYPE default_context_select = SELECT WITH (fsm_generic_state, functional_state_context);
    END_TYPE;
(*
```

4.2.19 Прикладной компонент definition_select

EXPRESS-описание:

```
*)
    TYPE definition_select = SELECT WITH (fsm_state, functional_state_context, general_function_
    definition, general_physical_definition, oo_view, system_view);
    END_TYPE;
(*
```

4.2.20 Прикладной компонент effective_element_select

EXPRESS-описание:

```
*)
    TYPE effective_element_select = SELECT WITH (instance_definition_select, person_organiza-
    tion_assignment);
    END_TYPE;
(*
```

4.2.21 Прикладной компонент event_or_date_select

EXPRESS-описание:

```
*)
    TYPE event_or_date_select = SELECT WITH (date_time, project_event_reference);
    END_TYPE;
(*
```

4.2.22 Прикладной компонент external_element_select

EXPRESS-описание:

```
*)
    TYPE external_element_select = SELECT WITH (function_instance, physical_instance);
    END_TYPE;
(*
```

4.2.23 Прикладной компонент fsm_interaction_select

EXPRESS-описание:

```
*)
    TYPE fsm_interaction_select = SELECT WITH (initial_state_transition_specification_assignment,
    specification_state_assignment, state_transition_specification_assignment);
    END_TYPE;
(*
```

4.2.24 Прикладной компонент function_role_enumeration

EXPRESS-описание:

```
*)
    TYPE function_role_enumeration = ENUMERATION OF (external_element,system_function);
    END_TYPE;
(*
```

4.2.25 Прикладной компонент identifier

EXPRESS-описание:

```
*)
    TYPE identifier = STRING;
    END_TYPE;
(*
```

4.2.26 Прикладной компонент instance_definition_select

EXPRESS-описание:

```
*)
    TYPE instance_definition_select = SELECT WITH (clock, clock_assignment_relationship, data_in-
    stance, documentation_reference, functionality_instance_reference, general_function_definition,
    general_functionality_instance, general_physical_definition, oo_model_element_select, physical_
    instance, physical_instance_reference, realized_systemp, requirement_definition, requirement_in-
    stance, system_instance, system_view);
    END_TYPE;
(*
```

4.2.27 Прикладной компонент integer_value

EXPRESS-описание:

```
*)
    TYPE integer_value = INTEGER;
    END_TYPE;
(*
```

4.2.28 Прикладной компонент issue_source_select

EXPRESS-описание:

```
*)
    TYPE issue_source_select = SELECT WITH (instance_definition_select);
    END_TYPE;
(*
```

4.2.29 Прикладной компонент justification_assignment_select

EXPRESS-описание:

```
*)
    TYPE justification_assignment_select = SELECT WITH (engineering_process_activityp, instance_
    definition_select, partial_system_ view_relationship, requirement_allocation_relationship);
    END_TYPE;
(*
```

4.2.30 Прикладной компонент label

EXPRESS-описание:

```
*)
    TYPE label = STRING;
    END_TYPE;
(*
```

4.2.31 Прикладной компонент link_select

EXPRESS-описание:

```
*)
    TYPE link_select = SELECT WITH (cb_input_relationship, cb_output_relationship, clock_as-
    signment_relationship, fsm_initial_state_transition, fsm_state_transition, functional_link, oo_ac-
    tion_staten_transition, oo_component_allocation, oo_constraint_model_element_relationship,
    oo_dependency, oo_element_import, oo_element_residencen, oo_extension, oo_generalization,
    oo_generic_association, oo_inclusion, oo_link, oo_message, oo_operation_interface_associa-
    tionn, oo_signal_behavioural_feature_relationship, physical_instance);
    END_TYPE;
(*
```

4.2.32 Прикладной компонент logical_value

EXPRESS-описание:

```
*)
    TYPE logical_value = LOGICAL;
    END_TYPE;
(*
```

4.2.33 Прикладной компонент natural_number
EXPRESS-описание:

```
*)
    TYPE natural_number = INTEGER;
    END_TYPE;
(*
```

4.2.34 Прикладной компонент node_select
EXPRESS-описание:

```
*)
    TYPE node_select = SELECT WITH (cb_place_reference, cb_transition_relationship, clock, fsm_
    state, general_functionality_instance, graphics_view, oo_action, oo_action_staten, oo_actor, oo_
    association_class, oo_class, oo_component, oo_constraint, oo_interfaces, oo_object, oo_pack-
    age, oo_send_action, oo_use_case, physical_instance, textual_paragraphn);
    END_TYPE;
(*
```

4.2.35 Прикладной компонент oo_classifier_or_operation_select
EXPRESS-описание:

```
*)
    TYPE oo_classifier_or_operation_select = SELECT WITH (oo_classifier_select, oo_operation);
    END_TYPE;
(*
```

4.2.36 Прикладной компонент oo_classifier_select
EXPRESS-описание:

```
*)
    TYPE oo_classifier_select = SELECT WITH (oo_actor, oo_class, oo_component, oo_interfaces,
    oo_signal, oo_use_case);
    END_TYPE;
(*
```

4.2.37 Прикладной компонент oo_extended_classifier_select
EXPRESS-описание:

```
*)
    TYPE oo_extended_classifier_select = SELECT WITH (oo_classifier_select, physical_instance);
    END_TYPE;
(*
```

4.2.38 Прикладной компонент oo_extended_model_element_select
EXPRESS-описание:

```
*)
    TYPE oo_extended_model_element_select = SELECT WITH (fsm_generic_state, generic_state_
    context, justification, oo_generalizable_element_select, oo_model_element_select);
    END_TYPE;
(*
```

4.2.39 Прикладной компонент oo_feature_select
EXPRESS-описание:

```
*)
    TYPE oo_feature_select = SELECT WITH (oo_attributen, oo_behavioural_featuree);
    END_TYPE;
(*
```

4.2.40 Прикладной компонент oo_generalizable_element_select

EXPRESS-описание:

```
*)
    TYPE oo_generalizable_element_select = SELECT WITH (oo_collaboration, oo_extended_classi-
    fier_select, oo_generic_association, oo_package, oo_stereotype);
    END_TYPE;
(*
```

4.2.41 Прикладной компонент oo_instance_select

EXPRESS-описание:

```
*)
    TYPE oo_instance_select = SELECT WITH (oo_attributen_instance, oo_extended_classifier_se-
    lect, oo_object);
    END_TYPE;
(*
```

4.2.42 Прикладной компонент oo_model_element_select

EXPRESS-описание:

```
*)
    TYPE oo_model_element_select = SELECT WITH (oo_action, oo_constraint, oo_extension_point,
    oo_feature_select, oo_generic_association_end, oo_interaction, oo_link, oo_link_end, oo_mes-
    sage, oo_parametere, oo_relationship_select, oo_stimulus);
    END_TYPE;
(*
```

4.2.43 Прикладной компонент oo_namespace_select

EXPRESS-описание:

```
*)
    TYPE oo_namespace_select = SELECT WITH (oo_collaboration, oo_extended_classifier_select,
    oo_package);
    END_TYPE;
(*
```

4.2.44 Прикладной компонент oo_relationship_select

EXPRESS-описание:

```
*)
    TYPE oo_relationship_select = SELECT WITH (oo_dependency, oo_extension, oo_generaliza-
    tion, oo_generic_association, oo_inclusion);
    END_TYPE;
(*
```

4.2.45 Прикладной компонент package_element_select

EXPRESS-описание:

```
*)
    TYPE package_element_select = SELECT WITH (configuration_element, configuration_element_
    version, engineering_process_activityp, instance_definition_select, project);
    END_TYPE;
(*
```

4.2.46 Прикладной компонент period_or_date_select

EXPRESS-описание:

```
*)
    TYPE period_or_date_select = SELECT WITH (date_time, project_event_reference, value_with_unite);
    END_TYPE;
(*
```

4.2.47 Прикладной компонент person_organization_assignment_select

EXPRESS-описание:

```
*)
    TYPE person_organization_assignment_select = SELECT WITH (approval, approval_assignment,
    assessment, change_report, configuration_element, configuration_element_version, critical_issue,
    critical_issue_impact, document_assignment, element_critical_issue_relationship, engineering_
    process_activityp, instance_definition_select, justification, justification_relationship, package_
    element_assignment, partial_system_view_relationship, project, requirement_allocation_relation-
    ship, requirement_system_view_assignment, work_ordert);
    END_TYPE;
(*
```

4.2.48 Прикладной компонент person_organization_select

EXPRESS-описание:

```
*)
    TYPE person_organization_select = SELECT WITH (organization, person_in_organization);
    END_TYPE;
(*
```

4.2.49 Прикладной компонент physical_element_role_enumeration

EXPRESS-описание:

```
*)
    TYPE physical_element_role_enumeration = ENUMERATION OF
    (external_element, system_element);
    END_TYPE;
(*
```

4.2.50 Прикладной компонент port_data_relation

EXPRESS-описание:

```
*)
    TYPE port_data_relation = ENUMERATION OF
    (consumer, producer);
    END_TYPE;
(*
```

4.2.51 Прикладной компонент port_position_select

EXPRESS-описание:

```
*)
    TYPE port_position_select = SELECT WITH (io_port, oo_generic_association_end, oo_link_end,
    physical_port);
    END_TYPE;
(*
```

4.2.52 Прикладной компонент port_type

EXPRESS-описание:

```
*)
    TYPE port_type = ENUMERATION OF
    (control, input, mechanism, output);
    END_TYPE;
(*
```

4.2.53 Прикладной компонент property_assignment_select

EXPRESS-описание:

```
*)
    TYPE property_assignment_select = SELECT WITH (data_instance, documentation_reference,
    functionality_instance_reference, general_function_definition, general_physical_definition, oo_
```

```
    model_element_select, package, physical_instance_reference, realized_systemp, requirement_
    definition, system_view);
    END_TYPE;
(*
```

4.2.54 Прикладной компонент property_value_select

EXPRESS-описание:

```
*)
    TYPE property_value_select = SELECT WITH (boolean_value, text_select, value_list, value_with_unite);
    END_TYPE;
(*
```

4.2.55 Прикладной компонент ranked_element_select

EXPRESS-описание:

```
*)
    TYPE ranked_element_select = SELECT WITH (critical_issue_impact, effectiveness_measure,
    instance_definition_select);
    END_TYPE;
(*
```

4.2.56 Прикладной компонент ranking_type

EXPRESS-описание:

```
*)
    TYPE ranking_type = ENUMERATION OF (cost, criticality, priority, project_criticality);
    END_TYPE;
(*
```

4.2.57 Прикладной компонент real_value

EXPRESS-описание:

```
*)
    TYPE real_value = REAL;
    END_TYPE;
(*
```

4.2.58 Прикладной компонент requirement_allocation_select

EXPRESS-описание:

```
*)
    TYPE requirement_allocation_select = SELECT WITH (data_instance, functional_link_reference,
    functionality_instance_reference, functionality_reference_relationship, oo_model_element_select,
    physical_instance_reference);
    END_TYPE;
(*
```

4.2.59 Прикладной компонент single_cardinality_select

EXPRESS-описание:

```
*)
    TYPE single_cardinalityyy_select = SELECT WITH (infinite_cardinality, natural_number);
    END_TYPE;
(*
```

4.2.60 Прикладной компонент specific_element_select

EXPRESS-описание:

```
*)
    TYPE specific_element_select = SELECT WITH (cb_place, fsm_generic_state);
    END_TYPE;
(*
```

4.2.61 Прикладной компонент specification_element_select

EXPRESS-описание:

```
*)
    TYPE specification_element_select = SELECT WITH (change_report, critical_issue, critical_is-
    sue_impact, instance_definition_select);
    END_TYPE;
(*
```

4.2.62 Прикладной компонент system_select

EXPRESS-описание:

```
*)
    TYPE system_select = SELECT WITH (s system_instance, system_view);
    END_TYPE;
(*
```

4.2.63 Прикладной компонент text

EXPRESS-описание:

```
*)
    TYPE text = STRING;
    END_TYPE;
(*
```

4.2.64 Прикладной компонент text_elements

EXPRESS-описание:

```
*)
    TYPE text_elements = SELECT WITH (control_characters, text);
    END_TYPE;
(*
```

4.2.65 Прикладной компонент text_select

EXPRESS-описание:

```
*)
    TYPE text_select = SELECT WITH (text, textual_paragraphn, textual_section, textual_table);
    END_TYPE;
(*
```

4.2.66 Прикладной компонент timing_type

EXPRESS-описание:

```
*)
    TYPE timing_type = ENUMERATION OF (best_case,nominal_case,worst_case);
    END_TYPE;
(*
```

4.2.67 Прикладной компонент trigger_type_enumeration

EXPRESS-описание:

```
*)
    TYPE trigger_type_enumeration = ENUMERATION OF (flank,level);
    END_TYPE;
(*
```

4.2.68 Прикладной компонент verification_allocation_select

EXPRESS-описание:

```
*)
    TYPE verification_allocation_select = SELECT WITH (functionality_instance_reference, physical_
    instance_reference, realized_systemp, requirement_instance, s system_instance);
    END_TYPE;
(*
```

4.3 Определения прикладных компонентов ARM-модели

В данном подразделе определены прикладные компоненты настоящего стандарта. При этом каждый из прикладных компонентов является элементом, который содержит уникальное прикладное понятие и атрибуты, определяющие элементы данных для соответствующего компонента. Все прикладные компоненты и их определения приведены ниже.

4.3.1 Прикладной компонент abstract_data_type_definition

Прикладной компонент abstract_data_type_definition принадлежит к тому же типу, что и прикладной компонент user_defined_data_type_definition, который может принимать следующие значения: любой, все или отсутствие элементов.

EXPRESS-описание:

```
*)
    ENTITY abstract_data_type_definition
    SUBTYPE OF (user_defined_data_type_definition) ;
    END_ENTITY;
(*
```

4.3.2 Прикладной компонент abstract_data_type_member

Прикладной компонент abstract_data_type_member определяет взаимосвязь между прикладными компонентами abstract_data_type_definition и data_instance, содержащимися в нем. Элементы прикладного компонента abstract_data_type_definition независимы друг от друга.

EXPRESS-описание:

```
*)
    ENTITY abstract_data_type_member;
    child : data_instance;
    parent : abstract_data_type_definition;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут child: Этот атрибут определяет прикладной компонент data_instance в указанной взаимосвязи.

Атрибут parent: Этот атрибут определяет прикладной компонент abstract_data_type_definition в указанной взаимосвязи.

4.3.3 Прикладной компонент actual_io_port

Прикладной компонент actual_io_port принадлежит к тому же типу, что и прикладной компонент io_port и элемент интерфейса прикладного компонента general_functionality_instance. Прикладной компонент actual_io_port определяет входной или выходной параметр для ссылочного прикладного компонента general_functionality_instance посредством атрибута port_of.

Примечание — Порты для информационных потоков классифицируются в соответствии с тремя нижеприведенными критериями модели данных

1. Является ли данный порт формальным (закрепленным за типом прикладного компонента general_function_definition) или фактическим (закрепленным за типом прикладного компонента general_functionality_instance).

2. Является ли данный порт входным или выходным.

3. Является ли данный порт информационным или контрольным (информационный порт служит для передачи данных, тогда как контрольный порт предназначен для передачи командных данных, например запуска, останова, приостановки, возобновления и т.п.). Этот вид порта ниже будет определен в прикладном компоненте control_io_port.

Согласно приведенной классификации назначение порта может быть получено в зависимости от объектов flow, имеющихся на этом порте. При этом порт может либо «потреблять» входные/выходные данные (данные в информационном потоке, поступающие на порт), либо «формировать» подобные данные (данные в информационном потоке, поступающие из порта).

Например, прикладной компонент formal_io_port, чьим атрибутом направления потока данных является input, будет формировать данные, тогда как прикладной компонент actual_io_port, чьим атрибутом направления потока является input, будет потреблять данные. В модели данных эта информация определяется с помощью предоставляемого атрибута role.

Указанный атрибут используется для гарантии того, что входные/выходные информационные объекты будут предоставляться должным образом (т. е. они должны иметь одного поставщика и одного потребителя), а также того, что прикладные компоненты io_port_binding будут применяться только к портам, для которых прикладной компонент actual_port в связке будет относиться к поставщику информации, а прикладной компонент formal_port в этой же связке — к потребителю информации.

EXPRESS-описание:

```
*)
    ENTITY actual_io_port
    SUBTYPE OF (io_port) ;
    port_of : general_functionality_instance;
    DERIVE
    SELF\io_port. RENAMED role: port_data_relation : determineactualportrole(SELF);
    INVERSE
    assigned_buffer : SET[0:1] OF io_buffer FOR assigned_to;
    UNIQUE
    UR1: port_of, io_port_number, port_type;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут port_of: Этот атрибут определяет прикладной компонент general_functionality_instance, для которого этот атрибут является частью компонента.

Атрибут role: Этот атрибут определяет поставщика или потребителя информации.

Атрибут assigned_buffer: Этот атрибут определяет прикладной компонент actual_io_port, за которым закрепляется прикладной компонент io_buffer.

Формальные выражения:

UR1:

4.3.4 Прикладной компонент actual_physical_port

Прикладной компонент actual_physical_port принадлежит тому же типу, что и прикладной компонент physical_port, и представляет элемент в интерфейсе прикладного компонента physical_instance. Направление передачи входных/выходных данных с помощью этого интерфейса не определяется прикладным компонентом actual_physical_port.

EXPRESS-описание:

```
*)
    ENTITY actual_physical_port
    SUBTYPE OF (physical_port);
    port_of : physical_instance;
    END_ENTITY;
(*
```

Определение атрибута:

Атрибут port_of: Этот атрибут определяет прикладной компонент physical_instance, для которого этот атрибут является частью интерфейса.

4.3.5 Прикладной компонент actual_port_position

Прикладной компонент actual_port_position принадлежит к тому же типу, что и прикладной компонент visual_element и является графическим представлением положения прикладного компонента actual_io_port или actual_physical_port. Это положение представляется с помощью прикладного компонента graphics_point, поскольку фактический порт может занимать множество различных положений (объектом на порте может быть физически протяженный объект, например прямоугольник, а не одиночная точка). Прикладной компонент actual_port_position должен указывать положение на границе прикладного компонента graphics_node для объекта на связанном с ним порте.

Примечание — Данное правило на языке EXPRESS не формализовано.

EXPRESS-описание:

```
*)
    ENTITY actual_port_position
    SUBTYPE OF (visual_element) ;
    assigned_to : graphics_node;
    position : graphics_point;
    positioned_port : port_position_select;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут assigned_to: Этот атрибут определяет прикладной компонент graphics_node, который обеспечивает визуальное представление информации для прикладного компонента actual_port_position.

Атрибут position: Этот атрибут визуально характеризует расположение порта.

Атрибут positioned_port: Этот атрибут определяет порт, для которого действительна визуальная информация о расположении порта.

4.3.6 Прикладной компонент address

Прикладной компонент address определяет адрес прикладного компонента person или organization. Все атрибуты прикладного компонента address являются необязательными, однако этот компонент должен иметь по крайней мере один ненулевой атрибут.

Примечание — Этот прикладной компонент непосредственно заимствуется из PDM-схемы и, возможно, извлекается из ARM-модели.

EXPRESS-описание:

```
*)
    ENTITY address;
    country : OPTIONAL label;
    electronic_mail_address : OPTIONAL label;
    facsimile_number : OPTIONAL label;
    internal_location : OPTIONAL label;
    postbox_number : OPTIONAL label;
    postcode : OPTIONAL label;
    region : OPTIONAL label;
    street : OPTIONAL label;
    street_number : OPTIONAL label;
    telephone_number : OPTIONAL label;
    telex_number : OPTIONAL label;
    town : OPTIONAL label;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут country: Этот атрибут определяет страну, связанную с прикладным компонентом address.

Атрибут electronic_mail_address: Этот атрибут определяет электронный адрес, связанный с данным прикладным компонентом address.

Атрибут facsimile_number: Этот атрибут определяет факс, связанный с данным прикладным компонентом address.

Атрибут internal_location: Этот атрибут устанавливает определенное местоположение объекта в прикладном компоненте address.

Пример 1 — Атрибутом internal_location может быть номер комнаты или этаж в здании.

Атрибут postbox_number: Этот атрибут определяет почтовый ящик, связанный с данным прикладным компонентом address.

Атрибут postcode: Этот атрибут определяет почтовый индекс, связанный с данным прикладным компонентом address.

Атрибут region: Этот атрибут определяет регион, связанный с данным прикладным компонентом address.

Атрибут street: Этот атрибут определяет улицу, связанную с данным прикладным компонентом address.

Атрибут street_number: Этот атрибут определяет номер улицы, связанный с данным прикладным компонентом address.

Атрибут telephone_number: Этот атрибут определяет номер телефона, связанный с данным прикладным компонентом address.

Атрибут telex_number: Этот атрибут определяет номер телекса, связанный с данным прикладным компонентом address.

Атрибут town: Этот атрибут определяет город, связанный с данным прикладным компонентом address.

4.3.7 Прикладной компонент aggregate_data_type_definition

Прикладной компонент aggregate_data_type_definition принадлежит к тому же типу, что и прикладной компонент user_defined_data_type_definition, и содержит перечень (перечни) значений.

EXPRESS-описание:

```
*)
    ENTITY aggregate_data_type_definition
    SUBTYPE OF (user_defined_data_type_definition);
    aggregate_type : data_type_definition_select;
    bound : LIST[1:?] OF integer_interval;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут aggregate_type: Этот атрибут определяет значения, которые может принимать каждый элемент в перечне.

Атрибут bound: Этот атрибут определяет упорядоченное множество элементов прикладного компонента aggregate_data_type_definition. Предполагается, что с целью присвоения составных значений они будут сохраняться в виде строки.

4.3.8 Прикладной компонент approval

Прикладной компонент approval является утверждением, касающимся качества тех данных о продукции, которые подлежат утверждению. Прикладной компонент approval представляет собой формальное утверждение, сделанное техническим или управленческим персоналом относительно того, будут ли некоторые требования восприниматься надзорным органом как выполненные.

Примечание 1 — Это необходимо согласовать для применимости текста.

Примечание 2 — Данный прикладной компонент заимствован из PDM-схемы.

EXPRESS-описание:

```
*)
    ENTITY approval;
    level : OPTIONAL label;
    status : label;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут level: Этот атрибут определяет вид операции, которая может выполняться и предоставляться с помощью прикладного компонента approval. Там, где это применимо, должны использоваться следующие состояния (значения) этого атрибута:

- состояние disposition: утвержденный объект утвержден и для серийной продукции;
- состояние equipment order: утвержденный объект достиг состояния, при котором изменениям будет подвергаться определенный процесс и оборудование, необходимое для производства (которые могут заказываться);
- состояние planning: утвержденный объект является технически законченным и достигшим достаточной стабильности, на базе которого могут основываться другие конструкции.

Атрибут status: Этот атрибут определяет утверждение, сделанное относительно производственных данных в рамках данного прикладного компонента approval.

4.3.9 Прикладной компонент approval_assignment

Прикладной компонент approval_assignment определяет присвоение прикладного компонента approval элементу системной спецификации.

EXPRESS-описание:

```
*)
    ENTITY approval_assignment;
    assigned_approval : approval;
    assigned_to : approval_assignment_select;
    description : OPTIONAL text_select;
    status : label;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут assigned_approval: Этот атрибут определяет присвоенный прикладной компонент approval.

Атрибут assigned_to: Этот атрибут определяет элемент системной инженерной спецификации, для которого действителен прикладной компонент approval_assignment.

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту approval_assignment.

Атрибут status: Этот атрибут определяет утверждение, сделанное относительно элемента системной инженерной спецификации в рамках данного прикладного компонента approval_assignment.

4.3.10 Прикладной компонент approval_person_organization

Прикладной компонент approval_person_organization определяет взаимосвязь между прикладным компонентом approval и юридическим лицом, связанным с утверждением данных. Семантика этой взаимосвязи будет определена ниже с помощью атрибута role.

EXPRESS-описание:

```
*)
    ENTITY approval_person_organization;
    authorized_approval : approval;
    person_organization : person_organization_select;
    role : label;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут authorized_approval: Этот атрибут определяет утверждение, сделанное для прикладного компонента approval_person_organization.

Атрибут person_organization: Этот атрибут определяет юридический орган, связанный с прикладным компонентом approval_person_organization.

Атрибут role: Этот атрибут определяет семантику прикладного компонента approval_person_organization.

Там, где это применимо, должны использоваться следующие состояния (значения) этого атрибута:

- состояние project responsible: утверждающее лицо будет нести ответственность за проект, с которым связана системная спецификация на элементы;
- состояние verification responsible: утверждающее лицо будет нести ответственность за верификацию системной спецификации на элементы.

4.3.11 Прикладной компонент approval_relationship

Прикладной компонент approval_relationship определяет взаимосвязь между двумя прикладными компонентами approval.

Примечание — Для определения семантики указанной взаимосвязи необходимо вводить атрибут.

EXPRESS-описание:

```
*)
    ENTITY approval_relationship;
    description : OPTIONAL text_select;
```

related_approval : approval;
relating_approval : approval;
relation_type : label;
END_ENTITY;
(*
Определения атрибутов:
Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту approval_relationship.
Атрибут related_approval: Этот атрибут определяет второй прикладной компонент approval в указанной взаимосвязи.
Атрибут relating_approval: Этот атрибут определяет первый прикладной компонент approval в указанной взаимосвязи.
Атрибут relation_type: Этот атрибут определяет тип прикладного компонента approval_relationship.

4.3.12 Прикладной компонент assessment

Прикладной компонент assessment определяет оценку состояния элемента в системной инженерной спецификации в процессе ее разработки. Семантика этой оценки содержится в атрибуте assessment_type.

EXPRESS-описание:

```
*)
    ENTITY assessment;
    assessed_level : label;
    assessment_type : label;
    assigned_to : assessment_assignment_select;
    description : OPTIONAL text_select;
    END_ENTITY;
(*
```

Определения атрибутов:
Атрибут assessed_level: Этот атрибут определяет оценочное значение.
Атрибут assessment_type: Этот атрибут определяет тип и назначение проводимой оценки. Там, где это применимо, должны использоваться следующие состояния (значения) этого атрибута:
- состояние maturity: прикладной компонент assessment характеризует оценку готовности элемента;
- состояние completeness: прикладной компонент assessment характеризует оценку уровня законченности присвоенного элемента;
- состояние risk: прикладной компонент assessment характеризует оценку проектного риска, связанного с присвоенным элементом;
- состояние stability: прикладной компонент assessment характеризует оценку стабильности присвоенного элемента.

Атрибут assigned_to: Этот атрибут определяет элемент системной инженерной спецификации, для которого действителен прикладной компонент assessment.
Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту assessment.

4.3.13 Прикладной компонент assessment_relationship

Прикладной компонент assessment_relationship определяет взаимосвязь между двумя прикладными компонентами assessment. Семантика этой взаимосвязи содержится в атрибуте relationship_type.

EXPRESS-описание:

```
*)
    ENTITY assessment_relationship;
    description : OPTIONAL text_select;
    related : assessment;
    relating : assessment;
    relationship_type : label;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту assessment_relationship.

Атрибут related (связанный): Этот атрибут определяет второй из двух прикладных компонентов assessment в указанной взаимосвязи.

Атрибут relating (связывающий): Этот атрибут определяет первый из двух прикладных компонентов assessment в указанной взаимосвязи.

Атрибут relationship_type: Этот атрибут определяет семантику прикладного компонента assessment_relationship. Там, где это применимо, должны использоваться следующие состояния (значения):

- состояние replacement: связывающий прикладной компонент assessment заменяет связанный прикладной компонент assessment;
- состояние conflict: связывающий прикладной компонент assessment идентифицируется как вступающий в конфликт со связанным прикладным компонентом assessment;
- состояние complement: связывающий прикладной компонент assessment идентифицируется как дополнительный к связанному прикладному компоненту assessment.

4.3.14 Прикладной компонент bi_directional_port_indicator

Прикладной компонент bi_directional_port_indicator определяет связь между двумя прикладными компонентами io_port, которые принимают и формируют один и тот же прикладной компонент data_instance object.

Примечание 1 — Прикладной компонент bi_directional_port_indicator определяет способ представления двунаправленных портов с использованием пары прикладных компонентов io_port. Атрибуты consuming_port и producing_port должны идентифицировать один и тот же прикладной компонент data_instance посредством атрибута data. Кроме того, прикладной компонент io_port, определяемый с помощью атрибута producing_port, должен формировать прикладные компоненты data_instance, а прикладной компонент io_port, определяемый с помощью атрибута consuming_port — формировать прикладные компоненты data_instance.

Примечание 2 — Атрибут port_of обоих прикладных компонентов io_port в прикладном компоненте bi_directional_port_indicator должен объединяться с тем же функциональным объектом, который должен быть либо прикладным компонентом general_functionality_instance, либо прикладным компонентом general_function_definition.

EXPRESS-описание:

```
*)
    ENTITY bi_directional_port_indicator;
    consuming_port : io_port;
    producing_port : io_port;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут consuming_port: Этот атрибут определяет прикладной компонент io_port в прикладном компоненте bi_directional_port_indicator, который идентифицирует порт, принимающий данные.

Атрибут producing_port: Этот атрибут определяет прикладной компонент io_port в прикладном компоненте bi_directional_port_indicator, который идентифицирует порт, выдающий данные.

4.3.15 Прикладной компонент binary_data_type_definition

Прикладной компонент binary_data_type_definition принадлежит к тому же типу, что и прикладной компонент elementary_maths_space, и содержит значения от 0 до $2n - 1$, где n — это размер (или число битов) для этого типа данных.

Примечание — Нередки такие случаи с сигналами на шине данных, когда ряд сегментов с информацией упаковывается в одно слово.

Пример 2 — Слово из восьми бит разбивают для передачи четырех сегментов с информацией о насосе.

- Биты 1–3 идентифицируют насос (система содержит 8 насосов, пронумерованных цифрами от 0 до 7).

- Биты 4–6 переносят информацию о состоянии насоса (насос может иметь 8 состояний, пронумерованных цифрами от 0 до 7).

- ***Бит 7 указывает направление нагнетания насоса (состояние 0 — вперед, состояние 1 — назад),***

- ***Бит 8 указывает состояние насоса (вкл/выкл) (состояние 0 — выкл, состояние 1 — вкл).***

Группа битов не объединяется. Биты формируют единственное значение и занимают определенную позицию в слове. Обычно отправитель и получатель данных используют битовые маски для работы с частями данных в каком-либо слове.

EXPRESS-описание:

```
*)
    ENTITY binary_data_type_definition
    SUBTYPE OF (elementary_maths_space) ;
    size : finite_integer_interval;
    END_ENTITY;
(*
```

Определение атрибута:

Атрибут size: Этот атрибут определяет конечный интервал, который определяет максимальную длину прикладного компонента binary_data_type_definition.

4.3.16 Прикладной компонент boolean_data_type_definition

Прикладной компонент boolean_data_type_definition принадлежит к тому же типу, что и прикладной компонент elementary_maths_space, и принимает состояние TRUE или FALSE.

EXPRESS-описание:

```
*)
    ENTITY boolean_data_type_definition
    SUBTYPE OF (elementary_maths_space) ;
    END_ENTITY;
(*
```

4.3.17 Прикладной компонент cardinality_list

Прикладной компонент cardinality_list определяет действующий диапазон значений.

EXPRESS-описание:

```
*)
    ENTITY cardinality_list;
    range : LIST[2:?] OF cardinality_range;
    END_ENTITY;
(*
```

Определение атрибута:

Атрибут range: Этот атрибут определяет неперекрывающийся перечень прикладных компонентов cardinality_range в строго возрастающем порядке, который указывает действующие интервалы для прикладного компонента cardinality_list.

4.3.18 Прикладной компонент cardinality_range

Прикладной компонент cardinality_range представляет собой пару значений, определяющих нижнюю и верхнюю границы их интервала. Значение для атрибута lower_bound должно быть меньше или равно значению атрибута higher_bound. Область значений для атрибутов состоит из натуральных чисел.

EXPRESS-описание:

```
*)
    ENTITY cardinality_range;
    lower_bound : single_cardinality;
    upper_bound : single_cardinality;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут lower_bound: Этот атрибут определяет нижнюю границу интервала значений.

Атрибут upper_bound: Этот атрибут определяет верхнюю границу интервала значений.

4.3.19 Прикладной компонент causal_block_bound

Прикладной компонент causal_block_bound определяет взаимосвязь между двумя прикладными компонентами cb_transition_relationship и указывает область фрагмента в прикладном компоненте cb_functional_behaviour_model.

Примечание — Этот прикладной компонент является расширением математического описания сетей Петри, который необходим для правильного представления параллельных и условных ветвей в причинно-следственной модели.

EXPRESS-описание:

```
*)
    ENTITY causal_block_bound;
    initial_transition : cb_transition_relationship;
    terminal_transition : cb_transition_relationship;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут initial_transition: Этот атрибут определяет исходный прикладной компонент cb_transition_relationship в модуле.

Атрибут terminal_transition: Этот атрибут определяет конечный прикладной компонент cb_transition_relationship в модуле.

4.3.20 Прикладной компонент cb_completion_alternative

Прикладной компонент cb_completion_alternative дает представление конечного прикладного компонента cb_functional_place для обеспечения потока данных в прикладном компоненте cb_functional_behaviour_model. Конечный прикладной компонент cb_functional_place в модели поведения идентифицируется с помощью атрибута final_ model_element. В тех случаях, когда существует два и более конечных прикладных компонента cb_functional_place (где каждый из них является исполняемым потоком данных), должен существовать один прикладной компонент cb_completion_alternative, характеризующий каждый из потоков.

Примечание — Прикладной компонент cb_completion_alternative вводится для преобразования множества выходных условий из декомпозируемой функции в соответствующие выходные условия.

EXPRESS-описание:

```
*)
    ENTITY cb_completion_alternative;
    completes_model : cb_functional_behaviour_model;
    final_model_element : cb_functional_place;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут completes_model: Этот атрибут определяет прикладной компонент cb_functional_behaviour_model, для которого данный атрибут определяет состояние выхода.

Атрибут final_model_element: Этот атрибут определяет прикладной компонент cb_functional_place, который является последним в выполняемом (исполняемом) потоке данных для прикладного компонента cb_functional_behaviour_model.

4.3.21 Прикладной компонент cb_completion_alternative_mapping

Прикладной компонент cb_completion_alternative_mapping определяет связь прикладного компонента cb_completion_alternative разделенной функции эквивалентного прикладного компонента cb_functional_place на один уровень исполнения выше.

Примечание — Прикладной компонент cb_completion_alternative_mapping определяет преобразование прикладного компонента cb_completion_alternative прикладного компонента cb_functional_behaviour_model в прикладной компонент cb_functional_transition. Семантика состоит в выборе окончательного альтернативного варианта сложной функции, которая должна интерпретироваться как выбор прикладного компонента cb_functional_transition, на уровне на один выше в иерархии.

EXPRESS-описание:

```
*)
    ENTITY cb_completion_alternative_mapping;
    child_completion_criterion : cb_completion_alternative;
    equivalent_transition : cb_functional_transition;
    scope : cb_place_function_association;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут child_completion_criterion: Этот атрибут определяет прикладной компонент cb_completion_alternative в прикладном компоненте cb_completion_alternative_mapping.

Атрибут equivalent_transition: Этот атрибут определяет прикладной компонент cb_functional_transition, для которого прикладной компонент child_completion_criterion преобразуется с помощью прикладного компонента cb_completion_alternative_mapping.

Атрибут scope: Этот атрибут определяет прикладной компонент cb_place_function_association прикладного компонента cb_completion_alternative_mapping.

4.3.22 Прикладной компонент cb_firing_condition

Прикладной компонент cb_firing_condition представляет собой спецификацию условий, которые должны выполняться для инициализации перехода к прикладному компоненту cb_functional_behaviour_model.

Примечание — Прикладной компонент cb_firing_condition связывается с прикладным компонентом cb_output_relationship, который предшествует прикладному компоненту cb_transition и который должен инициализироваться при выполнении поставленного условия.

EXPRESS-описание:

```
*)
    ENTITY cb_firing_condition;
    condition_definition : textual_specification;
    guarded_transition : cb_transition;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут condition_definition: Этот атрибут определяет условие, которое должно выполняться для инициализации перехода.

Атрибут guarded_transition: Этот атрибут определяет прикладной компонент cb_output_relationship, для которого применим прикладной компонент cb_transition.

4.3.23 Прикладной компонент cb_functional_behaviour_model

Прикладной компонент cb_functional_behaviour_model принадлежит к тому же типу, что и прикладной компонент functional_behaviour_model, а модель представления определяет частичное функциональное упорядочивание, например функциональную последовательность, взаимосовместимость или ветвление среди функций.

Примечание 1 — При присвоении прикладного компонента composite_function_definition функциональное представление определяется с помощью модели функционального взаимодействия для прикладных компонентов composite_function_definition и cb_functional_behaviour_model в их сочетании.

Примечание 2 — Компоновочные модули прикладного компонента cb_functional_behaviour_model эквивалентны таковым для сетей Петри. Они создаются из прикладных компонентов cb_place, cb_transition, cb_input_relationship и cb_output_relationship.

EXPRESS-описание:

```
*)
    ENTITY cb_functional_behaviour_model
    SUBTYPE OF (functional_behaviour_model);
    model_boundedness : label;
    model_type : label;
    INVERSE
```

constituent_places: SET[1:?] OF cb_functional_place FOR behaviour_model;
constituent_transitions: SET[2:?] OF cb_functional_transition FOR behaviour_model;
END_ENTITY;
(*
Определения атрибутов:

Атрибут model_boundedness: Этот атрибут определяет условие, накладываемое на максимальное число знаков, допускаемых в каждом прикладном компоненте cb_place прикладного компонента cb_functional_behaviour_model.

Примечание 3 — Для причинно-следственного формализма, популярного среди системных инженеров FFDB и диаграмм представления, модель ограничивается значением 1, за исключением случаев, когда в модели используется принцип репликации. В последнем случае атрибут model_boundedness будет иметь максимальное число потоков данных в реплицированной структуре.

Атрибут model_type: Этот атрибут определяет тип модели представления, которая реализуется с помощью прикладных компонентов cb_functional_behaviour_model и composite_function_definition.

Атрибут constituent_places: Этот атрибут определяет прикладной компонент cb_functional_behaviour_model, частью которого является прикладной компонент cb_functional_place.

Атрибут constituent_transitions: Этот атрибут определяет прикладной компонент cb_functional_behaviour_model, частью которого является прикладной компонент cb_functional_transition.

4.3.24 Прикладной компонент cb_functional_place

Прикладной компонент cb_functional_place принадлежит тому же типу, что и прикладной компонент cb_place и специализирован на фиксации статических состояний прикладного компонента functional_behaviour_model.

Примечание — Спецификация на причинно-следственное представление формируется путем соединения прикладных компонентов cb_functional_place и cb_functional_transition с помощью прикладных компонентов cb_input_relationship и cb_output_relationship. Необходимость соединения нескольких прикладных компонентов cb_output_relationship с прикладным компонентом cb_functional_transition будет означать, что он становится первым элементом в параллельной ветви в спецификации на представление.

Необходимость присоединения нескольких прикладных компонентов cb_input_relationship к прикладному компоненту cb_functional_transition будет означать, что параллельная ветвь будет прерываться. Альтернативные ветви могут определяться путем присвоения нескольких прикладных компонентов cb_output_relationship единственному прикладному компоненту cb_functional_place. Аналогично прерывание альтернативных потоков данных может задаваться путем объединения нескольких прикладных компонентов cb_input_relationship в единственный прикладной компонент cb_functional_place.

EXPRESS-описание:
*)

ENTITY cb_functional_place
SUBTYPE OF (cb_place) ;
behaviour_model: cb_functional_behaviour_model;
INVERSE
reference_information: SET[0:1] OF cb_place_reference FOR functional_place_reference;
END_ENTITY;
(*
Определения атрибутов:

Атрибут behaviour_model: Этот атрибут определяет прикладной компонент cb_functional_behaviour_model, частью которого является прикладной компонент cb_functional_place.

Атрибут reference_information: Этот атрибут определяет прикладной компонент cb_functional_place, для которого определен прикладной компонент cb_place_reference.

4.3.25 Прикладной компонент cb_functional_transition

Прикладной компонент cb_functional_transition принадлежит к тому же типу, что и прикладной компонент cb_transition и специализирован для определения изменяемого узла (точки) в прикладном компоненте cb_functional_behaviour_model.

Примечание — Спецификация на причинно-следственное представление формируется путем соединения прикладных компонентов cb_functional_place и cb_functional_transition с помощью прикладных компонентов

cb_input_relationship и cb_output_relationship. Необходимость объединения нескольких прикладных компонентов cb_output_relationship с прикладным компонентом cb_functional_transition будет означать, что он станет первым элементом на параллельной ветви в спецификации.

Необходимость объединения нескольких прикладных компонентов cb_input_relationship с прикладным компонентом cb_functional_transition будет означать, что параллельная ветвь будет прерываться. Альтернативные ветви могут определяться путем присвоения нескольких прикладных компонентов cb_output_relationship единственному прикладному компоненту cb_functional_place. Аналогично прерывание альтернативных потоков данных может задаваться путем объединения нескольких прикладных компонентов cb_input_relationship в единственный прикладной компонент cb_place.

EXPRESS-описание:

```
*)
    ENTITY cb_functional_transition
    SUBTYPE OF (cb_transition);
    behaviour_model: cb_functional_behaviour_model;
    INVERSE
    transition_relationship: SET[0:1] OF cb_transition_relationship FOR related_transition;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут behaviour_model: Этот атрибут определяет прикладной компонент cb_functional_behaviour_model, частью которого является прикладной компонент cb_functional_transition.

Атрибут transition_relationship: Этот атрибут определяет набор прикладных компонентов cb_transition, которые включаются в прикладной компонент cb_functional_transition.

4.3.26 Прикладной компонент cb_initial_marking

Прикладной компонент cb_initial_marking используется для указания того, что прикладной компонент cb_place в прикладном компоненте cb_functional_behaviour_model является исходным условием для модели поведения.

EXPRESS-описание:

```
*)
    ENTITY cb_initial_marking;
    marked_place : cb_place;
    number_of_tokens : INTEGER;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут marked_place: Этот атрибут указывает на то, что прикладной компонент cb_place является частью исходного условия для прикладного компонента cb_functional_behaviour_model.

Атрибут number_of_tokens: Этот атрибут определяет число знаков, присваиваемых атрибуту marked_place.

4.3.27 Прикладной компонент cb_input_relationship

Прикладной компонент cb_input_relationship определяет однонаправленную связь от прикладного компонента cb_place к прикладному компоненту cb_transition, указывающую на причинно-следственное ограничительное условие, накладываемое на прикладные компоненты cb_place и cb_transition.

Примечание 1 — Атрибут causal_weight определяет число знаков, необходимых для инициализации перехода.

Примечание 2 — По установившейся в сетях Петри терминологии прикладной компонент cb_input_relationship соответствует входной дуге.

EXPRESS-описание:

```
*)
    ENTITY cb_input_relationship;
    causal_weight : causal_weight_select;
    destination_transition : cb_transition;
```

source_place : cb_place;
END_ENTITY;
(*
Определения атрибутов:
Атрибут causal_weight: Этот атрибут определяет число знаков в прикладном компоненте cb_place, которое необходимо для инициализации перехода. Атрибут causal_weight выражается натуральным числом.
Атрибут destination_transition: Этот атрибут определяет прикладной компонент cb_transition в указанной связи.
Атрибут source_place: Этот атрибут определяет прикладной компонент cb_place в указанной связи.

4.3.28 Прикладной компонент cb_output_relationship

Прикладной компонент cb_output_relationship определяет однонаправленную связь от прикладного компонента cb_transition к прикладному компоненту cb_place, указывающую на причинно-следственное ограничительное условие, накладываемое на прикладные компоненты cb_transition и cb_place.

Примечание 1 — По установившейся в сетях Петри терминологии прикладной компонент cb_output_relationship соответствует входной дуге. При инициализации перехода атрибут causal_weight будет определять способ формирования знаков в атрибуте destination_place.

EXPRESS-описание:
*)
ENTITY cb_output_relationship;
causal_weight : causal_weight_select;
destination_place : cb_place;
source_transition : cb_transition;
END_ENTITY;
(*
Определения атрибутов:
Атрибут causal_weight: Этот атрибут определяет число знаков, формируемых в атрибуте destination_place.

Примечание 2 — Для применения не в сетях Петри значение атрибута causal_weight должно приниматься равным 1. То же самое будет относиться и к безопасным сетям Петри.

Атрибут destination_place: Этот атрибут определяет прикладной компонент cb_place, который вводится после инициализации прикладного компонента cb_transition, определенного с помощью атрибута source_transition.
Атрибут source_transition: Этот атрибут определяет прикладной компонент cb_transition, из которого формируется атрибут source_transition.

4.3.29 Прикладной компонент cb_place

Прикладной компонент cb_place представляет собой определение статического состояния в причинно-следственной функциональной модели представления.

Примечание — По терминологии, принятой в сетях Петри, прикладной компонент cb_place называется «местом».

EXPRESS-описание:
*)
ENTITY cb_place
ABSTRACT SUPERTYPE OF (ONEOF(cb_functional_place, oo_action_staten);
description: OPTIONAL text_select;
place_label: OPTIONAL label;
INVERSE
initial_marking: SET[0:1] OF cb_initial_marking FOR marked_place;
END_ENTITY;
(*
Определения атрибутов:
Атрибут description: Этот атрибут определяет дополнительную информацию относительно прикладного компонента cb_place.

Атрибут place_label: Этот атрибут определяет слово (или слова), которые используются для ссылки на прикладной компонент cb_place.

Атрибут initial_marking: Этот атрибут определяет прикладной компонент cb_place, который является частью исходного условия для прикладного компонента cb_functional_behaviour_model.

4.3.30 Прикладной компонент cb_place_function_association

Прикладной компонент cb_place_function_association определяет взаимосвязь между двумя прикладными компонентами function_instance и cb_place, при которой активация функции будет контролироваться с помощью прикладного компонента cb_place.

Примечание — Взаимно-однозначного соответствия между прикладными компонентами cb_place и function_instance не требуется. Прикладной компонент cb_place без присвоенного прикладного компонента function_instance будет указывать метку-заполнитель, вводимую для синхронизации потоков данных.

EXPRESS-описание:

```
*)
    ENTITY cb_place_function_association;
    causal_place : cb_functional_place;
    controls_function : function_instance;
    INVERSE
    completion_mapping: SET[0:?] OF cb_completion_alternative_mapping FOR scope;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут causal_place: Этот атрибут определяет прикладной компонент cb_place в указанной взаимосвязи.

Атрибут controls_function: Этот атрибут определяет прикладной компонент function_instance в указанной взаимосвязи.

Атрибут completion_mapping: Этот атрибут определяет прикладной компонент cb_place_function_association в прикладном компоненте cb_completion_alternative_mapping.

4.3.31 Прикладной компонент cb_place_reference

Прикладной компонент cb_place_reference определяет взаимосвязь между прикладным компонентом cb_place и дополнительной информацией. Семантика этой взаимосвязи содержится в атрибуте reference_type.

Примечание 1 — Прикладной компонент cb_place_reference вводится для указания того, что частный прикладной компонент cb_place должен быть аннотирован с помощью определенного синтаксического символа, который первоначально вводился в понятия FFBD и диаграммы представлений.

EXPRESS-описание:

```
*)
    ENTITY cb_place_reference;
    functional_place_reference : cb_functional_place;
    reference_type : label;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут functional_place_reference: Этот атрибут определяет прикладной компонент cb_functional_place, для которого определен прикладной компонент cb_place_reference.

Атрибут reference_type: Этот атрибут определяет собственную семантику. Там, где это применимо, должны использоваться следующие состояния (значения) этого атрибута:

Примечание 2 — Перечисленные ниже предпочтительные состояния (значения) введены для правильного выражения FFBD и диаграммы представлений с целью указания выхода из контуров или окончания исполняемого потока данных.

- Состояние thread_terminiation: Прикладной компонент cb_place, индицируемый с помощью атрибута functional_place_reference, является конечным элементом в исполняемом потоке данных;
- Состояние loop_exit: Прикладной компонент cb_place, индицируемый с помощью атрибута functional_place_reference, является конечным элементом контура.

4.3.32 Прикладной компонент cb_transition

Прикладной компонент cb_transition определяет временный пункт (точку) в причинно-следственной модели представления.

Примечание — По терминологии, принятой в сетях Петри, прикладной компонент cb_transition называется «переходом».

EXPRESS-описание:

```
*)
    ENTITY cb_transition
    ABSTRACT SUPERTYPE OF (ONEOF(cb_functional_transition, oo_action_staten_transition);
    description: OPTIONAL text_select;
    transition_label: OPTIONAL label;
    INVERSE
    guarded_by: SET[0:1] OF cb_firing_condition FOR guarded_transition;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут description: Этот атрибут определяет дополнительную текстовую информацию, относящуюся к прикладному компоненту cb_transition.

Атрибут transition_label: Этот атрибут определяет слово (или слова), которые используются для ссылки на прикладной компонент cb_transition.

Атрибут guarded_by: Этот атрибут определяет прикладной компонент cb_output_relationship, для которого применим прикладной компонент cb_transition.

4.3.33 Прикладной компонент cb_transition_relationship

Прикладной компонент cb_transition_relationship определяет взаимосвязь между двумя прикладными компонентами cb_transition.

Примечание 1 — Назначение этого прикладного компонента состоит в группировании разделенных ветвей в причинно-следственную цепочку. Семантика может также охватываться путем небольшого расширения прикладного компонента causal_block_bound.

Примечание 2 — Этот прикладной компонент, возможно, должен удаляться из модели.

EXPRESS-описание:

```
*)
    ENTITY cb_transition_relationship;
    related_transition : SET[1:?] OF cb_functional_transition;
    elationship_type : label;
    INVERSE
    end_bound : SET[0:1] OF causal_block_bound FOR terminal_transition;
    start_bound : SET[0:1] OF causal_block_bound FOR initial_transition;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут related_transition: Этот атрибут определяет набор прикладных компонентов cb_transition, который включается в прикладной компонент cb_functional_transition.

Атрибут relationship_type: Этот атрибут определяет характер указанной взаимосвязи. Там, где это применимо, должны использоваться следующие состояния (значения) этого атрибута:

- Состояние or: Прикладной компонент cb_transition_relationship представляет собой набор прикладных компонентов cb_functional_transition (см. раздел 4.2.25 ISO/WD PAS 20542) и указывает начало или окончание селективной структуры.

Примечание 3 — В языках программирования он соответствует состояниям (командам) IF .. THEN .. ELSE.

- Состояние and: Прикладной компонент cb_transition_relationship представляет собой набор прикладных компонентов cb_functional_transition и указывает начало или окончание параллельной структуры;

- Состояние loop: Прикладной компонент cb_transition_relationship представляет собой набор прикладных компонентов cb_functional_transition и указывает начало или окончание фиксированного контура.

Примечание 4 — Использование ключевого слова «контур» предполагает, что число итераций в этом контуре жестко закодировано.

- Состояние iteration: Прикладной компонент cb_transition_relationship представляет собой набор прикладных компонентов cb_functional_transition и указывает начало или окончание итеративной структуры, при которых выходное условие оценивается в динамическом режиме;
- Состояние replication: Прикладной компонент cb_transition_relationship представляет собой набор прикладных компонентов cb_functional_transition и указывает начало или окончание реплицированной структуры.

Примечание 5 — Репликация — это представление многофункциональных исполняемых потоков данных с помощью однофункциональной структуры.

Атрибут end_bound: Этот атрибут определяет конечный прикладной компонент cb_transition_relationship в модуле.

Атрибут start_bound: Этот атрибут определяет начальный прикладной компонент cb_transition_relationship в модуле.

4.3.34 Прикладной компонент cb_transition_unbounded_weight

Прикладной компонент cb_transition_unbounded_weight определяет способ индикации того, что ограниченность прикладного компонента cb_input_relationship или cb_output_relationship будет определяться во время работы. Ограничение, определяемое с помощью прикладного компонента cb_transition_unbounded_weight, выражается натуральным числом, большим или равным указанному в атрибуте minimal_weight.

Примечание — Это ограничение определяет число знаков, формируемых с помощью прикладного компонента cb_input_relationship, или число знаков, принимаемых с помощью прикладного компонента cb_output_relationship.

EXPRESS-описание:

```
*)
    ENTITY cb_transition_unbounded_weight;
    minimal_weight : natural_number;
    END_ENTITY;
(*
```

Определение атрибута:

Атрибут minimal_weight: Этот атрибут определяет минимальное число знаков (принимаемых или передаваемых).

4.3.35 Прикладной компонент change_order

Прикладной компонент change_order принадлежит к тому же типу, что и прикладной компонент work_order, который допускает внесение одного или нескольких изменений.

Примечание — Прикладной компонент change_order обычно предшествует прикладному компоненту change_request.

EXPRESS-описание:

```
*)
    ENTITY change_order
    SUBTYPE OF (work_ordert);
    change_element : element critical_issue_relationship;
    originates_from : change_request;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут change_element: Этот атрибут определяет элементы, которые подвергаются изменению с помощью прикладного компонента change_order.

Атрибут originates_from: Этот атрибут определяет прикладной компонент change_request, который дает обоснование для существования прикладного компонента change_order.

4.3.36 Прикладной компонент change_order_relationship

Прикладной компонент change_order_relationship определяет взаимосвязь между двумя прикладными компонентами change_order. Семантика этой взаимосвязи содержится в атрибуте change_order_relationship_type.

EXPRESS-описание:

```
*)
    ENTITY change_order_relationship;
    change_order_relationship_type : label;
    related : change_order;
    relating : change_order;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут change_order_relationship_type: Этот атрибут определяет его семантику. Там, где это применимо, должны использоваться следующие состояния (значения) этого атрибута:

- состояние temporal_order: Связывающий прикладной компонент change_order должен завершаться перед реализацией связанного прикладного компонента change_order;
- состояние alternative: Связывающий прикладной компонент change_order является альтернативой для связанного прикладного компонента change_order. Только один из двух идентифицированных прикладных компонентов change_order должен исполняться.

Атрибут related: Этот атрибут определяет второй прикладной компонент change_order в указанной взаимосвязи.

Атрибут relating: Этот атрибут определяет первый прикладной компонент change_order в указанной взаимосвязи.

4.3.37 Прикладной компонент change_report

Прикладной компонент change_report является совокупностью элементов, подтверждающих изменения, внесенные с помощью прикладного компонента change_request.

Примечание — Отчет об изменениях может представляться либо в виде текста в описательном атрибуте, либо путем присвоения других элементов спецификации прикладному компоненту change_report (с помощью прикладного компонента change_report_element_assignment).

EXPRESS-описание:

```
*)
    ENTITY change_report;
    associated_version : configuration_element_version;
    description : OPTIONAL text_select;
    name : label;
    originating_change_request : change_request;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут associated_version: Этот атрибут определяет прикладной компонент configuration_element_version для прикладного компонента change_report.

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту the change_report.

Атрибут name: Этот атрибут определяет слово (или слова), которые используются для ссылки на прикладной компонент change_report.

Атрибут originating_change_request: Этот атрибут определяет прикладной компонент change_order, который приводит к формированию прикладного компонента change_report.

4.3.38 Прикладной компонент change_report_element_assignment

Прикладной компонент change_report_element_assignment определяет способ присвоения описаниям изменений, вносимых в прикладной компонент change_report.

EXPRESS-описание:

```
*)
    ENTITY change_report_element_assignment;
    change_report : change_report;
    change_report_element : change_report_element_select;
    description : OPTIONAL text_select;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут change_report: Этот атрибут входит в прикладной компонент change_report_element_assignment.

Атрибут change_report_element: Этот атрибут определяет элемент, который является частью прикладного компонента change_report.

Атрибут description: Этот атрибут определяет дополнительную текстовую информацию, относящуюся к прикладному компоненту change_report_element_assignment.

4.3.39 Прикладной компонент change_request

Прикладной компонент change_request принадлежит к тому же типу, что и прикладной компонент work_request, и служит для запроса на внесение изменения.

Примечание — Элементы, на которые оказывает воздействие прикладной компонент change_request, идентифицируются с помощью атрибута scope, наследуемого из прикладного компонента work_order.

EXPRESS-описание:

```
*)
    ENTITY change_request
    SUBTYPE OF (work_request) ;
    response_to_issue : SET[1:?] OF critical_issue;
    END_ENTITY;
(*
```

Определение атрибута:

Атрибут response_to_issue: Этот атрибут определяет прикладной компонент critical_issue, который приводит к формированию прикладного компонента change_request.

4.3.40 Прикладной компонент clock

Прикладной компонент clock определяет устройство, выдающее контрольный сигнал с регулярными интервалами.

Примечание 1 — Прикладной компонент clock может определять синхронный режим работы для одного или нескольких прикладных компонентов function_instance.

Примечание 2 — Подкласс системных инженерных средств, поддерживающих синхронный режим работы, использует различные методы представления прикладного компонента clock, который обеспечивает синхронную активацию. Прикладной компонент clock представляют как разновидность функции, а другие — как свойство функции, которое контролируется с помощью прикладного компонента clock. Подход, принятый в настоящем стандарте, принимает оба варианта в том смысле, что прикладной компонент clock может представляться в виде подобной функции модуля или же рассматриваться как свойство контролируемой функции.

EXPRESS-описание:

```
*)
    ENTITY clock;
    control_signal : data_instance;
    description : OPTIONAL text_select;
    frequency : REAL;
    name : OPTIONAL label;
    WHERE
    correct_data_definition: 'SYSTEMS_ENGINEERING_DATA_REPRESENTATION.EVENT_DATA_
    TYPE_DEFINITION' IN TYPEOF(control_signal.definition);
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут control_signal: Этот атрибут определяет прикладной компонент data_instance (чье определение должно относиться к типу прикладного компонента event_data_type_definition), который прикладной компонент clock формирует с периодическими интервалами (задаваемыми с помощью атрибута frequency).

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту clock.

Атрибут frequency: Этот атрибут определяет число импульсов, выдаваемых за заданный промежуток времени.

Атрибут name: Этот атрибут определяет слово (или слова), которые используются для ссылки на прикладной компонент clock.

Формальные выражения:

correct_data_definition:

4.3.41 Прикладной компонент clock_assignment_relationship

Прикладной компонент clock_assignment_relationship определяет взаимосвязь между прикладными компонентами clock и control_ io_port, при которой периодический сигнал, связанный с прикладным компонентом clock, будет передаваться в прикладной компонент control_io_port.

Примечание — Прикладной компонент clock_assignment_relationship позволяет использовать единственный прикладной компонент clock с целью контроля нескольких функциональных объектов (посредством соответствующих прикладных компонентов control_io_port).

EXPRESS-описание:

```
*)
    ENTITY clock_assignment_relationship;
    clock : clock;
    trigger_for : control_io_port;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут clock: Этот атрибут определяет прикладной компонент clock в указанной взаимосвязи.

Атрибут trigger_for: Этот атрибут определяет прикладной компонент control_io_port в указанной взаимосвязи.

4.3.42 Прикладной компонент clock_reference_context_relationship

Прикладной компонент clock_reference_context_relationship определяет способ взаимосвязи прикладных компонентов clock и functionality_instance_reference, который определяет контекст прикладного компонента clock. Посредством использования прикладного компонента clock_reference_context_relationship становится возможным определять, что прикладной компонент clock является действующим только для прикладных компонентов functional_reference_configuration.

EXPRESS-описание:

```
*)
    ENTITY clock_reference_context_relationship;
    clock_signal : clock;
    relevant_functionality_context : functionality_instance_reference;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут clock_signal: Этот атрибут определяет прикладной компонент clock в указанной взаимосвязи.

Атрибут relevant_functionality_context: Этот атрибут определяет прикладной компонент functionality_instance_reference, с помощью которого прикладной компонент clock связывается с прикладным компонентом clock_assignment_relationship.

4.3.43 Прикладной компонент complex_data_type_definition

Прикладной компонент complex_data_type_definition принадлежит к тому же типу, что и прикладной компонент elementary_maths_space, который содержит все комплексные значения.

EXPRESS-описание:

```
*)
        ENTITY complex_data_type_definition
        SUBTYPE OF (elementary_maths_space) ;
        END_ENTITY;
(*
```

4.3.44 Прикладной компонент complex_value

Прикладной компонент complex_value определяет способ задания комплексного значения.

Примечание — Модель использует полярную, а не декартову систему координат для идентификации комплексной точки.

EXPRESS-описание:

```
*)
        ENTITY complex_value;
        radius : REAL;
        theta : REAL;
        END_ENTITY;
(*
```

Определения атрибутов:

Атрибут radius: Этот атрибут определяет расстояние от начала координат (0,0) до комплексной точки.

Атрибут theta: Этот атрибут определяет угловую координату для прикладного компонента complex_value.

4.3.45 Прикладной компонент composite_function_definition

Прикладной компонент composite_function_definition принадлежит к тому же типу, что и прикладной компонент general_function_definition, и представляет собой определение функции, которая в дальнейшем будет разделена. Он должен содержать по крайней мере один прикладной компонент general_functionality_instance.

EXPRESS-описание:

```
*)
        ENTITY composite_function_definition SUBTYPE OF (general_function_definition) ;
        INVERSE
        behaviour_constraint: SET[0:1] OF functional_behaviour_model_assignment FOR constrained_
        function;
        parent_of : SET[1:?] OF functional_decomposition_relationship FOR parent;
        END_ENTITY;
(*
```

Определения атрибутов:

Атрибут behaviour_constraint: Этот атрибут определяет прикладной компонент composite_function_definition, чье представление ограничивается с помощью прикладного компонента functional_behaviour_model.

Атрибут parent_of: Этот атрибут определяет прикладной компонент composite_function_definition, дочерний компонент которого является частью разделенной функции.

4.3.46 Прикладной компонент compound_value

Прикладной компонент compound_value определяет значение, содержащееся в перечне других значений. Этот перечень может содержать различные типы и другие перечни. При наличии в контрольной совокупности меньшего числа компонентов, чем в перечне, излишнее количество использоваться не будет, а при наличии большего числа компонентов — перечень будет применяться повторно до тех пор, пока эта совокупность не станет полной.

EXPRESS-описание:

```
*)
        ENTITY compound_value;
        value_list : LIST[0:?] OF data_type_value_select;
        END_ENTITY;
(*
```

Определение атрибута:

Атрибут value_list: Этот атрибут определяет перечень значений, содержащихся в прикладном компоненте compound_value.

4.3.47 Прикладной компонент configuration_element

Прикладной компонент configuration_element является либо одиночным компонентом, либо прикладным компонентом unit в группе компонентов. В нем собирается информация, которая является общей для всех вариантов реализаций компонентов.

Примечание — Прикладной компонент configuration_element является эффективным логическим хранилищем всех вариантов одного и того же объекта. Например, три варианта одной и той же функциональной модели могут быть собраны вместе с помощью прикладного компонента configuration_element (вместе с именем и описанием этой функциональной модели).

EXPRESS-описание:

```
*)
    ENTITY configuration_element;
    description : OPTIONAL text_select;
    id : element_identifier;
    name : label;
    INVERSE
    associated_version : SET[1:?] OF configuration_element_version FOR version_of;
    UNIQUE
    UR1: id;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут description: Этот атрибут определяет дополнительную текстовую информацию, относящуюся к прикладному компоненту configuration_element.

Атрибут id: Этот атрибут определяет идентификатор прикладного компонента configuration_element.

Атрибут name: Этот атрибут определяет слово (или слова), которые используются для ссылки на прикладной компонент configuration_element.

Атрибут associated_version: Этот атрибут определяет прикладной компонент configuration_element, для которого представлен его вариант.

Формальные выражения:

UR1:

4.3.48 Прикладной компонент configuration_element_relationship

Прикладной компонент configuration_element_relationship определяет способ взаимосвязи двух прикладных компонентов configuration_element. Характер этой взаимосвязи определяется с помощью атрибута relationship_type.

EXPRESS-описание:

```
*)
    ENTITY configuration_element_relationship;
    alternate : configuration_element;
    base : configuration_element;
    description : OPTIONAL text_select;
    relationship_type : label;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут alternate: Этот атрибут определяет второй прикладной компонент configuration_element в альтернативном варианте.

Атрибут base: Этот атрибут определяет первый прикладной компонент configuration_element в основном варианте.

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к этому атрибуту.

Атрибут relationship_type: Этот атрибут определяет семантику прикладного компонента configuration_element_relationship. Там, где это применимо, должны использоваться следующие состояния (значения) этого атрибута:

- состояние alternative: Основной и альтернативный прикладной компоненты configuration_element являются взаимозаменяемыми;
- состояние variant: Альтернативой является один из вариантов основного прикладного компонента configuration_element.

Примечание — Вариант компонента предполагает, что базовый и альтернативный варианты прикладного компонента configuration_element разделяют существенный объем информации.

4.3.49 Прикладной компонент configuration_element_version

Прикладной компонент configuration_element_version является «мгновенным изображением» любого конструктивного элемента, для которого сохраняется хронология изменений.

Примечание 1 — Перечень изменений сохраняется, начиная от начального выпуска элемента, и регистрируется с помощью прикладного компонента configuration_element_version, поэтому компонент с изображением будет содержать прикладной компонент configuration_element и набор прикладных компонентов configuration_element_version, в котором будут регистрироваться изменения данного прикладного компонента configuration_element. В настоящем стандарте конструктивные элементы могут получаться из модулей для функций, требований, поведения, документации и типов данных.

В дальнейшем это приведет к ситуации, в которой проект будет формировать экземпляры прикладного компонента configuration_element, относящиеся к представляемому конкретному варианту прикладного компонента configuration_element, поэтому проект может содержать набор прикладных компонентов configuration_element, представленный в отличающемся варианте. После этого новый вариант всего проекта может вносить одиночное изменение в проект.

Пример 3 — Например, вариант проекта 1 может содержать вариант 1 функции 1, вариант 3 функции 2 и вариант 1 функции 3. После этого следующий выпуск проекта (вариант 2) может содержать вариант 2 функции 1, вариант 3 функции 2 и вариант 1 функции 3. Этот пример иллюстрирует, что может существовать два уровня управления вариантами. Одна сторона проекта может желать управлять всеми вариантами проекта, а другая сторона проекта — лишь вариантами отдельных элементов.

Примечание 2 — Стандартное предположение состоит в том, что целостность будет сохраняться либо путем передачи инструмента и его получения, либо с помощью средств третьей стороны проекта, более специализированных для элементов управления конфигурацией.

Примечание 3 — Набор прикладных компонентов configuration_element_version прикладного компонента configuration_element представляет его хронологию в рамках определенного этапа жизненного цикла или, возможно, в течение всего жизненного цикла.

EXPRESS-описание:

```
*)
    ENTITY configuration_element_version;
    description : OPTIONAL text_select;
    id : element_identifier;
    version_of : configuration_element;
    UNIQUE;
    UR1: id;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут description: Этот атрибут определяет дополнительную текстовую информацию, относящуюся к прикладному компоненту configuration_element_version.

Атрибут id: Этот атрибут определяет идентификатор прикладного компонента configuration_element_version.

Атрибут version_of: Этот атрибут определяет прикладной компонент configuration_element, для которого прикладной компонент configuration_element представляет его вариант.

Формальные выражения:

UR1:

4.3.50 Прикладной компонент configuration_element_version_relationship

Прикладной компонент configuration_element_version_relationship определяет взаимосвязь между двумя прикладными компонентами configuration_element_version и устанавливает граф версии прикладных компонентов.

EXPRESS-описание:

```
*)
    ENTITY configuration_element_version_relationshipn;
    description : OPTIONAL text_select;
    related_version : configuration_element_version;
    relating_version : configuration_element_version;
    relationship_type : label;
    WHERE WR1: related_version :<>: relating_version;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут description: Этот атрибут определяет характер изменений между атрибутами related_version и relating_version.

Атрибут related_version: Этот атрибут определяет полученный прикладной компонент configuration_element_version, связываемый с помощью прикладного компонента configuration_element_version_relationship.

Атрибут relating_version: Этот атрибут определяет исходный прикладной компонент configuration_element_version, связываемый с помощью прикладного компонента configuration_element_version_relationship.

Атрибут relationship_type: Этот атрибут определяет семантику прикладного компонента configuration_element_version_relationship. Там, где это применимо, должны использоваться следующие состояния (значения) этого атрибута:

- состояние revision;
- состояние workspace_revision;
- состояние alternative.

Формальные выражения:

WR1:

4.3.51 Прикладной компонент context_function_relationship

Прикладной компонент context_function_relationship определяет взаимосвязь между прикладными компонентами function_instance и system_view, при которой прикладной компонент function_instance будет являться частью функционального описания прикладного компонента system_view.

Примечание 1 — Прикладной компонент function_instance может описывать функциональные аспекты прикладного компонента system_view или условий применения прикладного компонента system_view. В первом случае атрибут role должен устанавливаться в состояние system_function, а во втором случае этот же атрибут должен устанавливаться в состояние external_element.

EXPRESS-описание:

```
*)
    ENTITY context_function_relationship;
    associated_context : system_view;
    context_function : function_instance;
    description : OPTIONAL text_select;
    role : function_role_enumeration;
    INVERSE
    assigned_functional_configuration : SET[0:1] OF system_functional_configuration FOR system;
    UNIQUE
    UR1: associated_context, context_function;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут associated_context: Этот атрибут определяет прикладной компонент system_view, который действителен для этой взаимосвязи.

Атрибут context_function: Этот атрибут определяет прикладной компонент function_instance, который действителен для этой взаимосвязи.

Атрибут description: Этот атрибут определяет дополнительную текстовую информацию, относящуюся к данной взаимосвязи.

Атрибут role: Этот атрибут определяет то, что описывает прикладной компонент function_instance в контексте частного прикладного компонента system_view.

Примечание 2 — Существует ограничительное условие о единственности прикладного компонента function_instance, присваиваемого прикладному компоненту system_view в состоянии system_function.

Атрибут assigned_functional_configuration: Этот атрибут определяет прикладной компонент context_function_relationship, которому присваивается прикладной компонент functional_reference_configuration.

Формальные выражения:

UR1:

4.3.52 Прикладной компонент context_physical_relationship

Прикладной компонент context_physical_relationship определяет взаимосвязь между прикладными компонентами physical_instance и system_view, при которой прикладной компонент physical_instance является описанием структуры верхнего уровня для прикладного компонента system_view.

Примечание — Прикладной компонент physical_instance может описывать физические аспекты прикладного компонента system_view или условие применения прикладного компонента system_view. В первом случае атрибут role должен устанавливаться в состояние system_element, а во втором случае этот атрибут должен устанавливаться в состояние external_element.

EXPRESS-описание:

```
*)
ENTITY context_physical_relationship;
    description : OPTIONAL text_select;
    part_in_physical_context : system_view;
    объект-приложение physical_instance : physical_instance;
    role : physical_element_role_enumeration;
    INVERSE
    assigned_physical_configuration : SET[0:1] OF system_physical_configuration FOR system;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут description: Этот атрибут определяет дополнительную текстовую информацию, относящуюся к этой взаимосвязи.

Атрибут part_in_physical_context: Этот атрибут определяет прикладной компонент system_view, для которого эта взаимосвязь является значимой.

Атрибут physical_instance: Этот атрибут определяет прикладной компонент physical_instance, для которого эта взаимосвязь является действующей.

Атрибут role: Этот атрибут определяет характер атрибута physical_instance в прикладном компоненте system_view.

Атрибут assigned_physical_configuration: Этот атрибут определяет прикладной компонент context_physical_relationship, которому присваивается прикладной компонент physical_reference_configuration.

4.3.53 Прикладной компонент control_io_port

Прикладной компонент control_io_port принадлежит к тому же типу, что и прикладной компонент io_port, а элемент в интерфейсе — к прикладному компоненту function_instance. Прикладной компонент control_io_port всегда относится к входному порту и данным, поступающим от прикладных компонентов functional_link.

Примечание 1 — Его семантика такова, что прикладной компонент control_io_port активируется и деактивируется с помощью значения, связанного с прикладным компонентом control_io_port. Если прикладной компонент control_io_port активирован, то он будет способен активировать и прикладной компонент function_instance.

Примечание 2 — Если прикладной компонент data_instance, определенный с помощью атрибута data, принадлежит к типу прикладного компонента event_data_type_definition, то прикладной компонент control_io_port будет активироваться путем передачи события в прикладной компонент data_instance.

Примечание 3 — Если прикладной компонент data_instance, определенный с помощью атрибута data, принадлежит к типу прикладного компонента logical_data_type_definition, то прикладной компонент control_io_port будет активироваться тогда, когда состоянием прикладного компонента data_instance станет TRUE; в противном случае прикладной компонент control_io_port будет деактивироваться.

Примечание 4 — Если несколько прикладных компонентов control_io_port присоединяются к прикладному компоненту function_instance, то все они должны активироваться и для активации прикладного компонента function_instance.

EXPRESS-описание:

```
*)
    ENTITY control_io_port
    SUBTYPE OF (io_port) ;
    control_type : control_type_enumeration;
    offset : REAL;
    port_of : function_instance;
    trigger_type : trigger_type_enumeration;
    DERIVE
    SELF\ io_port. RENAMED role : port_data_relation : consumer;
    UNIQUE
    UR1: port_of, io_port_number, port_type;
    WHERE
    good_offset: offset >= 0.0;
    port_data_direction: (SELF\io_port.port_type <> output);
    WR1:  ('SYSTEMS_ENGINEERING_DATA_REPRESENTATION.EVENT_DATA_TYPE_DEFINITION' IN TYPEOF(data.definition)) OR ('SYSTEMS_ENGINEERING_DATA_REPRESENTATION.LOGICAL_DATA_TYPE_DEFINITION' IN TYPEOF(data.definition));
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут control_type: Этот атрибут определяет назначение прикладного компонента control_io_port.

Атрибут offset: Этот атрибут определяет время задержки, начиная с момента приема контрольного сигнала и до тех пор, пока прикладной компонент control_io_port будет оставаться активным.

Атрибут port_of: Этот атрибут определяет прикладной компонент function_instance, к которому присоединяется прикладной компонент control_io_port.

Атрибут trigger_type: Этот атрибут определяет способ активации прикладного компонента control_io_port.

Примечание 5 — Точная семантика инициализации определяется с помощью типа данных, связанных с контрольным портом.

Атрибут role:

Формальные положения:

UR1:

good_offset:

port_data_direction:

WR1:

4.3.54 Прикладной компонент coordinate_translation_information

Прикладной компонент coordinate_translation_information определяет способ преобразования информации о координатах из нормированной координатной системы в реально представляемую систему координат, используемую в приборе. Прикладной компонент coordinate_translation_information присваивается в качестве модификатора для предоставления реальной презентационной информации для прикладных компонентов graphics_view.

Размер поля зрения и положение отдельных элементов в нем рассчитывается по следующей формуле:

- y-координата:=normalized_x-position*scale_factor;
- x-координата:= normalized_x-position*aspect_ration*scale_factor;

Примечание — В настоящем стандарте используется нормированная координатная система, в которой вся информация о положении элементов представляется в диапазоне значений {0..1} для x- и y-координат. Коор-

дината {0,0} определяет верхний левый угол поля зрения системы, а координата {1,1} — нижний правый угол этого поля зрения.

EXPRESS-описание:

```
*)
    ENTITY coordinate_translation_information;
    measurement_unit : label;
    name : label;
    ratio : REAL;
    scale_factor : REAL;
    transformation_for : graphics_view;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут measurement_unit: Этот атрибут определяет единицу измерений, которая должна использоваться для декодирования размера поля зрения, определяемого с помощью прикладного компонента coordinate_translation_information.

Атрибут name: Этот атрибут определяет слово (или слова), которые используются для ссылки на прикладной компонент coordinate_translation_information.

Атрибут ratio: Этот атрибут определяет соотношения координат по осям x и y в координатной системе, определяемой с помощью прикладного компонента coordinate_translation_information. Значения, превышающие 1, будут указывать на то, что размер поля зрения больше по оси x, чем по оси y.

Атрибут scale_factor: Этот атрибут определяет множитель, который задает размер поля зрения.

Атрибут transformation_for: Этот атрибут определяет прикладной компонент graphics_view, для которого прикладной компонент coordinate_translation_information предоставляет информацию.

4.3.55 Прикладной компонент critical_issue

Прикладной компонент critical_issue является идентификатором воспринимаемой или реальной проблемы, касающейся элемента системной спецификации.

Примечание — Элемент, идентифицируемый с помощью прикладного компонента critical_issue, не требует подтверждения всеми заинтересованными сторонами в процессе разработки системы. Он является границей идентификатора для указания возможной проблемы или элемента.

EXPRESS-описание:

```
*)
    ENTITY critical_issue;
    description : OPTIONAL text_select;
    id : element_identifier;
    name : label;
    status : label;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут description: Этот атрибут определяет дополнительную информацию на прекращение выполнения операций: Никакие дополнительные операции не будут выполняться для анализа прикладного компонента critical_issue.

Атрибут id: Этот атрибут определяет собственный идентификатор.

Атрибут name: Этот атрибут определяет слово (или слова), которые используются для ссылки на прикладной компонент critical_issue.

Атрибут status: Этот атрибут определяет условие, накладываемое на прикладной компонент critical_issue. Там, где это применимо, должны использоваться следующие состояния (значения) этого компонента:

- состояние open: Достоверность прикладного компонента critical_issue пока не была оценена;
- состояние acknowledged: Достоверность прикладного компонента critical_issue была подтверждена;
- состояние closed: Никакие дополнительные операции не будут выполняться для анализа прикладного компонента critical_issue.

4.3.56 Прикладной компонент critical_issue_impact

Прикладной компонент critical_issue_impact определяет способ связи элементов, идентифицируемых как подверженные влиянию прикладного компонента critical_issue.

Примечание — За прикладным компонентом critical_issue_impact может закрепляться любое число элементов. Прикладной компонент critical_issue_impact является представлением всех совместно закрепленных элементов.

EXPRESS-описание:

```
*)
    ENTITY critical_issue_impact;
    description : OPTIONAL text_select;
    id : element_identifier;
    impact_of_issue : SET[1:?] OF critical_issue;
    name : label;
    INVERSE
    identified_issues : SET[0:?] OF critical_issue_relationship FOR issue_analysis;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту critical_issue_impact.

Атрибут id: Этот атрибут определяет идентификатор прикладного компонента critical_issue_impact.

Атрибут impact_of_issue: Этот атрибут определяет набор прикладных компонентов critical_issue, для которого действителен прикладной компонент critical_issue_impact.

Атрибут name: Этот атрибут определяет слово (или слова), которые используются для ссылки на прикладной компонент critical_issue_impact.

Атрибут identified_issues: Этот атрибут определяет прикладной компонент critical_issue_impact, для которого действителен прикладной компонент element_critical_issue_relationship.

4.3.57 Прикладной компонент data_field

Прикладной компонент data_field принадлежит к тому же типу, что и прикладной компонент data_instance, который использует тип данных прикладного компонента record_data_type_definition или union_data_type_definition.

EXPRESS-описание:

```
*)
    ENTITY data_field
    SUBTYPE OF (data_instance) ;
    role : OPTIONAL label;
    END_ENTITY;
(*
```

Определение атрибута:

Атрибут role: Этот атрибут определяет дополнительную информацию относительно прикладного компонента data_field.

4.3.58 Прикладной компонент data_instance

Прикладной компонент data_instance принадлежит к тому же типу, что и прикладной компонент user_defined_data_type_definition или maths_space.

Примечание 1 — В терминологии, используемой в компьютерной технике, прикладной компонент data_instance будет соответствовать переменной, а в области системного проектирования прикладной компонент data_instance будет интерпретироваться в более широком смысле.

EXPRESS-описание:

```
*)
    ENTITY data_instance
    SUPERTYPE OF (data_field);
    default_value : OPTIONAL data_type_value_select;
    definition : data_type_definition_select;
```

description : OPTIONAL text_select;
id : element_identifier;
initial_value : OPTIONAL data_type_value_select;
is_constant : BOOLEAN;
name : label;
unit_component : OPTIONAL unit;
UNIQUE
UR1: definition, id;
END_ENTITY;
(*

Определения атрибутов:

Атрибут default_value: Этот атрибут определяет значение, которое должно присваиваться прикладному компоненту data_instance, когда предоставляется недействительное значение, например когда значение выходит за ограниченный диапазон или имеет ненадлежащий тип.

Примечание 2 — Событие, когда для атрибута default_value действующие значения отсутствуют, может быть обусловлено тем фактом, что элемент, дающий значение, выпадает из последовательности.

Примечание 3 — Значение параметра default_value должно находиться в диапазоне допустимых значений, если он определен.

Атрибут definition: Этот атрибут определяет тип данных для прикладного компонента user_defined_data_type_definition или maths_space, который дает определение этого типа.

Атрибут description: Этот атрибут определяет дополнительную текстовую информацию, относящуюся к прикладному компоненту data_instance.

Атрибут id: Этот атрибут определяет идентификатор прикладного компонента data_instance.

Атрибут initial_value: Этот атрибут определяет значение, которое должно присваиваться прикладному компоненту data_instance, когда он будет сформирован.

Примечание 4 — Значение параметра initial_value должно находиться в диапазоне действующих значений для прикладного компонента initial_value, если он определен.

Атрибут is_constant: Этот атрибут определяет, является ли прикладной компонент data_instance постоянным. При наличии состояния TRUE атрибут is_constant может не обновляться. В этом случае значение атрибута is_constant будет определяться с помощью атрибута initial_value. Если для этого атрибута не предусмотрено никакого исходного значения, то его значение будет неопределенным.

Атрибут name: Этот атрибут определяет слово (или слова), которые используются для ссылки на прикладной компонент data_instance.

Атрибут unit_component: Этот атрибут определяет единицы измерения, которые должны связываться с прикладным компонентом data_instance.

Пример 4 — Для частного прикладного компонента data_instance атрибутом unit_component может быть Вольт.

Формальные выражения:

UR1:

4.3.59 Прикладной компонент data_transfer

Прикладной компонент data_transfer определяет характер взаимодействия, идентифицированного с помощью прикладного компонента implied_external_interaction.

Примечание — Область применения прикладного компонента data_transfer не требует ограничения данных; он может также использоваться для индикации взаимодействия физических элементов.

EXPRESS-описание:

*)

ENTITY data_transfer;
data : data_instance;
direction : data_direction;
transfer : implied_external_interactionr;
END_ENTITY;
(*

Определения атрибутов:

Атрибут data: Этот атрибут определяет прикладной компонент data_instance, который является частью взаимодействия между функциональными представлениями внешнего объекта и системы.

Атрибут direction: Этот атрибут определяет направление этого взаимодействия.

Атрибут transfer: Этот атрибут определяет прикладной компонент implied_external_interaction, для которого действующим является прикладной компонент data_transfer.

4.3.60 Прикладной компонент date_and_person_assignment

Прикладной компонент date_and_person_assignment определяет объект, который связывается с прикладным компонентом date_and_person_organization.

Примечание 1 — Подобное присвоение дает дополнительную информацию для связанного объекта. Предоставление подобных данных посредством такого присвоения имеет организационный характер, тогда как с целью сохранения семантической завершенности некоторые объекты требуют применения того же типа данных. Это присвоение не должно использоваться для связи соответствующих организационных данных с объектом, чьи атрибуты напрямую привязываются к указанным данным.

Примечание 2 — Заимствовано из Протокола AP-214.

EXPRESS-описание:

```
*)
    ENTITY date_and_person_assignment;
    assigned_date_and_person : date_and_person_organization;
    assigned_to : person_organization_assignment_select;
    description : OPTIONAL text_select;
    role : label;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут assigned_date_and_person: Этот атрибут определяет прикладной компонент date_and_person_organization.

Атрибут assigned_to: Этот атрибут определяет объект, к которому применим прикладной компонент date_and_person_assignment.

Атрибут description: Этот атрибут определяет дополнительную текстовую информацию, относящуюся к прикладному компоненту date_and_person_assignment.

Атрибут role: Этот атрибут определяет связь между датой (или временем) и прикладным компонентом person или organization в атрибуте role. Там, где это применимо, должны использоваться следующие состояния (значения) этого атрибута:

- состояние creation: Присвоение позволяет определять, что указанный объект был сформирован с помощью данного лица или организации в заданный день и время;
- состояние update: Присвоение позволяет определять, что указанный объект был изменен с помощью заданного лица.

4.3.61 Прикладной компонент date_and_person_organization

Прикладной компонент date_and_person_organization определяет прикладной компонент person_in_organization или organization, связанный с прикладным компонентом date_time.

EXPRESS-описание:

```
*)
    ENTITY date_and_person_organization;
    actual_date : date_time;
    person_organization : person_organization_select;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут actual_date: Этот атрибут определяет прикладной компонент date_time в прикладном компоненте date_and_person_organization.

Атрибут person_organization: Этот атрибут определяет прикладной компонент person_in_organization или organization в прикладном компоненте date_and_person_organization.

4.3.62 Прикладной компонент date_assignment

Прикладной компонент date_assignment определяет присвоение прикладного компонента date_time и временной информации.

EXPRESS-описание:

```
*)
    ENTITY date_assignment;
    assigned_to : date_assignment_select;
    date : date_time;
    role : label;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут assigned_to: Этот атрибут определяет присвоение прикладному компоненту date_time.

Атрибут date: Этот атрибут определяет дату при этом присвоении.

Атрибут role: Этот атрибут определяет семантику прикладного компонента date_time, присваиваемого элементу. Там, где это применимо, должны использоваться следующие состояния (значения) этого атрибута:

- состояние created: Присвоение определяет, что указанный объект был создан при заданном атрибуте date;
- состояние modified: Присвоение определяет, что указанный объект был изменен при заданном атрибуте date.

Примечание — Набор существующих значений атрибута может быть расширен для удовлетворения перспективных потребностей, однако в подобных случаях передающие и приемные средства должны согласовываться с действующим набором значений.

4.3.63 Прикладной компонент date_time

Прикладной компонент date_time определяет спецификацию для даты и времени дня.

EXPRESS-описание:

```
*)
    ENTITY date_time;
    day_component : INTEGER;
    hour_component : INTEGER;
    minute_component : INTEGER;
    month_component : INTEGER;
    second_component : INTEGER;
    year_component : INTEGER;
    WHERE
    WR1: {0 <= hour_component <= 23};
    WR2: {0 <= minute_component <= 59};
    WR3: {0 <= second_component <= 59};
    WR4: {1 <= month_component <= 12};
    WR5: {1 <= day_component <= 31};
    WR6: year_component >= 0;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут day_component: Этот атрибут определяет день месяца. Значение атрибута day_component не должно превышать числа дней в текущем месяце.

Атрибут hour_component: Этот атрибут определяет время, выражаемое в часах.

Значение атрибута hour_component должно находиться в диапазоне от 0 до 23.

Атрибут minute_component: Этот атрибут определяет время, выражаемое в минутах.

Значение атрибута minute_component должно находиться в диапазоне от 0 до 59.

Атрибут month_component: Этот атрибут определяет прикладной компонент date_time, выражаемый в месяцах.

Значение атрибута month_component должно находиться в диапазоне от 1 до 12.

Атрибут second_component: Этот атрибут определяет время, выражаемое в секундах.

Значение атрибута second_component должно находиться в диапазоне от 0 до 59.

Атрибут year_component: Этот атрибут определяет время года в прикладном компоненте date_time.

Значение атрибута year_component должно приводиться со всеми действующими значениями.

Пример 5 — Год 1999 должен представляться в виде целого числа «1999».

Формальные выражения:

WR1:

WR2:

WR3:

WR4:

WR5:

WR6:

4.3.64 Прикладной компонент derived_data_type_definition

Прикладной компонент derived_data_type_definition принадлежит к тому же типу, что и прикладной компонент user_defined_data_type_definition, чье значение рассчитывается по прикладным компонентам data_instance и/или data_field.

EXPRESS-описание:

```
*)
    ENTITY derived_data_type_definition
    SUBTYPE OF (user_defined_data_type_definition) ;
    expression : text_select;
    resultant_data_type : data_type_definition_select;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут expression: Этот атрибут определяет алгоритм расчета значения экземпляра прикладного компонента derived_data_type_definition.

Примечание — Синтаксис, необходимый для определения алгоритма расчета этого значения, не определен.

Пример 6 — Во избежание избыточности в конструкции базы данных и поддержания актуальности данных, поле для возраста должно рассчитываться по глобальному значению на сегодняшний день, а поле для возраста — по формуле birth. age = today — date_of_birth.

Атрибут resultant_data_type: Этот атрибут определяет ожидаемый тип результата выражения, основанный на типах значений, входящих в него.

4.3.65 Прикладной компонент digital_document

Прикладной компонент digital_document принадлежит к тому же типу, что и прикладной компонент documentation_reference и является ссылкой на документ, представляемый в цифровой форме.

EXPRESS-описание:

```
*)
    ENTITY digital_document
    SUBTYPE OF (documentation_reference) ;
    document_format : OPTIONAL label;
    document_size : OPTIONAL label;
    location : label;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут document_format: Этот атрибут определяет формат цифрового документа (например, MSWord, PDF, HTML).

Атрибут document_size: Этот атрибут определяет объем цифрового документа.

Атрибут location: Этот атрибут определяет местоположение цифрового документа (например, в структуре файлов, унифицированном указателе ресурсов — URL), откуда он может быть получен.

4.3.66 Прикладной компонент document_assignment

Прикладной компонент document_assignment определяет присвоение прикладного компонента documentation_reference элементу системного проектирования.

EXPRESS-описание:

```
*)
    ENTITY document_assignment
    SUPERTYPE OF (partial_document_assignment) ;
    description : OPTIONAL text_select;
    documentation : documentation_reference;
    documented_object : instance_definition_select;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту document_assignment.

Атрибут documentation: Этот атрибут определяет прикладной компонент documentation_reference.

Атрибут documented_object: Этот атрибут определяет объект, который документируется с помощью прикладного компонента document_assignment.

4.3.67 Прикладной компонент documentation_reference

Прикладной компонент documentation_reference определяет ссылку на любую существенную информацию, а прикладной компонент documentation_reference является абстрактным объектом, который никогда не должен реализовываться.

EXPRESS-описание:

```
*)
    ENTITY documentation_reference
    ABSTRACT SUPERTYPE OF (ONEOF(digital_document, non_digital_document);
    associated_version : configuration_element_version;
    description : OPTIONAL text_select;
    id : element_identifier;
    name : label;
    UNIQUE
    UR1: id;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут associated_version: Этот атрибут определяет прикладной компонент configuration_element_version для прикладного компонента documentation_reference.

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту documentation_reference.

Атрибут id: Этот атрибут определяет идентификатор прикладного компонента documentation_reference.

Атрибут name: Этот атрибут определяет слово (или слова), которые используются для ссылки на прикладной компонент documentation_reference.

Формальные выражения:

UR1:

4.3.68 Прикладной компонент documentation_relationship

Прикладной компонент documentation_relationship определяет взаимосвязь между двумя прикладными компонентами documentation_reference. Семантика этой взаимосвязи содержится в атрибуте description.

EXPRESS-описание:

```
*)
    ENTITY documentation_relationship;
    description : OPTIONAL text_select;
```

```
related_documentation : documentation_reference;
relating_documentation : documentation_reference;
relationship_type : OPTIONAL label;
WHERE
correct_relation: relating_documentation :<>: related_documentation;
END_ENTITY;
(*
```

Определения атрибутов:

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту documentation_relationship.

Атрибут related_documentation: Этот атрибут определяет второй прикладной компонент documentation_reference в указанной взаимосвязи.

Атрибут relating_documentation: Этот атрибут определяет первый прикладной компонент documentation_reference в указанной взаимосвязи.

Атрибут relationship_type: Этот атрибут определяет слово (или слова), которые используются для ссылки на прикладной компонент documentation_relationship.

Формальные выражения:

correct_relation:

4.3.69 Прикладной компонент effectiveness_measure

Прикладной компонент effectiveness_measure определяет критерий оптимизации для любого числа прикладных компонентов requirement_instance.

EXPRESS-описание:

```
*)
ENTITY effectiveness_measure;
optimization_function : textual_specification;
END_ENTITY;
(*
```

Определение атрибута:

Атрибут optimization_function: Этот атрибут определяет функцию оптимизации требований для прикладного компонента effectiveness_measure.

4.3.70 Прикладной компонент effectiveness_measure_assignment

Прикладной компонент effectiveness_measure_assignment определяет способ включения прикладного компонента requirement_instance в критерий оптимизации функций, представляемый с помощью прикладного компонента effectiveness_measure. Все присвоенные прикладные компоненты requirement_instance должны быть связаны с одним и тем же прикладным компонентом system_view.

EXPRESS-описание:

```
*)
ENTITY effectiveness_measure_assignment;
assigned_requirement : requirement_instance;
объект-приложение effectiveness_measure : effectiveness_measure;
weight : label;
END_ENTITY;
(*
```

Определения атрибутов:

Атрибут assigned_requirement: Этот атрибут определяет прикладной компонент requirement_instance, вводимый в функции оптимизации с помощью прикладного компонента effectiveness_measure_assignment.

Атрибут effectiveness_measure: Этот атрибут определяет прикладной компонент effectiveness_measure, который присваивается прикладному компоненту requirement_instance посредством прикладного компонента effectiveness_measure_assignment.

Атрибут weight: Этот атрибут определяет относительную важность прикладного компонента assigned_requirement_instance по отношению к другим прикладным компонентам requirement_instance, присваиваемых тому же прикладному компоненту effectiveness_measure.

4.3.71 Прикладной компонент effectiveness_measure_relationship

Прикладной компонент effectiveness_measure_relationship определяет способ указания взаимосвязи между двумя прикладными компонентами effectiveness_measure. Характер этой взаимосвязи будет определен ниже с помощью атрибута description.

Примечание — Прикладной компонент effectiveness_measure_relationship может использоваться и для указания взаимосвязи между двумя критериями оптимизации.

EXPRESS-описание:

```
*)
    ENTITY effectiveness_measure_relationship;
    description : OPTIONAL text_select;
    related : effectiveness_measure;
    relating : effectiveness_measure;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту effectiveness_measure_relationship.

Атрибут related: Этот атрибут определяет второй элемент в указанной взаимосвязи.

Атрибут relating: Этот атрибут определяет первый элемент в указанной взаимосвязи.

4.3.72 Прикладной компонент effectivity

Прикладной компонент effectivity является идентификатором эффективного использования свойства производственных данных, контролируемого с помощью прикладного компонента date_time или события.

Примечание 1 — Прикладной компонент effectivity может определять замкнутые или незамкнутые интервалы для проверки производственных данных.

Примечание 2 — Прикладной компонент effectivity должен определяться либо с помощью атрибута primary_definition attribute, либо с помощью другого прикладного компонента effectivity (связанного с использованием прикладного компонента effectivity_relationship), который имеет реализуемый атрибут primary_definition.

EXPRESS-описание:

```
*)
    ENTITY effectivity;
    concerned_organization : SET[0:?] OF organization;
    description : OPTIONAL text_select;
    id : OPTIONAL element_identifier;
    primary_definition : OPTIONAL event_or_date_select;
    secondary_definition : OPTIONAL period_or_date_select;
    version_id : OPTIONAL element_identifier;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут concerned_organization: Этот атрибут определяет прикладной компонент organization, в котором прикладной компонент effectivity является действующим.

Пример 7 — Прикладной компонент effectivity одного и того же элемента может быть различным в различных областях нахождения поставщика системы.

Атрибут description: Этот атрибут определяет дополнительную текстовую информацию, относящуюся к прикладному компоненту effectivity.

Атрибут id: Этот атрибут определяет идентификатор прикладного компонента effectivity.

Атрибут primary_definition: Этот атрибут определяет прикладной компонент date_time или событие тогда, когда идентифицированный компонент становится или перестает быть действующим.

Примечание 3 — Смысл данного атрибута будет подробно определяться с помощью атрибута role для прикладного компонента effectivity_assignment.

Атрибут secondary_definition: Этот атрибут используется для определения времени начала, указываемого для первичного определения. Данный атрибут должен использоваться лишь в том случае, когда атрибут role для прикладного компонента effectivity_assignment укажет, что этот атрибут определен как период времени.

Атрибут version_id: Этот атрибут определяет идентификатор конкретного варианта прикладного компонента effectivity.

4.3.73 Прикладной компонент effectivity_assignment

Прикладной компонент effectivity_assignment позволяет связывать прикладной компонент effectivity с объектом, чей прикладной компонент effectivity контролируется с помощью связанного с ним прикладного компонента effectivity.

EXPRESS-описание:

```
*)
    ENTITY effectivity_assignment;
    assigned_effectivity : effectivity;
    effective_element : effective_element_select;
    role : label;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут assigned_effectivity: Этот атрибут определяет присвоенный прикладной компонент effectivity.

Атрибут effective_element: Этот атрибут определяет объект, которому присваивается прикладной компонент effectivity.

Атрибут role: Этот атрибут определяет взаимосвязь между прикладным компонентом effectivity и объектом, который эффективно присваивается ему.

Если определено одно из состояний (значений) actual_start, actual_stop, planned_start или planned_stop, то в присвоенном прикладном компоненте effectivity не должно даваться никакое дополнительное определение.

Там, где это применимо, должны использоваться следующие состояния (значения) этого атрибута:

- состояние actual_period: Определяет фактический период, в течение которого действует прикладной компонент effectivity;
- состояние actual_start: Определяет действующий прикладной компонент date_time с того времени, когда действующий прикладной компонент effectivity стал действующим;
- состояние actual_stop: Определяет действующий прикладной компонент date_time с того времени, когда прикладной компонент effectivity перестал быть действующим;
- состояние planned_period: Определяет период, связанный с прикладным компонентом effectivity и указывающий плановый период времени, в течение которого связанный компонент является или являлся (по предположению) действующим.
- состояние planned_start: Определяет прикладной компонент date_time или событие, связанное с прикладным компонентом effectivity и определяющее прикладной компонент date_time или событие в то время, когда начинается или начиналось (по предположению) действие прикладного компонента effectivity;
- состояние planned_stop: Определяет прикладной компонент date_time или событие, связанное с прикладным компонентом effectivity и определяющее прикладной компонент date_time или событие в то время, когда заканчивается или заканчивалось (по предположению) действие прикладного компонента effectivity;
- состояние required_period: Определяет связанный компонент, который должен сохраняться действующим в течение этого периода;
- состояние required_start: Определяет прикладной компонент date_time или событие, связанное с прикладным компонентом effectivity и обусловленное началом действия прикладного компонента date. Предполагается наличие соответствия с этой границей.
- состояние required_stop: Определяет прикладной компонент date_time или событие, связанное с прикладным компонентом effectivity и обусловленное окончанием действия прикладного компонента date. Предполагается наличие соответствия с этой границей.

4.3.74 Прикладной компонент effectivity_relationship

Прикладной компонент effectivity_relationship определяет взаимосвязь между двумя прикладными компонентами effectivity.

EXPRESS-описание:

```
*)
    ENTITY effectivity_relationship;
    description : OPTIONAL text_select;
    related : effectivity;
    relating : effectivity;
    relation_type : label;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту effectivity_relationship.

Атрибут related: Этот атрибут определяет второй прикладной компонент effectivity в указанной взаимосвязи.

Атрибут relating: Этот атрибут определяет первый прикладной компонент effectivity в указанной взаимосвязи.

Атрибут relation_type: Этот атрибут определяет смысл указанной взаимосвязи. Там, где это применимо, должны использоваться следующие состояния (значения) этого атрибута:

- состояние constraint: Промежуток времени между началом и окончанием действия определения связанного прикладного компонента effectivity должен находиться в пределах периода времени для прикладного компонента effectivity;
- состояние inheritance: Связанный прикладной компонент effectivity не должен иметь какого-либо определенного «первичного определения» и «вторичного определения», однако должен наследовать даты действия от связанного прикладного компонента effectivity.

4.3.75 Прикладной компонент element_critical_issue_relationship

Прикладной компонент element_critical_issue_relationship определяет способ указания того, что элемент (определяемый с помощью атрибута impact_on_element) в некотором отношении находится под воздействием прикладного компонента critical_issue, который отбирается с помощью прикладного компонента critical_issue_impact. Само это воздействие не определяется, поскольку определенные прикладные компоненты critical_issue_impact могут формироваться в результате воздействия прикладного компонента critical_issue. Характер воздействия определяется с помощью атрибута description.

EXPRESS-описание:

```
*)
    ENTITY element_critical_issue_relationship;
    description : OPTIONAL text_select;
    impact_on_element : change_element_select;
    issue_analysis : critical_issue_impact;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут description: Этот атрибут определяет характер воздействия на элемент, иденфицируемый с помощью атрибута impact_on_element для прикладного компонента element_critical_issue_relationship.

Атрибут impact_on_element: Этот атрибут определяет элемент, находящийся под воздействием прикладного компонента critical_issue.

Атрибут issue_analysis: Этот атрибут определяет прикладной компонент critical_issue_impact, для которого действителен прикладной компонент element_critical_issue_relationship.

4.3.76 Прикладной компонент element_identifier

Прикладной компонент element_identifier является идентификатором, используемым для представления уникальной метки для элементов одного и того же типа.

EXPRESS-описание:

```
*)
    ENTITY element_identifier;
    identifier_context : person_organization_select;
    identifier_value : identifier;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут identifier_context: Этот атрибут определяет область для организации, в которой атрибут identifier_value, как предполагается, будет являться уникальным для прикладного компонента element_identifier.

Атрибут identifier_value: Этот атрибут определяет строку из буквенно-численных символов, которая присваивается прикладному компоненту element_identifier и является уникальной в его области для организации.

4.3.77 Прикладной компонент elementary_maths_space

Прикладной компонент elementary_maths_space определяет набор значений, который будет иметь точный математический смысл.

EXPRESS-описание:

```
*)
    ENTITY elementary_maths_space
    ABSTRACT SUPERTYPE OF (ONEOF (binary_data_type_definition, boolean_data_type_definition, complex_data_type_definition, event_data_type_definition, integer_data_type_definition, logical_data_type_definition, real_data_type_definition, string_data_type_definition) )
    SUBTYPE OF (maths_space) ;
    END_ENTITY;
(*
```

4.3.78 Прикладной компонент engineering_process_activity

Прикладной компонент engineering_process_activity определяет ту часть процесса разработки системы, которая может планироваться в процессе работы (или уже быть выполненной).

Примечание — В области действия настоящего стандарта прикладной компонент engineering_process_activity в процессе инженерного проектирования считается дискретным или составным элементом.

Пример 8 — В процессе инженерного проектирования фаза анализа требований, а также мер безопасности в отношении какой-либо подсистемы будет рассматриваться как соответствующая прикладному компоненту engineering_process_activity.

EXPRESS-описание:

```
*)
    ENTITY engineering_process_activityp;
    activity_type : label;
    actual_end_date : OPTIONAL date_time;
    actual_start_date : OPTIONAL date_time;
    description : OPTIONAL text_select;
    id : element_identifier;
    name : label;
    planned_end_date : OPTIONAL period_or_date_select;
    planned_start_date : OPTIONAL event_or_date_select;
    resolved_request : SET[0:?] OF work_request;
    status : OPTIONAL text_select;
    INVERSE
    authorization : SET[0:1] OF work_ordert FOR is_controlling;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут activity_type: Этот атрибут определяет назначение прикладного компонента engineering_process_activity. Там, где это применимо, должны использоваться следующие состояния (значения) этого атрибута:

- состояние analysis: Прикладной компонент engineering_process_activity представляет анализ операций;

- состояние design: Прикладной компонент engineering_process_activity представляет операцию разработки;

- состояние review: Прикладной компонент engineering_process_activity представляет операцию проверки, осуществляемой персоналом;

- состояние design_change: Прикладной компонент engineering_process_activity представляет операцию внесения изменений;

- состояние trade_off_analysis: Прикладной компонент engineering_process_activity представляет операцию выбора решений.

Атрибут actual_end_date: Этот атрибут определяет дату, когда прикладной компонент engineering_process_activity фактически был закончен.

Атрибут actual_start_date: Этот атрибут определяет дату, когда прикладной компонент engineering_process_activity фактически был начат.

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту engineering_process_activity.

Атрибут id: Этот атрибут определяет идентификатор прикладного компонента engineering_process_activity.

Атрибут name: Этот атрибут определяет слово (или слова), которые используются для ссылки на прикладной компонент engineering_process_activity.

Атрибут planned_end_date: Этот атрибут определяет прикладной компонент date_time, когда прикладной компонент engineering_process_activity завершается или был (по предположению) уже завершен.

Атрибут planned_start_date: Этот атрибут определяет прикладной компонент date_time, когда прикладной компонент engineering_process_activity начинается или был начат (по предположению).

Атрибут resolved_request: Этот атрибут определяет набор прикладных компонентов work_request, который допускается с помощью прикладного компонента engineering_process_activity.

Атрибут status: Этот атрибут определяет уровень исполнения прикладного компонента engineering_process_activity.

Атрибут authorization: Этот атрибут определяет прикладные компоненты engineering_process_activity, которые контролируются с помощью данного частного прикладного компонента work_order.

4.3.79 Прикладной компонент engineering_process_activity_element_assignment

Прикладной компонент engineering_process_activity_element_assignment определяет взаимосвязь между данными системного проектирования и прикладным компонентом engineering_process_activity, в котором данными являются состояния created in, modified in или referenced in.

EXPRESS-описание:

```
*)
    ENTITY engineering_process_activityp_element_assignment;
    activity : engineering_process_activityp;
    description : OPTIONAL text_select;
    element : specification_element_select;
    role : label;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут activity: Этот атрибут определяет прикладной компонент engineering_process_activity, к которому принадлежит прикладной компонент engineering_process_activity_element_assignment.

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту engineering_process_activity_element_assignment.

Атрибут element: Этот атрибут определяет фрагмент данных системного проектирования, на которые дается ссылка с помощью прикладного компонента engineering_process_activity.

Атрибут role: Этот атрибут определяет функцию, которая выполняется с помощью прикладного компонента engineering_process_activity_element_assignment в контексте рассматриваемого прикладного компонента engineering_process_activity. Там, где это применимо, должны использоваться следующие состояния (значения) этого компонента:

- состояние control: Указанный элемент оказывает влияние на завершение действия прикладного компонента engineering_process_activity;

- состояние created: Указанный элемент был сформирован в прикладном компоненте engineering_process_activity;
- состояние modified: Указанный элемент был изменен в прикладном компоненте engineering_process_activity.

4.3.80 Прикладной компонент engineering_process_activity_relationship

Прикладной компонент engineering_process_activity_relationship определяет взаимосвязь между двумя прикладными компонентами engineering_process_activity.

EXPRESS-описание:

```
*)
    ENTITY engineering_process_activityp_relationship;
    description : OPTIONAL text_select;
    related_activity : engineering_process_activityp;
    relating_activity : engineering_process_activityp;
    relation_type : label;
    WHERE
    WR1: relating_activity :<>: related_activity;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту engineering_process_activity_relationship.

Атрибут related_activity: Этот атрибут определяет второй из двух прикладных компонентов engineering_process_activity, связанных с помощью прикладного компонента engineering_process_activity_relationship.

Атрибут relating_activity: Этот атрибут определяет первый из двух прикладных компонентов engineering_process_activity, связанных с помощью прикладного компонента engineering_process_activity_relationship.

Атрибут relation_type: Этот атрибут определяет текстовое описание характера взаимосвязи между двумя прикладными компонентами engineering_process_activity, включенными в эту взаимосвязь. Там, где это применимо, должны использоваться следующие состояния (значения) этого атрибута:

- состояние alternative: Должен использоваться либо атрибут relating_activity, либо атрибут related_activity;
- состояние hierarchy: Связь между атрибутами relating_activity и related_activity такова, что атрибут relating_activity отделяется от атрибута related_activity;
- состояние sequence: Атрибуты relating_activity и related_activity должны использоваться последовательно, причем атрибут relating_activity должен находиться перед атрибутом related_activity;
- состояние simultaneous: Атрибуты relating_activity и related_activity могут использоваться одновременно.

Формальные выражения:

WR1:

4.3.81 Прикладной компонент event_data_type_definition

Прикладной компонент event_data_type_definition принадлежит к тому же типу, что и прикладной компонент elementary_maths_space, которые содержат все значения событий.

EXPRESS-описание:

```
*)
    ENTITY event_data_type_definition;
    SUBTYPE OF (elementary_maths_space) ;
    END_ENTITY;
(*
```

4.3.82 Прикладной компонент execution_time

Прикладной компонент execution_time является оценкой времени, которое необходимо функции для формирования выходных данных либо после их активации, либо сразу же после подготовки всего массива данных.

Примечание 1 — Время исполнения не следует путать с временем выполнения физическим компонентом своих функций, которые связаны с определением физической архитектуры в физическом функциональном модуле.

Пример 9 — Точная семантика данного прикладного компонента определяется с помощью атрибута role. Указанное в нем время исполнения регистрируется, например, в обрабатывающей промышленности при выполнении той или иной операции. При этом может понадобиться, например, информация о работе с данной кислотной ванной в течение определенного промежутка времени, обеспечивающей получение определенных свойств продукции.

Примечание 2 — Прикладной компонент execution_time может связываться со всеми типами прикладных компонентов general_function_definition.

Примечание 3 — Прикладной компонент execution_time определяет время, затрачиваемое на выполнение функции, без учета влияния других функций в системе.

EXPRESS-описание:

```
*)
    ENTITY execution_time;
    role : timing_type;
    time : REAL;
    timing : functionality_instance_reference;
    unit : label;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут role: Этот атрибут определяет время, необходимое для его интерпретации.

Пример 10 — Функция calculate_time_to_destination может иметь время, определяемое функцией best_case execution_time, равное 0 секунд; время, определяемое функцией worst_case execution_time — 0.020 секунд, а время, определяемое функцией nominal_case execution_time, — не выражаться.

Атрибут time: Этот атрибут определяет время выполнения функции.

Атрибут timing: Этот атрибут определяет прикладной компонент general_function_definition, для которого применим прикладной компонент execution_time.

Атрибут unit: Этот атрибут определяет единицу измерений прикладного компонента execution_time при использовании атрибута time.

4.3.83 Прикладной компонент finite_integer_interval

Прикладной компонент finite_integer_interval принадлежит к тому же типу, что и прикладной компонент integer_interval и имеет нижнюю и верхнюю границы, ограничивающие содержащееся в них подмножество целых чисел.

EXPRESS-описание:

```
*)
    ENTITY finite_integer_interval
    SUBTYPE OF (integer_interval) ;
    low_index : INTEGER;
    size : INTEGER;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут low_index: Этот атрибут определяет наименьшее значение, которое может принимать прикладной компонент data_instance данного типа.

Атрибут size: Этот атрибут определяет область значений для прикладного компонента finite_integer_interval в виде последовательности целых значений, отсчитываемых от атрибута low_index, а прикладной компонент data_instance этого типа может отбираться.

Примечание — Верхняя граница области значений для прикладного компонента finite_integer_interval задается следующим выражением: значение атрибута low_index + значение атрибута size — 1.

4.3.84 Прикладной компонент finite_real_interval

Прикладной компонент finite_real_interval принадлежит к тому же типу, что и прикладной компонент real_interval, который имеет нижнюю и верхнюю границы, ограничивающие содержащееся в них подмножество действительных чисел.

EXPRESS-описание:

```
*)
        ENTITY finite_real_interval
        SUBTYPE OF (real_interval) ;
        high_closure : BOOLEAN;
        high_index : REAL;
        low_closure : BOOLEAN;
        low_index : REAL;
        END_ENTITY;
(*
```

Определения атрибутов:

Атрибут high_closure: Этот атрибут определяет, включается ли атрибут high_index в прикладной компонент maths_space. При наличии состояния TRUE оно включается в этот компонент; в противном случае оно исключается из него.

Атрибут high_index: Этот атрибут определяет максимальное значение, которое может принимать прикладной компонент data_instance данного типа.

Атрибут low_closure: Этот атрибут определяет, включается ли атрибут low_index в прикладной компонент maths_space. При наличии состояния TRUE оно включается в этот компонент; в противном случае оно исключается из него.

Атрибут low_index: Этот атрибут определяет максимальное значение, которое может принимать прикладной компонент data_instance данного типа.

4.3.85 Прикладной компонент finite_space

Прикладной компонент finite_space принадлежит к тому же типу, что и прикладной компонент maths_space и содержит точный набор значений.

Примечание — Прикладные компоненты elementary_maths_space, которые являются конечными (как, например, прикладные компоненты binary_data_type_definition и logical_data_type_definition), должны использоваться, имея приоритет перед прикладными компонентами finite_space.

EXPRESS-описание:

```
*)
        ENTITY finite_space
        SUBTYPE OF (maths_space) ;
        member : SET[0:?] OF data_type_value_select;
        END_ENTITY;
(*
```

Определение атрибута:

Атрибут member: Этот атрибут определяет множество элементов, формирующих прикладной компонент finite_space.

4.3.86 Прикладной компонент formal_data_interaction_port

Прикладной компонент formal_data_interaction_port является элементом интерфейса к прикладному компоненту functional_state_context.

EXPRESS-описание:

```
*)
        ENTITY formal_data_interaction_port;
        data : data_instance;
        port_of : functional_state_context;
        END_ENTITY;
(*
```

Определения атрибутов:

Атрибут data: Этот атрибут определяет прикладной компонент data_instance в интерфейсе.

Атрибут port_of: Этот атрибут определяет прикладной компонент functional_state_context, для которого прикладной компонент formal_data_interaction_port является частью интерфейса.

4.3.87 Прикладной компонент formal_io_port

Прикладной компонент formal_io_port принадлежит к тому же типу, что и прикладной компонент io_port и определяет элемент интерфейса к прикладному компоненту general_function_definition.

Примечание 1 — Если прикладной компонент general_function_definition имеет интерфейс, то его будет определять один или несколько прикладных компонентов formal_io_port.

Примечание 2 — Порты для информационных потоков классифицируются в соответствии со следующими тремя критериями для модели данных.

1. Является ли порт формальным (закрепленным за прикладным компонентом general_function_definition) или реальным (закрепленным за прикладным компонентом function_instance, fsm_model, persistent_storage или io_split_join);
2. Является ли порт входным или выходным;
3. Является ли порт предназначенным для информационного потока или контрольным портом (порт для информационного потока переносит данные, тогда как контрольный порт служит для подачи управляющих сигналов, например запуска, останова, приостановки и возобновления выполнения операции).

Согласно вышеприведенной классификации может определяться та роль, которую должен играть порт по отношению к прикладным компонентам functional_link. Порт может принимать информационные потоки (т. е. данные, подаваемые на порт) или же он может формировать информационные потоки (т. е. данные, которые будут выдаваться из порта). Например, прикладной компонент formal_io_port, чьим состоянием атрибута direction является output, будет формировать данные, тогда как прикладной компонент actual_io_port, чьим состоянием атрибута direction является input, будет принимать данные.

В модели данных мы фиксируем указанную информацию в получаемом атрибуте role, который используется для гарантии того, что прикладные компоненты functional_link будут правильно объединены (прикладной компонент functional_link должен иметь одного получателя и одного отправителя), а прикладные компоненты io_port_binding будут применимы только к портам, для которых прикладной компонент actual_port в этой связи будет являться отправителем информации, а прикладной компонент в связи — получателем информации, и наоборот.

Примечание 3 — Если (и только если) прикладной компонент function_instance для прикладного компонента general_function_definition создает прикладной компонент actual_io_port, то будет существовать соответствующий каждому прикладному компоненту formal_io_port информационный поток, связанный с действующим портом (а этот порт связан с соответствующим прикладным компонентом formal_io_port посредством прикладного компонента io_port_binding).

EXPRESS-описание:

```
*)
    ENTITY formal_io_port
    SUBTYPE OF (io_port) ;
    port_of : general_function_definition;
    DERIVE
    SELF\io_port. RENAMED role : port_data_relation : determineformalportrole(SELF);
    UNIQUE
    UR1: port_of, io_port_number, port_type;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут port_of: Этот атрибут определяет прикладной компонент general_function_definition, для которого существует данный порт, усиливающийся с помощью обратной взаимосвязи и указывающий, что этот атрибут port_of может только вводиться при определении, однако это определение будет относиться к нескольким портам.

Атрибут role:

Формальные выражения:

UR1:

4.3.88 Прикладной компонент formal_physical_port

Прикладной компонент formal_physical_port принадлежит к тому же типу, что и прикладной компонент physical_port и представляет собой либо входные и выходные данные, либо входные данные или выходные данные для прикладного компонента general_physical_definition.

EXPRESS-описание:

```
*)
        ENTITY formal_physical_port
        SUBTYPE OF (physical_port) ;
        port_of : general_physical_definition;
        END_ENTITY;
(*
```

Определение атрибута:

Атрибут port_of: Этот атрибут определяет прикладной компонент general_physical_defintion, для которого прикладной компонент formal_physical_port является частью интерфейса.

4.3.89 Прикладной компонент formal_port_position

Прикладной компонент formal_port_position является экземпляром прикладного компонента visual_element и графическим представлением положения прикладного компонента formal_port, formal_io_port или formal_physical_port.

EXPRESS-описание:

```
*)
        ENTITY formal_port_position
        SUBTYPE OF (visual_element) ;
        position : graphics_point;
        positioned_port : port_position_select;
        END_ENTITY;
(*
```

Определения атрибутов:

Атрибут position: Этот атрибут определяет положение порта.

Атрибут positioned_port: Этот атрибут определяет формальный порт, который размещается с помощью прикладного компонента formal_port_position.

4.3.90 Прикладной компонент fsm_and_state

Прикладной компонент fsm_and_state принадлежит к тому же типу, что и прикладной компонент fsm_state и является представлением разделенного состояния в автомате с конечным числом состояний, в котором все дочерние прикладные компоненты fsm_state одновременно активны. Прикладной компонент and_state состоит по крайней мере из двух дочерних прикладных компонентов fsm_state.

EXPRESS-описание:

```
*)
        ENTITY fsm_and_state
        SUBTYPE OF (fsm_state) ;
        INVERSE
        SELF\fsm_state. RENAMED child_states : SET[2:?] OF fsm_state_composition_relationship FOR
        parent_state;
        END_ENTITY;
(*
```

Определение атрибута:

Атрибут child_states:

4.3.91 Прикладной компонент fsm_command_interaction_relationship

Прикладной компонент fsm_command_interaction_relationship определяет способ указания того, что команда будет выдаваться для ссылки на функцию (в виде прикладного компонента state_function_interaction_port) с помощью прикладного компонента textual_specification. Прикладной компонент fsm_command_interaction_relationship используется только в области применения автомата с конечным числом состояний. Характер этой команды определен в атрибуте interaction_type.

Примечание 1 — Прикладной компонент fsm_command_interaction_relationship указывает на то, что команда (функция: запуск, останов, приостановка, продолжение выполнения операции) выдается с помощью прикладного компонента fsm_transition или fsm_state. Обозначения дополняют текстовое определение операций в состояниях или при переходах.

EXPRESS-описание:

```
*)
    ENTITY fsm_command_interaction_relationship;
    defined_in : fsm_interaction_select;
    interaction_port : state_function_interaction_port;
    interaction_type : label;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут defined_in: Этот атрибут определяет прикладной компонент textual_specification, который будет выдавать команду.

Атрибут interaction_port: Этот атрибут определяет прикладной компонент state_function_interaction_port, посредством которого выдается команда, изменяемая с помощью прикладного компонента fsm_command_interaction_relationship.

Атрибут interaction_type: Этот атрибут определяет семантику прикладного компонента fsm_command_interaction_relationship. Там, где это применимо, должны использоваться следующие состояния (значения) этого атрибута:

- состояние start: Указанная функция инициализируется с помощью данной команды;
- состояние stop: Указанная функция деактивируется с помощью данной команды;
- состояние resume: Указанная функция повторно инициализируется с помощью данной команды из состояния приостановки;
- состояние suspend: Указанная функция временно приостанавливается с помощью данной команды.

Примечание 2 — Все предварительно заданные команды являются идемпотентными, т. е. выдача команды start для уже активированной функции не будет давать никакого эффекта.

4.3.92 Прикладной компонент fsm_data_interaction_binding

Прикладной компонент fsm_data_interaction_binding определяет параметр согласования взаимосвязи между элементом actual_interface в реальном интерфейсе к прикладному компоненту actual_io_port и элементу formal_interface прикладного компонента formal_data_interaction_port.

EXPRESS-описание:

```
*)
    ENTITY fsm_data_interaction_binding;
    actual_port : actual_io_port;
    formal_port : formal_data_interaction_port;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут actual_port: Этот атрибут определяет прикладной компонент actual_io_port в указанной взаимосвязи.

Атрибут formal_port: Этот атрибут определяет прикладной компонент formal_data_interaction_port в указанной взаимосвязи.

4.3.93 Прикладной компонент fsm_data_interaction_relationship

Прикладной компонент fsm_data_interaction_relationship определяет способ указания привязки прикладного компонента data_instance к прикладному компоненту textual_specification. Характер этой привязки определяется атрибутом interaction_type.

Примечание — Прикладной компонент fsm_data_interation_relationship, который связан с элементом данных прикладного компонента io_port fsm_interaction_port, обновляется или считывается в процессе выполнения части операции в данном состоянии или при переходе.

EXPRESS-описание:

```
*)
    ENTITY fsm_data_interaction_relationship;
    defined_in : fsm_interaction_select;
    interaction_port : formal_data_interaction_port;
```

```
    interaction_type : label;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут defined_in: Этот атрибут определяет прикладной компонент textual_specification, который будет выдавать команду.

Атрибут interaction_port: Этот атрибут определяет прикладной компонент formal_data_interaction_port в указанной взаимосвязи.

Атрибут interaction_type: Этот атрибут определяет семантику прикладного компонента fsm_data_interaction_relationship. Там, где это применимо, должны использоваться следующие состояния (значения) этого атрибута:

- состояние read: Значение прикладного компонента data_instance в прикладном компоненте formal_data_interaction_port указывается в прикладном компоненте textual_specification, идентифицированном с помощью атрибута defined_in;
- состояние write: Значение прикладного компонента data_instance в прикладном компоненте formal_data_interaction_port записывается в прикладном компоненте textual_specification, идентифицированном с помощью атрибута defined_in.

4.3.94 Прикладной компонент fsm_generic_state

Прикладной компонент fsm_generic_state является абстрактным суперклассом либо прикладного компонента fsm_state, либо прикладного компонента fsm_generic_state.

EXPRESS-описание:

```
*)
    ENTITY fsm_generic_state
    ABSTRACT SUPERTYPE OF ( ONEOF(fsm_state, fsm_transient_state) ) ;
    INVERSE
    destination_transition : SET[0:?] OF fsm_state_transition FOR destination_state;
    END_ENTITY;
(*
```

Определение атрибута:

Атрибут destination_transition: Этот атрибут определяет прикладной компонент fsm_state, активированный при условии инициализации прикладного компонента fsm_state_transition.

4.3.95 Прикладной компонент fsm_initial_state_transition

Прикладной компонент fsm_initial_state_transition определяет начальный переход в исходном состоянии прикладного компонента fsm_or_state в автомате с конечным числом состояний.

EXPRESS-описание:

```
*)
    ENTITY fsm_initial_state_transition;
    initial_state : fsm_generic_state;
    transition_context : default_context_select;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут initial_state: Этот атрибут определяет начальное состояние.

Атрибут transition_context: Этот атрибут определяет среду, для которой прикладной компонент initial_state_context является действующим.

4.3.96 Прикладной компонент fsm_model

Прикладной компонент fsm_model принадлежит к тому же типу, что и прикладной компонент general_functionality_instance и представляет собой точку входа в автомат с конечным числом состояний.

Примечание 1 — Прикладные компоненты fsm_model могут относиться к функциональной структуре разделения и представлять активацию/деактивацию логики работы функций в этой структуре.

Примечание 2 — Данный прикладной компонент включается в модель для ввода FSM в прикладной компонент general_function_definition, что будет обеспечивать связь между функциональным представлением и представлением о состоянии системы.

EXPRESS-описание:

```
*)
        ENTITY fsm_model
        SUBTYPE OF (general_functionality_instance) ;
        behaviour_model : state_machine_functional_behaviour_model;
        definition : functional_state_context;
        id : element_identifier;
        name : label;
        presentation_id : OPTIONAL label;
        END_ENTITY;
(*
```

Определения атрибутов:

Атрибут behaviour_model: Этот атрибут определяет прикладной компонент state_machine_functional_behaviour_model, для которого прикладной компонент fsm_model предоставляет спецификацию поведения.

Атрибут definition: Этот атрибут определяет прикладной компонент functional_state_context, содержащий модель автомата с конечным числом состояний.

Атрибут id: Этот атрибут определяет идентификатор прикладного компонента fsm_model.

Атрибут name: Этот атрибут определяет слово (или слова), которые используются для ссылки на прикладной компонент fsm_model.

Атрибут presentation_id: Этот атрибут определяет информацию, подтверждающую идентичность прикладного компонента fsm_model и представляемую пользователю.

4.3.97 Прикладной компонент fsm_or_state

Прикладной компонент fsm_or_state принадлежит к тому же типу, что и прикладной компонент fsm_state для автомата с конечным числом состояний, который в случае разделения будет ограничивать семантику дочерних прикладных компонентов fsm_state, так что только одно из дочерних состояний может быть активировано (если активно родительское состояние).

Примечание 1 — Определение подкласса, возможно, является оптимальным способом документирования данного свойства.

Примечание 2 — Прикладной компонент fsm_or_state соответствует или является картой состояний и нормального состояния автомата с конечным числом состояний.

EXPRESS-описание:

```
*)
        ENTITY fsm_or_state
        SUBTYPE OF (fsm_state);
        END_ENTITY;
(*
```

4.3.98 Прикладной компонент fsm_state

Прикладной компонент fsm_state принадлежит к тому же типу, что и прикладной компонент fsm_generic_state и характеризует статическое состояние автомата с конечным числом состояний.

Примечание — В рамках протокола AP 233 прикладной компонент fsm_state всегда будет являться элементом автомата с конечным числом состояний. Понятие «состояние» в EACM содержится в протоколе AP-233 (по терминологии прикладного компонента partial_system_view).

EXPRESS-описание:

```
*)
        ENTITY fsm_state
        ABSTRACT SUPERTYPE OF ( ONEOF(fsm_and_state, fsm_or_state) )
        SUBTYPE OF (fsm_generic_state) ;
        description : OPTIONAL text_select;
        name : label;
        presentation_id : OPTIONAL label;
        INVERSE
```

```
child_states : SET[0:?] OF fsm_state_composition_relationship FOR parent_state;
END_ENTITY;
(*
```

Определения атрибутов:

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к описанию.

Атрибут name: Этот атрибут определяет слово (или слова), которые используются для ссылки на него.

Атрибут presentation_id: Этот атрибут определяет информацию, подтверждающую его идентичность и предоставляемую пользователю.

Атрибут child_states: Этот атрибут определяет родительский прикладной компонент fsm_state в указанной связи.

4.3.99 Прикладной компонент fsm_state_composition_relationship

Прикладной компонент fsm_state_composition_relationship определяет взаимосвязь между двумя прикладными компонентами fsm_state, а также связи между родительскими и дочерними компонентами.

Примечание 1 — Родительский прикладной компонент fsm_state идентифицируется с помощью атрибута parent_state, а дочерний прикладной компонент fsm_state — с помощью атрибута child_state.

Примечание 2 — Прикладной компонент fsm_state_composition_relationship включается в модель для поддержки расширений карты состояний на традиционное представление автомата с конечным числом состояний.

EXPRESS-описание:

```
*)
    ENTITY fsm_state_composition_relationship;
    child_state : fsm_state;
    parent_state : fsm_state;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут child_state: Этот атрибут определяет дочерний прикладной компонент fsm_state в указанной взаимосвязи.

Атрибут parent_state: Этот атрибут определяет родительский прикладной компонент fsm_state в указанной взаимосвязи.

4.3.100 Прикладной компонент fsm_state_transition

Прикладной компонент fsm_state_transition является представлением возможной исполняемой ветви между двумя прикладными компонентами fsm_generic_state.

Примечание — Прикладной компонент fsm_state_transition применим для использования только в автоматах с конечным числом состояний.

EXPRESS-описание:

```
*)
    ENTITY fsm_state_transition;
    destination_state : fsm_generic_state;
    source_state : fsm_generic_state;
    INVERSE
    transition_label : SET[0:1] OF state_transition_specification_assignment FOR assigned_to;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут destination_state: Этот атрибут определяет прикладной компонент fsm_state, активируемый только при инициализации прикладного компонента fsm_state_transition.

Атрибут source_state: Этот атрибут определяет прикладной компонент fsm_state, который первоначально был активным в указанной взаимосвязи.

Атрибут transition_label: Этот атрибут определяет прикладной компонент fsm_state_transition, которому присваивается текстовая спецификация.

4.3.101 Прикладной компонент fsm_transient_state

Прикладной компонент fsm_transient_state принадлежит к тому же типу, что и прикладной компонент fsm_generic_state при отсутствии дискриминатора состояния, при любых условиях и статическом состоянии автомата с конечным числом состояний.

Примечание — В терминах карты состояний прикладной компонент fsm_transient_state является псевдосостоянием. Ни при каких условиях прикладной компонент fsm_transient_state в системе не должен находиться в статическом состоянии. Если существуют переходы вне прикладного компонента fsm_transient_state, то они должны быть либо незащищенными, либо должны быть совокупностью условий защиты, т. е. по крайней мере одна степень защиты всегда должна расцениваться как находящаяся в состоянии TRUE.

EXPRESS-описание:

```
*)
    ENTITY fsm_transient_state
    SUBTYPE OF (fsm_generic_state);
    state_type : label;
    END_ENTITY;
(*
```

Определение атрибута:

Атрибут state_type: Этот атрибут определяет тип прикладного компонента fsm_transient_state. Там, где это применимо, должны использоваться следующие состояния (значения) этого атрибута:

- состояние history;
- состояние deep-history;
- состояние condition;
- состояние select.

4.3.102 Прикладной компонент fsm_transient_state_composition_relationship

Прикладной компонент fsm_transient_state_composition_relationship определяет взаимосвязь между прикладными компонентами fsm_state и fsm_transient_state, указывающую на то, что прикладной компонент fsm_transient_state является субсостоянием прикладного компонента fsm_state.

Примечание — Прикладной компонент fsm_transient_state_composition_relationship может быть реализован только при наличии по крайней мере одного прикладного компонента fsm_state_composition_relationship, существующего для прикладного компонента fsm_state. Прикладной компонент fsm_state не может состоять только из прикладных компонентов fsm_transient_state.

EXPRESS-описание:

```
*)
    ENTITY fsm_transient_state_composition_relationship;
    child_state : fsm_transient_state;
    parent_state : fsm_state;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут child_state: Этот атрибут определяет прикладной компонент fsm_transient_state в прикладном компоненте fsm_transient_state_composition_relationship.

Атрибут parent_state: Этот атрибут определяет прикладной компонент fsm_state в прикладном компоненте fsm_transient_state_composition_relationship.

4.3.103 Прикладной компонент function_instance

Прикладной компонент function_instance принадлежит к тому же типу, что и прикладной компонент general_functionality_instance и представляет собой объект, который выполняет операцию в системе.

Примечание — Функциональная область описывает характер действий системы.

EXPRESS-описание:

```
*)
    ENTITY function_instance
    SUBTYPE OF (general_functionality_instance) ;
    definition : general_function_definition;
    id : element_identifier;
    name : label;
```

```
        presentation_id : OPTIONAL label;
        INVERSE
        control_port : SET[0:?] OF control_io_port FOR port_of;
        UNIQUE
        UR1: definition, id;
        END_ENTITY;
(*
```

Определения атрибутов:

Атрибут definition: Этот атрибут определяет прикладной компонент general_function_definition, который дает описание прикладного компонента function_instance.

Атрибут id: Этот атрибут определяет идентификатор прикладного компонента function_instance.

Атрибут name: Этот атрибут определяет слово (или слова), которые используются для ссылки на прикладной компонент function_instance.

Атрибут presentation_id: Этот атрибут определяет информацию, подтверждающую идентичность прикладного компонента function_instance и предоставляемую пользователю.

Атрибут control_port: Этот атрибут определяет прикладной компонент function_instance, к которому присоединяется прикладной компонент control_io_port.

Формальные выражения:

UR1:

4.3.104 Прикладной компонент function_reference

Прикладной компонент function_reference определяет взаимосвязь между прикладными компонентами function_instance и fsm_model.

Примечание 1 — Прикладной компонент function_reference дает способ сопоставления элементов в пространстве имен прикладного компонента composite_function_definition с пространством имен для автомата с конечным числом состояний.

Примечание 2 — Для реализации способа сопоставления этих элементов должны использоваться прикладные компоненты state_function_interaction_port и name_binding.

EXPRESS-описание:

```
*)
        ENTITY function_reference;
        function_link : function_instance;
        port_of : fsm_model;
        END_ENTITY;
(*
```

Определения атрибутов:

Атрибут function_link: Этот атрибут определяет прикладной компонент function_instance, который будет импортироваться в автомат с конечным числом состояний.

Атрибут port_of: Этот атрибут определяет прикладной компонент fsm_model, в который будет импортироваться прикладной компонент function_instance.

4.3.105 Прикладной компонент functional_behaviour_model

Прикладной компонент functional_behaviour_model является определением функциональной модели поведения системы. Прикладной компонент functional_behaviour_model является абстрактным супертипом, который никогда не должен реализовываться.

EXPRESS-описание:

```
*)
        ENTITY functional_behaviour_model
        ABSTRACT SUPERTYPE OF ( ONEOF(cb_functional_behaviour_model, state_machine_ functional_behaviour_model) ) ;
        description : OPTIONAL text_select;
        INVERSE
        defines_behaviour_for : SET[0:1] OF functional_behaviour_model_assignment FOR assigned_behaviour_model;
        END_ENTITY;
(*
```

Определения атрибутов:

Атрибут description: Этот атрибут определяет дополнительную текстовую информацию, относящуюся к прикладному компоненту functional_behaviour_model.

Атрибут defines_behaviour_for: Этот атрибут определяет присвоенный прикладной компонент functional_behaviour_model.

4.3.106 Прикладной компонент functional_behaviour_model_assignment

Прикладной компонент functional_behaviour_model_assignment является присваиваемой взаимосвязью между прикладными компонентами functional_behaviour_model и composite_function_definition, чье поведение определяется с помощью прикладного компонента functional_behaviour_model.

EXPRESS-описание:

```
*)
    ENTITY functional_behaviour_model_assignment;
    assigned_behaviour_model : functional_behaviour_model;
    constrained_function : composite_function_definition;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут assigned_behaviour_model: Этот атрибут определяет присвоенный прикладной компонент functional_behaviour_model.

Атрибут constrained_function: Этот атрибут определяет прикладной компонент composite_function_definition, чье поведение ограничивается присвоенным прикладным компонентом functional_behaviour_model.

4.3.107 Прикладной компонент functional_decomposition_relationship

Прикладной компонент functional_decomposition_relationship является иерархическим соотношением между прикладными компонентами composite_function_definition и general_functionality_instance.

Примечание 1 — Прикладной компонент composite_function_definition в этом соотношении является родительским, а прикладной компонент general_functionality_instance — дочерним.

EXPRESS-описание:

```
*)
    ENTITY functional_decomposition_relationship;
    child : general_functionality_instance;
    description : OPTIONAL text_select;
    parent : composite_function_definition;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут child: Этот атрибут определяет прикладной компонент general_functionality_instance, который является частью составного родительского компонента.

Атрибут description: Этот атрибут определяет ту роль, которую дочерний компонент должен играть в прикладном компоненте composite_function_definition, определенном с помощью атрибута parent.

Примечание 2 — Поскольку дочерний компонент может входить во многие прикладные компоненты composite_function_definition, атрибут description позволяет давать описание назначения дочернего компонента в соответствующем прикладном компоненте composite_function_definition.

Атрибут parent: Этот атрибут определяет прикладной компонент composite_function_definition, для которого дочерний компонент является частью разделения.

4.3.108 Прикладной компонент functional_link

Прикладной компонент functional_link определяет взаимосвязь между двумя прикладными компонентами io_port, при которой будет существовать путь взаимодействия данных между двумя портами.

Примечание 1 — Направление этого пути для данных: от атрибута source_port к атрибуту destination_port.

Примечание 2 — Данные, переносимые прикладным компонентом functional_link, определяются с помощью данных, содержащихся в атрибутах source_port и destination_port. Один и тот же прикладной компонент data_instance должен быть на портах источника и пункта назначения.

EXPRESS-описание:

```
*)
    ENTITY functional_link;
    control_link : BOOLEAN;
    description : OPTIONAL text_select;
    destination_port : io_port;
    name : label;
    source_port : io_port;
    DERIVE
    data_on_link : data_instance : source_port.data;
    WHERE
    correct_ports: ((destination_port.role = consumer) AND (source_port.role = producer));
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут control_link: Этот атрибут определяет, переносит ли прикладной компонент functional_link контрольную информацию.

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту functional_link.

Атрибут destination_port: Этот атрибут определяет прикладной компонент io_port, который является получателем данных.

Атрибут name: Этот атрибут определяет слово (или слова), которые используются для ссылки на прикладной компонент functional_link.

Атрибут source_port: Этот атрибут определяет прикладной компонент io_port, который является отправителем данных.

Атрибут data_on_link:

Формальные выражения:

correct_ports:

4.3.109 Прикладной компонент functional_link_allocation_relationship

Прикладной компонент functional_link_allocation_relationship определяет способ указания того, что прикладной компонент functional_link_reference должен направляться в соответствующий прикладной компонент physical_instance_reference.

Примечание — Необходимое предварительное условие подобного размещения — это то, что размещенный прикладной компонент functional_link будет относиться к прикладному компоненту physical_instance_reference, который должен являться частью того же прикладного компонента system_view.

EXPRESS-описание:

```
*)
    ENTITY functional_link_allocation_relationship;
    allocated_functional_link : functional_link_reference;
    allocated_to : physical_instance_reference;
    description : OPTIONAL text_select;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут allocated_functional_link: Этот атрибут определяет прикладной компонент functional_link_reference в прикладном компоненте functional_link_allocation_relationship.

Атрибут allocated_to: Этот атрибут определяет прикладной компонент physical_instance_reference в указанном способе.

Атрибут description: Этот атрибут определяет дополнительную текстовую информацию, относящуюся к прикладному компоненту functional_link_allocation_relationship.

4.3.110 Прикладной компонент functional_link_group

Прикладной компонент functional_link_group является хранилищем для группирования связанных прикладных компонентов functional_link.

Примечание — Прикладной компонент functional_link_group позволяет разработчику обращаться к группе прикладных компонентов functional_link, как если бы они составляли единый информационный поток. Одной из причин использования прикладного компонента functional_link_group в модели является возможность производить присвоение одних и тех же графических свойств объектам группы, которые относятся к элементам прикладного компонента functional_link_group.

EXPRESS-описание:

```
*)
    ENTITY functional_link_group;
    elements : SET[2:?] OF functional_link;
    name : label;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут elements: Этот атрибут определяет множество прикладных компонентов functional_link, которые формируют прикладной компонент functional_link_group.

Атрибут name: Этот атрибут определяет слово (или слова), которые используются для ссылки на прикладной компонент functional_link_group.

4.3.111 Прикладной компонент functional_link_reference

Прикладной компонент functional_link_reference определяет однозначную ссылку на прикладные компоненты functional_link в контексте прикладного компонента general_functionality_instance, который в дальнейшем будет разделен.

EXPRESS-описание:

```
*)
    ENTITY functional_link_reference;
    in_scope_of : functionality_instance_reference;
    reference_functional_link : functional_link;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут in_scope_of: Этот атрибут определяет прикладной компонент functionality_instance_reference, который указывает контекст прикладного компонента functional_link и идентифицируется с помощью прикладного компонента functional_link_reference.

Атрибут reference_functional_link: Этот атрибут определяет прикладной компонент functional_link, для которого прикладной компонент functional_link_reference является однозначной ссылкой.

4.3.112 Прикладной компонент functional_reference_configuration

Прикладной компонент functional_reference_configuration определяет совокупность всех прикладных компонентов functionality_reference_composition_relationship, которая применима для конкретного функционального представления системы.

Примечание 1 — Прикладной компонент functional_reference_configuration может присваиваться любому числу систем посредством использования нескольких прикладных компонентов system_functional_configuration.

Примечание 2 — Прикладной компонент functional_reference_configuration дает способ определения нескольких нефункциональных представлений архитектуры функциональной модели одиночной системы.

EXPRESS-описание:

```
*)
    ENTITY functional_reference_configuration;
    description : OPTIONAL text_select;
    END_ENTITY;
(*
```

Определение атрибута:

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту functional_reference_configuration.

4.3.113 Прикладной компонент functional_representation_relationship

Прикладной компонент functional_representation_relationship определяет способ указания того, как прикладной компонент data_instance, используемый в функциональном представлении спецификации, будет псевдонимом для прикладного компонента physical_instance, исходя из физического представления спецификации.

Примечание — Наличие прикладного компонента functional_representation_relationship мотивируется необходимостью ссылки функциональных моделей на другие элементы, кроме типов данных (при условии поддержания точной структуры в данной модели).

Альтернативным вариантом может стать разрешение прикладным компонентам io_port напрямую передавать компоненты других типов, кроме типа прикладного компонента data_instance. Хотя напрямую этот подход будет приводить к серьезным проблемам в интерпретации средств, неспособных представлять все компоненты, кроме других прикладных компонентов data_instance.

EXPRESS-описание:

```
*)
    ENTITY functional_representation_relationship;
    description : OPTIONAL text_select;
    functional_representation : data_instance;
    physical_element : physical_instance;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту functional_representation_relationship.

Атрибут functional_representation: Этот атрибут определяет прикладной компонент data_instance, задаваемый как псевдоним для прикладного компонента physical_instance и определяемый с помощью атрибута physical_element прикладного компонента functional_representation_relationship.

Атрибут physical_element: Этот атрибут определяет прикладной компонент physical_instance, входящий в прикладной компонент functional_representation_relationship.

4.3.114 Прикладной компонент functional_state_context

Прикладной компонент functional_state_context принадлежит к тому же типу, что и прикладной компонент generic_state_context и входит в автомат с конечным числом состояний.

Примечание 1 — Существует определенный прикладной компонент functional_state_context, поскольку использование FSM-моделей отличается в структурированном анализе и при объектно-ориентированном инженерном проектировании.

Примечание 2 — Прикладной компонент functional_state_context может содержать любое число состояний и переходов, но никакой переход не может пересекать границы, установленной для прикладного компонента functional_state_context.

EXPRESS-описание:

```
*)
    ENTITY functional_state_context
    SUBTYPE OF (generic_state_context) ;
    END_ENTITY;
(*
```

4.3.115 Прикладной компонент functionality_allocation_relationship

Прикладной компонент functionality_allocation_relationship определяет способ преобразования прикладного компонента functionality_instance_reference в конкретный прикладной компонент physical_instance_reference, который будет реализовывать функциональные характеристики.

EXPRESS-описание:

```
*)
    ENTITY functionality_allocation_relationship;
    allocated_functionality : functionality_instance_reference;
    allocated_to : physical_instance_reference;
```

```
        description : OPTIONAL text_select;
        END_ENTITY;
(*
```

Определения атрибутов:

Атрибут allocated_functionality: Этот атрибут определяет размещенный прикладной компонент functionality_instance_reference.

Атрибут allocated_to: Этот атрибут определяет прикладной компонент physical_instance_reference, к которому относится прикладной компонент functionality_allocation_relationship.

Атрибут description: Этот атрибут определяет дополнительную текстовую информацию, относящуюся к прикладному компоненту functionality_allocation_relationship.

4.3.116 Прикладной компонент functionality_instance_reference

Прикладной компонент functionality_instance_reference является однозначной ссылкой на элемент general_functionality_instance в структуре прикладного компонента composite_function_definition.

Примечание — Прикладной компонент functionality_instance_reference вводится для обеспечения размещения и обращения зависящих от системы нефункциональных характеристик системы к ее функциональному описанию.

EXPRESS-описание:

```
*)
        ENTITY functionality_instance_reference
        SUPERTYPE OF (persistent_storage_reference) ;
        description : OPTIONAL text_select;
        id : element_identifier;
        name : label;
        referenced_functionality_instance : general_functionality_instance;
        END_ENTITY;
(*
```

Определения атрибутов:

Атрибут description: Этот атрибут определяет дополнительную текстовую информацию, относящуюся к прикладному компоненту functionality_instance_reference.

Атрибут id: Этот атрибут определяет идентификатор прикладного компонента functionality_instance_reference.

Атрибут name: Этот атрибут определяет слово (или слова), которые используются для ссылки на прикладной компонент functionality_instance_reference.

Атрибут referenced_functionality_instance: Этот атрибут определяет прикладной компонент general_functionality_instance, чей прикладной компонент functionality_instance_reference является ссылкой.

4.3.117 Прикладной компонент functionality_reference_composition_relationship

Прикладной компонент functionality_reference_composition_relationship определяет способ указания иерархической связи между двумя прикладными компонентами functionality_instance_reference.

EXPRESS-описание:

```
*)
        ENTITY functionality_reference_composition_relationship;
        child : functionality_instance_reference;
        mirror_of : functional_decomposition_relationship;
        parent : functionality_instance_reference;
        reference_configuration : functional_reference_configuration;
        END_ENTITY;
(*
```

Определения атрибутов:

Атрибут child: Этот атрибут определяет функциональные характеристики компонента в прикладном компоненте functionality_reference_composition_relationship.

Атрибут mirror_of: Этот атрибут определяет прикладной компонент functional_decomposition_relationship, для которого прикладной компонент functionality_reference_composition_relationship обеспечивает однозначную ссылку.

Атрибут parent: Этот атрибут определяет разделенные функциональные характеристики в прикладном компоненте functionality_reference_composition_relationship.

Атрибут reference_configuration: Этот атрибут определяет прикладной компонент functional_reference_configuration, для которого действующими являются родительские и дочерние прикладные компоненты functionality_instance_reference.

4.3.118 Прикладной компонент functionality_reference_relationship

Прикладной компонент functionality_reference_relationship определяет способ фиксации двухсторонних взаимосвязей между двумя прикладными компонентами functionality_instance_reference с целью фиксации информации.

Пример 11 — Требования к временным взаимосвязям между двумя прикладными компонентами functionality_instance_reference.

EXPRESS-описание:

```
*)
    ENTITY functionality_reference_relationship;
    first_reference : functionality_instance_reference;
    second_reference : functionality_instance_reference;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут first_reference: Этот атрибут определяет первый прикладной компонент functionality_instance_reference в указанных взаимосвязях.

Атрибут second_reference: Этот атрибут определяет второй прикладной компонент functionality_instance_reference в указанных взаимосвязях.

4.3.119 Прикладной компонент general_function_definition

Прикладной компонент general_function_definition дает структурированное определение функции (или процесса) на определенном абстрактном уровне в системе. Каждый прикладной компонент general_function_definition при этом будет являться либо прикладным компонентом composite_function_definition, либо прикладным компонентом leaf_function_definition.

EXPRESS-описание:

```
*)
    ENTITY general_function_definition
    ABSTRACT SUPERTYPE OF ( ONEOF(composite_function_definition, leaf_function_definition)
    ) ;
    associated_version : configuration_element_version;
    description : OPTIONAL text_select;
    id : element_identifier;
    name : OPTIONAL label;
    INVERSE
    formal_port : SET[0:?] OF formal_io_port FOR port_of;
    UNIQUE
    UR1: id;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут associated_version: Этот атрибут определяет прикладной компонент configuration_element_version.

Атрибут description: Этот атрибут определяет дополнительную текстовую спецификацию на прикладной компонент general_function_definition.

Атрибут id: Этот атрибут определяет собственный идентификатор.

Атрибут name: Этот атрибут определяет слово (или слова), которые используются для ссылки на прикладной компонент general_function_definition.

Атрибут formal_port: Этот атрибут определяет прикладной компонент general_function_definition, для которого он является портом. Этот атрибут усиливается путем обратной взаимосвязи, которая ука-

зывает на то, что компонент port_of может быть только пунктом в определении, однако определение может ссылаться на несколько портов.

Формальные выражения:

UR1:

4.3.120 Прикладной компонент general_functionality_instance

Каждый прикладной компонент general_functionality_instance является либо прикладным компонентом fsm_model, либо прикладным компонентом function_instance, либо прикладным компонентом io_split_join, либо прикладным компонентом persistent_storage.

EXPRESS-описание:

```
*)
    ENTITY general_functionality_instance
    ABSTRACT SUPERTYPE OF ( ONEOF(sm_model, function_instance, io_split_join, persistent_
    storage) ) ;
    description : OPTIONAL text_select;
    INVERSE
    actual_port : SET[0:?] OF actual_io_port FOR port_of;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут description: Этот атрибут определяет дополнительную информацию для описания.

Атрибут actual_port: Этот атрибут определяет прикладной компонент general_functionality_instance, частью которого является прикладной компонент port_of.

4.3.121 Прикладной компонент general_physical_definition

Каждый прикладной компонент general_physical_definition является либо прикладным компонентом physical_link_definition, либо прикладным компонентом physical_node_definition.

Примечание 1 — Прикладной компонент general_physical_definition является спецификацией объекта, который при его реализации становится физическим элементом и которого a) можно касаться, b) определять его вес и c) измерять его ненулевые физические размеры.

Примечание 2 — Спецификация PAS 20542 дает только абстрактное представление физических компонентов, входящих в систему. Обоснование подобного представления состоит в том, что каждый физический компонент системы может иметь различные характеристики, наилучшим образом представляемые с помощью множества прикладных протоколов, доступных в системе STEP.

Предоставляемые ограниченные функциональные характеристики предназначены для того, чтобы дать возможность системным инженерам преобразовывать функции и информационные потоки в компоненты с целью связать требования к применяемым компонентам, а также для фиксации других свойств физического компонента.

EXPRESS-описание:

```
*)
    ENTITY general_physical_definition
    ABSTRACT SUPERTYPE OF ( ONEOF(physical_link_definition, physical_node_definition) ) ;
    associated_version : configuration_element_version;
    description : OPTIONAL text_select;
    id : element_identifier;
    name : OPTIONAL label;
    INVERSE
    formal_port : SET[0:?] OF formal_physical_port FOR port_of;
    UNIQUE
    UR1: id;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут associated_version: Этот атрибут определяет прикладной компонент configuration_element_version для прикладного компонента general_physical_definition.

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту general_physical_definition.

Атрибут id: Этот атрибут определяет идентификатор прикладного компонента general_physical_definition.

Атрибут name: Этот атрибут определяет слово (или слова), которые используются для ссылки на прикладной компонент general_physical_definition.

Атрибут formal_port: Этот атрибут определяет прикладной компонент general_physical_defintion, для которого прикладной компонент formal_physical_port является частью интерфейса.

Формальные выражения:

UR1:

4.3.122 Прикладной компонент generic_state_context

Прикладной компонент generic_state_context определяет оболочку автомата с конечным числом состояний или карту состояний. Каждый прикладной компонент generic_state_context является и прикладным компонентом functional_state_context.

EXPRESS-описание:

```
*)
    ENTITY generic_state_context
    ABSTRACT SUPERTYPE ;
    associated_version : configuration_element_version;
    description : OPTIONAL text_select;
    id : element_identifier;
    name : OPTIONAL label;
    original_representation : label;
    state_machine_model : label;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут associated_version: Этот атрибут определяет прикладной компонент configuration_element_version для прикладного компонента generic_state_context.

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту generic_state_context.

Атрибут id: Этот атрибут определяет идентификатор прикладного компонента generic_state_context.

Атрибут name: Этот атрибут определяет слово (или слова), которые используются для ссылки на прикладной компонент generic_state_context.

Атрибут original_representation: Этот атрибут определяет способ, с помощью которого был представлен автомат с конечным числом состояний, ограничиваемый с помощью прикладного компонента generic_state_context. Там, где это применимо, для представления атрибута original_representation должно использоваться одно из следующих состояний (значений) этого компонента:

- состояние table: Состояние автомата представляется в исходном средстве в виде таблицы;
- состояние graph: Состояние автомата представляется в исходном средстве в графическом виде.

Атрибут state_machine_model: Этот атрибут определяет тип конечного автомата, ограничиваемый с помощью прикладного компонента generic_state_context. Там, где это применимо, должны использоваться следующие состояния (значения) этого атрибута:

- состояние moore: Закрытое состояние автомата является единообразным. Никакие состояния не могут разделяться. Выходной алфавит конечного автомата формируется с помощью прикладных компонентов fsm_generic_state;
- состояние mealy: Закрытое состояние автомата является единообразным. Никакие состояния не могут разделяться. Выходной алфавит конечного автомата формируется с помощью прикладных компонентов fsm_state_transition и fsm_initial_state_transition;
- состояние hierarchical: Состояния в закрытом конечном автомате могут разбиваться на другие субсостояния. Выходной алфавит может формироваться с помощью прикладных компонентов fsm_generic_state, fsm_state_transition и fsm_initial_state_transition.

4.3.123 Прикладной компонент graphics_link

Прикладной компонент graphics_link принадлежит к тому же типу, что и прикладной компонент visual_element и представляется в открытой графической форме, содержащей перечень прикладных компонентов graphics_point.

Примечание — Графическая информация напрямую не связывается с компонентами, которые представляются с помощью прикладного компонента graphics_link, поскольку эти компоненты в различных контекстах будут иметь полностью отличающиеся представления.

EXPRESS-описание:

```
*)
    ENTITY graphics_link
    SUBTYPE OF (visual_element) ;
    associated_with : link_select;
    point : LIST[2:?] OF graphics_point;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут associated_with: Этот атрибут определяет прикладной компонент configuration_element, для которого прикладной компонент graphics_link дает визуальную компоновку.

Атрибут point: Этот атрибут определяет перечень координат для прикладного компонента graphics_link.

4.3.124 Прикладной компонент graphics_node

Прикладной компонент graphics_node принадлежит к тому же типу, что и прикладной компонент visual_element и является представлением замкнутого прямоугольника при указании положения объекта.

EXPRESS-описание:

```
*)
    ENTITY graphics_node
    SUBTYPE OF (visual_element) ;
    associated_with : node_select;
    bottom_right : graphics_point;
    top_left : graphics_point;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут associated_with: Этот атрибут определяет прикладной компонент, для которого он дает визуальную информацию о положении объекта.

Атрибут bottom_right: Этот атрибут определяет прикладной компонент graphics_point, дающий информацию о положении нижнего правого угла прямоугольника.

Атрибут top_left: Этот атрибут определяет прикладной компонент graphics_point, дающий информацию о положении верхнего левого угла прямоугольника.

4.3.125 Прикладной компонент graphics_point

Прикладной компонент graphics_point определяет положение точки на координатной плоскости.

Примечание 1 — Координатная плоскость занимает область {0<= x, y <= 1}. Никакой информации относительно соотношения геометрических размеров области не предоставляется.

Примечание 2 — Данная модель должна выбираться, поскольку размеры элементов при графическом представлении изображения не будут иметь какого-либо смысла в указанной области. Если соотношение геометрических размеров будет установлено таким образом, что изображение будет восприниматься как искаженное, то наблюдатель должен скорректировать это соотношение.

EXPRESS-описание:

```
*)
    ENTITY graphics_point;
    x_coordinate : REAL;
    y_coordinate : REAL;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут x_coordinate: Этот атрибут определяет горизонтальное положение объекта в координатной плоскости. Значение этого атрибута 0 определяет крайне левое положение объекта, а значение 1 — крайне правое положение этого объекта.

Атрибут y_coordinate: Этот атрибут определяет вертикальное положение объекта в координатной плоскости. Значение этого атрибута 0 определяет крайне верхнее положение объекта, а значение 1 — крайне нижнее положение этого объекта.

4.3.126 Прикладной компонент graphics_view

Прикладной компонент graphics_view определяет совокупность всех элементов, несущих графическую информацию и применимую к содержимому прикладного компонента general_function_definition, general_physical_definition, functional_state_context или fsm_state.

EXPRESS-описание:

```
*)
    ENTITY graphics_view;
    definition_for : definition_select;
    INVERSE
    coordinate_definition : SET[0:?] OF coordinate_translation_information FOR transformation_for;
    root_view : SET[0:1] OF multi_level_view FOR top_view;
    view_element : SET[1:?] OF visual_element FOR view;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут definition_for: Этот атрибут определяет компонент, для которого действующим является прикладной компонент graphics_view.

Атрибут coordinate_definition: Этот атрибут определяет прикладной компонент graphics_view, для которого информацию дает прикладной компонент coordinate_translation_information.

Атрибут root_view: Этот атрибут определяет прикладной компонент graphics_view, для которого прикладным компонентом multi_level_view является top_view.

Атрибут view_element: Этот атрибут определяет прикладной компонент graphics_view, для которого задан прикладной компонент visual_element.

4.3.127 Прикладной компонент hibound_integer_interval

Прикладной компонент hibound_integer_interval определяет прикладной компонент integer_interval, который имеет верхнюю границу, ограничивающую подмножество содержащихся в нем целых значений.

EXPRESS-описание:

```
*)
    ENTITY hibound_integer_interval
    SUBTYPE OF (integer_interval) ;
    high_index : INTEGER;
    END_ENTITY;
(*
```

Определение атрибута:

Атрибут high_index: Этот атрибут определяет максимальное значение, которое может принимать прикладной компонент data_instance данного типа.

4.3.128 Прикладной компонент hibound_real_interval

Прикладной компонент hibound_real_interval принадлежит к тому же типу, что и прикладной компонент real_interval и имеет верхнюю границу, ограничивающую подмножество содержащихся в нем действительных значений.

EXPRESS-описание:

```
*)
    ENTITY hibound_real_interval
    SUBTYPE OF (real_interval);
```

high_closure : BOOLEAN;
high_index : REAL;
END_ENTITY;
(*
Определения атрибутов:
Атрибут high_closure: Этот атрибут определяет, включен ли он в прикладной компонент maths_space. Состояние TRUE соответствует его включению, а состояние FALSE — невключению.
Атрибут high_index: Этот атрибут определяет максимальное значение, которое может принимать прикладной компонент data_instance данного типа.

4.3.129 Прикладной компонент implied_external_interaction
Прикладной компонент implied_external_interaction определяет взаимосвязь между прикладными компонентами requirement_instance и function_instance или physical_instance, которая предполагает наличие функционального или физического элемента в требовании.
EXPRESS-описание:
*)
ENTITY implied_external_interaction;
associated_requirement : requirement_instance;
implied_external_element : external_element_select;
INVERSE
associated_data : SET[0:1] OF data_transfer FOR transfer;
END_ENTITY;
(*
Определения атрибутов:
Атрибут associated_requirement: Этот атрибут определяет прикладной компонент requirement_instance в указанной взаимосвязи.
Атрибут implied_external_element: Этот атрибут определяет прикладной компонент configuration_element, выражаемый в требовании.
Атрибут associated_data: Этот атрибут определяет прикладной компонент implied_external_interaction, для которого действующим является прикладной компонент data_transfer.

4.3.130 Прикладной компонент infinite_cardinality
Прикладной компонент infinite_cardinality является представлением положительного бесконечно большого действительного числа.
EXPRESS-описание:
*)
ENTITY infinite_cardinality;
END_ENTITY;
(*

4.3.131 Прикладной компонент initial_state_transition_specification_assignment
Прикладной компонент initial_state_transition_specification_assignment определяет способ связи прикладного компонента textual_specification, определяющего операции, которые будут выполняться при активации прикладного компонента fsm_initial_state_transition.
EXPRESS-описание:
*)
ENTITY initial_state_transition_specification_assignment;
assigned_to : fsm_initial_state_transition;
specification : textual_specification;
END_ENTITY;
(*
Определения атрибутов:
Атрибут assigned_to: Этот атрибут определяет прикладной компонент fsm_initial_state_transition, с которым связана спецификация.
Атрибут specification: Этот атрибут определяет прикладной компонент textual_specification, присваиваемый с помощью прикладного компонента initial_state_transition_specification_assignment.

4.3.132 Прикладной компонент integer_data_type_definition

Прикладной компонент integer_data_type_definition принадлежит к тому же типу, что и прикладной компонент elementary_maths_space, и включает все целые числа.

EXPRESS-описание:

```
*)
    ENTITY integer_data_type_definition
    SUBTYPE OF (elementary_maths_space) ;
    END_ENTITY;
(*
```

4.3.133 Прикладной компонент integer_interval

Прикладной компонент integer_interval принадлежит к тому же типу, что и прикладной компонент elementary_maths_space, и ограничивает подмножество всех целых значений.

EXPRESS-описание:

```
*)
    ENTITY integer_interval
    ABSTRACT SUPERTYPE OF ( ONEOF(finite_integer_interval,hibound_integer_interval, lobound_
    integer_interval) )
    SUBTYPE OF (maths_space) ;
    END_ENTITY;
(*
```

4.3.134 Прикладной компонент io_buffer

Прикладной компонент io_buffer является модификатором поведения прикладного компонента actual_io_port во времени.

Примечание — Если прикладной компонент io_buffer присваивается прикладному компоненту actual_io_port, дающему данные при их дискретизации, то он будет считываться лишь один раз. Если прикладной компонент io_buffer присваивается прикладному компоненту actual_io_port (принимающему данные), то получаемые данные будут дискретизироваться и сохраняться только при их считывании с порта. Поведение буфера дополнительно определяется с помощью атрибута continuously_active. Для принимающих данные портов прикладной компонент io_buffer может контролировать их непрерывную активацию или активацию только через определенные интервалы. Атрибут continuously_active не изменяет семантику прикладных компонентов actual_io_port, предоставляющих данные.

EXPRESS-описание:

```
*)
    ENTITY io_buffer;
    assigned_to : actual io_port;
    continuously_active : BOOLEAN;
    synchronisation_semantics : buffer_synchronisation_enumeration;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут assigned_to: Этот атрибут определяет прикладной компонент actual_io_port, которому присваивается прикладной компонент io_buffer.

Атрибут continuously_active: Этот атрибут определяет, непрерывно ли активируется прикладной компонент io_buffer.

Атрибут synchronisation_semantics: Этот атрибут определяет условия синхронизации, устанавливаемые с помощью прикладного компонента io_buffer, и может принимать одно из следующих состояний (значений) этого атрибута:

- состояние synchronous: Функция, передающая информацию посредством прикладного компонента actual_io_port, к которому присоединяется прикладной компонент io_buffer, должна оставаться активной до тех пор, пока информация не будет принята получателем;

- состояние asynchronous: Функция, передающая информацию посредством прикладного компонента actual_io_port, к которому присоединяется прикладной компонент io_buffer, может прерываться без ограничения на то, получена информация или нет.

4.3.135 Прикладной компонент io_composition_port

Прикладной компонент io_composition_port принадлежит к тому же типу, что и прикладной компонент io_port и TBD-разделение.

Примечание 1 — Семантика прикладного компонента functional_link состоит в передаче прикладного компонента data_instance от порождающего прикладного компонента io_port к приемному прикладному компоненту io_port. Прикладной компонент data_instance должен быть одинаковым для обоих портов. По этим соображениям вводится требование активации средства для разделения прикладного компонента io_port таким образом, чтобы прикладные компоненты functional_link могли считывать или записывать подмножества данных прикладного компонента io_port. Прикладной компонент io_composition_port является средством получения прикладного компонента data_instance типа composite_data_type для прикладного компонента io_port и создания промежуточного порта, данными для которого является подмножество для объединенного порта и либо источника, либо адресата для прикладного компонента functional_link.

Примечание 2 — Благодаря функции абстрактного представления данных, включенной в ARM-модель, объединение может выполняться единовременно только на одном уровне. Для определения того, что прикладной компонент functional_link будет порождать или принимать прикладной компонент data_instance (который является частью структуры типов данных на уровне n), необходимо, чтобы прикладной компонент io_composition_port точно определял это.

Примечание 3 — Прикладной компонент io_composition_port соответствует точечному обозначению, используемому в большинстве обязательных языков программирования.

Пример — a:=a_data_type.a_data_field.

EXPRESS-описание:

```
*)
    ENTITY io_composition_port
    SUBTYPE OF (io_port) ;
    is_alias_for : data_instance;
    port_of : data_composition_select;
    DERIVE
    SELF\io_port.role : port_data_relation := port_of.role;
    UNIQUE
    UR1: port_of, io_port_number;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут is_alias_for: Этот атрибут определяет прикладной компонент data_instance, атрибут port_of которого присваивается прикладному компоненту io_composition_port. Прикладной компонент data_instance должен быть на один уровень ниже в иерархии типов абстрактных данных, чем прикладной компонент io_port, который определяется с помощью атрибута port_of.

Примечание 4 — Этот случай соответствует точечному обозначению в языках программирования. Рассмотрим следующий тип составных данных:

```
type a_type = record
a_field : another_type;
another_field : another_type;
end type;
```

Примечание 5 — Атрибут is_alias_for определяет область для типа составных данных, которые присваиваются величине данных (прикладному компоненту data_instance) с помощью атрибута is_alias_for.

Примечание 6 — Атрибут data заимствуется из прикладного компонента io_port.

Атрибут port_of: Этот атрибут определяет прикладной компонент io_port, частью которого является прикладной компонент io_composition_port.

Атрибут role:

Формальные выражения:

UR1:

4.3.136 Прикладной компонент io_port

Прикладной компонент io_port является частью интерфейса функционального объекта.

Примечание 1 — Прикладной компонент io_port является абстрактным объектом и не должен реализовываться (создавать экземпляры).

EXPRESS-описание:

```
*)
    ENTITY io_port
    ABSTRACT SUPERTYPE OF ( ONEOF(actual_io_port, control_io_port, formal_io_port, io_com-
    position_port) ) ;
    data : data_instance;
    io_port_number : INTEGER;
    port_type : port_type;
    role : port_data_relation;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут data: Этот атрибут определяет прикладной компонент data_instance, который указывает формат элемента в интерфейсе.

Атрибут io_port_number: Этот атрибут определяет прикладной компонент io_port_number, который в сочетании с атрибутами port_type и port_of должен быть уникальным для совокупности портов, связанных с данным элементом. Атрибут io_port_number является идентификатором прикладного компонента io_port.

Атрибут port_type: Этот атрибут определяет характер прикладного компонента io_port и может принимать одно из следующих состояний (значений):

- состояние input: Порт предназначен для обработки информации, исходящей от элементов вне контекста функциональных характеристик прикладного компонента io_port, и является элементом интерфейса;
- состояние output: Порт предназначен для обработки информации, исходящей от элементов в контексте функциональных характеристик прикладного компонента io_port, и является элементом интерфейса;
- состояние control: Порт предназначен для специализации входа, при которой наличие предоставляемой для данной части интерфейса информации является необходимым предварительным условием активации прикладного компонента io_port и частью интерфейса.

Примечание 2 — Для представления контрольных входов IDEF0 предусмотрено состояние control. Семантически нет разницы между атрибутами control и port_type;

- состояние mechanism: Порт предназначен для специализации входа, при которой наличие предоставляемых для данной части ресурсов является необходимым предварительным условием активации прикладного компонента io_port и частью интерфейса.

Примечание 3 — Для представления контрольных входов IDEF0 предусмотрено состояние mechanism. Семантически нет разницы между атрибутами mechanism и port_type.

Атрибут role: Этот атрибут определяет взаимодействие между прикладными компонентами io_port, служит для выдачи или получения информации и может принимать одно из следующих состояний (значений).

4.3.137 Прикладной компонент io_port_binding

Прикладной компонент io_port_binding определяет преобразование одного прикладного компонента actual_io_port в прикладной компонент formal_io_port.

Примечание 1 — В языках программирования это преобразование соответствует преобразованию между формальными и фактическими параметрами при обращении к функции.

Примечание 2 — Прикладной компонент io_port_binding является действующим лишь в случае, когда прикладной компонент actual_io_port объединяется с прикладным компонентом function_instance, поскольку его атрибут definition относится к прикладному компоненту general_function_definition, а атрибут port_of относится к прикладному компоненту formal_io_port.

EXPRESS-описание:

```
*)
    ENTITY io_port_binding;
    actual_port : actual_io_port;
```

```
        formal_port : formal_io_port;
        WHERE
        WR1: (SELF.formal_port.data :<>: SELF.actual_port.data);
        WR2: formal_port.role <> actual_port.role;
        WR3: correct_binding(SELF);
        END_ENTITY;
(*
```

Определения атрибутов:

Атрибут actual_port: Этот атрибут определяет прикладной компонент actual_io_port в указанном преобразовании.

Атрибут formal_port: Этот атрибут определяет прикладной компонент formal_io_port в указанном преобразовании.

Формальные выражения:

WR1:

WR2:

WR3:

4.3.138 Прикладной компонент io_split_join

Прикладной компонент io_split_join принадлежит к тому же типу, что и прикладной компонент general_functionality_instance, и характеризует способ разделения элементов прикладного компонента functional_link, передающего композитные данные в несколько прикладных компонентов functional_link, которые передают или объединяют первичные данные в прикладной компонент functional_link, который передает первичные данные в прикладной компонент functional_link, передающий композитные данные.

Примечание — Судьба этого компонента неясна, поскольку он может быть удален из модели, а функциональные характеристики могут быть заменены обыкновенной функцией.

EXPRESS-описание:

```
*)
        ENTITY io_split_join
        SUBTYPE OF (general_functionality_instance) ;
        END_ENTITY;
(*
```

4.3.139 Прикладной компонент issue_source_relationship

Прикладной компонент issue_source_relationship определяет способ связи элементов, которые действуют как часть постановки вопроса к прикладному компоненту critical_issue.

EXPRESS-описание:

```
*)
        ENTITY issue_source_relationship;
        description : OPTIONAL text_select;
        issue : critical_issue;
        issue_source : issue_source_select;
        END_ENTITY;
(*
```

Определения атрибутов:

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту issue_source_relationship.

Атрибут issue: Этот атрибут определяет прикладной компонент critical_issue, которому присваивается атрибут issue_source.

Атрибут issue_source: Этот атрибут определяет элемент, который поднимает проблему.

4.3.140 Прикладной компонент issue_system_assignment

Прикладной компонент issue_system_assignment определяет способ связи прикладного компонента critical_issue с прикладным компонентом system_view или system_instance, для которого проблема является значимой.

EXPRESS-описание:

```
*)
    ENTITY issue_system_assignment;
    description : OPTIONAL text_select;
    identified_system : system_select;
    issue : critical_issue;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту issue_system_assignment.

Атрибут identified_system: Этот атрибут определяет прикладной компонент system_view или system_instance в указанной связи.

Атрибут issue: Этот атрибут определяет прикладной компонент critical_issue в указанной связи.

4.3.141 Прикладной компонент justification

Прикладной компонент justification определяет текстовое описание причин существования части системы или обоснование проектирования.

EXPRESS-описание:

```
*)
    ENTITY justification;
    assigned_to : justification_assignment_select;
    justification_text : text_select;
    role : label;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут assigned_to: Этот атрибут определяет компонент, к которому применим прикладной компонент justification.

Атрибут justification_text: Этот атрибут определяет текстовый комментарий к компоненту, ссылка на который дается с помощью атрибута assigned_to.

Атрибут role: Этот атрибут определяет слово (или слова), которые используются для ссылки на прикладной компонент justification.

Пример 12 — Значением для прикладного компонента justification может быть «comment», «rationale» или «justification».

4.3.142 Прикладной компонент justification_relationship

Прикладной компонент justification_relationship определяет ряд взаимосвязей между двумя прикладными компонентами justification.

Примечание — Точная семантика этих взаимосвязей зависит от значения атрибута relationship_type.

Пример 13 — Взаимосвязь между двумя прикладными компонентами justification может возникать в той ситуации, когда выбор одного альтернативного варианта в определенной части разработки может оказывать влияние на последующий выбор вариантов и делать возможные альтернативные варианты недействительными.

EXPRESS-описание:

```
*)
    ENTITY justification_relationship;
    description : OPTIONAL text_select;
    related : justification;
    relating : justification;
    relationship_type : label;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут description: Этот атрибут определяет дополнительную текстовую информацию, относящуюся к прикладному компоненту justification_relationship.

Атрибут related: Этот атрибут определяет второй прикладной компонент justification в указанной взаимосвязи.

Атрибут relating: Этот атрибут определяет первый прикладной компонент justification в указанной взаимосвязи.

Атрибут relationship_type: Этот атрибут определяет семантику указанной взаимосвязи.

4.3.143 Прикладной компонент leaf_function_definition

Прикладной компонент leaf_function_definition принадлежит к тому же типу, что и прикладной компонент general_function_definition, который в дальнейшем не может разделяться.

Примечание 1 — Иерархия функций всегда будет ограничиваться прикладным компонентом leaf_function_definition. При более поздней проектной реализации системы прикладной компонент leaf_function может преобразоваться в прикладной компонент composite_function_definition. В этом случае будут использоваться прикладные компоненты configuration_element_version, идентифицируемые двумя вариантами одного и того же компонента.

EXPRESS-описание:

```
*)
    ENTITY leaf_function_definition
    SUBTYPE OF (general_function_definition) ;
    definition : textual_specification;
    function_type : OPTIONAL label;
    predefined : BOOLEAN;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут definition: Этот атрибут определяет текст, который описывает, какая услуга, преобразование, изменение состояния и т. п. будут выполняться с помощью данной функции, а также прикладной компонент textual_specification, дающий спецификацию и язык ее описания для прикладного компонента leaf_function_definition.

Примечание 2 — В настоящем стандарте не описывается какой-либо способ, гарантирующий непротиворечивость и корректность содержания этого определения.

Атрибут function_type: Этот атрибут определяет слово (или слова), которые используются для ссылки на прикладной компонент leaf_function_definition, и в основном смысле должен интерпретироваться как определяющий класс функций.

Примечание 3 — Допустимыми состояниями (значениями) атрибута function_type могут быть mathematical, transformational или trigonometric.

Примечание 4 — В настоящем стандарте даются определения предпочтительных состояний (значений) этого атрибута.

Атрибут predefined: Этот атрибут определяет, имеет ли предварительно определенный компонент формально задаваемую семантику в экспортируемом средстве.

Примечание 5 — Данный атрибут определяет простой способ, позволяющий средствам, обладающим большими библиотеками предварительно задаваемых функций, обмениваться информацией без преобразования слишком большого объема информации с помощью определяемых пользователем функций.

4.3.144 Прикладной компонент lobound_integer_interval

Прикладной компонент lobound_integer_interval принадлежит к тому же типу, что и прикладной компонент integer_interval, и имеет нижнюю границу, ограничивающую подмножество содержащихся в нем целых чисел.

EXPRESS-описание:

```
*)
    ENTITY lobound_integer_interval
    SUBTYPE OF (integer_interval) ;
```

```
low_index : INTEGER;
END_ENTITY;
(*
```

Определение атрибута:

Атрибут low_index: Этот атрибут определяет минимальное значение, которое может принимать прикладной компонент data_instance данного типа.

4.3.145 Прикладной компонент lobound_real_interval

Прикладной компонент lobound_real_interval принадлежит к тому же типу, что и прикладной компонент real_interval, и имеет нижнюю границу, ограничивающую подмножество содержащихся в нем действительных чисел.

EXPRESS-описание:

```
*)
ENTITY lobound_real_interval
SUBTYPE OF (real_interval) ;
low_closure : BOOLEAN;
low_index : REAL;
END_ENTITY;
(*
```

Определения атрибутов:

Атрибут low_closure: Этот атрибут определяет, включается ли атрибут low_index в прикладной компонент maths_space. Состояние TRUE соответствует его включению, а состояние FALSE — невключению этого атрибута.

Атрибут low_index: Этот атрибут определяет минимальное значение, которое может принимать прикладной компонент data_instance данного типа.

4.3.146 Прикладной компонент logical_data_type_definition

Прикладной компонент logical_data_type_definition принадлежит к тому же типу, что и прикладной компонент elementary_maths_space, чьими состояниями являются TRUE, FALSE или UNKNOWN.

Примечание — Прикладной компонент logical_data_type_definition не моделируется как прикладной компонент finite_space, так и понятие прикладного компонента logical_data_type_definition, которые могут иметь более точный смысл, чем это будет допускаться прикладным компонентом finite_space.

EXPRESS-описание:

```
*)
ENTITY logical_data_type_definition
SUBTYPE OF (elementary_maths_space) ;
END_ENTITY;
(*
```

4.3.147 Прикладной компонент maths_space

Прикладной компонент maths_space определяет множество математических значений, которое может принимать прикладной компонент data_instance в данной области значений.

Примечание — Для целей контроля типов данных должны быть предусмотрены строки, состоящие из математических значений.

EXPRESS-описание:

```
*)
ENTITY maths_space
ABSTRACT SUPERTYPE OF ( ONEOF(elementary_ maths_space, finite_space, integer_interval,
real_interval) ) ;
description : OPTIONAL text_select;
id : element_identifier;
name : OPTIONAL label;
UNIQUE
UR1: id;
END_ENTITY;
(*
```

Определения атрибутов:

Атрибут description: Этот атрибут определяет дополнительную текстовую информацию, относящуюся к прикладному компоненту maths_space.

Атрибут id: Этот атрибут определяет идентификатор прикладного компонента maths_space.

Атрибут name: Этот атрибут определяет слово (или слова), которые используются для ссылки на прикладной компонент maths_space.

Формальные выражения:

UR1:

4.3.148 Прикладной компонент model_defined_requirement_definition

Прикладной компонент model_defined_requirement_definition принадлежит к тому же типу, что и прикладной компонент requirement_definition, у которого требуемые функциональные возможности устанавливаются с использованием определенного вида модели, выражаемого в формате настоящего стандарта.

Примечание — Модель может выражаться в любом формате, поддерживаемом спецификацией PAS 20542.

EXPRESS-описание:

```
*)
    ENTITY model_defined_requirement_definition
    SUBTYPE OF (requirement_definition) ;
    assigned_model : system_view;
    model_relevance : OPTIONAL text_select;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут assigned_model: Этот атрибут определяет модель, которая задает требование.

Атрибут model_relevance: Этот атрибут определяет в текстовом виде способ интерпретации требования, выражаемого с помощью модели, или то, какая часть атрибута assigned_model выражает это требование.

4.3.149 Прикладной компонент multi_level_view

Прикладной компонент multi_level_view определяет множество прикладных компонентов graphics_view, которые дают информационно-ознакомительное описание нескольких уровней разделения в рамках единого представления.

Примечание 1 — Прикладной компонент multi_level_view не должен реализовываться при отсутствии для частного прикладного компонента graphics_view иерархии прикладного компонента graphics_view.

Примечание 2 — Прикладной компонент multi_level_view предусмотрен для представления нескольких уровней разделения проектной схемы.

EXPRESS-описание:

```
*)
    ENTITY multi_level_view;
    description : OPTIONAL text_select;
    reference_name : OPTIONAL label;
    top_view : graphics_view;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут description: Этот атрибут определяет дополнительную текстовую информацию, относящуюся к прикладному компоненту multi_level_view.

Атрибут reference_name: Этот атрибут определяет слово (или слова), которые используются для ссылки на прикладной компонент multi_level_view.

Атрибут top_view: Этот атрибут определяет прикладной компонент graphics_view, для которого прикладным компонентом multi_level_view является элемент top_view.

4.3.150 Прикладной компонент name_binding

Прикладной компонент name_binding определяет преобразование, связывающее прикладной компонент function_reference в области имен (в части иерархической классификации) для прикладных компонентов composite_function_definition и state_function_interaction_port.

Примечание — Прикладной компонент name_binding преобразует ссылку на прикладной компонент function_instance, используемый в прикладном компоненте functional_state_context, в действующий прикладной компонент function_instance, для чего вводятся два следующих объекта порта:

1) прикладной компонент function_reference: Обеспечивает ссылку на прикладной компонент function_instance;

2) прикладной компонент state_function_interaction_port: Определяет локальную ссылку в прикладном компоненте functional_state_context.

EXPRESS-описание:

```
*)
    ENTITY name_binding;
    actual_port : function_reference;
    formal_port : state_function_interaction_port;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут actual_port: Этот атрибут определяет прикладной компонент function_reference в указанной связи.

Атрибут formal_port: Этот атрибут определяет прикладной компонент state_function_interaction_port в указанной связи.

4.3.151 Прикладной компонент nominal_value

Прикладной компонент nominal_value принадлежит к тому же типу, что и прикладной компонент value_with_unit, и количественно выражается численным значением вместе с единицей измерения.

EXPRESS-описание:

```
*)
    ENTITY nominal_value
    SUBTYPE OF (value_with_unite) ;
    value_component : NUMBER;
    INVERSE
    limitation : SET[0:1] OF plus_minus_bounds FOR limited_value;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут value_component: Этот атрибут определяет значение nominal.

Примечание — Это определение заимствовано из протокола AP-214 ARM-модели.

Атрибут limitation: Этот атрибут определяет прикладной компонент nominal_value, дополнительно определяемый прикладным компонентом plus_minus_bound.

4.3.152 Прикладной компонент non_digital_document

Прикладной компонент non_digital_document принадлежит к тому же типу, что и прикладной компонент document_reference, и обеспечивает ссылку на документ, представляемый не в цифровой форме.

EXPRESS-описание:

```
*)
    ENTITY non_ digital_document
    SUBTYPE OF (documentation_reference) ;
    location : label;
    END_ENTITY;
(*
```

Определение атрибута:

Атрибут location: Этот атрибут определяет физическое положение, в котором может находиться прикладной компонент non_digital_document.

4.3.153 Прикладной компонент oo_action

Прикладной компонент oo_action является выполняемым оператором, который дает абстрактное представление о процедуре вычислений.

EXPRESS-описание:

```
*)
    ENTITY oo_action
    SUPERTYPE OF ( ONEOF(oo_call_action, oo_create_action, oo_send_action) ) ;
    description : OPTIONAL text_select;
    is_asynchronous : BOOLEAN;
    name : label;
    script : textual_specification;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту oo_action.

Атрибут is_asynchronous: Этот атрибут определяет, будет ли прикладной компонент oo_action выполняться синхронно или асинхронно.

Атрибут name: Этот атрибут определяет слово (или слова), которые используются для ссылки на прикладной компонент oo_action.

Атрибут script: Этот атрибут определяет прикладной компонент textual_specification, который дает спецификацию и ее язык описания для прикладного компонента oo_action.

4.3.154 Прикладной компонент oo_action_state

Прикладной компонент oo_action_state принадлежит к тому же типу, что и прикладной компонент cb_place, и характеризует выполнение элементарного действия, обычно связанного с инициализацией прикладного компонента oo_operation.

EXPRESS-описание:

```
*)
    ENTITY oo_action_staten
    SUBTYPE OF (cb_place) ;
    END_ENTITY;
(*
```

4.3.155 Прикладной компонент oo_action_staten_transition

Прикладной компонент oo_action_staten_transition принадлежит к тому же типу, что и прикладной компонент cb_transition, и определяет временную взаимосвязь между двумя прикладными компонентами oo_action_state.

EXPRESS-описание:

```
*)
    ENTITY oo_action_staten_transition
    SUBTYPE OF (cb_transition) ;
    END_ENTITY;
(*
```

4.3.156 Прикладной компонент oo_action_temporal_relationship

Прикладной компонент oo_action_temporal_relationship определяет взаимосвязь между двумя прикладными компонентами oo_action и указывает временной порядок их выполнения.

EXPRESS-описание:

```
*)
    ENTITY oo_action_temporal_relationshipn;
    description : OPTIONAL text_select;
    predecessor : oo_action;
    successor : oo_action;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту oo_action_temporal_relationship.

Атрибут predecessor: Этот атрибут определяет более ранний прикладной компонент oo_action.

Атрибут successor: Этот атрибут определяет более поздний прикладной компонент oo_action.

4.3.157 Прикладной компонент oo_actor

Прикладной компонент oo_actor определяет взаимосвязанное множество ролей, которые пользователь прикладного компонента oo_use_case может играть при взаимодействии с ним.

EXPRESS-описание:

```
*)
    ENTITY oo_actor;
    description : OPTIONAL text_select
    id : element_identifier;
    name : label;
    namespace : OPTIONAL oo_namespace_select;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту oo_actor.

Атрибут id: Этот атрибут определяет идентификатор прикладного компонента oo_actor.

Атрибут name: Этот атрибут определяет слово (или слова), которые используются для ссылки на прикладной компонент oo_actor.

Атрибут namespace: Этот атрибут определяет элемент, которому принадлежит прикладной компонент oo_actor. Имена элементов, принадлежащие одной и той же области имен, должны быть уникальными в данной области.

4.3.158 Прикладной компонент oo_argument

Прикладной компонент oo_argument определяется атрибутом initial_value, соответствующим прикладному компоненту oo_parameter.

EXPRESS-описание:

```
*)
    ENTITY oo_argument;
    action : oo_action;
    initial_value : text_select;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут action: Этот атрибут определяет прикладной компонент oo_action, которому принадлежит этот атрибут.

Атрибут initial_value: Этот атрибут определяет свое действующее значение.

4.3.159 Прикладной компонент oo_association

Прикладной компонент oo_association принадлежит к тому же типу, что и прикладной компонент oo_generic_association, и обеспечивает семантическую взаимосвязь между элементами системы.

EXPRESS-описание:

```
*)
    ENTITY oo_association
    SUBTYPE OF (oo_generic_association) ;
    WHERE
    WR1: SELF\oo_generic_association.reading_direction IN SELF\oo_generic_association.connection;
    END_ENTITY;
(*
```

Формальные выражения:

WR1:

4.3.160 Прикладной компонент oo_association_class

Прикладной компонент oo_association_class определяет объект, который обладает свойствами прикладных компонентов oo_class_association и oo_class и представляет собой множество прикладных компонентов oo_object, которые одновременно могут реализовываться вместе со связанным с ним представителем (экземпляром).

EXPRESS-описание:

```
*)
    ENTITY oo_association_class;
    class : oo_class;
    class_association : oo_generic_association;
    description : OPTIONAL text_select;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут class: Этот атрибут определяет связанный прикладной компонент oo_object.

Атрибут class_association: Этот атрибут определяет связанный прикладной компонент oo_association.

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту oo_association_class.

4.3.161 Прикладной компонент oo_association_end

Прикладной компонент oo_association_end принадлежит к тому же типу, что и прикладной компонент oo_generic_association_end, и указывает конечную точку для прикладного компонента oo_association.

EXPRESS-описание:

```
*)
    ENTITY oo_association_end
    SUBTYPE OF (oo_generic_association_end) ;
    SELF\oo_generic_association_end. RENAMED association : oo_association;
    END_ENTITY;
(*
```

Определение атрибута:

Атрибут association:

4.3.162 Прикладной компонент oo_association_end_classifier_relationship

Прикладной компонент oo_association_end_classifier_relationship определяет взаимосвязь между прикладным компонентом oo_association_end и классификатором, определяющим прикладной компонент oo_association_end_classifier_relationship.

EXPRESS-описание:

```
*)
    ENTITY oo_association_end_classifier_relationship;
    association_end : oo_generic_association_end;
    description : OPTIONAL text_select;
    specification : oo_extended_classifier_select;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут association_end: Этот атрибут определяет классифицированный прикладной компонент oo_association_end.

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту oo_association_end_classifier_relationship.

Атрибут specification: Этот атрибут определяет классификатор.

4.3.163 Прикладной компонент oo_association_end_qualifier_association

Прикладной компонент oo_association_end_qualifier_association определяет взаимосвязь между прикладными компонентами oo_association_end и oo_attribute.

EXPRESS-описание:

```
*)
    ENTITY oo_association_end_qualifier_association;
    association_end : oo_generic_association_end;
    qualifier : OPTIONAL oo_attributen;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут association_end: Этот атрибут определяет классифицированный прикладной компонент oo_association_end.

Атрибут qualifier: Этот атрибут определяет классификационный прикладной компонент oo_attribute object.

4.3.164 Прикладной компонент oo_association_end_role

Прикладной компонент oo_association_end_role принадлежит к тому же типу, что и прикладной компонент oo_generic_association_end, и определяет конечную точку для прикладного компонента oo_association_role.

EXPRESS-описание:

```
*)
    ENTITY oo_association_end_role
    SUBTYPE OF (oo_generic_association_end) ;
    base : OPTIONAL oo_association_end;
    collaboration_multiplicity : cardinality_association_select;
    role_type : oo_classifier_rolen;
    SELF\oo_generic_association_end. RENAMED association: oo_association_role;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут base: Этот атрибут определяет прикладной компонент oo_generic_association_end, на котором базируется прикладной компонент oo_association_end_role.

Атрибут collaboration_multiplicity: Этот атрибут определяет число адресных реализаций, которые могут связываться с единственной исходной реализацией в заданном прикладном компоненте oo_association_role с помощью прикладного компонента oo_association_end_role.

Атрибут role_type: Этот атрибут определяет семантику прикладного компонента oo_association_end_role.

Атрибут association:

4.3.165 Прикладной компонент oo_association_role

Прикладной компонент oo_association_role принадлежит к тому же типу, что и прикладной компонент oo_generic_association, и определяет специфическое поведение прикладного компонента oo_association в конкретном контексте.

EXPRESS-описание:

```
*)
    ENTITY oo_association_role
    SUBTYPE OF (oo_generic_association) ;
    base : OPTIONAL oo_association;
    multiplicity : cardinality_association_select;
    WHERE
    WR1: SELF\oo_generic_association.reading_direction IN SELF\oo_generic_association.connection;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут base: Этот атрибут определяет прикладной компонент oo_association, на котором основывается прикладной компонент oo_association_role.

Атрибут multiplicity: Этот атрибут определяет число адресных реализаций, которые могут связываться с единственной исходной реализацией в заданном прикладном компоненте oo_association.

Формальные выражения:

WR1:

4.3.166 Прикладной компонент oo_attribute

Прикладной компонент oo_attribute является поименованной частью классификатора, которая определяет диапазон значений, которые могут принимать экземпляры этого классификатора.

EXPRESS-описание:

```
*)
    ENTITY oo_attributen;
    definition : oo_extended_classifier_select;
    description : OPTIONAL text_select;
    id : element_identifier;
    name : label;
    owner : oo_extended_classifier_select;
    visibility : label;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут definition: Этот атрибут определяет тип определения прикладного компонента oo_attribute.

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту oo_attribute.

Атрибут id: Этот атрибут определяет идентификатор прикладного компонента oo_attribute.

Атрибут name: Этот атрибут определяет слово (или слова), которые используются для ссылки на прикладной компонент oo_attribute.

Атрибут owner: Этот атрибут определяет классификатор, к которому принадлежит прикладной компонент oo_attribute.

Атрибут visibility: Этот атрибут определяет способ, с помощью которого прикладной компонент oo_attribute может быть «видимым» вне его замкнутой области имен. Там, где это применимо, должны использоваться следующие состояния (значения) этого атрибута:

- состояние private: Определяет, что прикладной компонент oo_attribute будет «видим» только для элементов в той же области имен;
- состояние protected: Определяет, что прикладной компонент oo_attribute будет «видим» только для элементов в той же области имен, а также в наследуемых им областях имен;
- состояние public: Определяет, что доступ к прикладному компоненту oo_attribute может осуществляться из других элементов, включая и те, которые находятся вне области имен прикладного компонента oo_attribute.

4.3.167 Прикладной компонент oo_attribute_instance

Прикладной компонент oo_attribute_instance является представителем (экземпляром) прикладного компонента oo_attribute.

EXPRESS-описание:

```
*)
    ENTITY oo_attributen_instance;
    attribute_value : label;
    definition : oo_attributen;
    owner : oo_extended_classifier_select;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут attribute_value: Этот атрибут определяет фактическое значение прикладного компонента oo_attribute_instance.

Атрибут definition: Этот атрибут определяет прикладной компонент oo_attribute, который дает определение прикладному компоненту oo_attribute_instance.

Атрибут owner: Этот атрибут определяет классификатор, к которому принадлежит прикладной компонент oo_attribute_instance.

4.3.168 Прикладной компонент oo_attribute_link_end_association

Прикладной компонент oo_attribute_link_end_association позволяет соотносить прикладные компоненты oo_attribute_instance с прикладными компонентами oo_attribute_link_end.

EXPRESS-описание:

```
*)
    ENTITY oo_attributen_link_end_associatione;
    attribute_instance : oo_attributen_instance;
    description : text_select;
    link_end : oo_link_end;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут attribute_instance: Этот атрибут определяет прикладной компонент oo_attribute_instance, который присоединяется к прикладному компоненту oo_link_end и задается с помощью атрибута link_end.

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту oo_attribute_link_end_association.

Атрибут link_end: Этот атрибут определяет прикладной компонент oo_link_end, к которому присоединяется атрибут attribute_instance.

4.3.169 Прикладной компонент oo_behavioural_feature

Прикладной компонент oo_behavioural_feature определяет динамическое свойство классификатора.

EXPRESS-описание:

```
*)
    ENTITY oo_behavioural_featuree
    ABSTRACT SUPERTYPE OF ( ONEOF(oo_method, oo_operation, name : label;
    owner : oo_classifier_select;
    visibility : label;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут name: Этот атрибут определяет слово (или слова), которые используются для ссылки на прикладной компонент oo_behavioural_feature.

Атрибут owner: Этот атрибут определяет классификатор, к которому принадлежит прикладной компонент oo_behavioural_feature.

Атрибут visibility: Этот атрибут определяет способ, с помощью которого прикладной компонент oo_behavioural_feature может быть «видимым» вне его замкнутой области имен. Там, где это применимо, должны использоваться следующие состояния (значения) этого атрибута:

- состояние private: Определяет, что прикладной компонент oo_behavioural_feature будет «видим» только для элементов в той же области имен;
- состояние protected: Определяет, что прикладной компонент oo_behavioural_feature будет «видимым» только для элементов в той же области имен, а также в наследуемых им областях имен;
- состояние public: Определяет, что доступ к прикладному компоненту oo_behavioural_feature может осуществляться из других элементов, включая и те, которые находятся вне области имен прикладного компонента oo_behavioural_feature.

4.3.170 Прикладной компонент oo_call_action

Прикладной компонент oo_call_action принадлежит к тому же типу, что и прикладной компонент oo_action, и обеспечивает запрос операции, определяемой как прикладной компонент oo_operation object.

EXPRESS-описание:

```
*)
    ENTITY oo_call_action
    SUBTYPE OF (oo_action) ;
    operation : oo_operation;
    END_ENTITY;
(*
```

Определение атрибута:

Атрибут operation: Этот атрибут определяет прикладной компонент oo_operation, который будет запрашиваться.

4.3.171 Прикладной компонент oo_class

Прикладной компонент oo_class определяет основное объектно-ориентированное обозначение, представляющее собой множество прикладных компонентов oo_object, имеющих одну и ту же совокупность общих свойств, что и у прикладных компонентов oo_attribute, oo_operation, oo_method и relationships, в форме прикладного компонента oo_association, oo_generalization, oo_dependency, oo_extension или oo_inclusion.

EXPRESS-описание:

```
*)
    ENTITY oo_class;
    associated_version : configuration_element_version;
    description : OPTIONAL text_select;
    id : element_identifier;
    is_active : BOOLEAN;
    name : label;
    namespace : OPTIONAL oo_namespace_select;
    visibility : label;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут associated_version: Этот атрибут определяет прикладной компонент configuration_element_version для прикладного компонента oo_class.

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту oo_class.

Атрибут id: Этот атрибут определяет идентификатор прикладного компонента oo_class.

Атрибут is_active: Этот атрибут определяет, принадлежит ли представитель прикладного компонента oo_class (в виде прикладного компонента oo_object) к потоку контрольных данных. Если этот атрибут находится в состоянии TRUE, то представитель прикладного компонента oo_object с атрибутом is_active будет принадлежать этому потоку и может выполняться одновременно с другими активированными представителями прикладного компонента oo_class. Если этот атрибут находится в состоянии FALSE, то запросы прикладного компонента oo_operation будут выполняться под контролем запрашивающего активного прикладного компонента oo_object.

Атрибут name: Этот атрибут определяет слово (или слова), которые используются для ссылки на прикладной компонент oo_class.

Атрибут namespace: Этот атрибут определяет элемент, к которому принадлежит прикладной компонент oo_class. Имена элементов, принадлежащих к одной и той же области имен, должны быть уникальными в совокупности этих элементов.

Атрибут visibility: Этот атрибут определяет способ, с помощью которого прикладной компонент oo_class может быть «видимым» вне его замкнутой области имен. Там, где это применимо, должны использоваться следующие состояния (значения) этого атрибута:

- состояние private: Определяет, что прикладной компонент oo_class oo_class будет «видим» только для элементов в той же области имен;
- состояние protected: Определяет, что прикладной компонент oo_class будет «видим» только для элементов в той же области имен, а также в наследуемых им областях имен;
- состояние public: Определяет, что доступ к прикладному компоненту oo_class может осуществляться из других элементов, включая и те, которые находятся вне области имен прикладного компонента oo_class.

4.3.172 Прикладной компонент oo_classifier_role

Прикладной компонент oo_classifier_role является специфическим представлением классификатора конкретного контекста.

EXPRESS-описание:

```
*)
    ENTITY oo_classifier_rolen;
    multiplicity : cardinality_association_select;
    INVERSE
    association_end_role : SET[0:?] OF oo_association_end_role FOR role_type;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут multiplicity: Этот атрибут определяет число прикладных компонентов oo_classifier_role, которые могут быть связаны с одиночным отправителем ссылочного прикладного компонента oo_message.

Атрибут association_end_role: Этот атрибут определяет семантику прикладного компонента oo_association_end_role.

4.3.173 Прикладной компонент oo_collaboration

Прикладной компонент oo_collaboration определяет способ, с помощью которого различные элементы должны использоваться при выполнении конкретного задания.

EXPRESS-описание:

```
*)
    ENTITY oo_collaboration;
    description : OPTIONAL text_select;
    id : element_identifier;
    name : label;
    representing : oo_classifier_or_operation_select;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту oo_collaboration.

Атрибут id: Этот атрибут определяет идентификатор прикладного компонента oo_collaboration.

Атрибут name: Этот атрибут определяет слово (или слова), которые используются для ссылки на прикладной компонент oo_collaboration.

Атрибут representing: Этот атрибут определяет элемент, чье представление описывается с помощью прикладного компонента oo_collaboration.

4.3.174 Прикладной компонент oo_component

Прикладной компонент oo_component представляет подлежащую распределению часть реализуемой системы.

EXPRESS-описание:

```
*)
    ENTITY oo_component;
    description : OPTIONAL text_select;
    id : element_identifier;
    name : label;
    namespace : OPTIONAL oo_namespace_select;
    visibility : label;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту oo_component.

Атрибут id: Этот атрибут определяет идентификатор прикладного компонента oo_component.

Атрибут name: Этот атрибут определяет слово (или слова), которые используются для ссылки на прикладной компонент oo_component.

Атрибут namespace: Этот атрибут определяет элемент, к которому принадлежит прикладной компонент oo_component. Имена элементов, принадлежащих к одной и той же области имен, должны быть уникальными в совокупности этих элементов.

Атрибут visibility: Этот атрибут определяет способ, с помощью которого прикладной компонент oo_component может быть «видимым» вне ограниченной области имен. Там, где это применимо, должны использоваться следующие состояния (значения) этого атрибута:

- состояние private: Определяет, что прикладной компонент oo_component будет «видим» для элементов в той же области имен;
- состояние protected: Определяет, что прикладной компонент oo_component будет «видим» для элементов в той же области имен, а также в наследуемой для этих элементов области имен;

- состояние public: Определяет, что прикладной компонент oo_component может быть доступен из других элементов, включая и те, что находятся вне области имен прикладного компонента oo_component.

4.3.175 Прикладной компонент oo_component_allocation

Прикладной компонент oo_component_allocation определяет размещение прикладного компонента oo_component в прикладном компоненте physical_instance_reference.

EXPRESS-описание:

```
*)
    ENTITY oo_component_allocation;
    deployment_location : physical_instance_reference;
    description : OPTIONAL text_select;
    resident : oo_component;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут deployment_location: Этот атрибут определяет прикладной компонент physical_instance_reference, в котором прикладной компонент oo_component определяется постоянно проживающими лицами.

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту oo_component_allocation.

Атрибут resident: Этот атрибут определяет размещенный прикладной компонент oo_component.

4.3.176 Прикладной компонент oo_constraint

Прикладной компонент oo_constraint определяет семантическое условие или ограничение.

EXPRESS-описание:

```
*)
    ENTITY oo_constraint;
    body : textual_specification;
    END_ENTITY;
(*
```

Определение атрибута:

Атрибут body: Этот атрибут определяет прикладной компонент textual_specification, который задает спецификацию и язык ее описания прикладного компонента oo_constraint.

4.3.177 Прикладной компонент oo_constraint_model_element_relationship

Прикладной компонент oo_constraint_model_element_relationship определяет размещение прикладного компонента oo_constraint в элементе.

EXPRESS-описание:

```
*)
    ENTITY oo_constraint_model_element_relationship;
    constraint : oo_constraint;
    model_element : oo_model_element_select;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут constraint: Этот атрибут определяет ограничение прикладного компонента oo_constraint.

Атрибут model_element: Этот атрибут определяет ограниченный элемент.

4.3.178 Прикладной компонент oo_create_action

Прикладной компонент oo_create_action принадлежит к тому же типу, что и прикладной компонент oo_action, и определяет формирование классификатора.

EXPRESS-описание:

```
*)
    ENTITY oo_create_action
    SUBTYPE OF (oo_action);
```

```
    instantiation : oo_classifier_select;
    END_ENTITY;
(*
```

Определение атрибута:

Атрибут instantiation: Этот атрибут определяет классификатор, который был сформирован с помощью прикладного компонента oo_create_action.

4.3.179 Прикладной компонент oo_dependency

Прикладной компонент oo_dependency определяет взаимосвязь между двумя прикладными компонентами, в которых изменение элемента поставщика будет оказывать влияние на элемент заказчика.

EXPRESS-описание:

```
*)
    ENTITY oo_dependency;
    client : oo_model_element_select;
    description : text_select;
    supplier : oo_model_element_select;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут client: Этот атрибут определяет зависимый элемент прикладного компонента oo_dependency.

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту oo_dependency.

Атрибут supplier: Этот атрибут определяет независимый элемент прикладного компонента oo_dependency.

4.3.180 Прикладной компонент oo_element_import

Прикладной компонент oo_element_import позволяет использовать прикладной компонент model_element в прикладном компоненте oo_package.

EXPRESS-описание:

```
*)
    ENTITY oo_element_import;
    alias_name : OPTIONAL label;
    container : oo_package;
    model_element : oo_model_element_select;
    name : label;
    visibility : label;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут alias_name: Этот атрибут определяет альтернативное имя для импортированного прикладного компонента model_element, которое может использоваться для ссылки на прикладной компонент model_element.

Атрибут container: Этот атрибут определяет прикладной компонент oo_package, который импортирует прикладной компонент model_element через хранилище.

Атрибут model_element: Этот атрибут определяет импортируемый элемент.

Атрибут name: Этот атрибут определяет слово (или слова), которые используются для ссылки на прикладной компонент oo_element_import.

Атрибут visibility: Этот атрибут определяет способ, с помощью которого прикладной компонент oo_element_import может быть «видимым» вне ограниченной области имен. Там, где это применимо, должны использоваться следующие состояния (значения) этого атрибута:

- состояние private: Определяет, что прикладной компонент oo_element_import будет «видим» для элементов в той же области имен;
- состояние protected: Определяет, что прикладной компонент oo_element_import будет «видим» для элементов в той же области имен, а также в наследуемой им областях имен;
- состояние public: Определяет, что прикладной компонент oo_element_import может быть доступен из других элементов, включая и те, что находятся вне области имен прикладного компонента oo_element_import.

4.3.181 Прикладной компонент oo_element_residence

Прикладной компонент oo_element_residence идентифицирует прикладной компонент oo_component, в котором хранится реализуемый элемент.

EXPRESS-описание:

```
*)
    ENTITY oo_element_residencen;
    description : OPTIONAL text_select;
    implementation_location : oo_component;
    resident : oo_model_element_select;
    visibility : label;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту oo_element_residence.

Атрибут implementation_location: Этот атрибут определяет прикладной компонент oo_component, в котором хранится элемент.

Атрибут resident: Этот атрибут определяет элемент, хранящийся в прикладном компоненте oo_component и определенный с помощью атрибута implementation_location.

Атрибут visibility: Этот атрибут определяет способ, с помощью которого прикладной компонент oo_element_residence может быть «видимым» вне ограниченной области имен. Там, где это применимо, должны использоваться следующие состояния (значения) этого атрибута:

- состояние private: Определяет, что прикладной компонент oo_element_residence будет «видим» для элементов в той же области имен;
- состояние protected: Определяет, что прикладной компонент oo_element_residence будет «видим» только для элементов в той же области имен, а также в наследуемых им областях имен;
- состояние public: Определяет, что прикладной компонент oo_element_residence может быть доступен из других элементов, включая и те, что находятся вне области имен прикладного компонента oo_element_residence.

4.3.182 Прикладной компонент oo_extension

Прикладной компонент oo_extension определяет связь расширения прикладного компонента oo_use_case с тем же основным прикладным компонентом.

EXPRESS-описание:

```
*)
    ENTITY oo_extension;
    base : oo_use_case;
    condition : text_select;
    extension : oo_use_case;
    extension_point : oo_extension_point;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут base: Этот атрибут определяет исходный прикладной компонент oo_use_case в прикладном компоненте oo_extension.

Атрибут condition: Этот атрибут определяет прикладной компонент textual_specification, определяющий условие, которое должно выполняться для существования расширения.

Атрибут extension: Этот атрибут определяет расширенный прикладной компонент oo_use_case в прикладном компоненте oo_extension.

Атрибут extension_point: Этот атрибут определяет прикладной компонент oo_extension_point, определяющий место ввода информации.

4.3.183 Прикладной компонент oo_extension_point

Прикладной компонент oo_extension_point определяет положение, в котором прикладной компонент oo_use_case может расширяться посредством прикладного компонента oo_extension.

EXPRESS-описание:

```
*)
        ENTITY oo_extension_point;
        location : text_select;
        use_case : oo_use_case;
        END_ENTITY;
(*
```

Определения атрибутов:

Атрибут location: Этот атрибут определяет прикладной компонент textual_specification, указывающий положение прикладного компонента oo_extension_point в тексте.

Атрибут use_case: Этот атрибут определяет прикладной компонент oo_use_case, для которого прикладной компонент oo_extension_point является точкой ввода расширения.

4.3.184 Прикладной компонент oo_generalization

Прикладной компонент oo_generalization определяет таксономическую связь между более общим и более специфическим элементом.

EXPRESS-описание:

```
*)
        ENTITY oo_generalization;
        child : oo_generalizable_element_select;
        discriminator : label;
        parent : oo_generalizable_element_select;
        END_ENTITY;
(*
```

Определения атрибутов:

Атрибут child: Этот атрибут определяет более специфический элемент. При этом дочерний элемент является полностью совместимым с родительским элементом и способен содержать дополнительную информацию.

Атрибут discriminator: Этот атрибут определяет имя сегмента для всех дочерних элементов, имеющих один и тот же родительский элемент.

Атрибут parent: Этот атрибут определяет более общий элемент.

4.3.185 Прикладной компонент oo_generic_association

Прикладной компонент oo_generic_association определяет либо прикладной компонент oo_association, либо прикладной компонент oo_association_role, а также определяет семантическую связь между элементами и объектно-ориентированной системной спецификацией.

EXPRESS-описание:

```
*)
        ENTITY oo_generic_association
        ABSTRACT SUPERTYPE OF ( ONEOF(oo_association, oo_association_role) ) ;
        description : OPTIONAL text_select;
        id : element_identifier;
        name : label;
        reading_direction : oo_generic_association_end;
        visibility : label;
        INVERSE
        connection : SET[2:?] OF oo_generic_association_end FOR association;
        END_ENTITY;
(*
```

Определения атрибутов:

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту oo_generic_association.

Атрибут id: Этот атрибут определяет идентификатор прикладного компонента oo_generic_association.

Атрибут name: Этот атрибут определяет слово (или слова), которые используются для ссылки на прикладной компонент oo_generic_association.

Атрибут reading_direction: Этот атрибут определяет прикладной компонент oo_generic_association_end, на который указывает имя прикладного компонента oo_generic_association.

Пример 14 — Атрибут reading_direction для прикладного компонента oo_class называется adult, а другой прикладной компонент oo_class, именуемый child, называется parent. Атрибут reading_direction определяет child как направляемый компонент; например, связь будет читаться как «Adult is Parent of Child».

Атрибут visibility: Этот атрибут определяет способ, с помощью которого прикладной компонент oo_generic_association будет «видим» вне ограниченной области имен. Там, где это применимо, должны использоваться следующие состояния (значения) этого атрибута:

- состояние private: Определяет, что прикладной компонент oo_generic_association будет «видим» для элементов в той же области имен;
- состояние protected: Определяет, что прикладной компонент oo_generic_association будет «видим» только для элементов в той же области имен, а также в наследуемых им областях имен.

Атрибут connection: Этот атрибут определяет прикладной компонент oo_association, для которого прикладной компонент oo_generic_association_end является конечным.

4.3.186 Прикладной компонент oo_generic_association_end

Прикладной компонент oo_generic_association_end определяет конечную точку для прикладного компонента oo_association.

EXPRESS-описание:

```
*)
    ENTITY oo_generic_association_end
    ABSTRACT SUPERTYPE OF ( ONEOF(oo_association_end, oo_association_end_role) ) ;
    aggregation : label;
    association : oo_generic_association;
    description : OPTIONAL text_select;
    id : element_identifier;
    is_navigable : BOOLEAN;
    multiplicity : cardinality_association_select;
    name : label;
    visibility : label;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут aggregation: Этот атрибут определяет отношение целого к части для связанных элементов.

Атрибут association: Этот атрибут определяет прикладной компонент oo_association, для которого прикладной компонент oo_generic_association_end является конечным.

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту oo_generic_association_end.

Атрибут id: Этот атрибут определяет идентификатор прикладного компонента oo_generic_association_end.

Атрибут is_navigable: Этот атрибут определяет, возможно ли перемещение от представителя источника к связанному с ним получателю.

Атрибут multiplicity: Этот атрибут определяет число представителей получателя, которое может связываться с единственным представителем источника посредством прикладного компонента oo_generic_association, определенного с помощью связанного с ним атрибута.

Атрибут name: Этот атрибут определяет слово (или слова), которые используются для ссылки на прикладной компонент oo_generic_association_end.

Атрибут visibility: Этот атрибут определяет способ, с помощью которого прикладной компонент oo_generic_association_end может быть «видимым» вне ограниченной области имен. Там, где это применимо, должны использоваться следующие состояния (значения):

- состояние private: Определяет, что прикладной компонент oo_generic_association_end будет «видим» только для элементов в той же области имен;
- состояние protected: Определяет, что прикладной компонент oo_generic_association_end будет «видим» только для элементов в той же области имен, а также в наследуемых им областях имен;

- состояние public: Определяет, что прикладной компонент oo_generic_association_end может быть доступен из других элементов, включая и те, что находятся вне области имен прикладного компонента oo_generic_association_end.

4.3.187 Прикладной компонент oo_inclusion

Прикладной компонент oo_inclusion указывает, что представитель прикладного компонента oo_use_case также будет содержать представление, определенное с помощью другого прикладного компонента oo_use_case.

EXPRESS-описание:

```
*)
    ENTITY oo_inclusion;
    addition : oo_use_case;
    base : oo_use_case;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут addition: Этот атрибут определяет включенный прикладной компонент oo_use_case.

Атрибут base: Этот атрибут определяет включаемый прикладной компонент oo_use_case object.

4.3.188 Прикладной компонент oo_instance_classifier_relationship

Прикладной компонент oo_instance_classifier_relationship определяет взаимосвязь между прикладным компонентом и классификатором, прикладной компонент которого oo_instance_classifier_relationship является его представителем.

EXPRESS-описание:

```
*)
    ENTITY oo_instance_classifier_relationship;
    classifier : oo_extended_classifier_select;
    description : text_select;
    instance : oo_instance_select;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут classifier: Этот атрибут определяет прикладной компонент, который дает определение прикладного компонента oo_instance_classifier_relationship.

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту oo_instance_classifier_relationship.

Атрибут instance: Этот атрибут определяет прикладной компонент oo_instance_classifier_relationship, который должен связываться с определенной операцией.

4.3.189 Прикладной компонент oo_interaction

Прикладной компонент oo_interaction определяет способ посылки сообщения между различными пунктами для выполнения определенного задания.

EXPRESS-описание:

```
*)
    ENTITY oo_interaction;
    description : OPTIONAL text_select;
    id : element_identifier;
    interaction_context : oo_collaboration;
    name : label;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту oo_interaction.

Атрибут id: Этот атрибут определяет идентификатор прикладного компонента oo_interaction.

Атрибут interaction_context: Этот атрибут определяет прикладной компонент oo_collaboration, указывающий контекст прикладного компонента oo_interaction.

Атрибут name: Этот атрибут определяет слово (или слова), которые используются для ссылки на прикладной компонент oo_interaction.

4.3.190 Прикладной компонент oo_interfaces

Прикладной компонент oo_interfaces определяет совокупность прикладных компонентов oo_operation, которая может использоваться для определения услуги, предоставляемой данной инстанцией.

EXPRESS-описание:

```
*)
    ENTITY oo_interfaces;
    associated_version : configuration_element_version;
    description : OPTIONAL text_select;
    id : element_identifier;
    name : label;
    namespace : OPTIONAL oo_namespace_select;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут associated_version: Этот атрибут определяет прикладной компонент configuration_element_version для прикладного компонента oo_interfaces.

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту oo_interfaces.

Атрибут id: Этот атрибут определяет идентификатор прикладного компонента oo_interfaces.

Атрибут name: Этот атрибут определяет слово (или слова), которые используются для ссылки на прикладной компонент oo_interfaces.

Атрибут namespace: Этот атрибут определяет элемент, к которому принадлежит прикладной компонент oo_class. Имена элементов, принадлежащих к одной и той же области имен, должны быть уникальными для совокупности этих элементов.

4.3.191 Прикладной компонент oo_link

Прикладной компонент oo_link определяет связь между инстанциями и является представителем прикладного компонента oo_association object.

EXPRESS-описание:

```
*)
    ENTITY oo_link;
    definition : oo_generic_association;
    INVERSE
    connection : SET[2:?] OF oo_link_end FOR link;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут definition: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту oo_generic _association.

Атрибут connection: Этот атрибут определяет связь для размещения прикладных компонентов oo_attribute_instance в прикладных компонентах oo_link_end.

4.3.192 Прикладной компонент oo_link_end

Прикладной компонент oo_link_end является конечной точкой для прикладного компонента oo_link и представителем прикладного компонента oo_association_end.

EXPRESS-описание:

```
*)
    ENTITY oo_link_end;
    definition : oo_generic_association_end;
    instance : oo_instance_select;
    link : oo_link;
```

WHERE
WR1: definition.association :=: link.definition;
END_ENTITY;
(*
Определения атрибутов:
Атрибут definition: Этот атрибут определяет прикладной компонент oo_association_end, который дает описание прикладного компонента oo_link_end.
Атрибут instance: Этот атрибут определяет инстанцию, за которой закреплен прикладной компонент oo_link_end.
Атрибут link: Этот атрибут позволяет размещать прикладные компоненты oo_attribute_instance в прикладных компонентах oo_link_end.
Формальные выражения:
WR1:

4.3.193 Прикладной компонент oo_message
Прикладной компонент oo_message определяет связь между инстанциями, которая будет обеспечивать передачу информации, ожидая, что эта операция будет выполнена.
EXPRESS-описание:
*)
ENTITY oo_message;
action : oo_action;
communication : OPTIONAL oo_association_role;
interaction : oo_interaction;
name : label;
receiver : oo_classifier_rolen;
sender : oo_classifier_rolen;
sequence_number : LIST[1:?] OF natural_number;
END_ENTITY;
(*
Определения атрибутов:
Атрибут action: Этот атрибут определяет прикладной компонент oo_action, который после выполнения будет передавать представитель прикладного компонента oo_message.
Атрибут communication: Этот атрибут определяет прикладной компонент oo_association_role, который указывает назначение, которое должны подтверждать отправитель и получатель.
Атрибут interaction: Этот атрибут определяет прикладной компонент oo_interaction, который является частью указанной связи.
Атрибут name: Этот атрибут определяет слово (или слова), которые используются для ссылки на прикладной компонент oo_message.
Атрибут receiver: Этот атрибут определяет получаемый прикладной компонент oo_classifier_role.
Атрибут sender: Этот атрибут определяет отправляемый прикладной компонент oo_classifier_role.
Атрибут sequence_number: Этот атрибут определяет порядок следования прикладного компонента oo_message в последовательности прикладных компонентов oo_message в процессе их полного взаимодействия.

4.3.194 Прикладной компонент oo_message_temporal_relationship
Прикладной компонент oo_message_temporal_relationship определяет взаимосвязь между двумя прикладными компонентами oo_message и указывает временной порядок их следования.
EXPRESS-описание:
*)
ENTITY oo_message_temporal_relationship;
predecessor : oo_message;
successor : oo_message;
END_ENTITY;
(*

Определения атрибутов:

Атрибут predecessor: Этот атрибут определяет предыдущий в последовательности прикладной компонент oo_message.

Атрибут successor: Этот атрибут определяет последующий в последовательности прикладной компонент oo_message.

4.3.195 Прикладной компонент oo_method

Прикладной компонент oo_method определяет способ реализации прикладного компонента oo_operation.

EXPRESS-описание:

```
*)
    ENTITY oo_method
    SUBTYPE OF (oo_behavioural_featuree) ;
    body : textual_specification;
    description : OPTIONAL text_select;
    id : element_identifier;
    specification : oo_operation;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут body: Этот атрибут определяет прикладной компонент textual_specification, который задает спецификацию и язык ее описания для прикладного компонента oo_method.

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту oo_method.

Атрибут id: Этот атрибут определяет идентификатор прикладного компонента oo_method.

Атрибут specification: Этот атрибут определяет прикладной компонент oo_operation, для которого прикладной компонент oo_method является реализацией.

4.3.196 Прикладной компонент oo_model_element_stereotype_relationship

Прикладной компонент oo_model_element_stereotype_relationship определяет взаимосвязь между прикладным компонентом oo_stereotype и другим прикладным компонентом.

EXPRESS-описание:

```
*)
    ENTITY oo_model_element_stereotype_relationship;
    model_element : oo_model_element_select;
    stereotype : oo_stereotype;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут model_element: Этот атрибут определяет связанный элемент.

Атрибут stereotype: Этот атрибут определяет связывающий прикладной компонент oo_stereotype.

4.3.197 Прикладной компонент oo_model_element_tagged_value_relationship

Прикладной компонент oo_model_element_tagged_value_relationship определяет взаимосвязь между прикладным компонентом oo_tagged_value и другим прикладным компонентом, использующим прикладной компонент oo_tagged_value.

EXPRESS-описание:

```
*)
    ENTITY oo_model_element_tagged_value_relationship;
    model_element : oo_model_element_select;
    tagged_value : oo_tagged_valuee;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут model_element: Этот атрибут определяет связывающий элемент.

Атрибут tagged_value: Этот атрибут определяет связанный прикладной компонент oo_tagged_value.

4.3.198 Прикладной компонент oo_object

Прикладной компонент oo_object является представителем прикладного компонента oo_class.

EXPRESS-описание:

```
*)
    ENTITY oo_object;
    definition : oo_class;
    description : OPTIONAL text_select;
    id : element_identifier;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут definition: Этот атрибут определяет прикладной компонент oo_class, для которого прикладной компонент oo_object является представителем.

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту oo_object.

Атрибут id: Этот атрибут определяет идентификатор прикладного компонента oo_object.

4.3.199 Прикладной компонент oo_operation

Прикладной компонент oo_operation определяет услугу, которая может запрашиваться для выполнения представления.

EXPRESS-описание:

```
*)
    ENTITY oo_operation
    SUBTYPE OF (oo_behavioural_featuree) ;
    concurrency : label;
    description : OPTIONAL text_select;
    id : element_identifier;
    is_abstract : BOOLEAN;
    specification : textual_specification;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут concurrency: Этот атрибут определяет способ вызова прикладного компонента oo_operation для его выполнения.

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту oo_operation.

Атрибут id: Этот атрибут определяет идентификатор прикладного компонента oo_operation.

Атрибут is_abstract: Этот атрибут определяет, возможна ли реализация прикладного компонента oo_operation. Если он находится в состоянии TRUE, то прикладной компонент oo_operation будет абстрактным и не может быть непосредственно реализован, а реализованы могут быть только производные атрибуты is_abstract. Если же он находится в состоянии FALSE, то атрибут is_abstract не будет абстрактным и может быть реализован.

Атрибут specification: Этот атрибут определяет прикладной компонент textual_specification, который дает спецификацию и язык ее описания для прикладного компонента oo_operation.

4.3.200 Прикладной компонент oo_operation_interface_association

Прикладной компонент oo_operation_interface_association определяет взаимосвязь между прикладными компонентами oo_operation и oo_interface.

EXPRESS-описание:

```
*)
    ENTITY oo_operation_interface_association;
    description : OPTIONAL text_select;
    interface : oo_interfaces;
    operation : oo_operation;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту oo_operation_interface_association.

Атрибут interface: Этот атрибут определяет прикладной компонент oo_interface, чьей частью он является.

Атрибут operation: Этот атрибут определяет прикладной компонент oo_operation, который является частью прикладного компонента oo_interface, определяемого с помощью интерфейса.

4.3.201 Прикладной компонент oo_package

Прикладной компонент oo_package определяет универсальный способ группирования прикладных компонентов.

EXPRESS-описание:

```
*)
    ENTITY oo_package
    SUPERTYPE OF (oo_view ) ;
    associated_version : configuration_element_version;
    description : OPTIONAL text_select;
    id : element_identifier;
    name : label;
    visibility : label;
    INVERSE
    element_import : SET[0:?] OF oo_element_import FOR container;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут associated_version: Этот атрибут определяет прикладной компонент configuration_element_version для прикладного компонента oo_package.

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту oo_package.

Атрибут id: Этот атрибут определяет идентификатор прикладного компонента oo_package.

Атрибут name: Этот атрибут определяет слово (или слова), которые используются для ссылки на прикладной компонент oo_package.

Атрибут visibility: Этот атрибут определяет способ, с помощью которого прикладной компонент oo_package может быть «видимым» вне ограниченной области имен. Там, где это применимо, должны использоваться следующие состояния (значения) этого атрибута:

- состояние private: Определяет, что прикладной компонент oo_package будет «видим» только для элементов в той же области имен;
- состояние protected: Определяет, что прикладной компонент oo_package будет «видим» только для элементов в той же области имен, а также в наследуемых им областях имен;
- состояние public: Определяет, что прикладной компонент oo_package может быть доступен и для других элементов, включая и те, что находятся вне области имен прикладного компонента oo_package.

Атрибут element_import: Этот атрибут определяет прикладной компонент oo_package, который будет импортировать прикладной компонент model_element в хранилище.

4.3.202 Прикладной компонент oo_parameter

Прикладной компонент oo_parameter является переменной, которая может изменяться, пропускаться или возвращаться.

EXPRESS-описание:

```
*)
    ENTITY oo_parametere;
    behavioural_feature : oo_behavioural_feature;
    default_value : label;
    kind : label;
    name : label;
    parameter_type : oo_classifier_select;
```

visibility : label;
END_ENTITY;
(*
Определения атрибутов:
Атрибут behavioural_feature: Этот атрибут определяет прикладной компонент oo_behavioural_feature, к которому принадлежит прикладной компонент behavioural_feature.
Атрибут default_value: Этот атрибут определяет состояние (значение), которое придается атрибуту default_value, когда он создается или если его состояние выпадает из допустимого диапазона значений.
Атрибут kind: Этот атрибут определяет его направление изменений. Там, где это применимо, должны использоваться следующие состояния (значения) этого атрибута.
Атрибут name: Этот атрибут определяет слово (или слова), которые используются для ссылки на него.
Атрибут parameter_type: Этот атрибут определяет тип его определения.
Атрибут visibility: Этот атрибут определяет способ, с помощью которого этот атрибут может быть «видимым» вне ограниченной области имен.

4.3.203 Прикладной компонент oo_reception
Прикладной компонент oo_reception указывает, что определенный заказчиком классификатор подготовлен к получению прикладного компонента oo_signal. Классификатором является один из следующих прикладных компонентов: physical_instance, oo_class, oo_actor, oo_use_case, oo_signal, oo_component или oo_interface.
EXPRESS-описание:
*)
ENTITY oo_reception
SUBTYPE OF (oo_behavioural_featuree) ;
signal : oo_signal;
specification : text_select;
END_ENTITY;
(*
Определения атрибутов:
Атрибут signal: Этот атрибут определяет прикладной компонент oo_signal, который заказчик приготавливает для реагирования.
Атрибут specification: Этот атрибут определяется для спецификации.

4.3.204 Прикладной компонент oo_send_action
Прикладной компонент oo_send_action определяет операцию асинхронной передачи сигналов.
EXPRESS-описание:
*)
ENTITY oo_send_action
SUBTYPE OF (oo_action) ;
signal : oo_signal;
END_ENTITY;
(*
Определение атрибута:
Атрибут signal: Этот атрибут определяет прикладной компонент oo_signal, который должен быть возрастающим.

4.3.205 Прикладной компонент oo_signal
Прикладной компонент oo_signal определяет асинхронную связь между инстанциями.
EXPRESS-описание:
*)
ENTITY oo_signal;
name : label;
namespace : OPTIONAL oo_namespace_select;
END_ENTITY;
(*

Определения атрибутов:

Атрибут name: Этот атрибут определяет слово (или слова), которые используются для ссылки на этот атрибут.

Атрибут namespace: Этот атрибут определяет элемент, к которому принадлежит этот атрибут. Имена элементов, принадлежащих к одной и той же области имен, должны быть уникальными для совокупности этих элементов.

4.3.206 Прикладной компонент oo_signal_behavioural_feature_relationship

Прикладной компонент oo_signal_behavioural_feature_relationship определяет взаимосвязь между прикладными компонентами oo_signal и oo_behavioral_feature, как и его прикладной компонент behavioural_feature_context.

EXPRESS-описание:

```
*)
    ENTITY oo_signal_behavioural_feature_relationship;
    behavioural_feature_context : oo_behavioural_featuree;
    raised_signal : oo_signal;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут behavioural_feature_context: Этот атрибут определяет прикладной компонент behavioural_feature, который определяет контекст прикладного компонента oo_signal, определенного с помощью атрибута raised_signal.

Атрибут raised_signal: Этот атрибут определяет прикладной компонент oo_signal.

4.3.207 Прикладной компонент oo_stereotype

Прикладной компонент oo_stereotype определяет новый тип элемента, который будет расширять семантику мета-модели.

EXPRESS-описание:

```
*)
    ENTITY oo_stereotype;
    associated_version : configuration_element_version;
    description : OPTIONAL text_select;
    id : element_identifier;
    name : label;
    visibility : label;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут associated_version: Этот атрибут определяет прикладной компонент configuration_element_version для прикладного компонента associated_version.

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к этому атрибуту.

Атрибут id: Этот атрибут определяет его идентификатор прикладного компонента.

Атрибут name: Этот атрибут определяет слово (или слова), которые используются для ссылки на имя этого атрибута.

Атрибут visibility: Этот атрибут определяет способ, с помощью которого этот атрибут становится «видимым» вне ограниченной области имен.

4.3.208 Прикладной компонент oo_stimulus

Прикладной компонент oo_stimulus определяет связь между двумя инстанциями.

EXPRESS-описание:

```
*)
    ENTITY oo_stimulus;
    dispatch_action : oo_action;
    receiver : oo_instance_select;
```

```
        sender : oo_instance_select;
        END_ENTITY;
(*
```

Определения атрибутов:

Атрибут dispatch_action: Этот атрибут определяет операцию, выполняемую с помощью данного атрибута.

Атрибут receiver: Этот атрибут определяет представителя, получаемого при приеме.

Атрибут sender: Этот атрибут определяет представителя, посылаемого отправителем.

4.3.209 Прикладной компонент oo_stimulus_argument

Прикладной компонент oo_stimulus_argument определяет взаимосвязь между прикладным компонентом oo_stimulus object и представителем, выполняющим функцию аргумента воздействия.

EXPRESS-описание:

```
*)
        ENTITY oo_stimulus_argument;
        argument : oo_instance_select;
        stimulus : oo_stimulus;
        END_ENTITY;
(*
```

Определения атрибутов:

Атрибут argument: Этот атрибут определяет представителя, который выполняет функцию аргумента для прикладного компонента oo_stimulus, определенного с помощью воздействия.

Атрибут stimulus: Этот атрибут определяет прикладной компонент oo_stimulus, чьим стимулом является агрумент.

4.3.210 Прикладной компонент oo_tagged_value

Прикладной компонент oo_tagged_value определяет свойство пары свойств прикладных компонентов name-initial_value.

EXPRESS-описание:

```
*)
        ENTITY oo_tagged_valuee;
        id : element_identifier;
        initial_value : text_select;
        name : label;
        END_ENTITY;
(*
```

Определения атрибутов:

Атрибут id: Этот атрибут определяет собственный идентификатор.

Атрибут initial_value: Этот атрибут определяет собственное фактическое состояние (значение).

Атрибут name: Этот атрибут определяет слово (или слова), которые используются для ссылки на этот атрибут как на тэг.

4.3.211 Прикладной компонент oo_use_case

Прикладной компонент oo_use_case определяет последовательность операций, которые могут выполняться системой или другими объектами, взаимодействуя с прикладным компонентом oo_actor системы.

EXPRESS-описание:

```
*)
        ENTITY oo_use_case;
        description : OPTIONAL text_select;
        id : element_identifier;
        name : label;
        namespace : OPTIONAL oo_namespace_select;
        END_ENTITY;
(*
```

Определения атрибутов:

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту oo_use_case.

Атрибут id: Этот атрибут определяет идентификатор прикладного компонента oo_use_case.

Атрибут name: Этот атрибут определяет слово (или слова), которые используются для ссылки на прикладной компонент oo_use_case.

Атрибут namespace: Этот атрибут определяет элемент, к которому принадлежит прикладной компонент oo_use_case. Имена элементов, принадлежащих к одной и той же области имен, должны быть уникальными для совокупности этих элементов.

4.3.212 Прикладной компонент oo_view

Прикладной компонент oo_view является графическим представлением совокупности прикладных элементов model_elements.

EXPRESS-описание:

```
*)
    ENTITY oo_view
    SUBTYPE OF (oo_package) ;
    view_type : label;
    END_ENTITY;
(*
```

Определение атрибута:

Атрибут view_type: Этот атрибут определяет тип прикладного компонента oo_view. Там, где это применимо, необходимо использовать следующие состояния (значения) этого атрибута:

- состояние activity: Прикладной компонент oo_view представляет состояние установки, в которой всем или большинством состояний прикладных компонентов oo_action_state являются прикладные компоненты oo_action_state_transition;
- состояние collaboration: Прикладной компонент oo_view представляет собой взаимодействие и связи между совокупностью прикладных компонентов;
- состояние component: Прикладной компонент oo_view представляет собой организации и взаимозависимости прикладных компонентов oo_component;
- состояние deployment: Прикладной компонент oo_view представляет собой конфигурацию исполнительной программы, предназначенной для выполнения прикладных компонентов physical_instance и oo_component;
- состояние sequence: Прикладной компонент oo_view представляет собой взаимодействие совокупности прикладных компонентов во временной последовательности;
- состояние statechart: Прикладной компонент oo_view представляет собой состояние автомата;
- состояние static structure: Прикладной компонент oo_view представляет собой совокупность декларативных (статических) прикладных компонентов;
- состояние use case: Прикладной компонент oo_view указывает связь между прикладными компонентами oo_actor и oo_use_case в системе.

4.3.213 Прикладной компонент oo_view_context_element_relationship

Прикладной компонент oo_view_context_element_relationship определяет размещение объекта в прикладном компоненте oo_view.

EXPRESS-описание:

```
*)
    ENTITY oo_view_context_element_relationship;
    represented_model_element : oo_extended_model_element_select;
    view : oo_view;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут represented_model_element: Этот атрибут определяет прикладной компонент, который должен быть преобразован.

Атрибут view: Этот атрибут определяет прикладной компонент oo_view, который относится к прикладному компоненту model_element.

4.3.214 Прикладной компонент oo_view_relationship

Прикладной компонент oo_view_relationship определяет взаимосвязь между двумя прикладными компонентами oo_view.

EXPRESS-описание:

```
*)
    ENTITY oo_view_relationship;
    base : oo_view;
    description : text_select;
    reference_view : oo_view;
    relationship_type : label;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут base: Этот атрибут определяет связанный прикладной компонент oo_view.

Атрибут description: Этот атрибут определяет дополнительную информацию относительно прикладного компонента oo_view_relationship.

Атрибут reference_view: Этот атрибут определяет связанный прикладной компонент oo_view.

Атрибут relationship_type: Этот атрибут определяет тип прикладного компонента oo_view_relationship.

Пример 15 — Прикладной компонент oo_view в последовательности объектов view_type может быть особым типом другого используемого прикладного компонента oo_view. Например, этот тип будет конкретизировать значения.

4.3.215 Прикладной компонент oo_view_system_view_relationship

Прикладной компонент oo_view_system_view_relationship определяет взаимосвязь между двумя прикладными компонентами oo_view and a system_view.

EXPRESS-описание:

```
*)
    ENTITY oo_view_system_view_relationship;
    объект-приложение oo_view : oo_view;
    system_context : system_view;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут oo_view: Этот атрибут определяет прикладной компонент oo_view, для которого задается атрибут system_context.

Атрибут system_context: Этот атрибут определяет прикладной компонент system_view, который задает контекст атрибута oo_view.

4.3.216 Прикладной компонент organization

Прикладной компонент organization определяет группу лиц, принимающих участие в конкретном бизнес-процессе.

Примечание 1 — Это определение содержится в протоколе AP 214.

EXPRESS-описание:

```
*)
    ENTITY organization;
    delivery_address : OPTIONAL address;
    description : OPTIONAL text_select;
    id : basic_identifier;
    name : label;
    postal_address : OPTIONAL address;
    visitor_address : OPTIONAL address;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут delivery_address: Этот атрибут определяет адрес, куда должны поставляться товары.

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту organization.

Атрибут id: Этот атрибут определяет идентификатор прикладного компонента organization.

Примечание 2 — Присвоение данного атрибута обычно контролируется органом по регистрации, которой может быть общественная организация, присваивающая идентификаторы корпорациям, или же им может быть корпорация-учредитель, присвающая идентификаторы его компонентам.

Пример 16 — Атрибутом id может быть код, присваиваемый в перечне идентификатору на фондовой бирже (или же номер депртамента).

Атрибут name: Этот атрибут определяет слово (или слова), которые используются для ссылки на прикладной компонент organization.

Атрибут postal_address: Этот атрибут определяет адрес почтовой службы, куда должны доставляться письма.

Атрибут visitor_address: Этот атрибут определяет адрес, по которому прикладной компонент organization будет получать посетителей.

4.3.217 Прикладной компонент organization_relationship

Прикладной компонент organization_relationship используется для представления произвольной связи между двумя прикладными компонентами organization.

Примечание 1 — Этот компонент также позволяет определять соответствующую роль прикладного компонента organization в прикладном компоненте project (генеральный подрядчик, партнеры, субподрядчики).

Пример 17 — Связью между двумя прикладными компонентами organization может быть «исполнитель системы» или «партнер».

EXPRESS-описание:

```
*)
    ENTITY organization_relationship;
    description : OPTIONAL text_select;
    name : label;
    related_organization : organization;
    relating_organization : organization;
    WHERE
    WR1: related_organization :<>: relating_organization;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут description: Этот атрибут определяет текстовую строку, которая получает информацию относительно характера связи между двумя прикладными компонентами organization.

Атрибут name: Этот атрибут определяет имя, которое связывается с взаимоотношением между двумя прикладными компонентами organization.

Атрибут related_organization: Этот атрибут определяет второй из двух прикладных компонентов organization, находящийся в этом взаимоотношении.

Примечание 2 — Точная семантика этого атрибута зависит от значения атрибута description.

Атрибут relating_organization: Этот атрибут определяет первый из двух прикладных компонентов organization, находящийся в этом взаимоотношении.

Примечание 3 — Точная семантика этого атрибута зависит от значения атрибута description.

Формальные выражения:

WR1:

4.3.218 Прикладной компонент package

Прикладной компонент package определяет множество данных, используемых для групповых прикладных компонентов, определенным образом связанных между собой.

Примечание 1 — Критерии классификации этих компонентов задаются пользователем.

Примечание 2 — Прикладной компонент package может использоваться для получения групповой информации на любом уровне абстрактного проектирования, начиная с небольшого множества прикладных компонентов requirement_instance, которые тем или иным образом связаны с проектной информацией и являются порождением проектного прикладного компонента engineering_process_activity.

EXPRESS-описание:

```
*)
    ENTITY package
    SUPERTYPE OF (selection_package ) ;
    description : OPTIONAL text_select;
    discriminator : text;
    id : element_identifier;
    name : label;
    INVERSE
    element : SET[0:?] OF package_element_assignment FOR package;
    UNIQUE
    UR1: id;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту package.

Атрибут discriminator: Этот атрибут определяет общий критерий классификации, который применим ко всем элементам прикладного компонента package.

Примечание 3 — Критерий, определяемый с помощью дискриминатора, формально не является строгим. Данный стандарт не освобождает классификацию от конфликта с дискриминатором.

Атрибут id: Этот атрибут определяет идентификатор прикладного компонента package.

Атрибут name: Этот атрибут определяет слово (или слова), которые используются для ссылки на прикладной компонент package.

Атрибут element: Этот атрибут определяет прикладной компонент package, которому присваивается элемент.

Формальные выражения:

UR1:

4.3.219 Прикладной компонент package_classification_assignment

Прикладной компонент package_classification_assignment определяет способ присвоения прикладного компонента package прикладному компоненту package_classification_system, при котором прикладной компонент package становится в системе классификации элементом высшего уровня.

EXPRESS-описание:

```
*)
    ENTITY package_classification_assignment;
    assigned_package : package;
    classification_system : package_classification_system;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут assigned_package: Этот атрибут определяет прикладной компонент package во взаимосвязи компонентов.

Атрибут classification_system: Этот атрибут определяет прикладной компонент package_classification_system во взаимосвязи компонентов.

4.3.220 Прикладной компонент package_classification_system

Прикладной компонент package_classification_system является представлением системы классификации, содержащей множество прикладных компонентов package.

Примечание — Целью введения прикладного компонента package_classification_system является обеспечение его повторного использования в проектных базах данных. Классификация может распространяться на произвольную глубину путем встраивания прикладных компонентов package.

EXPRESS-описание:

```
*)
    ENTITY package_classification_system;
    description : OPTIONAL text_select;
    id : element_identifier;
    name : label;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут description: Этот атрибут определяет дополнительную текстовую информацию, относящуюся к прикладному компоненту package_classification_system.

Атрибут id: Этот атрибут определяет идентификатор прикладного компонента package_classification_system.

Атрибут name: Этот атрибут определяет слово (или слова), которые используются для ссылки на прикладной компонент package_classification_system.

4.3.221 Прикладной компонент package_element_assignment

Прикладной компонент package_element_assignment определяет присвоение элемента прикладному компоненту package.

Примечание — Необходимо численно определить точный перечень тех элементов, которые могут присваиваться прикладному компоненту package.

EXPRESS-описание:

```
*)
    ENTITY package_element_assignment;
    description : OPTIONAL text_select;
    element : package_element_select;
    package : package;
    reference_name : OPTIONAL label;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к нему.

Атрибут element: Этот атрибут определяет прикладной компонент instance_definition_select, configuration_element_version, configuration_element, engineering_process_activity или project, которые присваиваются прикладному компоненту package.

Атрибут package: Этот атрибут определяет прикладной компонент package, которому присваивается данный элемент.

Атрибут reference_name: Этот атрибут определяет слово (или слова), с комощью которого элемент, заданный с помощью атрибута element, дает ссылку на контекст прикладного компонента package, определяемый с помощью атрибута package.

4.3.222 Прикладной компонент package_hierarchy_relationship

Прикладной компонент package_hierarchy_relationship определяет отношение типа «родитель — потомок» между двумя прикладными компонентами package.

Примечание 1 — Целью введения этого компонента является создание систем классификации, начиная от обобщенных и заканчивая специализированными, например системы классификации транспортных средств по более специализированным классам.

Примечание 2 — Классификация в данном контексте не обязательно предполагает наследование характеристик, поскольку оно определяется объектно-ориентированными программными методами проектирования и языками программирования.

EXPRESS-описание:

```
*)
    ENTITY package_hierarchy_relationship;
    sub_package : package;
```

```
super_package : package;
END_ENTITY;
(*
```

Определения атрибутов:

Атрибут sub_package: Этот атрибут определяет дочерний прикладной компонент package в указанном отношении.

Атрибут super_package: Этот атрибут определяет родительский прикладной компонент package в указанном отношении.

4.3.223 Прикладной компонент partial_document_assignment

Прикладной компонент partial_document_assignment принадлежит к тому же типу, что и прикладной компонент document_assignment, с дикриминатором, в котором только часть представляет интерес для элемента документа, которому он присваивается.

EXPRESS-описание:

```
*)
ENTITY partial_document_assignment
SUBTYPE OF (document_assignment) ;
document_portion : label;
END_ENTITY;
(*
```

Определение атрибута:

Атрибут document_portion: Этот атрибут определяет подмножество для присвоенного документа, который представляет интерес для прикладного компонента partial_document_assignment.

4.3.224 Прикладной компонент partial_system_view

Прикладной компонент partial_system_view принадлежит к тому же типу, что и прикладной компонент system_view, а также к области, режиму работы или ориентированному на выбранный сценарий представлению системы.

Примечание 1 — В модели не делается никаких предположений относительно сложности этого представления. Оно может быть каким угодно, начиная от совершенно простого до весьма сложного представления. Прикладной компонент partial_system_view предназначен для сбора требований о них с целью получения информации относительно области, состояния или зависящих от выбранного сценария данных о системе. Прикладной компонент partial_system_definition сам по себе не является спецификацией на конкретную систему.

Примечание 2 — Для включения элементов, содержащихся в прикладном компоненте partial_system_view, в спецификацию на систему, она должна быть связана с прикладным компонентом system_definition путем использования для этого прикладного компонента system_view_assignment. Для прикладных компонентов partial_system_view можно отбирать такую информацию, как, например, мнение о предполагаемом сроке службы конкретной системы или точности воспроизведения (посредством атрибута is_relevant_for).

EXPRESS-описание:

```
*)
ENTITY partial_system_view
SUBTYPE OF (system_view) ;
is_relevant_for : system_view_context;
INVERSE
assigned_to_systems : SET[1:?] OF system_view_assignment FOR assigned_view;
END_ENTITY;
(*
```

Определения атрибутов:

Атрибут is_relevant_for: Этот атрибут определяет прикладной компонент system_view_context, который устанавливает жизненный цикл, точность воспроизведения и мнение в этом атрибуте.

Атрибут assigned_to_systems: Этот атрибут определяет присвоенный прикладной компонент partial_system_view.

4.3.225 Прикладной компонент partial_system_view_relationship

Прикладной компонент partial_system_view_relationship определяет взаимосвязь между двумя прикладными компонентами partial_system_view. Семантика этой взаимосвязи определяется в атрибуте relationship_type.

EXPRESS-описание:

```
*)
    ENTITY partial_system_view_relationship
    SUPERTYPE OF ( triggered_system_view_relationship ) ;
    description : OPTIONAL text_select;
    related : partial_system_view;
    relating : partial_system_view;
    relationship_type : label;
    system_definition_context : system_definition;
    WHERE
    correct_relationship: related :<>: relating;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту partial_system_view.

Атрибут related: Этот атрибут определяет второй прикладной компонент partial_system_view в указанной взаимосвязи.

Атрибут relating: Этот атрибут определяет первый прикладной компонент partial_system_view в указанной взаимосвязи.

Атрибут relationship_type: Этот атрибут определяет собственную семантику.

Пример 18 — Для пусковой ракетной установки приемлемое разграничение состоит в определении системы в двух представлениях (состояниях): «на земле» и «запущена».

Прикладной компонент partial_system_view_relationship может устанавливаться между прикладным компонентом partial_system_view и атрибутом relationship_type в состоянии precedence, что будет указывать на то, что состояние «на земле» всегда будет предшествовать состоянию «запущено», если система находится в рабочем состоянии.

Там, где это применимо, должны использоваться следующие состояния (значения) этого атрибута:

- состояние overlap: Связанный и связывающий прикладные компоненты partial_system_view содержат спецификации, которые являются избыточными (предполагается, что эти спецификации обладают одним и тем же уровнем детализации);
- состояние detail: Связанный прикладной компонент partial_system_view дает более детальное представление о системе, нежели связывающие прикладные компоненты;
- состояние precedence: Связанный прикладной компонент partial_system_view описывает состояние системы, которое будет предшествовать связанному состоянию системы.

Атрибут system_definition_context: Этот атрибут определяет прикладной компонент system_definition, для которого действителен прикладной компонент partial_system_view.

Формальные выражения:

correct_relationship:

4.3.226 Прикладной компонент persistent_storage

Прикладной компонент persistent_storage принадлежит к тому же типу, что и прикладной компонент general_functionality_instance, и позволяет постоянно хранить дискретные прикладные компоненты data_instance.

Примечание 1 — Постоянное хранение компонентов не следует путать с их непрерывным хранением, например в водяном резервуаре, поскольку последнее будет зависеть от права собственности и поэтому должно представляться с помощью прикладного компонента general_function_definition, определяющего функциональные характеристики этого резервуара.

Примечание 2 — Прикладные компоненты data_instance используются для представления объектов хранения, хотя прикладной компонент persistent_storage предназначался для описания этого хранения. Интерфейс к прикладному компоненту persistent_storage определяется с помощью прикладных компонентов actual_io_port. То, что фактически сохраняется в прикладном компоненте persistent_storage, определяется данными, связанными с используемым портом.

EXPRESS-описание:

```
*)
        ENTITY persistent_storage
        SUBTYPE OF (general_functionality_instance) ;
        id : element_identifier;
        name : label;
        permanent : LOGICAL;
        presentation_id : OPTIONAL label;
        read_only : LOGICAL;
        storage_access : label;
        store_size : OPTIONAL INTEGER;
        UNIQUE
        UR1: id;
        END_ENTITY;
(*
```

Определения атрибутов:

Атрибут id: Этот атрибут определяет идентификатор прикладного компонента opersistent_storage.

Атрибут name: Этот атрибут определяет слово (или слова), которые используются для ссылки на прикладной компонент opersistent_storage.

Атрибут permanent: Этот атрибут определяет, является ли хранение постоянным или временным.

Примечание 3 — Временность хранения в данном контексте зависит от области применения. Назначение данного атрибута состоит в определении того, останется ли сохраненная информация доступной после определенного вида состояния системы типа «повторный запуск» или «сброс».

Атрибут presentation_id: Этот атрибут определяет информацию, подтверждающую идентичность прикладного компонента opersistent_storage и предоставляемую пользователю.

Атрибут read_only: Этот атрибут определяет возможность/невозможность записи информации для ее сохранения.

Примечание 4 — Способ считывания только полученных значений в настоящем стандарте не определен, однако определяется средствами проектирования. Допускается, чтобы в различных режимах работы запоминающее устройство имело различные значения атрибута read_only (т. е. рабочие данные могут загружаться в течение времени прогрева системы на земле, однако считываться только в период полета ракеты).

Атрибут storage_access: Этот атрибут определяет способ доступа к элементам в прикладном компоненте opersistent_storage. Там, где это применимо, должны использоваться следующие состояния (значения) этого атрибута:

- состояние queue: Применима семантика памяти типа «первые данные вошли/первые данные вышли»;
- состояние stack: Применима семантика памяти типа «последние данные вошли/первые данные вышли»;
- состояние random_access: Семантика доступа не ограничена;
- состояние other: Семантика доступа не определена.

Атрибут store_size: Этот атрибут определяет верхнюю границу для элементов, которые могут содержаться в прикладном компоненте opersistent_storage.

Формальные выражения:

UR1:

4.3.227 Прикладной компонент persistent_storage_equivalence_relationship

Прикладной компонент persistent_storage_equivalence_relationship определяет взаимосвязь между двумя прикладными компонентами persistent_storage, предназначенную для индикации того, что два прикладных компонента являются представителями одного и того же объекта хранения.

Примечание — Целью введения прикладного компонента persistent_storage_equivalence_relationship является указание того, что два сохраняемых прикладных компонента относятся к одному и тому же накопителю, поскольку можно видеть иерархию функций, однако способ хранения в общем случае будет оставаться единообразным.

EXPRESS-описание:

```
*)
    ENTITY persistent_storage_equivalence_relationship;
    equivalent_storage : SET[2:?] OF persistent_storage_reference;
    valid_context : functional_reference_configuration;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут equivalent_storage: Этот атрибут определяет множество прикладных компонентов persistent_storage_reference, которые должны рассматриваться как относящиеся к области применения прикладного компонента functional_reference_configuration.

Атрибут valid_context: Этот атрибут определяет прикладной компонент system_view, для которого действует соотношение эквивалентности.

4.3.228 Прикладной компонент persistent_storage_reference

Прикладной компонент persistent_storage_reference принадлежит к тому же типу, что и прикладной компонент functionality_instance_reference, который дает способ однозначной ссылки на прикладной компонент persistent_storage.

EXPRESS-описание:

```
*)
    ENTITY persistent_storage_reference
    SUBTYPE OF (functionality_instance_reference) ;
    SELF\functionality_instance_reference. RENAMED referenced_functionality_instance : persistent_storage;
    END_ENTITY;
(*
```

Определение атрибута:

Атрибут referenced_functionality_instance:

4.3.229 Прикладной компонент person

Прикладной компонент person определяет отдельное лицо, которое обладает определенной связью с производственными данными.

EXPRESS-описание:

```
*)
    ENTITY person;
    address : OPTIONAL address;
    first_name : OPTIONAL label;
    id : basic_identifier;
    last_name : OPTIONAL label;
    middle_names : OPTIONAL LIST[1:?] OF label;
    prefix_titles : OPTIONAL LIST[1:?] OF label;
    suffix_titles : OPTIONAL LIST[1:?] OF label;UNIQUE
    UR1: id;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут address: Этот атрибут определяет место, где можно получить доступ к прикладному компоненту person.

Атрибут first_name: Этот атрибут определяет первое имя этого лица.

Атрибут id: Этот атрибут определяет идентификатор прикладного компонента person.

Атрибут last_name: Этот атрибут определяет фамилию этого лица.

Атрибут middle_names: Этот атрибут определяет любое второе имя этого лица.

Атрибут prefix_titles: Этот атрибут определяет любой префикс наименования этого лица.

Атрибут suffix_titles: Этот атрибут определяет любой суффффикс наименования этого лица.

Формальные выражения:

UR1:

4.3.230 Прикладной компонент person_in_organization

Прикладной компонент person_in_organization определяет спецификацию прикладного компонента person в контексте прикладного компонента organization.

EXPRESS-описание:

```
*)
    ENTITY person_in_organization;
    associated_organization : organization;
    associated_person : person;
    description : OPTIONAL text_select;
    role : label;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут associated_organization: Этот атрибут определяет прикладной компонент organization.

Атрибут associated_person: Этот атрибут определяет прикладной компонент person.

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту person_in_organization.

Атрибут role: Этот атрибут определяет взаимосвязь между прикладными компонентами person и organization.

4.3.231 Прикладной компонент person_organization_assignment

Прикладной компонент person_organization_assignment определяет объект, который связывает прикладной компонент organization или person_in_organization с производственными данными.

Примечание — Подобное закрепление дает дополнительную информацию для связанного с ним прикладного компонента. Предоставление этой информации последством указанного присвоения имеет организационный характер, хотя некоторые прикладные компоненты для достижения семантической законченности требуют обязательных данных того же типа, хотя это представление не должно использоваться для связи соответствующих организационных данных с прикладным компонентом, чьи атрибуты позволяют давать непосредственную ссылку на эти данные.

EXPRESS-описание:

```
*)
    ENTITY person_organization_assignment;
    assigned_person_organization : person_organization_select;
    assigned_to : person_organization_assignment_select;
    description : OPTIONAL text_select;
    role : label;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут assigned_person_organization: Этот атрибут определяет рассматриваемый прикладной компонент person_in_organization или organization.

Атрибут assigned_to: Этот атрибут определяет элемент, к которому применим прикладной компонент person_organization_assignment.

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту person_organization_assignment.

Атрибут role: Этот атрибут определяет ответственность присвоенного прикладного компонента person_in_organization или organization в отношении компонента, к которому он применим. Там, где это применимо, должны использоваться следующие состояния (значения) этого атрибута:

- состояние creator: Упоминаемый объект формируется с помощью присвоенного прикладного компонента person или organization;
- состояние custodian: Присвоенный прикладной компонент person или organization ответственен за существование и целостность упоминаемого объекта;
- состояние customer: Присвоенный прикладной компонент person или organization действует как получатель или потребитель упоминаемого объекта;
- состояние designer: Присвоенный прикладной компонент person или organization является компонентом, который предоставляет данные, описывающие упоминаемый объект (например, системную спецификацию);

- состояние editor: Присвоенный прикладной компонент person или organization ответственен за внесение любых изменений в любой атрибут упоминаемого компонента;
- состояние id_owner: Присвоенный прикладной компонент person или organization — это компонент, ответственный за обозначение идентификатора;
- состояние location: Присвоенный прикладной компонент organization определяет место, где упоминаемый компонент может быть найден или находится;
- состояние manufacturer: Присвоенный прикладной компонент person или organization — это компонент, формирующий реальный (физический) объект;
- состояние owner: Присвоенный прикладной компонент person или organization обладают упоминаемый объект и имеющий последнее слово относительно своего размещения или любого его изменения;
- состояние supplier: Присвоенный прикладной компонент person или organization — это компонент, который предоставляет реальный (физический) объект;
- состояние wholesaler: Присвоенный прикладной компонент person или organization определяет сбытовую цепочку между изготовителем и поставщиком.

4.3.232 Прикладной компонент physical_binding

Прикладной компонент physical_binding определяет преобразование между прикладными компонентами formal_physical_port и actual_physical_port, при которых порты соединяются друг с другом.

EXPRESS-описание:

```
*)
    ENTITY physical_binding;
    actual_port : actual_physical_port;
    formal_port : formal_physical_port;
    WHERE
    WR1: formal_port.port_of :=: actual_port.port_of.definition;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут actual_port: Этот атрибут определяет прикладной компонент actual_physical_port в указанном преобразовании.

Атрибут formal_port: Этот атрибут определяет прикладной компонент formal_physical_port в указанном преобразовании.

Формальные выражения:

WR1:

4.3.233 Прикладной компонент physical_composition_relationship

Прикладной компонент physical_composition_relationship определяет способ связи прикладного компонента physical_instance (в роли компонента) с ансамблем компонентов, проедставляемых с помощью прикладного компонента general_physical_definition.

Примечание — Прикладной компонент physical_composition_relationship используется для описания структуры разделения компонента. Прикладной компонент general_physical_definition может, таким образом, состоять из множества субкомпонентов, а взаимосвязь будет указывать, с одной стороны, на определение, а с другой стороны, на представителей, которые принадлежат к разделению компонента.

EXPRESS-описание:

```
*)
    ENTITY physical_composition_relationshipg;
    assembly : general_physical_definition;
    component : physical_instance;
    description : OPTIONAL text_select;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут assembly: Этот атрибут определяет прикладной компонент general_physical_definition в указанной связи.

Атрибут component: Этот атрибут определяет прикладной компонент physical_instance в указанной связи.

Description: Этот атрибут определяет дополнительную информацию относительно указанной связи.

4.3.234 Прикладной компонент physical_connection

Прикладной компонент physical_connection определяет способ объединения двух прикладных компонентов physical_port друг с другом.

Примечание 1 — Прикладной компонент physical_connection определяет идеальное объединение двух портов. При этом предполагается отсутствие потерь энергии в этом объединении. В случае неидеального соединения желательно, чтобы оно само по себе представлялось с помощью прикладного компонента general_physical_definition или physical_instance.

Примечание 2 — В функциональной спецификации, наиболее точно соответствующей прикладному компоненту physical_connection, объектом является прикладной компонент functional_link.

Примечание 3 — Прикладной компонент physical_connection является двухнаправленным.

EXPRESS-описание:

```
*)
    ENTITY physical_connection;
    connected : physical_port;
    connecting : physical_port;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут connected: Этот атрибут определяет второй прикладной компонент physical_port в указанном объединении.

Атрибут connecting: Этот атрибут определяет первый прикладной компонент physical_port в указанном объединении.

4.3.235 Прикладной компонент physical_instance

Прикладной компонент physical_instance определяет нахождение прикладного компонента general_physical_definition в системной спецификации.

EXPRESS-описание:

```
*)
    ENTITY physical_instance;
    definition : general_physical_definition;
    description : OPTIONAL text_select;
    id : element_identifier;
    name : label;
    presentation_id : OPTIONAL label;
    INVERSE
    actual_port : SET[0:?] OF actual_physical_port FOR port_of;
    UNIQUE
    UR1: definition, id;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут definition: Этот атрибут определяет прикладной компонент general_physical_definition, который предоставляет спецификацию на прикладной компонент physical_instance.

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту physical_instance.

Атрибут id: Этот атрибут определяет идентификатор прикладного компонента physical_instance.

Атрибут name: Этот атрибут определяет слово (или слова), которые используются для ссылки на прикладной компонент physical_instance.

Атрибут presentation_id: Этот атрибут определяет информацию, подтверждающую идентичность прикладного компонента physical_instance и предоставляемую пользователю.

Атрибут actual_port: Этот атрибут определяет прикладной компонент physical_instance, для которого компонент port_of является частью интерфейса.

Формальные выражения:

UR1:

4.3.236 Прикладной компонент physical_instance_reference

Прикладной компонент physical_instance_reference определяет однозначную ссылку на прикладной компонент physical_instance в структуре прикладного компонента physical_node_definition.

Примечание — Прикладной компонент physical_instance_reference вводится для обеспечения размещения и указания системно-зависимых нефункциональных характеристик в физическом описании системы.

EXPRESS-описание:

```
*)
    ENTITY physical_instance_reference;
    description : OPTIONAL text_select;
    id : element_identifier;
    name : label;
    reference_for_instance : physical_instance;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут description: Этот атрибут определяет дополнительную текстовую информацию, относящуюся к прикладному компоненту physical_instance_reference.

Атрибут id: Этот атрибут определяет идентификатор прикладного компонента physical_instance_reference.

Атрибут name: Этот атрибут определяет слово (или слова), которые используются для ссылки на прикладной компонент physical_instance_reference.

Атрибут reference_for_instance: Этот атрибут определяет прикладной компонент physical_instance, который является ссылкой для прикладного компонента physical_instance_reference.

4.3.237 Прикладной компонент physical_link_definition

Прикладной компонент physical_link_definition принадлежит к тому же типу, что и прикладной компонент general_physical_definition, и имеет дискриминатор, основной задачей которого является передача информации или материалов.

EXPRESS-описание:

```
*)
    ENTITY physical_link_definition
    SUBTYPE OF (general_physical_definition) ;
    END_ENTITY;
(*
```

4.3.238 Прикладной компонент physical_node_definition

Прикладной компонент physical_node_definition принадлежит к тому же типу, что и прикладной компонент general_physical_definition, и имеет дискриминатор, который используется для обработки (преобразования) информации или материалов из одной формы в другую.

Примечание — Элемент, определяемый с помощью прикладного компонента physical_node_definition, будет практически независящим от методики. В рамках настоящего стандарта не дается никаких предположений относительно характера его физической сущности.

Пример 19 — Например, для выполнения определенного количества операций за заданный промежуток времени может быть использован человек (достаточно надежный и аккуратный), тогда как для некоторых других видов обработки потребуется мощный компьютер. Проектировщик процесса может использовать любые методики для удовлетворения установленных требований.

Пример 20 — Для выполнения теплоизоляции можно использовать несколько вариантов технологий, которые будут основываться на применении различных видов ма-

териалов. Прикладной компонент physical_node_definition при этом будет описывать эксплуатационные характеристики материала, которым должна удовлетворять теплоизоляция (диапазон температур, срок службы, стоимость и т. п.), а при реальном изготовлении компонента будет использоваться специфический материал, отвечающий этим характеристикам.

EXPRESS-описание:

```
*)
    ENTITY physical_node_definition
    SUBTYPE OF (general_physical_definition) ;
    END_ENTITY;
(*
```

4.3.239 Прикладной компонент physical_port

Прикладной компонент physical_port является представлением элемента в интерфейсе к прикладному компоненту physical_instance или general_physical_definition.

EXPRESS-описание:

```
*)
    ENTITY physical_port
    ABSTRACT SUPERTYPE OF ( ONEOF(actual_physical_port, formal_physical_port) ) ;
    description : OPTIONAL text_select;
    direction : label;
    name : label;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту physical_port.

Атрибут direction: Этот атрибут определяет направление передачи данных (материалов, энергии и т. п.), поддерживаемое прикладным компонентом physical_port в соответствии с одним из следующих состояний:

- состояние input: Прикладной компонент physical_port соответствует вводу данных и т.п. при передаче;
- состояние output: Прикладной компонент physical_port соответствует выводу данных и т.п. при передаче;
- состояние bi-directional: Прикладной компонент physical_port соответствует вводу или выводу данных и т. п. при передаче.

Атрибут name: Этот атрибут определяет слово (или слова), которые используются для ссылки на прикладной компонент physical_port.

4.3.240 Прикладной компонент physical_reference_configuration

Прикладной компонент physical_reference_configuration определяет совокупность всех прикладных компонентов physical_reference_relationship, которые применимы для какого-либо физического представления системы.

Примечание 1 — Прикладной компонент physical_reference_configuration может закрепляться за любым числом систем посредством использования прикладных компонентов system_physical_configuration.

Примечание 2 — Прикладной компонент physical_reference_configuration дает способ определения множества нефункциональных представлений физической модели архитектуры одиночной системы.

EXPRESS-описание:

```
*)
    ENTITY physical_reference_configuration;
    description : OPTIONAL text_select;
    END_ENTITY;
(*
```

Определение атрибута:

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту physical_reference_configuration.

4.3.241 Прикладной компонент physical_reference_relationship

Прикладной компонент physical_reference_relationship определяет способ указания иерархической связи между двумя прикладными компонентами physical_instance_reference.

EXPRESS-описание:

```
*)
    ENTITY physical_reference_relationship;
    child : physical_instance_reference;
    mirror_of : physical_composition_relationshipg;
    parent : physical_instance_reference;
    valid_configuration : physical_reference_configuration;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут child: Этот атрибут определяет физический элемент в прикладном компоненте physical_instance_reference.

Атрибут mirror_of: Этот атрибут определяет прикладной компонент physical_composition_relationship, для которого прикладной компонент physical_reference_relationship предоставляет однозначную ссылку.

Атрибут parent: Этот атрибут определяет разложенный прикладной компонент physical_instance_reference в прикладном компоненте physical_reference_relationship.

Атрибут valid_configuration: Этот атрибут определяет прикладной компонент physical_reference_configuration, для которого действующими являются родительский и дочерний прикладные компоненты physical_instance_reference.

4.3.242 Прикладной компонент plus_minus_bounds

Прикладной компонент plus_minus_bounds является спецификацией на допустимые отклонения от численного значения, относящегося к прикладному компоненту nominal_value.

EXPRESS-описание:

```
*)
    ENTITY plus_minus_bounds;
    distribution_function : OPTIONAL textual_specification;
    limited_value : nominal_valuet;
    lower_bound : NUMBER;
    significant_digits : OPTIONAL INTEGER;
    upper_bound : NUMBER;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут distribution_function: Этот атрибут определяет математическую функцию, применяемую для описания предполагаемого распределения измеряемых значений.

Атрибут limited_value: Этот атрибут определяет прикладной компонент nominal_value, дополнительно определяемый с помощью прикладного компонента plus_minus_bounds.

Атрибут lower_bound: Этот атрибут определяет значение допуска, которое должно вычитаться из точного значения для установления минимально допустимого значения величины.

Атрибут significant_digits: Этот атрибут определяет число значащих цифр, характеризующее точность определения нижней и верхней границ диапазона значений.

Атрибут upper_bound: Этот атрибут определяет значение допуска, которое должно прибавляться к точному значению для установления максимально допустимого значения величины.

4.3.243 Прикладной компонент project

Прикладной компонент project определяет идентифицированную программу работ.

Примечание — Прикладной компонент project может дополнительно характеризоваться с помощью запланированных и фактических дат выполнения работ, их допустимого бюджета или ресурсов.

Пример 21 — Для проектирования новой системы проект устанавливает ответственность за принятие проектных решений и калькуляцию стоимости работ.

EXPRESS-описание:

```
*)
    ENTITY project;
    actual_end_date : OPTIONAL date_time;
    actual_start_date : OPTIONAL date_time;
    description : OPTIONAL text_select;
    id : element_identifier;
    name : label;
    planned_end_date : OPTIONAL period_or_date_select;
    planned_start_date : OPTIONAL event_or_date_select;
    work_program : SET[0:?] OF engineering_process_activity;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут actual_end_date: Этот атрибут определяет дату, когда прикладной компонент project был фактически завершен.

Атрибут actual_start_date: Этот атрибут определяет дату, когда прикладной компонент project был фактически начат.

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту project.

Атрибут id: Этот атрибут определяет идентификатор прикладного компонента project.

Атрибут name: Этот атрибут определяет слово (или слова), которые используются для ссылки на прикладной компонент project.

Атрибут planned_end_date: Этот атрибут определяет либо дату, когда прикладной компонент project был завершен (или предполагается, что был завершен), либо продолжительность действия прикладного компонента project.

Атрибут planned_start_date: Этот атрибут определяет либо дату, когда прикладной компонент project был начат (или предполагается, что был начат).

Атрибут work_program: Этот атрибут определяет прикладные компоненты engineering_process_activity, которые выполняются в рамках прикладного компонента project.

4.3.244 Прикладной компонент project_event_reference

Прикладной компонент project_event_reference является определением момента времени, установленным относительно какого-либо события.

EXPRESS-описание:

```
*)
    ENTITY project_event_reference;
    description : OPTIONAL text_select;
    event_type : label;
    offset : value_with_unite;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту project_event_reference.

Атрибут event_type: Этот атрибут определяет вид события, служащего в качестве точки отсчета по времени.

Пример 22 — Примером атрибута event_type могут служить два события: «начало анализа системы» и «окончание рабочего проектирования».

Атрибут offset: Этот атрибут определяет промежуток времени до или после какого-либо события, который должен использоваться для расчета фактического момента времени.

4.3.245 Прикладной компонент project_relationship

Прикладной компонент project_relationship определяет взаимосвязь между двумя прикладными компонентами project.

EXPRESS-описание:

```
*)
        ENTITY project_relationshipe;
        description : OPTIONAL text_select;
        related : project;
        relating : project;
        relation_type : label;
        END_ENTITY;
(*
```

Определения атрибутов:

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к нему.

Атрибут related: Этот атрибут определяет второй из двух прикладных компонентов project в указанной взаимосвязи.

Примечание 1 — Семантика этого атрибута определяется атрибутом relation_type.

Атрибут relating: Этот атрибут определяет первый из двух прикладных компонентов project в указанной взаимосвязи.

Примечание 2 — Семантика этого атрибута определяется атрибутом relation_type.

Атрибут relation_type: Этот атрибут определяет смысл указанной взаимосвязи. Там, где это применимо, должны использоваться следующие состояния (значения этого атрибута):

4.3.246 Прикладной компонент property_assignment

Прикладной компонент property_assignment определяет способ объединения прикладного компонента property_value с элементом.

EXPRESS-описание:

```
*)
        ENTITY property_assignment;
        assigned_to : property_assignment_select;
        assignment_rationale : OPTIONAL text_select;
        measurement_method : label;
        property : property_value;
        property_name : label;
        END_ENTITY;
(*
```

Определения атрибутов:

Атрибут assigned_to: Этот атрибут определяет элемент, с которым связывается то или иное свойство.

Примечание — Перечень объектов должен быть тщательно проанализирован с помощью спецификаций PAS 20542.

Атрибут assignment_rationale: Этот атрибут определяет причину его присвоения прикладному компоненту property_value.

Атрибут measurement_method: Этот атрибут определяет метод, используемый для получения прикладного компонента property_assignment. Там, где это применимо, должны использоваться следующие состояния (значения) этого атрибута:

- состояние measurement;
- состояние estimation.

Атрибут property: Этот атрибут определяет присваиваемое свойство.

Атрибут property_name: Этот атрибут определяет слово (или слова), которые используются для ссылки на прикладной компонент property_value в контексте элемента, которому присваивается прикладной компонент property_value.

4.3.247 Прикладной компонент property_definition

Прикладной компонент property_definition является определением особого качества.

Примечание — Свойство может отражать физическое или любое определенное пользователем измерение.

EXPRESS-описание:

```
*)
    ENTITY property_definition;
    allowed_unit : SET[0:?] OF unit;
    description : OPTIONAL text_select;
    property_type : label;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут allowed_unit: Этот атрибут определяет принятую единицу (единицы) измерений.

Пример 23 — Компания в качестве единицы измерений может принять килограммы или тонны, а не граммы или фунты.

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту property_definition.

Атрибут property_type: Этот атрибут определяет характер свойства.

Пример 24 — Возможными значениями для атрибута property_type могут быть «электрические свойства» и «время».

4.3.248 Прикладной компонент property_relationship

Прикладной компонент property_relationship определяет взаимосвязь между двумя прикладными компонентами property.

Пример 25 — Прикладной компонент property_relationship может использоваться для указания того, что значение одного прикладного компонента property_definition может получаться из значения другого прикладного компонента property_definition.

EXPRESS-описание:

```
*)
    ENTITY property_relationship;
    description : OPTIONAL text_select;
    related : property_definition;
    relating : property_definition;
    relation_type : label;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту property_relationship.

Атрибут related: Этот атрибут определяет второй из двух прикладных компонентов property_definition, связываемых с помощью прикладного компонента property_relationship.

Атрибут relating: Этот атрибут определяет первый из двух прикладных компонентов property_definition, связываемых с помощью прикладного компонента property_relationship.

Атрибут relation_type: Этот атрибут определяет смысл указанной взаимосвязи. Там, где это применимо, должны использоваться следующие состояния (значения) этого атрибута:

- состояние dependency: Связанный прикладной компонент property_definition зависит от связывающего прикладного компонента property_definition;
- состояние hierarchy: Прикладной компонет определяет иерархическую связь, в которой связанный прикладной компонент property_definition находится на более низком уровне в этой иерархии по отношению к связывающему прикладному компоненту property_definition.

4.3.249 Прикладной компонент property_value

Прикладной компонент property_value определяет представление характеристики прикладного компонента property definition.

EXPRESS-описание:

```
*)
    ENTITY property_value;
    definition : property_definition;
    объект-приложение property_value_name : label;
    specified_value : property_value_select;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут definition: Этот атрибут определяет прикладной компонент property definition, который характеризует прикладной компонент property_value.

Атрибут property_value_name: Этот атрибут определяет слово (или слова), которые используются для ссылки на прикладной компонент property_value.

Пример 26 — Примерами этих слов (имен) прикладного компонента property_value_name является «I1» или «vol2».

Атрибут specified_value: Этот атрибут определяет либо текстовое описание, либо численное значение, представляющие прикладной компонент property_value.

Примечание — Если установленное значение выражается как численная величина, то ею может быть действительное число, диапазон допустимых значений, предельное значение для прикладного компонента property_value или их перечни.

4.3.250 Прикладной компонент property_value_function

Прикладной компонент property_value_function определяет способ представления функции, используемой для расчета относительного значения (по отношению к оптимальному) прикладного компонента property_value.

EXPRESS-описание:

```
*)
    ENTITY property_value_function;
    defines_property_value_merit : property_value;
    function_specification : text_select;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут defines_property_value_merit: Этот атрибут определяет прикладной компонент property_value, чье относительное значение рассчитывается с помощью атрибута function_specification.

Атрибут function_specification: Этот атрибут определяет функцию, используемую для определения относительного значения прикладного компонента property_value и определяемую с помощью атрибута defines_property_value_merit.

4.3.251 Прикладной компонент property_value_relationship

Прикладной компонент property_value_ определяет взаимосвязь между двумя прикладными компонентами property_value.

EXPRESS-описание:

```
*)
    ENTITY property_value_relationship;
    related : property_value;
    relating : property_value;
    relation_description : OPTIONAL text_select;
    relation_type : label;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут related: Этот атрибут определяет второй прикладной компонент property_value в указанной взаимосвязи.

Атрибут relating: Этот атрибут определяет первый прикладной компонент property_value в указанной взаимосвязи.

Атрибут relation_description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту property_value_relationship.

Атрибут relation_type: Этот атрибут определяет смысл указанной взаимосвязи.

Примечание — В настоящем стандарте должен приниматься в расчет перечень предпочтительных значений этого атрибута.

4.3.252 Прикладной компонент rank_assignment

Прикладной компонент rank_assignment определяет присвоение элемента какому-либо прикладному компоненту ranking_element.

Примечание — Предполагается, что присвоение с конкретным приоритетом должно производиться по отношению к набору атрибутов.

EXPRESS-описание:

```
*)
    ENTITY rank_assignment;
    assigned_rank : ranking_element;
    assigned_to_element : ranked_element_select;
    assignment_criteria : OPTIONAL label;
    assignment_description : OPTIONAL text_select;
    relevant_elements : rank_group;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут assigned_rank: Этот атрибут определяет прикладной компонент ranking_element, которому присваивается элемент.

Атрибут assigned_to_element: Этот атрибут определяет присваиваемый элемент.

Атрибут assignment_criteria: Этот атрибут определяет критерий, который применялся при присвоении элемента какому-либо прикладному компоненту ranking_element.

Атрибут assignment_description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту ranking_element.

Атрибут relevant_elements: Этот атрибут определяет прикладной компонент rank_group, который содержит ряд элементов, ранжированных в соответствии с тем же критерием.

4.3.253 Прикладной компонент rank_group

Прикладной компонент rank_group является меткой-заполнителем для элементов, которые сравниваются и ранжируются в соответствии с установленным критерием.

EXPRESS-описание:

```
*)
    ENTITY rank_group;
    classification_criteria : ranking_type;
    name : label;
    INVERSE
    compared_element : SET[0:?] OF rank_assignment FOR relevant_elements;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут classification_criteria: Этот атрибут определяет цель его введения. Он устанавливает критерий, который будет общим для всех элементов, присваиваемых прикладному компоненту rank_group.

Атрибут name: Этот атрибут определяет слово (или слова), которые используются для ссылки на прикладной компонент rank_group.

Атрибут compared_element: Этот атрибут определяет прикладной компонент rank_group, который содержит ряд элементов, ранжированных в соответствии с тем же критерием.

4.3.254 Прикладной компонент rank_relation

Прикладной компонент rank_relation определяет способ представления частичных соотношений между двумя прикладными компонентами ranking_element.

EXPRESS-описание:

```
*)
        ENTITY rank_relation;
        higher_rank : ranking_element;
        lower_rank : ranking_element;
        END_ENTITY;
(*
```

Определения атрибутов:

Атрибут higher_rank: Этот атрибут определяет прикладной компонент ranking_element с более высоким приоритетом.

Атрибут lower_rank: Этот атрибут определяет прикладной компонент ranking_element с более низким приоритетом.

4.3.255 Прикладной компонент ranking_element

Прикладной компонент ranking_element является представлением определенного уровня прикладного компонента ranking_system.

EXPRESS-описание:

```
*)
        ENTITY ranking_element;
        description : OPTIONAL text_select;
        name : label;
        INVERSE
        higher_ranked_element : SET[0:?] OF rank_relation FOR lower_rank;
        lower_ranked_element : SET[0:?] OF rank_relation FOR higher_rank;
        END_ENTITY;
(*
```

Определения атрибутов:

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту ranking_element.

Атрибут name: Этот атрибут определяет слово (или слова), которые используются для ссылки на прикладной компонент ranking_element.

Атрибут higher_ranked_element: Этот атрибут определяет прикладной компонент ranking_element с более высоким приоритетом.

Атрибут lower_ranked_element: Этот атрибут определяет прикладной компонент ranking_element с более низким приоритетом.

4.3.256 Прикладной компонент ranking_system

Прикладной компонент ranking_system определяет способ представления дискретных диапазонов для уровней приоритетов.

EXPRESS-описание:

```
*)
        ENTITY ranking_system;
        description : OPTIONAL text_select;
        highest_rank : ranking_element;
        name : label;
        END_ENTITY;
(*
```

Определения атрибутов:

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту ranking_system.

Атрибут highest_rank: Этот атрибут определяет прикладной компонент ranking_system с более высоким приоритетом.

Атрибут name: Этот атрибут определяет слово (или слова), которые используются для ссылки на прикладной компонент ranking_system.

4.3.257 Прикладной компонент real_data_type_definition

Прикладной компонент real_data_type_definition принадлежит к тому же типу, что и прикладной компонент elementary_maths_space, и включает в себя все действительные значения.

EXPRESS-описание:

```
*)
    ENTITY real_data_type_definition
    SUBTYPE OF (elementary_maths_space) ;
    END_ENTITY;
(*
```

4.3.258 Прикладной компонент real_interval

Прикладной компонент real_interval принадлежит к тому же типу, что и прикладной компонент maths_space, и ограничивает подмножество всех действительных значений.

EXPRESS-описание:

```
*)
    ENTITY real_interval
    ABSTRACT SUPERTYPE OF ( ONEOF(finite_real_interval, hibound_real_interval, lobound_real_
    interval) );
    SUBTYPE OF (maths_space) ;
    END_ENTITY;
(*
```

4.3.259 Прикладной компонент realized_system

Прикладной компонент realized_system является ссылкой на физически реализуемую систему и реализуется в соответствии со спецификацией, идентифицируемой с помощью атрибута realisation_of.

Примечание 1 — Модель данных должна быть расширена для ее распространения на реальные системы с целью точного представления программ верификации и их результатов.

EXPRESS-описание:

```
*)
    ENTITY realized_systemp;
    description : text_select;
    id : element_identifier;
    name : label;
    realization_of : system_instance;
    UNIQUE
    UR1: id, realization_of;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту realized_system.

Атрибут id: Этот атрибут определяет идентификатор прикладного компонента realized_system.

Примечание 2 — Атрибутом id обычно является серийный номер прикладного компонента realized_system.

Атрибут name: Этот атрибут определяет слово (или слова), которые используются для ссылки на прикладной компонент realized_system.

Атрибут realization_of: Этот атрибут определяет прикладной компонент system_instance, который дает спецификацию на экземпляр прикладного компонента realized_system.

Формальные выражения:

UR1:

4.3.260 Прикладной компонент realized_system_composition_relationship

Прикладной компонент realized_system_composition_relationship определяет иерархическую связь между двумя прикладными компонентами realized_system. Суперсистема в этой связи идентифицируется посредством атрибута системы, а субсистема — посредством атрибута компонента.

Примечание — Прикладной компонент realized_system_composition_relationship определяет, что структура системы формируется для конкретной цели. В области применения в настоящем стандарте наиболее вероятной причиной этого является проверка того, что реализованные системы совместимы со спецификацией, на которой основывалась реализация этих систем.

EXPRESS-описание:

```
*)
    ENTITY realized_system_composition_relationship;
    component : realized_systemp;
    description : text_select;
    mirror_of : system_composition_relationship;
    system : realized_systemp;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут component: Этот атрибут определяет субсистему в прикладном компоненте realized_system_composition_relationship.

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту realized_system_composition_relationship.

Атрибут mirror_of: Этот атрибут определяет прикладной компонент system_composition_relationship, который реализуется в физической системе и индицируется с помощью прикладного компонента realized_system_composition_relationship.

Атрибут system: Этот атрибут определяет суперсистему в прикладном компоненте realized_system_composition_relationship.

4.3.261 Прикладной компонент record_data_type_definition

Прикладной компонент record_data_type_definition принадлежит к тому же типу, что и прикладной компонент user_defined_data_type_definition, и содержит ряд значений в одном элементе (записи). Существует зависимость между связанными значениями, которые ограничивают их независимое применение.

Пример 27 — Рассмотрим следующее определение:

```
record EmployeeDetails
name : String;
date_of_birth : Date;
age : Integer Derived = TODAY - date_of_birth;
end record;
```

Атрибут date_of_birth определяет дату рождения сотрудника, имя которого указано в записи. Если эта дата отделена от остальных данных о сотруднике, то она становится просто датой.

EXPRESS-описание:

```
*)
    ENTITY record_data_type_definition
    SUBTYPE OF (user_defined_data_type_definition) ;
    END_ENTITY;
(*
```

4.3.262 Прикладной компонент record_data_type_member

Прикладной компонент record_data_type_member определяет взаимосвязь между прикладными компонентами record_data_type_definition и data_field. При этом их порядок в связи будет отсутствовать, однако должны иметься все члены прикладного компонента record_data_type_definition.

EXPRESS-описание:

```
*)
    ENTITY record_data_type_member;
    child : data_field;
    parent : record_data_type_definition;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут child: Этот атрибут определяет прикладной компонент data_field в указанной взаимосвязи.

Атрибут parent: Этот атрибут определяет прикладной компонент record_data_type_definition в указанной взаимосвязи.

4.3.263 Прикладной компонент recursive_data_type_definition

Прикладной компонент recursive_data_type_definition принадлежит к тому же типу, что и прикладной компонент user_defined_data_type_definition, и является повторным определением другого типа данных. В своей простейшей форме прикладной компонент recursive_data_type_definition позволяет переименовывать или повторно описывать другой тип данных, а в своей наиболее сложной форме прикладной компонент recursive_data_type_definition будет приобретать новые свойства, которые, возможно, будут замещать свойства повторно определенного прикладного компонента recursive_data_type_definition или maths_space.

Тип рекурсивных данных наследует свойства повторно определенного типа данных.

Тип рекурсивных данных может обладать свойствами, которыми не обладает повторно определяем тип данных.

Тип рекурсивных данных может обладать альтернативными значениями для свойств по сравнению с повторно определенным типом данных. Повторно определенные свойства могут вступать в конфликт с существующими свойствами, однако в случае возникновения этого конфликта должно предполагаться, что свойства прикладного компонента recursive_data_type_definition обладают преимуществом перед повторно определенным типом данных.

EXPRESS-описание:

```
*)
    ENTITY recursive_data_type_definition
    SUBTYPE OF (user_defined_data_type_definition) ;
    redefines : data_type_definition_select;
    END_ENTITY;
(*
```

Определение атрибута:

Атрибут redefines: Этот атрибут определяет прикладной компонент maths_space или user_defined_data_type_definition, на котором основывается прикладной компонент recursive_data_type_definition.

4.3.264 Прикладной компонент requirement_allocation_property_relationship

Прикладной компонент requirement_allocation_property_relationship определяет способ связи присвоенного прикладного компонента requirement_instance с элементом прикладного компонента property_value, присваиваемого этому элементу.

Примечание — Подобная конструкция позволяет отслеживать последствия или значимость размещения требования к элементу.

EXPRESS-описание:

```
*)
    ENTITY requirement_allocation_property_relationship;
    allocated_requirement : requirement_allocation_relationship;
    define_property_value : property_value;
    description : OPTIONAL text_select;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут allocated_requirement: Этот атрибут определяет требование к указанной связи.

Атрибут define_property_value: Этот атрибут определяет прикладной компонент property_value, на который оказывает влияние данное требование (или определяется им).

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту requirement_allocation_property_relationship.

4.3.265 Прикладной компонент requirement_allocation_relationship

Прикладной компонент requirement_allocation_relationship определяет способ размещения прикладного компонента requirement_instance в другом элементе, так что этот элемент будет удовлетво-

рять заданному требованию или, как будет установлено, будет совместимым с функциональными характеристиками.

Примечание — Семантика прикладного компонента requirement_allocation_relationship дополнительно определяется с помощью атрибута role.

EXPRESS-описание:

```
*)
    ENTITY requirement_allocation_relationship
    SUPERTYPE OF ( specific_requirement_allocation_relationship ) ;
    description : OPTIONAL text_select;
    relation_to : requirement_allocation_select;
    requirement : requirement_instance;
    role : label;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту requirement_allocation_relationship.

Атрибут relation_to: Этот атрибут определяет элемент, на который распределен прикладной компонент requirement_instance.

Атрибут requirement: Этот атрибут определяет прикладной компонент requirement_instance в связи.

Атрибут role: Этот атрибут определяет семантику связи. Там, где это применимо, должны использоваться следующие состояния (значения) этого атрибута:

- состояние allocation: Прикладной компонент определяет связь при размещении требуемого компонента в атрибуте relation_to. Эта связь означает, что атрибут relation_to должен быть спроектирован так, чтобы установленное требование выполнялось;
- состояние fulfils: Прикладной компонент определяет связь, при которой установленное для компонента требование будет удовлетворяться с помощью атрибута relation_to. Эта связь означает, что атрибут relation_to должен оцениваться и отвечать установленным требованием.

4.3.266 Прикладной компонент requirement_class

Прикладной компонент requirement_class является совокупностью требований к идентичным характеристикам.

Примечание 1 — Данный стандарт не определяет какую-либо фиксированную структуру для классификации требований. Вместо динамической структуры в этом стандарте предусмотрены прикладные компоненты requirement_class и requirement_class_relationship.

EXPRESS-описание:

```
*)
    ENTITY requirement_class;
    description : OPTIONAL text_select;
    id : element_identifier;
    name : label;
    UNIQUE
    UR1: id;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту requirement_class.

Примечание 2 — Его описание содержит информацию, идентифицирующую характеристики требований, содержащихся в области описания.

Атрибут id: Этот атрибут определяет идентификатор прикладного компонента requirement_class.

Атрибут name: Этот атрибут определяет слово (или слова), которые используются для ссылки на прикладной компонент requirement_class.

Формальные выражения:

UR1:

4.3.267 Прикладной компонент requirement_class_relationship

Прикладной компонент requirement_class_relationship определяет взаимосвязь между двумя прикладными компонентами requirement_class.

Примечание — Семантика этой взаимосвязи определяется с помощью атрибута relationship_type.

EXPRESS-описание:

```
*)
    ENTITY requirement_class_relationships;
    description : OPTIONAL text_select;
    related_class : requirement_class;
    relating_class : requirement_class;
    relationship_type : label;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут description: Этот атрибут определяет дополнительную текстовую информацию, относящуюся к специализации.

Атрибут related_class: Этот атрибут определяет второй прикладной компонент requirement_class в указанной взаимосвязи.

Атрибут relating_class: Этот атрибут определяет первый прикладной компонент requirement_class в указанной взаимосвязи.

Атрибут relationship_type: Этот атрибут определяет характер указанной взаимосвязи. Там, где это применимо, должны использоваться следующие состояния (значения) в этом компоненте:

- состояние specialization: Атрибут related_class для прикладного компонента requirement_class является экземпляром атрибута relating_class для прикладного компонента requirement_class;
- состояние equivalence: Атрибут related_class для прикладного компонента requirement_class является синонимом атрибута relating_class для прикладного компонента requirement_class.

4.3.268 Прикладной компонент requirement_composition_relationship

Прикладной компонент requirement_composition_relationship определяет связь при разбиении прикладных компонентов requirement_definition и requirement_occurence.

Примечание 1 — Прикладной компонент requirement_definition может быть разделен на любое число прикладных компонентов requirement_occurence.

Примечание 2 — Одно важное предположение состоит в том, что модель будет применяться для представления множества вариантов системной спецификации. Соответственно, существует значительный выигрыш от совместного или повторного использования в этих вариантах общих проектных элементов. Указанные предположения приводят к модели, в которой четыре объекта входят в представление единственного требования, каким он видится с точки зрения пользователя. Структура модели будет описана ниже.

Прикладной компонент requirement_definition содержит описание всего того, что требуется. В модели содержатся три субтипа компонента, которые поддерживают различные представления требования. При этом никаких предположений не делается в отношении контекста, в котором используется требование и которое означает отсутствие связей между требованиями, определяемыми прикладным компонентом requirement_definition. Прикладной компонент requirement_definition может давать определение любому числу прикладных компонентов requirement_occurence.

Прикладной компонент requirement_instance является контекстно-зависимым представлением требования и может присваиваться прикладному компоненту system_view (в архитектуре системных функциональных модулей). Все взаимосвязи между требованиями определены в прикладном компоненте requirement_instance.

Прикладной компонент requirement_composition_relationship является средством разделения требований. Для каждого вводимого в состав требования должен быть конкретизирован новый прикладной компонент requirement_composition_relationship.

Прикладной компонент requirement_occurence является промежуточным представлением требования, которое включается в модель для обеспечения максимальной отслеживаемости и усиления поддержки повторного использования этого требования в нескольких различных контекстах (в различных вариантах системы или системах). Прикладной компонент requirement_occurence, использующий только один прикладной компонент requirement_definition в качестве его определения, может служить в качестве определения любого числа прикладных компонентов requirement_instance.

Соответственно, прикладной компонент requirement_occurence позволяет отделять структуру статических требований к прерыванию от динамических, зависящих от системы требований (представляемых с помощью прикладного компонента requirement_instance), а также обеспечивать слабую связь между родительскими и дочерними требованиями в структуре требований к разделению.

Первый из этих аспектов важен из-за того, что он позволяет давать ссылку многих контекстно-зависимых компонентов (прикладной компонент requirement_instance) на соответствующий контекстно-независимый компонент.

Второй из этих аспектов состоит в том, что прикладной компонент requirement_definition, являющийся частью совокупности (посредством прикладных компонентов requirement_occurence и requirement_composition_relationship), неплотно связан с родительским прикладным компонентом requirement_definition. Данная конструкция позволяет информационной модели в одной ситуации представлять какое-либо требование как часть общей иерархии требований, тогда как в другой ситуации оно может рассматриваться как требование высшего уровня.

EXPRESS-описание:

```
*)
    ENTITY requirement_composition_relationship;
    child_requirement : requirement_occurence;
    description : OPTIONAL text_select;
    index : label;
    parent_definition : requirement_definition;
    UNIQUE
    UR1: index, parent_definition;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут child_requirement: Этот атрибут определяет разделяемый прикладной компонент requirement_occurence в указанной связи.

Атрибут description: Этот атрибут определяет дополнительную текстовую информацию, относящуюся к прикладному компоненту requirement_composition_relationship.

Атрибут index: Этот атрибут определяет идентификатор прикладного компонента requirement_occurence, идентифицируемый с помощью атрибута child_requirement в контексте прикладного компонента requirement_definition, который определяется с помощью атрибута parent_definition.

Примечание 3 — Атрибут index является средством представления идентификатора, часто используемым для идентификации требования в средствах управления требованиями. В настоящем стандарте данный идентификатор формируется поэтапно по времени. Если, например, используется цифровая индексация, а какой-либо атрибут child_requirement является третьим в составе, определяемом с помощью атрибута parent_definition, то индексу атрибута должно быть присвоено значение 3.

Атрибут parent_definition: Этот атрибут определяет разделенный прикладной компонент requirement_definition в указанной связи.

Формальные выражения:

UR1:

4.3.269 Прикладной компонент requirement_definition

Прикладной компонент requirement_definition является контекстно-независимым определением прикладного компонента requirement.

Примечание — Сочетание прикладных компонентов requirement_definition, requirement_occurence и requirement_composition_relationship определяет способ разделения комплексных требований на их основные части. Использование трех раздельных компонентов позволяет представлять, что какой-либо прикладной компонент requirement_definition является частью нескольких иерархий разделения, а использование прикладного компонента requirement_occurence для каждой отдельной иерархии будет давать однозначное и изолированное представление каждой иерархии разделения.

EXPRESS-описание:

```
*)
    ENTITY requirement_definition
    ABSTRACT SUPERTYPE OF ( ONEOF(model_defined_requirement_definition, structured_
```

requirement_definition, textual_requirement_definition)) ;
associated_version : configuration_element_version;
id : element_identifier;
name : OPTIONAL label;
INVERSE
composed_of : SET[0:?] OF requirement_composition_relationship FOR parent_definition;
in_requirement_class : SET[0:1] OF requirement_requirement_class_assignment FOR requirement;
UNIQUE
UR1: id;
END_ENTITY;

(*

Определения атрибутов:

Атрибут associated_version: Этот атрибут определяет прикладной компонент configuration_element_version для прикладного компонента requirement_definition.

Атрибут id: Этот атрибут определяет идентификатор прикладного компонента requirement_definition.

Атрибут name: Этот атрибут определяет слово (или слова), которые используются для ссылки на прикладной компонент requirement_definition.

Атрибут composed_of: Этот атрибут определяет разделенный прикладной компонент requirement_definition в указанной связи.

Атрибут in_requirement_class: Этот атрибут определяет прикладной компонент requirement_definition в указанном присвоении.

Формальные выражения:

UR1:

4.3.270 Прикладной компонент requirement_instance

Прикладной компонент requirement_instance является контексно-зависимым представлением требования к конкретной системе.

Примечание — Прикладные компоненты requirement_definition, requirement_occurence и requirement_composition_relationship дают определение требования.

EXPRESS-описание:

*)

ENTITY requirement_instance;
definition : requirement_occurence;
id : element_identifier;
name : label;
INVERSE
implied_external : SET[0:?] OF implied_external_interactionr FOR associated_requirement;
UNIQUE
UR1: definition, id;
END_ENTITY;

(*

Определения атрибутов:

Атрибут definition: Этот атрибут определяет прикладной компонент requirement_definition, который дает дополнительную информацию относительно требования.

Атрибут id: Этот атрибут определяет идентификатор прикладного компонента requirement_instance.

Атрибут name: Этот атрибут определяет слово (или слова), которые используются для ссылки на прикладной компонент requirement_instance.

Атрибут implied_external: Этот атрибут определяет прикладной компонент requirement_instance в указанной связи.

Формальные выражения:

UR1:

4.3.271 Прикладной компонент requirement_occurence

Прикладной компонент requirement_occurence определяет метку-заполнитель для статического представителя требования в иерархии разделения требований.

EXPRESS-описание:

```
*)
    ENTITY requirement_occurence;
    definition : requirement_definition;
    id : element_identifier;
    name : label;
    INVERSE
    child_of : SET[0:?] OF requirement_composition_relationship FOR child_requirement;
    UNIQUE
    UR1: definition, id;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут definition: Этот атрибут определяет прикладной компонент requirement_definition, который дает определение требованию.

Атрибут id: Этот атрибут определяет идентификатор прикладного компонента requirement_occurence.

Атрибут name: Этот атрибут определяет слово (или слова), которые используются для ссылки на прикладной компонент requirement_occurence.

Атрибут child_of: Этот атрибут определяет разделенный прикладной компонент requirement_occurence в указанной связи.

Формальные выражения:

UR1:

4.3.272 Прикладной компонент requirement_relationship

Прикладной компонент requirement_relationship определяет способ представления взаимосвязей между прикладными компонентами requirement_instance.

Примечание 1 — Прикладной компонент requirement_relationship может регистрировать существование взаимосвязей между набором требований или же может использоваться для получения взаимосвязей между наборами требований и отдельными требованиями.

Примечание 2 — Прикладной компонент requirement_relationship может использоваться, например, для идентификации измененных или производных требований в различных прикладных компонентах system_view.

Примечание 3 — Предусмотрено два следующих способа использования прикладного компонента requirement_relationship.

1. Использование для связи нескольких требований с целью ее определения с помощью атрибута relationship_type. Любое число прикладных компонентов requirement_relationship_input_assignment, связывающих прикладные компоненты requirement_relationship и ряд прикладных компонентов requirement_instance, может поддерживать подобную связь. При таком использовании никакой новый прикладной компонент requirement_relationship_resulting_relationship не реализуется в результате установления с помощью прикладного компонента requirement_relationship.

2. Использование для создания или идентификации требований, которые зависят от установленного прикладного компонента requirement_relationship. Как и в первом случае, любое число прикладных компонентов requirement_relationship_input_assignment, связывающих прикладной компонент requirement_relationship и ряд прикладных компонентов requirement_instance формирует входные данные для связи. Для индикации выходных данных при анализе любое число прикладных компонентов requirement_relationship_resulting_relationship может использоваться с целью указания требований, созданных или идентифицированных на основе этой связи требований.

EXPRESS-описание:

```
*)
    ENTITY requirement_relationship;
    description : OPTIONAL text_select;
```

relationship_type : label;
END_ENTITY;
(*
Определения атрибутов:
Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту requirement_relationship.
Атрибут relationship_type: Этот атрибут определяет тип связи. Там, где это применимо, должны использоваться следующие состояния (значения) этого атрибута:
- состояние alternate: Требования, присоединяемые посредством прикладного компонента requirement_relationship_input_assignment к прикладному компоненту requirement_relationship, являются взаимоисключающими или эквивалентными;
- состояние derived: Требования, объединяемые посредством прикладного компонента requirement_relationship_input_assignment, используются для получения дополнительных прикладных компонентов requirement_instance. Новые требования индицируются с использованием прикладных компонентов requirement_relationship.

4.3.273 Прикладной компонент requirement_relationship_context_assignment
Прикладной компонент requirement_relationship_context_assignment определяет способ объединения прикладного компонента requirement_relationship или requirement_allocation_relationship с прикладным компонентом system_view, для которого связь является действующей.

Примечание — Прикладной компонент requirement_relationship_input_assignment позволяет определять действительность нескольких прикладных компонентов system_view и повторно использовать элементы спецификации для нескольких вариантов системной спецификации.

EXPRESS-описание:
*)
ENTITY requirement_relationship_context_assignment;
assigned_requirement_relationship : assigned_requirement_relationship_select;
description : OPTIONAL text_select;
system_context : system_view;
END_ENTITY;
(*
Определения атрибутов:
Атрибут assigned_requirement_relationship: Этот атрибут определяет связь, присваиваемую прикладному компоненту system_view с помощью прикладного компонента assigned_requirement_relationship.
Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту requirement_relationship_context_assignment.
Атрибут system_context: Этот атрибут определяет прикладной компонент system_view, который присваивается связи требований.

4.3.274 Прикладной компонент requirement_relationship_input_assignment
Прикладной компонент requirement_relationship_input_assignment определяет способ присвоения прикладного компонента requirement_instance прикладному компоненту requirement_relationship.
EXPRESS-описание:
*)
ENTITY requirement_relationship_input_assignment;
assigned_instance : requirement_instance;
input_requirement : requirement_relationship;
END_ENTITY;
(*

Определения атрибутов:
Атрибут assigned_instance: Этот атрибут определяет прикладной компонент requirement_instance в указанном способе.

Атрибут input_requirement: Этот атрибут определяет прикладной компонент requirement_relationship в указанном способе.

4.3.275 Прикладной компонент requirement_relationship_resulting_relationship

Прикладной компонент requirement_relationship_resulting_relationship определяет способ связи прикладного компонента requirement_instance с прикладным компонентом requirement_relationship, который указывает, что прикладной компонент requirement_instance выражается с помощью прикладного компонента requirement_relationship.

EXPRESS-описание:

```
*)
    ENTITY requirement_relationship_resulting_relationship;
    motivation : OPTIONAL text_select;
    requirement_relationship : requirement_relationship;
    resulting_requirement : requirement_instance;
    role : label;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут motivation: Этот атрибут определяет дополнительную информацию о причинах создания прикладного компонента requirement_relationship_resulting_relationship.

Атрибут requirement_relationship: Этот атрибут определяет прикладной компонент requirement_relationship.

Атрибут resulting_requirement: Этот атрибут определяет вновь созданное требование.

Атрибут role: Этот атрибут определяет объект, который должен быть удален.

4.3.276 Прикладной компонент requirement_requirement_class_assignment

Прикладной компонент requirement_requirement_class_assignment определяет способ присвоения прикладного компонента requirement_definition прикладному компоненту requirement_class, при котором характеристики прикладного компонента requirement_definition будут совпадать с определенными в прикладном компоненте requirement_class.

EXPRESS-описание:

```
*)
    ENTITY requirement_requirement_class_assignment;
    class : requirement_class;
    motivation : OPTIONAL text_select;
    requirement : requirement_definition;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут class: Этот атрибут определяет прикладной компонент requirement_class в данном присвоении.

Атрибут motivation: Этот атрибут определяет причину присвоения.

Атрибут requirement: Этот атрибут определяет прикладной компонент requirement_definition в данном присвоении.

4.3.277 Прикладной компонент requirement_system_view_assignment

Прикладной компонент requirement_system_view_assignment определяет способ присвоения прикладного компонента requirement_instance прикладному компоненту system_view, указывающий на применимость требований к прикладному компоненту system_view.

EXPRESS-описание:

```
*)
    ENTITY requirement_system_view_assignment
    SUPERTYPE OF ( root_requirement_system_view_assignment ) ;
    description : OPTIONAL text_select;
    requirement : requirement_instance;
```

system_view : system_view;
END_ENTITY;
(*
Определения атрибутов:
Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту requirement_system_view_assignment.
Атрибут requirement: Этот атрибут определяет прикладной компонент requirement_instance в указанном способе.
Атрибут system_view: Этот атрибут определяет прикладной компонент system_view в указанном способе.

4.3.278 Прикладной компонент requirement_traces_to_ requirement_relationship
Прикладной компонент requirement_traces_to_requirement_relationship определяет взаимосвязь между двумя прикладными компонентами requirement_instance, присваиваемыми различным прикладным компонентам partial_system_view, которая устанавливает подконтрольное объединение двух требований.

Примечание 1 — Эта ваимосвязь действует только для конкретного прикладного компонента system_view.

Примечание 2 — Эта взаимосвязь мотивируется необходимостью поддержания оперативного контроля между различными прикладными компонентами system_view системы. Кроме того, предполагается, что источник и отслеживаемые требования к источнику не будут присваиваться одному и тому же прикладному компоненту system_view.

EXPRESS-описание:
*)
ENTITY requirement_traces_to_requirement_relationship;
motivation : OPTIONAL text_select;
source_requirement : requirement_instance;
traced_requirement : requirement_instance;
valid_context : system_definition;
END_ENTITY;
(*
Определения атрибутов:
Атрибут motivation: Этот атрибут определяет дополнительную информацию, относящуюся к характеру прикладного компонента requirement_traces_to_requirement_relationship.
Атрибут source_requirement: Этот атрибут определяет первый прикладной компонент requirement_instance в указанной взаимосвязи.
Атрибут traced_requirement: Этот атрибут определяет второй прикладной компонент requirement_instance в указанной взаимосвязи.
Атрибут valid_context: Этот атрибут определяет прикладной компонент system_view, для которого действующим является прикладной компонент requirement_traces_to_requirement_relationship.

4.3.279 Прикладной компонент root_requirement_system_view_assignment
Прикладной компонент root_requirement_system_view_assignment принадлежит к тому же типу, что и прикладной компонент requirement_system_view_assignment и к тому же способу присвоения прикладного компонента requirement_instance, который является основой для структуры требований к прикладному компоненту system_view. Например, присваиваемое требование не должно быть дочерним в указанной структуре.
Индексный атрибут определяет первый элемент в презентационном индексе для требования в конкретной системе.
EXPRESS-описание:
*)
ENTITY root_requirement_system_view_assignment
SUBTYPE OF (requirement_system_view_assignment) ;
index : label;

UNIQUE
UR1: index, system_view;
END_ENTITY;
(*
Определения атрибутов:
Атрибут index: Этот атрибут определяет идентификатор для представления прикладного компонента requirement_instance, идентифицируемого с помощью атрибута requirement, который наследуется из прикладного компонента requirement_system_view_assignment в контексте прикладного компонента system_view, определяемого с помощью атрибута system_view.

Пример 28 — Индекс для конкретного, например, присвоенного требования не должен быть дочерним в структуре требований, может быть равным 5 и указывать на то, что идентифицированный с помощью атрибута requirement (наследуемый из прикладного компонента requirement_system_view_assignment) прикладной компонент requirement_instance будет пятым требованием к прикладному компоненту system_view, идентифицированному с помощью атрибута system_view (подобно наследуемому из прикладного компонента requirement_system_view_assignment).

Примечание — Индексация является способом представления идентификатора, часто используемого для идентификации требования в средствах управления изменениями. В настоящем стандарте данный идентификатор формируется поэтапно, по одному этапу в каждый момент времени для каждого уровня в составе. Например, при использовании цифровой индексации и конкретного атрибута child_requirement, являющегося третьим в составе и определенным с помощью атрибута system_view, индекс атрибута должен быть равен «3».

Последующее разделение требования должно в каждый момент времени добавлять информацию на одну позицию индекса. Например, если основное требование разделяется на четыре частичных требования, то последние могут снабжаться индексами 1..4 с использованием атрибута index прикладного компонента requirement_composition_relationship.

Индекс требования в составной структуре требований реконструируется путем включения атрибутов index обнаруживаемого прикладного компонента requirement_composition_relationship при перемещении разделенного требования от наивысшего требования к наинизшему.

Формальные выражения:
UR1:

4.3.280 Прикладной компонент selection_package

Прикладной компонент selection_package принадлежит к тому же типу, что и прикладной компонент package, с дискриминатором того, что ряд элементов, который может быть выбран из прикладного компонента package для конкретной системы, является ограниченным.

Примечание 1 — Атрибут selection_type определяет семантику прикладного компонента selection_package.

Примечание 2 — Прикладной компонент selection_package предоставляет спецификацию на ограничительные условия, в которых только один из элементов может выбираться из элементов, содержащихся в прикладном компоненте package.

EXPRESS-описание:
*)
ENTITY selection_package
SUBTYPE OF (package) ;
selection_type : label;
END_ENTITY;
(*
Определения атрибутов:
Атрибут selection_type: Этот атрибут определяет выбранное ограничительное условие. Там, где это применимо, должны использоваться следующие состояния (значения) этого атрибута:

- Состояние and: Один элемент в прикладном компоненте package может выбираться для конкретной системы;
- Состояние or: Любое число элементов в прикладном компоненте package может выбираться для конкретной системы.

4.3.281 Прикладной компонент single_cardinality

Прикладной компонент single_cardinality определяет единственное целое число, представляющее численное ограничительное условие.

EXPRESS-описание:

```
*)
    ENTITY single_cardinalityyy;
    defined_value : single_cardinality_select;
    END_ENTITY;
(*
```

Определение атрибута:

Атрибут defined_value: Этот атрибут определяет численное ограничительное условие.

4.3.282 Прикладной компонент specific_requirement_allocation_relationship

Прикладной компонент specific_requirement_allocation_relationship принадлежит к тому же типу, что и прикладной компонент requirement_allocation_relationship и служит в качестве способа распределения или отслеживания прикладного компонента requirement_instance для элемента, который не может быть неоднозначно идентифицирован в функциональной структуре разделения, например прикладного компонента fsm_generic_state или cb_place.

Примечание 1 — Метод идентификации элементов в указанной структуре предназначен для идентификации прикладного компонента functionality_instance_reference, наиболее близкого в структуре разделения к прикладному компоненту fsm_generic_state или cb_place (посредством атрибута related_to), а также для специальной идентификации отслеживаемого объекта (посредством атрибута specific_element).

Примечание 2 — В тех случаях, когда прикладной компонент specific_requirement_allocation_relationship указывает на отношение к прикладному компоненту fsm_generic_state, атрибут related_to должен идентифицировать прикладной компонент fsm_model, который определяется контекстом автомата с конечным числом состояний.

Примечание 3 — В тех случаях, когда прикладной компонент specific_requirement_allocation_relationship указывает на требование, относящееся к прикладному компоненту cb_place, атрибут related_to должен идентифицировать прикладной компонент functionality_instance_reference, который, в свою очередь, идентифицирует прикладной компонент composite_function_definition (посредством прикладного компонента function_instance), который ограничивается прикладным компонентом functional_behaviour_model. Прикладной компонент cb_place, идентифицируемый с помощью атрибута specific_element, является элементом.

EXPRESS-описание:

```
*)
    ENTITY specific_requirement_allocation_relationship
    SUBTYPE OF (requirement_allocation_relationship) ;
    specific_element : specific_element_select;
    WHERE
    WR1: 'SYSTEMS_ENGINEERING_DATA_REPRESENTATION.FUNCTIONALITY_INSTANCE_
    REFERENCE' IN TYPEOF(SELF\ requirement_allocation_relationship.relation_to);
    END_ENTITY;
(*
```

Определение атрибута:

Атрибут specific_element: Этот атрибут определяет компонент, чье требование распределяется посредством прикладного компонента specific_requirement_allocation_relationship.

Формальные выражения:

WR1:

4.3.283 Прикладной компонент specification_state_assignment

Прикладной компонент specification_state_assignment определяет способ присвоения прикладного компонента textual_specification прикладному компоненту fsm_state, при котором спецификация применяется с помощью данного присвоения.

EXPRESS-описание:

```
*)
    ENTITY specification_state_assignment;
    assigned_to : fsm_state;
```

```
        specification : textual_specification;
        END_ENTITY;
(*
```

Определения атрибутов:

Атрибут assigned_to: Этот атрибут определяет прикладной компонент fsm_state в указанном присвоении.

Атрибут specification: Этот атрибут определяет прикладной компонент textual_specification в указанном присвоении.

4.3.284 Прикладной компонент start_order

Прикладной компонент start_order принадлежит к тому же типу, что и прикладной компонент work_order, и служит в качестве разрешения на исполнение одного или нескольких прикладных компонентов engineering_process_activity.

Примечание — Прикладной компонент start_order охватывает все виды заказов на работы, за исключением заказа на внесение изменений, который должен обрабатываться с помощью прикладного компонента change_order.

EXPRESS-описание:

```
*)
        ENTITY start_order
        SUBTYPE OF (work_ordert) ;
        start_order_type : label;
        END_ENTITY;
(*
```

Определение атрибута:

Атрибут start_order_type: Этот атрибут определяет тип работы, допускаемой прикладным компонентом start_order. Там, где это применимо, должны использоваться следующие состояния (значения) этого атрибута:

- состояние design_activity: Определяет разрешение на начало работ по проектированию системы;
- состояние impact_analysis: Определяет разрешение на начало работ по анализу работ и определению влияния наиболее значимого проектного показателя.

4.3.285 Прикладной компонент start_request

Прикладной компонент start_request принадлежит к тому же типу, что и прикладной компонент work_request, который требует указания начала определенной работы.

Примечание — Прикладной компонент start_request содержит все виды запросов для начала работ, за исключением изменения связанной с ним информации.

EXPRESS-описание:

```
*)
        ENTITY start_request
        SUBTYPE OF (work_request) ;
        request_type : label;END_ENTITY;
(*
```

Определения атрибутов:

Атрибут request_type: Этот атрибут определяет характер прикладного компонента start_request. Там, где это применимо, должны использоваться следующие состояния (значения) этого атрибута:

- состояние cost_reduction: Запрос, служащий для снижения инженерных и производственных затрат на элемент;
- состояние customer_rejection: Запрос заказчика, обусловленный отбраковкой элемента;
- состояние customer_request: Запрос на работу, которая необходима для выполнения запроса заказчика;
- состояние durability_improvement: Запрос, служащий для увеличения срока службы элемента;
- состояние government_regulation: Запрос заказчика, обусловленный юридическим требованием к элементу;
- состояние procurement_alignment: Запрос для корректировки процесса закупки различных элементов;

- состояние security_reason: Запрос на работу, которая необходима с точки зрения безопасности;
- состояние standardization: Запрос для унификации различных вариантов исполнения элемента;
- состояние supplier_request: Запрос на работу, которая необходима для выполнения запроса поставщика;
- состояние technical_improvement: Запрос, служащий для улучшения технических характеристик элемента.

4.3.286 Прикладной компонент state_context_relationship

Прикладной компонент state_context_relationship определяет способ связи прикладного компонента fsm_generic_state с прикладным компонентом functional_state_context, указывающий на то, что прикладной компонент fsm_state находится в состоянии высшего уровня в прикладном компоненте functional_state_context.

Примечание — Этот компонент определяет связь между прикладным компонентом fsm_state (состояние высшего уровня в случае использования карты состояний) и средой, в которой действует автомат с конечным числом состояний.

EXPRESS-описание:

```
*)
    ENTITY state_context_relationship;
    in_context : generic_state_context;
    state : fsm_generic_state;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут in_context: Этот атрибут определяет прикладной компонент functional_state_context в указанном способе.

Атрибут state: Этот атрибут определяет прикладной компонент fsm_generic_state в указанном способе.

4.3.287 Прикладной компонент state_function_interaction_port

Прикладной компонент state_function_interaction_port определяет элемент интерфейса к прикладному компоненту functional_state_context.

Примечание — Прикладной компонент state_function_interaction_port является средством импортирования сведений о функциях из прикладного компонента functional_state_context в прикладной компонент functional_state_context. Точный импорт подобных сведений необходим, поскольку прикладной компонент functional_state_context поддерживает собственную область имен.

EXPRESS-описание:

```
*)
    ENTITY state_function_interaction_port;
    port_of : functional_state_context;
    END_ENTITY;
(*
```

Определение атрибута:

Атрибут port_of: Этот атрибут определяет прикладной компонент functional_state_context, интерфейс к которому является частью прикладного компонента state_function_interaction_port.

4.3.288 Прикладной компонент state_machine_ functional_behaviour_model

Прикладной компонент state_machine_ functional_behaviour_model принадлежит к тому же типу, что и прикладной компонент functional_behaviour_model с дискриминатором, динамическое поведение которого может фиксироваться с помощью автомата с конечным числом состояний и связанным с ним формальным математическим описанием.

EXPRESS-описание:

```
*)
    ENTITY state_machine_functional_behaviour_model
    SUBTYPE OF (functional_behaviour_model) ;
    INVERSE
```

```
    behaviour_constraint : SET[1:?] OF fsm_model FOR behaviour_model;
    END_ENTITY;
(*
```

Определение атрибута:

Атрибут behaviour_constraint: Этот атрибут определяет прикладной компонент state_machine_functional_behaviour_model, для которого прикладной компонент fsm_model дает спецификацию на характеристики работ.

4.3.289 Прикладной компонент state_transition_specification_assignment

Прикладной компонент state_transition_specification_assignment определяет присвоение текстовой спецификации прикладному компоненту fsm_state_transition.

Примечание 1 — Прикладной компонент state_transition_specification_assignment определяет способ присвоения контрольного выражения прикладному компоненту fsm_state_transition (в форме Мура для автомата с конечным числом состояний) или способа присвоения контрольного выражения и установки требований к работам для прикладного компонента fsm_state_transition (в форме Мили для автомата с конечным числом состояний).

Примечание 2 — Язык описания присвоенной спецификации протоколом AP-233 не предписывается.

EXPRESS-описание:

```
*)
    ENTITY state_transition_specification_assignment;
    assigned_to : fsm_state_transition;
    specification : textual_specification;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут assigned_to: Этот атрибут определяет прикладной компонент fsm_state_transition, которому присваивается текстовая спецификация.

Атрибут specification: Этот атрибут определяет текстовую спецификацию, присваемую прикладному компоненту fsm_state_transition.

4.3.290 Прикладной компонент string_data_type_definition

Прикладной компонент string_data_type_definition принадлежит к тому же типу, что и прикладной компонент elementary_maths_space, содержащий все строки вплоть до строк с длиной n, которые могут формироваться из комбинации допустимых символов. Если значение n не задано, то строка может иметь бесконечную длину.

EXPRESS-описание:

```
*)
    ENTITY string_data_type_definition
    SUBTYPE OF (elementary_maths_space) ;
    size : OPTIONAL finite_integer_interval;
    END_ENTITY;
(*
```

Определение атрибута:

Атрибут size: Этот атрибут определяет пространство, занимаемое прикладным компонентом string_data_type_definition.

Примечание 1 — Нижняя граница для прикладного компонента finite_integer_interval в общем случае не достигается, однако пригодна в том случае, когда последующее использование становится очевидным (например, идентификация подстрок в строках). Эта нижняя граница должна устанавливаться на нуль.

Примечание 2 — Размер прикладного компонента finite_integer_interval равен максимальному числу символов и возможному числу экземпляров прикладного компонента string_data_type_definition.

Размер компонента не обязательно должен задаваться для прикладного компонента string_data_type_definition в тех случаях, когда он имеет неизвестную длину, а также не обязательно должен задаваться для конкретного прикладного компонента string_data_type_definition object.

4.3.291 Прикладной компонент structured_requirement_definition

Прикладной компонент structured_requirement_definition принадлежит к тому же типу, что и прикладной компонент requirement_definition, с дискриминатором, требуемые возможности которого выражаются числом.

Примечание 1 — Данный компонент дает возможность представления требований, которые структурированы и типизированы, в отличие от требований, выражаемых в текстовом виде.

Примечание 2 — Имя этого компонента должно пересматриваться. Его текущее имя отражает тот факт, что модель свойства используется для сохранения требования. Оптимальное имя для прикладного компонента — structured_requirement_definition.

EXPRESS-описание:

```
*)
    ENTITY structured_requirement_definition
    SUBTYPE OF (requirement_definition) ;
    required_characteristic : property_value;
    END_ENTITY;
(*
```

Определение атрибута:

Атрибут required_characteristic: Этот атрибут определяет прикладной компонент property_value, который дает структурированное определение требования.

4.3.292 Прикладной компонент system_composition_relationship

Прикладной компонент system_composition_relationship определяет соотношение разделения между прикладными компонентами system_definition и system_instance, который является подсистемой прикладного компонента system_definition.

EXPRESS-описание:

```
*)
    ENTITY system_composition_relationship;
    component_system : s system_instance;
    decomposed_system : system_definition;
    description : OPTIONAL text_select;
    relationship_type : label;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут component_system: Этот атрибут определяет субсистему в указанном соотношении.

Атрибут decomposed_system: Этот атрибут определяет систему в указанном соотношении.

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к указанному соотношению.

Атрибут relationship_type: Этот атрибут определяет соотношение (связь), описывающее тип прикладного компонента system_composition_relationship. Там, где это применимо, должны использоваться следующие состояния (значения) этого атрибута:

- состояние mandatory: Субсистема должна быть в структуре разделения реальной системы, которая является действующей для нее;
- состояние optional: Субсистема может быть в любой структуре разделения любой реальной системы (однако этого не требуется).

4.3.293 Прикладной компонент system_definition

Прикладной компонент system_definition принадлежит к тому же типу, что и прикладной компонент system_view, и является представлением системной спецификации в течение всего жизненного цикла системы.

Примечание — Эта спецификация может иметь любую степень законченности. Прикладной компонент system_definition посредством соотношений для сбора подробных компонентов (например, function_definition и др.) формирует системную спецификацию в точке разворачивания системы. При ее завершении прикладной компонент system_definition будет формировать системную спецификацию для всего жизненного цикла системы, который определяется прикладным компонентом system_definition.

EXPRESS-описание:

```
*)
    ENTITY system_definition
    SUBTYPE OF (system_view) ;
    life_cycle_stage : label;
    INVERSE
    assigned_to_system : SET[0:?] OF partial_system_объект-приложение view_relationship FOR
    system_definition_context;
    scenarios_for_system : SET[0:?] OF system_view_assignment FOR system_specification;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут life_cycle_stage: Этот атрибут определяет фазу в течение жизненного цикла системы, в которой действующим является прикладной компонент system_definition.

Пример 29 — Значением этого атрибута может быть состояние «операционный», указывающее на то, что прикладной компонент system_definition дает спецификацию на стадию эксплуатации системы.

Атрибут assigned_to_system: Этот атрибут определяет прикладной компонент system_definition, для которого действующим является прикладной компонент partial_system_view.

Атрибут scenarios_for_system: Этот атрибут определяет прикладной компонент system_definition, которому присваивается прикладной компонент partial_system_view.

4.3.294 Прикладной компонент system_functional_configuration

Прикладной компонент system_functional_configuration определяет способ присвоения прикладного компонента functional_reference_configuration прикладному компоненту context_function_relationship.

Примечание — Прикладной компонент system_functional_configuration определяет нефункциональные характеристики, например значения свойств и информации о размещении, которые являются действующими в одном конкретном представлении системы.

EXPRESS-описание:

```
*)
    ENTITY system_functional_configuration;
    functional_configuration : functional_reference_configuration;
    system : context_function_relationship;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут functional_configuration: Этот атрибут определяет прикладной компонент functional_reference_configuration, который присваивается с помощью этого атрибута.

Атрибут system: Этот атрибут определяет прикладной компонент context_function_relationship, которому присваивается прикладной компонент functional_reference_configuration.

4.3.295 Прикладной компонент system_instance

Прикладной компонент system_instance определяет реализацию системной спецификации в частном контексте.

Примечание 1 — Прикладной компонент system_instance не содержит реализованную систему; планируется только такая конкретная спецификация, которая должна использоваться для реализации системы в структуре компонентов системы.

Примечание 2 — Каждый прикладной компонент system_instance получает свое определение только из одного прикладного компонента system_instance, определенного с помощью атрибута definition.

EXPRESS-описание:

```
*)
    ENTITY s system_instance;
    definition : system_definition;
    description : OPTIONAL text_select;
```

```
id : element_identifier;
name : label;UNIQUE
UR1: definition, id;
END_ENTITY;
(*
```

Определения атрибутов:

Атрибут definition: Этот атрибут определяет прикладной компонент system_definition, который дает точно один прикладной компонент system_definition, действующий в качестве частного определения.

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту system_instance.

Атрибут id: Этот атрибут определяет прикладной компонент element_identifier для прикладного компонента system_instance.

Атрибут name: Этот атрибут определяет слово (или слова), которые используются для ссылки на прикладной компонент system_instance.

Формальные выражения:

UR1:

4.3.296 Прикладной компонент system_instance_relationship

Прикладной компонент system_instance_relationship является представлением взаимодействия между двумя или несколькими прикладными компонентами system_instance.

Примечание — Прикладной компонент system_instance_relationship используется для получения контекста системы путем моделирования взаимосвязи компонентов. Идея состоит в моделировании множества взаимодействующих систем и соотношений между ними. Последнее означает определение кардинальных ограничительных условий между системами, находящимися в связи. Все объекты, входящие в прикладной компонент system_instance_relationship, должны быть частью той же структуры системы разделения.

Это является новым принципом моделирования с точки зрения модели и соответствует системно-контекстному представлению, предложенному Дэвидом Оливером. Пример системной контекстной диаграммы приведен ниже, в комплексных инженерных системах, содержащих модели и объекты, предложенные Дэвидом Оливером. Действующая взаимосвязь формируется с помощью прикладных компонентов system_instance, system_instance_relationship_port и system_instance_relationship.

EXPRESS-описание:

```
*)
    ENTITY s system_instance_relationship;
    description : text_select;
    name : label;
    INVERSE
    connected_port : SET[2:?] OF system_instance_relationship_port FOR defined_relationship;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к описанию.

Атрибут name: Этот атрибут определяет слово (или слова), которые используются для ссылки на имя.

Атрибут connected_port: Этот атрибут определяет упоминаемый прикладной компонент system_instance_relationship.

4.3.297 Прикладной компонент system_instance_relationship_port

Прикладной компонент system_instance_relationship_port определяет способ объединения прикладных компонентов system_instance и system_instance_relationship.

Примечание — Прикладной компонент system_instance_relationship_port также определяет мощность связи прикладного компонента system_instance, участвующего в связи.

Пример 30 — В сотовой телефонной системе может оказаться приемлемым выражать то, что каждый сотовый элемент может одновременно находиться в диапазоне номеров оконечных пунктов 0–5000. В настоящем стандарте это выражается путем конкретизации

связи прикладного компонента system_instance систем сотовой связи с прикладными компонентами system_instance_relationship_port, которая фиксирует кардинальную информацию для каждой системы в этой связи.

Например, прикладной компонент system_instance_relationship_port для прикладного компонента system_instance станции сотовой связи должен иметь кардинальное число в интервале 0–5000.

EXPRESS-описание:

```
*)
    ENTITY s system_instance_relationship_port;
    cardinality : cardinality_association_select;
    defined_relationship : s system_instance_relationship;
    port_of : s system_instance;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут cardinality: Этот атрибут определяет диапазон значений для прикладных компонентов system_instance, которые находятся во взаимной связи. Независимо от определенной кардинальности в точности один прикладной компонент system_instance всегда будет определяться с помощью атрибута port_of.

Кардинальность определяет число прикладных компонентов system_instance, которые входят в прикладной компонент system_instance_relationship, определенный с помощью атрибута defined_relationship.

Атрибут defined_relationship: Этот атрибут определяет указанный прикладной компонент system_instance_relationship.

Атрибут port_of: Этот атрибут определяет указанный прикладной компонент system_instance.

4.3.298 Прикладной компонент system_instance_replication_relationship

Прикладной компонент system_instance_replication_relationship определяет связь между двумя прикладными компонентами system_instance, в которой новый прикладной компонент system_instance заменяет в системе исходный.

Примечание — Этот компонент неадекватно удовлетворяет требование, поэтому требуется дополнительная работа.

EXPRESS-описание:

```
*)
    ENTITY s system_instance_replication_relationship;
    rationale : OPTIONAL text_select;
    replaced_application : s system_instance;
    replacing_application : s system_instance;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут rationale: Этот атрибут определяет причину замены прикладного компонента system_instance.

Атрибут replaced_application: Этот атрибут определяет исходный прикладной компонент system_instance.

Атрибут replacing_application: Этот атрибут определяет замененный прикладной компонент system_instance.

4.3.299 Прикладной компонент system_physical_configuration

Прикладной компонент system_physical_configuration определяет способ присвоения прикладного компонента physical_reference_configuration прикладному компоненту context_physical_relationship.

EXPRESS-описание:

```
*)
    ENTITY system_physical_configuration;
    physical_configuration : physical_reference_configuration;
```

```
system : context_physical_relationship;
END_ENTITY;
(*
```

Определения атрибутов:

Атрибут physical_configuration: Этот атрибут определяет прикладной компонент physical_reference_configuration, который присваивается с помощью прикладного компонента context_physical_relationship.

Атрибут system: Этот атрибут определяет прикладной компонент context_physical_relationship, которому присваивается прикладной компонент context_physical_relationship.

4.3.300 Прикладной компонент system_substitution_relationship

Прикладной компонент system_substitution_relationship определяет связь, указывающую способ замещения одного прикладного компонента system_composition_relationship другим прикладным компонентом system_composition_relationship. Объектами этого замещения являются связанные прикладные компоненты component_system_instance для обоих прикладных компонентов system_composition_relationship. При этом прикладной компонент decomposed_system_definition должен быть одним и тем же для обоих прикладных компонентов system_composition_relationship.

EXPRESS-описание:

```
*)
ENTITY system_substitution_relationship;
base : system_composition_relationship;
description : OPTIONAL text_select;
substitute : system_composition_relationship;
END_ENTITY;
(*
```

Определения атрибутов:

Атрибут base: Этот атрибут определяет прикладной компонент system_composition_relationship, который допускает замену.

Атрибут description: Этот атрибут определяет дополнительную текстовую информацию, относящуюся к прикладному компоненту system_composition_relationship.

Атрибут substitute: Этот атрибут определяет прикладной компонент system_composition_relationship, который может использоваться вместо основного прикладного компонента system_composition_relationship.

4.3.301 Прикладной компонент system_view

Прикладной компонент system_view определяет частичное или полное представление системы согласно спецификации.

Примечание 1 — Прикладной компонент system_view содержит совокупность всей информации, представляющей интерес для системной спецификации.

Примечание 2 — Прикладной компонент system_view содержит понятие конкретного представления для модели системы, как если бы она рассматривалась какой-либо заинтересованной стороной. Он также дает все связанные с системой объекты с базовыми функциональными характеристиками, например, управлением версиями, постоянными идентификаторами, именами и т.п.

EXPRESS-описание:

```
*)
ENTITY system_view
ABSTRACT SUPERTYPE OF ( ONEOF(partial_system_view, system_definition) ) ;
associated_version : configuration_element_version;
description : OPTIONAL text_select;
id : element_identifier;
name : label;
INVERSE
system : SET[0:?] OF context_function_relationship FOR associated_context;
UNIQUE
UR1: id;
```

WHERE
WR1: at_most_one_system_function_assigned(SELF);
END_ENTITY;
(*
Определения атрибутов:
Атрибут associated_version: Этот атрибут определяет прикладной компонент configuration_element, для которого прикладной компонент configuration_element_version представляет собой версию.
Атрибут description: Этот атрибут определяет дополнительную текстовую информацию, относящуюся к прикладному компоненту system_view.
Атрибут id: Этот атрибут определяет идентификатор прикладного компонента system_view.
Атрибут name: Этот атрибут определяет слово (или слова), которые используются для ссылки на прикладной компонент system_view.
Атрибут system: Этот атрибут определяет прикладной компонент system_view, который действителен для существующей связи.
UR1:
WR1:

4.3.302 Прикладной компонент system_view_assignment

Прикладной компонент system_view_assignment определяет присвоение прикладного компонента partial_system_view прикладному компоненту system_definition.

Примечание — Прикладной компонент partial_system_view, присваиваемый прикладному компоненту system_definition, представляет собой часть полной системной спецификации.

EXPRESS-описание:
*)
ENTITY system_view_assignment;
assigned_view : partial_system_view;
assignment_comment : OPTIONAL text_select;
system_specification : system_definition;
END_ENTITY;
(*
Определения атрибутов:
Атрибут assigned_view: Этот атрибут определяет присвоенный прикладной компонент partial_system_view.
Атрибут assignment_comment: Этот атрибут определяет дополнительную информацию, относящуюся к данному присвоению.
Атрибут system_specification: Этот атрибут определяет прикладной компонент system_definition, которому присваивается прикладной компонент partial_system_view.

4.3.303 Прикладной компонент system_view_context

Прикладной компонент system_view_context дает описание верности воспроизведения и мнения о системе для прикладного компонента partial_system_view.

Примечание — Прикладной компонент system_view_context содержит ряд атрибутов, которые будут более эффективными, если они будут присваиваться в индивидуальном порядке.

EXPRESS-описание:
*)
ENTITY system_view_context;
description : OPTIONAL text_select;
fidelity : label;
объект-приложение system_viewpoint : label;
END_ENTITY;
(*
Определения атрибутов:
Атрибут description: Этот атрибут определяет дополнительную текстовую информацию, относящуюся к прикладному компоненту system_view_context.

Атрибут fidelity: Этот атрибут определяет классификацию уровней детализации прикладного компонента partial_system_view.

Пример 31 — Значением верности может быть «пользователь» для указания того, что связанный прикладной компонент partial_system_view определяет системную спецификацию на пользовательском уровне детализации.

Атрибут system_viewpoint: Этот атрибут определяет частное мнение, которое является действующим для прикладного компонента system_view_context.

Пример 32 — Атрибутом system_viewpoint может быть «эксплуатационный персонал», указывающий на то, что прикладной компонент partial_system_view, связывающий этот компонент, содержит технические требования, которые являются действительными с точки зрения этого персонала.

Атрибут system_viewpoint может также использоваться для укаания того, что ссылочный прикладной компонент partial_system_view является значимым для очень специфичной области.

Пример 33 — Атрибутом system_viewpoint может быть «temporal_safety_analysis», указывающий на то, что прикладной компонент partial_system_view, относящийся к атрибуту system_viewpoint, содержит подмножество технических требований к системе, которое определяет временные требования к системе согласно спецификации.

4.3.304 Прикладной компонент textual_paragraph

Прикладной компонент textual_paragraph определяет структурированное представление текста.

Примечание 1 — Имя компонента, вероятно, должно изменяться для отражения того факта, что объект дает не только шаблон, но и текстовую информацию.

EXPRESS-описание:

```
*)
    ENTITY textual_paragraph;
    element_value : LIST[1:?] OF text_elements;
    name : label;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут element_value: Этот атрибут определяет текстовый элемент прикладного компонента textual_paragraph.

Атрибут name: Этот атрибут определяет слово (или слова), которые используются для ссылки на прикладной компонент textual_paragraph.

Примечание 2 — Атрибут name может быть ссылкой на информацию о форматировании текста.

4.3.305 Прикладной компонент textual_requirement_definition

Прикладной компонент textual_requirement_definition определяет тип прикладного компонента requirement_definition и определение требования, выражаемого в тексте.

Примечание — Этот компонент является наиболее «примитивным» представлением требований в настоящем стандарте.

EXPRESS-описание:

```
*)
    ENTITY textual_requirement_definition
    SUBTYPE OF (requirement_definition) ;
    description : text_select;
    END_ENTITY;
(*
```

Определение атрибута:

Атрибут description: Этот атрибут определяет требуемые функциональные характеристики.

4.3.306 Прикладной компонент textual_section

Прикладной компонент textual_section определяет текст, в котором указываются составные элементы (параграфы) текста.

Примечание — Прикладной компонент textual_section предусмотрен для тех ситуаций, когда структура текстовой информации требует последующего форматирования.

EXPRESS-описание:

```
*)
    ENTITY textual_section;
    element : LIST[1:?] OF textual_paragraphn;
    name : label;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут element: Этот атрибут определяет отдельный элемент шаблона.

Атрибут name: Этот атрибут определяет слово (или слова), которые используются для ссылки на прикладной компонент textual_section.

4.3.307 Прикладной компонент textual_specification

Прикладной компонент textual_specification определяет текст, выражаемый с использованием заданного языка.

Примечание 1 — Этот язык может или не может иметь компьютерную интерпретацию.

EXPRESS-описание:

```
*)
    ENTITY textual_specification;
    definition : text_select;
    definition_language : label;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут definition:

Атрибут definition_language: Этот атрибут определяет с помощью слова (или нескольких слов) язык записи спецификации.

Примечание 2 — Возможным значением атрибута definition_language может быть «ANSI C», «структурированный текст» и т. п.

Примечание 3 — Необходимо заново пересмотреть перечень предпочтительных языков.

4.3.308 Прикладной компонент textual_table

Прикладной компонент textual_table является представлением структуры строки, где прикладной компонент textual_section представляет каждую позицию в этой структуре.

Примечание 1 — Прикладной компонент textual_section может содержать перечень прикладных компонентов textual_paragraph, каждый из которых представляет собой столбец прикладного компонента textual_table.

EXPRESS-описание:

```
*)
    ENTITY textual_table;
    name : label;
    table_row : LIST[1:?] OF textual_section;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут name: Этот атрибут определяет слово (или слова), которые используются для ссылки на прикладной компонент textual_table.

Атрибут table_row: Этот атрибут определяет число строк в прикладном компоненте textual_table.

Примечание 2 — Число строк должно быть больше нуля.

4.3.309 Прикладной компонент triggered_system_view_relationship

Прикладной компонент triggered_system_view_relationship принадлежит к тому же типу, что и прикладной компонент partial_system_view_relationship и является TBD-индикатором.

Примечание 1 — Поддерживающая модель спецификация на режимы системы (каждый прикладной компонент partial_system_view) может рассматриваться в системе как режим высокого уровня. Прикладной компонент triggered_system_view_relationship является прикладным компонентом partial_system_view_relationship, для которых связь между связанным и связывающим прикладными компонентами partial_system_view такова, что система может совершать переход из одного режима в другой, когда выполняется атрибут transition_condition.

Примечание 2 — Эта связь является однонаправленной. Прикладной компонент triggered_system_view_relationship определяет только переход от связанного прикладного компонента к связывающему прикладному компоненту triggered_system_view.

EXPRESS-описание:

```
*)
    ENTITY triggered_system_view_relationship
    SUBTYPE OF (partial_system_view_relationship) ;
    transition_condition : text;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут transition_condition: Этот атрибут определяет в текстовом виде условие, которое должно выполняться.

Примечание 3 — Если переход выполняет TBD.

4.3.310 Прикладной компонент undefined_data_type_definition

Прикладной компонент undefined_data_type_definition принадлежит к тому же типу, что и прикладной компонент user_defined_data_type_definition, чья область и диапазон значений не определены.

Примечание — Прикладной компонент user_defined_data_type_definition должен использоваться в тех случаях, когда тип данных прикладного компонента data_instance неизвестен.

EXPRESS-описание:

```
*)
    ENTITY undefined_data_type_definition
    SUBTYPE OF (user_defined_data_type_definition) ;
    END_ENTITY;
(*
```

4.3.311 Прикладной компонент union_data_type_definition

Прикладной компонент union_data_type_definition принадлежит к тому же типу, что и прикладной компонент user_defined_data_type_definition, который в любой момент времени может принимать значение одного из компонентов.

EXPRESS-описание:

```
*)
    ENTITY union_data_type_definition
    SUBTYPE OF (user_defined_data_type_definition) ;
    END_ENTITY;
(*
```

4.3.312 Прикладной компонент union_data_type_member

Прикладной компонент union_data_type_member определяет взаимосвязь между прикладными компонентами union_data_type_definition и data_field, который он содержит. Элементы прикладного компонента union_data_type_definition являются взаимно исключающими друг друга.

EXPRESS-описание:

```
*)
    ENTITY union_data_type_member;
    child : data_field;
```

```
        parent : union_data_type_definition;
        END_ENTITY;
(*
```

Определения атрибутов:

Атрибут child: Этот атрибут определяет прикладной компонент data_field в указанной взаимосвязи.

Атрибут parent: Этот атрибут определяет прикладной компонент union_data_type_definition в указанной взаимосвязи.

4.3.313 Прикладной компонент unit

Прикладной компонент unit является представителем единицы измерений.

EXPRESS-описание:

```
*)
        ENTITY unit;
        unit_name : label;
        END_ENTITY;
(*
```

Определения атрибутов:

Атрибут unit_name: Этот атрибут определяет имя единицы измерений с использованием (в идеальном случае) соответствующих терминов единиц в ИСО.

Примечание 1 — Примерами могут служить единицы измерений: с (секунды), м (метры); кг (килограммы) и т. п.

Примечание 2 — Атрибут unit_name должен находиться в соответствии со стандартными типами измерений.

4.3.314 Прикладной компонент user_defined_data_type_definition

Прикладной компонент user_defined_data_type_definition определяет тип данных вместе с областью определения их значений, которая напрямую не была определена в настоящем стандарте.

EXPRESS-описание:

```
*)
        ENTITY user_defined_data_type_definition
        ABSTRACT SUPERTYPE OF ( ONEOF(abstract_data_type_definition, aggregate_data_type_
        definition, derived_data_type_definition, record_data_type_definition, recursive_data_type_defini-
        tion, undefined_data_type_definition, union_data_type_definition) ) ;
        associated_version : configuration_element_version;
        description : OPTIONAL text_select;
        id : element_identifier;
        name : label;
        UNIQUE
        UR1: id;
        END_ENTITY;
(*
```

Определения атрибутов:

Атрибут associated_version: Этот атрибут определяет прикладной компонент configuration_element_version для прикладного компонента user_defined_data_type_definition.

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту user_defined_data_type_definition.

Атрибут id: Этот атрибут определяет идентификатор прикладного компонента user_defined_data_type_definition.

Атрибут name: Этот атрибут определяет слово (или слова), которые используются для ссылки на прикладной компонент user_defined_data_type_definition.

Формальные выражения:

UR1:

4.3.315 Прикладной компонент value_limit

Прикладной компонент value_limit принадлежит к тому же типу, что и прикладной компонент value_with_unit и классифицированное численное значение, представляющее либо нижний предел, либо верхний предел какой-либо характеристики.

Пример 34 — Примерами прикладного компонента value_limit являются «30.5 max» и «5 min».

EXPRESS-описание:

```
*)
    ENTITY value_limit
    SUBTYPE OF (value_with_unite) ;
    limit : NUMBER;
    limit_qualifier : label;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут limit: Этот атрибут определяет предельное значение.

Атрибут limit_qualifier: Этот атрибут определяет тип предела. Там, где это применимо, должны использоваться следующие состояния (значения) этого атрибута:

- состояние max: Нормативным пределом является верхний предел;
- состояние min: Нормативным пределом является нижний предел.

4.3.316 Прикладной компонент value_list

Прикладной компонент value_list определяет упорядоченную совокупность прикладных компонентов value_with_unit для прикладного компонента property_value.

Пример 35 — Прикладной компонент property_definition может содержать различные значения свойств, например «массы», «скорости» и «возраста», которые все необходимы в данном контексте. Прикладной компонент value_list объединяет все эти значения в заданном порядке, так что каждое из них поддается идентификации в перечне по их индексам.

EXPRESS-описание:

```
*)
    ENTITY value_list;
    values : LIST[1:?] OF value_with_unite;
    END_ENTITY;
(*
```

Определение атрибута:

Атрибут values: Этот атрибут определяет упорядоченную совокупность прикладных компонентов value_with_unit, которые совместно предоставляются для прикладного компонента property_value.

4.3.317 Прикладной компонент value_range

Прикладной компонент value_range принадлежит к тому же типу, что и прикладной компонент value_with_unit и пара численных значений, представляющих их диапазон, которому должно принадлежать значение прикладного компонента property_value.

Примечание — Данный вид допуска может заменять номинальное значение прикладного компонента property_value. Все значения в этом диапазоне будут считаться одинаково «хорошими». Обычно этот способ определения размеров используется для задания допусков на большое число объектов с конкретной технической характеристикой.

EXPRESS-описание:

```
*)
    ENTITY value_range
    SUBTYPE OF (value_with_unite) ;
    distribution_function : OPTIONAL textual_specification;
    lower_limit : NUMBER;
    upper_limit : NUMBER;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут distribution_function: Этот атрибут определяет прогнозируемую функцию распределения для большого множества результатов измерений.

Атрибут lower_limit: Этот атрибут определяет минимально допустимое значение прикладного компонента property_value, которое ограничивается диапазоном значений прикладного компонента value.

Атрибут upper_limit: Этот атрибут определяет максимально допустимое значение прикладного компонента property_value, которое ограничивается диапазоном значений прикладного компонента value.

4.3.318 Прикладной компонент value_with_unit

Прикладной компонент value_with_unit определяет либо единственную численную меру, либо диапазон численных значений, имеющий верхнюю или нижнюю границы.

Примечание — Каждый прикладной компонент value_with_unit является прикладным компонентом value_limit, value_range или nominal_value.

EXPRESS-описание:

```
*)
    ENTITY value_with_unite
    ABSTRACT SUPERTYPE OF ( ONEOF(nominal_valuet, value_limit, value_range) ) ;
    significant_digits : OPTIONAL INTEGER;
    unit_component : OPTIONAL unit;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут significant_digits: Этот атрибут определяет число значащих чисел для конкретного прикладного компонента value_with_unit.

Атрибут unit_component: Этот атрибут определяет единицу измерений для прикладного компонента value_with_unit.

4.3.319 Прикладной компонент verification_report_for_verification_specification

Прикладной компонент verification_report_for_verification_specification определяет способ связи результатов, получаемых с помощью прикладного компонента verification_specification_allocation с прикладным компонентом verification_result. Результат работ по верификации отбирается с помощью атрибута verification_report_entry.

EXPRESS-описание:

```
*)
    ENTITY verification_report_for_verification_specificationIt;
    description : text_select;
    specific_verification_element : verification_specificationIt_allocation;
    verification_report_entry : verification_result;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту verification_report_for_verification_specification.

Атрибут specific_verification_element: Этот атрибут определяет прикладной компонент verification_specification_allocation, который используется для верификации.

Атрибут verification_report_entry: Этот атрибут определяет описание результатов верификации.

4.3.320 Прикладной компонент verification_result

Прикладной компонент verification_result определяет заключение по результатам выполнения операции верификации.

EXPRESS-описание:

```
*)
    ENTITY verification_result;
    description : text_select;
    id : element_identifier;
    name : label;
    system_under_verification : verification_specificationIt_system_view_relationship;
```

UNIQUE
UR1: id, system_under_verification;
END_ENTITY;
(*
Определения атрибутов:

Атрибут description: Этот атрибут определяет полное заключение по результатам проведения проверочной операции. Результат применения отдельного прикладного компонента verification_specification к проверяемым элементам попадает в прикладной компонент verification_report_for_verification_specification. В описании определена дополнительная информация относительно прикладного компонента verification_result.

Атрибут id: Этот атрибут определяет идентификатор прикладного компонента verification_result.

Атрибут name: Этот атрибут определяет слово (или слова), которые используются для ссылки на прикладной компонент verification_result.

Атрибут system_under_verification: Этот атрибут определяет прикладной компонент verification_specification_system_view_relationship, для которого действующим является прикладной компонент verification_result.

Формальные выражения:

UR1:

4.3.321 Прикладной компонент verification_specification

Прикладной компонент verification_specification является спецификацией на способ, с помощью которого должны проверяться система, ее функциональные возможности или физические элементы.

Примечание 1 — Прикладной компонент verification_specification использует структуру требований для определения способа испытаний системы для того, чтобы эта структура давала возможность ограничения представления только текстовой информацией. Посредством прикладного компонента model_defined_requirement_definition можно определять модели или внешние файлы, которые должны использоваться в проверочной операции.

Примечание 2 — Прикладной компонент verification_specification не подразумевает традиционного испытания, поскольку вместо него может применяться формальная верификация, анализ или инспекция системы (или любой другой метод).

EXPRESS-описание:

*)
ENTITY verification_specificationIt;
definition : requirement_occurence;
description : OPTIONAL text_select;
id : element_identifier;
name : label;
verification_method : label;
END_ENTITY;
(*
Определения атрибутов:

Атрибут definition: Этот атрибут определяет прикладной компонент requirement_definition, который должен быть протестирован.

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту verification_specification.

Атрибут id: Этот атрибут определяет идентификатор прикладного компонента verification_specification.

Атрибут name: Этот атрибут определяет слово (или слова), которые используются для ссылки на прикладной компонент verification_specification.

Атрибут verification_method: Этот атрибут определяет способ проведения испытания конкретной системы.

4.3.322 Прикладной компонент verification_specification_allocation

Прикладной компонент verification_specification_allocation определяет способ соотнесения прикладного компонента verification_specification с прикладным компонентом system_instance, physical_instance, function_instance или requirement_instance таким образом, чтобы прикладной компонент verification_specification применялся к предназначенному компоненту.

EXPRESS-описание:

```
*)
    ENTITY verification_specificationIt_allocation;
    description : OPTIONAL text_select;
    relevant_for : verification_allocation_select;
    specification : verification_specificationIt;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту verification_specification_allocation.

Атрибут relevant_for: Этот атрибут определяет прикладной компонент, с которым соотносится прикладной компонент verification_specification.

Атрибут specification: Этот атрибут определяет прикладной компонент verification_specification.

4.3.323 Прикладной компонент verification_specification_system_view_relationship

Прикладной компонент verification_specification_system_view_relationship определяет способ присвоения прикладного компонента verification_specification прикладному компоненту system_view, который указывает технические требования к испытаниям, которые должны проводиться для верификации или валидации системы.

EXPRESS-описание:

```
*)
    ENTITY verification_specificationIt_system_view_relationship;
    assigned_verification : verification_specificationIt;
    description : OPTIONAL text_select;
    index : label;
    system_view : system_view;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут assigned_verification: Этот атрибут определяет присвоенный прикладной компонент verification_specification.

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту verification_specification_system_view_relationship.

Атрибут index: Этот атрибут определяет идентификатор прикладного компонента verification_specification, идентифицируемый с помощью атрибута assigned_verification в контексте прикладного компонента system_view, который определяется с помощью атрибута system_view.

Атрибут system_view: Этот атрибут определяется присвоенным прикладным компонентом verification_specification.

4.3.324 Прикладной компонент view_relationship

Прикладной компонент view_relationship определяет иерархическую связь между двумя прикладными компонентами visual_element, при которой дочерний прикладной компонент является частью родительского прикладного компонента graphics_view.

Примечание 1 — Если прикладной компонент view_relationship существует между двумя прикладными компонентами graphics_view, то аналогичная иерархическая связь должна существовать и между прикладными компонентами general_function_definition. Прикладные компоненты general_physical_definition или fsm_generic_state входят в эту иерархическую связь.

Примечание 2 — Каждый прикладной компонент graphics_view может являться частью многих прикладных компонентов multi_level_view. Для перекрытия представлений для каждого прикладного компонента view_relationship необходимо с помощью атрибута valid_in определить прикладной компонент multi_level_view, для которого указанная связь является действующей.

EXPRESS-описание:

```
*)
    ENTITY view_relationship;
    child : graphics_view;
```

```
        parent : graphics_view;
        valid_in : multi_level_view;
        END_ENTITY;
(*
```

Определения атрибутов:

Атрибут child: Этот атрибут определяет прикладной компонент graphics_view, который накладывается в родительском прикладном компоненте graphics_view.

Атрибут parent: Этот атрибут определяет прикладной компонент graphics_view, который действует как контекст дочернего прикладного компонента graphics_view.

Атрибут valid_in: Этот атрибут определяет прикладной компонент multi_level_view, для которого указанная связь является действующей.

4.3.325 Прикладной компонент visual_element

Прикладной компонент visual_element является супертипом для всех объектов, несущих графическую информацию для прикладного элемента. Прикладным компонентом visual_element является прикладной компонент graphics_node, actual_port_position, formal_port_position или graphics_link.

EXPRESS-описание:

```
*)
        ENTITY visual_element
        ABSTRACT SUPERTYPE OF ( ONEOF(actual_port_position, formal_port_position, graphics_link,
        graphics_node) ) ;
        view : graphics_view;
END_ENTITY;
(*
```

Определение атрибута:

Атрибут view: Этот атрибут определяет прикладной компонент graphics_view, для которого задается прикладной компонент visual_element.

4.3.326 Прикладной компонент work_order

Прикладной компонент work_order определяет полномочия для одного или нескольких прикладных компонентов engineering_process_activity, которые должны выполняться.

EXPRESS-описание:

```
*)
        ENTITY work_ordert
        ABSTRACT SUPERTYPE OF ( ONEOF(change_order, start_order) );
        description : OPTIONAL text_select;
        id : element_identifier;
        is_controlling : SET[1:?] OF engineering_process_activityp;
        status : label;
        version_id : OPTIONAL element_identifier;
        END_ENTITY;
(*
```

Определения атрибутов:

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту work_order.

Атрибут id: Этот атрибут определяет идентификатор прикладного компонента work_order.

Атрибут is_controlling: Этот атрибут определяет прикладные компоненты engineering_process_activity, которые контролируются с помощью данного частного прикладного компонента work_order.

Атрибут status: Этот атрибут определяет состояние прикладного компонента work_order. Там, где это применимо, должны использоваться следующие состояния (значения) этого атрибута:

- состояние in_work: Запрос разрабатывается;
- состояние issued: Запрос завершен, просмотрен и немедленно приводит к выполнению операции;
- состояние resolved: Запрос разрешен; операция, определенная с помощью запроса, завершена, и никакой другой работы в дальнейшем не требуется.

Атрибут version_id: Этот атрибут определяет идентификатор конкретной версии прикладного компонента work_order.

4.3.327 Прикладной компонент work_request

Прикладной компонент work_request определяет требование к определенной работе, которая должна быть выполнена.

EXPRESS-описание:

```
*)
    ENTITY work_request
    ABSTRACT SUPERTYPE OF (ONEOF(change_request, start_request) ) ;
    description : text_select;
    id : element_identifier;
    notified_person : SET[0:?] OF date_and_person_organization;
    requestor : date_and_person_organization;
    scope : SET[0:?] OF specification_element_select;
    status : label;
    version_id : OPTIONAL element_identifier;
    END_ENTITY;
(*
```

Определения атрибутов:

Атрибут description: Этот атрибут определяет дополнительную информацию, относящуюся к прикладному компоненту work_request.

Атрибут id: Этот атрибут определяет идентификатор прикладного компонента work_request.

Атрибут notified_person: Этот атрибут определяет прикладной компонент person или organization, который должен давать информацию относительно прикладного компонента work_request, а также о дате, когда прикладной компонент person или organization должны получать информацию.

Атрибут requestor: Этот атрибут определяет прикладной компонент person или organization, который должен обеспечивать выдачу запроса и даты, когда этот прикладной компонент person или organization был выдан.

Атрибут scope: Этот атрибут определяет прикладные компоненты, которые зависят от прикладного компонента work_request.

Атрибут status: Этот атрибут определяет состояние прикладного компонента work_request. Там, где это применимо, должны использоваться следующие состояния (значения) этого атрибута:

- состояние in_work: Запрос разрабатывается;
- состояние issued: Запрос завершен, просмотрен и немедленно приводит к выполнению операции;
- состояние resolved: Запрос разрешен; операция, определенная с помощью запроса, завершена, и никакой другой работы в дальнейшем не требуется.

Атрибут version_id: Этот атрибут определяет идентификатор конкретной версии прикладного компонента work_request.

```
*)
    END_SCHEMA;
(*
```

4.4 Функции в ARM-модели

В данном подразделе определяются функции ARM-модели, предназначенные для PAS-спецификации.

4.4.1 Функция at_most_one_system_function_assigned

EXPRESS-описание:

```
*)
    FUNCTION at_most_one_system_function_assigned (a_system_view : system_view): LOGICAL;
    LOCAL
    no_of_system_functions : INTEGER := 0;
    END_LOCAL;
    IF SIZEOF(a_system_view.system) > 0 THEN
    REPEAT i := 1 TO SIZEOF(a_system_view.system);
    IF a_system_view.system[i].role = system_function THEN
    no_of_system_functions := no_of_system_functions + 1;
```

```
        END_IF;
        END_REPEAT;
        END_IF;
        RETURN (no_of_system_functions <= 1);
        END_FUNCTION;
(*
```

4.4.2 Функция correct_binding

EXPRESS-описание:

```
*)
        FUNCTION correct_binding (binding : io_port_binding): BOOLEAN;
        LOCAL
        function_interface : function_instance;
        END_LOCAL;
        IF ('SYSTEM_ENGINERING_AND_DESIGN.FUNCTION_INSTANCE' IN TYPEOF(binding.actu-
        al_port.port_of)) THEN
        RETURN (FALSE);
        END_IF;
        function_interface := binding.actual_port.port_of;
        IF (binding.formal_port.port_of :=: function_interface.definition) THEN
        RETURN (TRUE);
        ELSE
        RETURN (FALSE);
        END_IF;
        END_FUNCTION;
(*
```

4.4.3 Функция determineactualportrole

EXPRESS-описание:

```
*)
        FUNCTION determineactualportrole (p : actual_io_port): port_data_relation;
        IF (p.port_type = input) OR (p.port_type = control) OR (p.port_type = mechanism) THEN RETURN
        (consumer);
        ELSE
        RETURN (producer);
        END_IF;
        END_FUNCTION;
(*
```

4.4.4 Функция determineformalportrole

EXPRESS-описание:

```
*)
        FUNCTION determineformalportrole (p : formal_io_port): port_data_relation;
        IF (p.port_type = input) OR (p.port_type = control) OR (p.port_type = mechanism) THEN
        RETURN (producer);
        ELSE
        RETURN (consumer);
        END_IF;
        END_FUNCTION;
(*
```

5 Требования соответствия

Соответствие настоящему стандарту означает выполнение требований, установленных в настоящем стандарте, требований к поддерживаемому методу (методам) и соответствующих требований,

содержащихся в нормативных ссылочных документах. Их практическая реализация должна поддерживать по крайней мере один из методов, указанных в следующих стандартах:

- ИСО 10303-21;
- ИСО 10303-22.

Требования с учетом зависящих от метода практической реализации требований определены в приложении С.

Протокол практической реализации (PICS) содержит перечни опций или их комбинаций, которые могут включаться в практическую реализацию (см. приложение В).

Данная спецификация предоставляет ряд опций, которые могут поддерживаться с помощью практической реализации.

Нижеуказаные опции сгруппированы в следующие классы соответствия:

- Административная информация;
- Управление работами;
- Управление внесением изменений;
- Ссылки на документацию;
- Классификация элементов системы;
- Задание приоритетности элементов системы;
- Графическое представление информации;
- Представление требований к текстам;
- Представление требований к текстам и соответствующим понятиям;
- Формы представления структурных требований;
- Диаграммы потоков данных;
- Функциональные блок-схемы потоков;
- Диаграммы поведения систем;
- Структурный анализ систем;
- Физическая архитектура систем;
- Объектно-ориентированный анализ систем;
- Объектно-ориентированные статические структуры систем;
- Объектно-ориентированное поведение систем;
- Объектно-ориентированная практическая реализация систем;
- Представление систем;
- Верификация систем;
- Среда функциональной архитектуры и требования к ней;
- Среда физической архитектуры и требования к ней;
- Среда полного инженерного проектирования систем;
- Среда объектно-ориентированного инженерного проектирования систем;
- Среда расширенного инженерного проектирования систем.

Поддержка определенного класса соответствия требует поддержки всех опций этого класса.

Соответствие определенному классу требует поддержки всех прикладных компонентов, заданных в этом классе. Таблица соответствия прикладных компонентов классам соответствия приведена ниже.

5.1 Класс А: Административная информация

Данный класс соответствия распространяется на административную информацию и содержит бизнес-информацию, необходимую для ее включения в спецификацию.

В данный класс соответствия полностью или частично входит следующий функциональный модуль:

- System_Validation.

5.2 Класс В: Управление работами

Данный класс соответствия распространяется на управление работами и содержит информацию об обозначениях для ее включения в инженерный проект системы и соответствующие работы. Поддерживает связь содержащейся в спецификации информации с работами по инженерному проектированию систем; также позволяет фиксировать запросы на начало работ по проектированию систем.

В данный класс соответствия полностью или частично входит следующий функциональный модуль:

- Work_management.

5.3 Класс C: Управление внесением изменений

Данный класс соответствия распространяется на процесс управления внесением изменений и содержит информацию, необходимую для связи спецификации с процессом внесения изменений.

В данный класс соответствия полностью или частично входят следующие функциональные модули:

- Change_management;
- Work_management.

5.4 Класс D: Ссылки на документацию

Данный класс соответствия распространяется на ссылки на документацию и содержит информацию, необходимую для связи нестандартизованной, согласно ИСО 10303, документации с системными спецификациями. Эта документация может существовать в цифровой или бумажной форме.

В данный класс соответствия полностью или частично входит следующий функциональный модуль:

- External_document_reference_mechanism.

5.5 Класс E: Классификация элементов системы

Данный класс соответствия распространяется на классификацию элементов системы и содержит информацию, необходимую для связи элементов системной спецификации с общей системой классификации.

В данный класс соответствия полностью или частично входит следующий функциональный модуль:

- Element_classification.

5.6 Класс F: Задание приоритетности элементов

Данный класс соответствия распространяется на классификацию элементов системы и содержит информацию, необходимую для связи элементов системной спецификации с общей системой задания приоритетов.

В данный класс соответствия полностью или частично входит следующий функциональный модуль:

- Element_prioritization.

5.7 Класс G: Графическое представление информации

Данный класс соответствия распространяется на графическое представление информации и содержит информацию, необходимую для связи элементов системной спецификации с общей двухмерной системой координат с целью обеспечения визуального размещения элементов спецификации.

В данный класс соответствия полностью или частично входит следующий функциональный модуль:

- График.

5.8 Класс 01: Представление требований к текстам

Данный класс соответствия распространяется на представление требований к текстам и поддерживает представление требований, выражаемых в тексте, совокупности требований и фиксацию взаимосвязей между ними. Поддерживается также и произвольная классификация этих требований.

В данный класс соответствия полностью или частично входят следующие функциональные модули:

- Person_organization;
- Requirement_representation;
- Structured_text;
- Version_management.

5.9 Класс 02: Представление требований к текстам и соответствующим понятиям

Данный класс соответствия распространяется на представление требований к текстам и соответствующим понятиям, является расширением предыдущего класса соответствия и позволяет фиксировать элементы фунциональной спецификации, подразумеваемые требованиями к текстам.

В данный класс соответствия полностью или частично входят следующие функциональные модули:

- Functional_Hierarchy;
- Person_organization;

- Requirement_representation;
- Structured_text;
- Version_management.

5.10 Класс 03: Формы представления структурных требований

Данный класс соответствия распространяется на формы представления структурных требований, является расширением класса соответствия 01 и позволяет фиксировать требования, выражаемые как модельные или же во внешней документации.

В данный класс соответствия полностью или частично входят следующие функциональные модули:

- Explicit_Functional_reference;
- Explicit_Physical_reference;
- Functional_Allocation;
- Functional_Behaviour_basics;
- Functional_Behaviour_Causal_chain;
- Functional_Behaviour_finite_State_Machine;
- Functional_Behaviour_Interaction;
- Functional_Hierarchy;
- Functional_Performance;
- Person_organization;
- Physical_Architecture;
- Свойств;
- Requirement_representation;
- Requirement_representation_Structured_Formats;
- Structured_text;
- System_Architecture;
- Version_management.

5.11 Класс 04: Диаграммы потоков данных

Данный класс соответствия распространяется на диаграммы потоков данных, содержащий информацию, которая позволяет фиксировать функциональную структуру разделения и функциональное взаимодействие в функциональной спецификации системы.

В данный класс соответствия полностью или частично входят следующие функциональные модули:

- Functional_Behaviour_Interaction;
- Functional_Hierarchy;
- Measurement_Unit;
- Person_organization;
- Structured_text;
- Version_management.

5.12 Класс 05: Функциональные блок-схемы потоков

Данный класс соответствия распространяется на функциональные блок-схемы потоков, содержит информацию, которая позволяет фиксировать функциональную структуру разделения и причинно-следственные связи между функциями в функциональной спецификации системы.

В данный класс соответствия полностью или частично входят следующие функциональные модули:

- Functional_Behaviour_basics;
- Functional_Behaviour_Causal_chain;
- Functional_Hierarchy;
- Measurement_Unit;
- Person_organization;
- Structured_text;
- Version_management.

5.13 Класс 06: Диаграммы поведения систем

Данный класс соответствия распространяется на диаграммы поведения систем и объединяет классы соответствия 04 и 05.

В данный класс соответствия полностью или частично входят следующие функциональные модули:
- Functional_Behaviour_basics;
- Functional_Behaviour_Causal_chain;
- Functional_Behaviour_Interaction;
- Functional_Hierarchy;
- Measurement_Unit;
- Person_organization;
- Structured_text;
- Version_management.

5.14 Класс 07: Структурный анализ систем

Данный класс соответствия распространяется на структурный анализ систем, является расширением класса соответствия 04 и поддерживает представление функционального поведения в модели автомата с конечным числом состояний.

В данный класс соответствия полностью или частично входят следующие функциональные модули:
- Functional_Behaviour_basics;
- Functional_Behaviour_finite_State_Machine;
- Functional_Behaviour_Interaction;
- Functional_Hierarchy;
- Measurement_Unit;
- Person_organization;
- Structured_text;
- Version_management.

5.15 Класс 08: Физическая архитектура систем

Данный класс соответствия распространяется на физическую архитектуру систем, содержит информацию, которая позволяет фиксировать абстрактное представление физического вида системы согласно спецификации.

В данный класс соответствия полностью или частично входят следующие функциональные модули:
- Person_organization;
- Physical_Architecture;
- Structured_text;
- Version_management.

5.16 Класс 09: Объектно-ориентированный анализ систем

Данный класс соответствия распространяется на объектно-ориентированный анализ систем и является расширением класса соответствия 11 путем предоставления примеров использования системы согласно спецификации.

В данный класс соответствия полностью или частично входят следующие функциональные модули:
- OO_Behaviour;
- OO_Common;
- OO_Use_Case;
- Person_organization;
- Structured_text;
- Version_management.

5.17 Класс 10: Объектно-ориентированные статические структуры систем

Данный класс соответствия распространяется на объектно-ориентированные статические структуры систем, содержит информацию, которая позволяет фиксировать объектно-ориентированное представление статических свойств системы согласно спецификации, и включает в себя информацию, необходимую для представления диаграмм классов и блоков.

В данный класс соответствия полностью или частично входят следующие функциональные модули:
- OO_Common;
- OO_Static_Structure;
- Person_organization;

- Relationship_Cardinality;
- Structured_text;
- Version_management.

5.18 Класс 11: Объектно-ориентированное поведение систем

Данный класс соответствия распространяется на объектно-ориентированное поведение систем, обеспечивает поддержку объектно-ориентированого представления поведения, выражаемую в форме диаграмм взаимодействия и кооперирования в системе согласно спецификации.

В данный класс соответствия полностью или частично входят следующие функциональные модули:

- OO_Behaviour;
- OO_Common;
- Person_organization;
- Structured_text;
- Version_management.

5.19 Класс 12: Объектно-ориентированная практическая реализация систем

Данный класс соответствия распространяется на объектно-ориентированную практическую реализацию систем и обеспечивает поддержку фиксации объектно-ориентированного представления реализации программного обеспечения в виде диаграмм компонентов и размещения.

В данный класс соответствия полностью или частично входят следующие функциональные модули:

- OO_Common;
- OO_Implementation;
- Person_organization;
- Structured_text;
- Version_management.

5.20 Класс 13: Представление систем

Данный класс соответствия распространяется на представление систем, содержит информацию, которая позволяет группировать набор спецификаций со взаимосвязанными обозначениями и представлять системы в структурах систем.

В данный класс соответствия полностью или частично входят следующие функциональные модули:

- Person_organization;
- Relationship_Cardinality;
- Structured_text;
- System_Architecture;
- Version_management.

5.21 Класс 14: Верификация систем

Данный класс соответствия распространяется на верификацию систем, содержит информацию, которая позволяет представлять требования к валидации и верификации системы согласно спецификации.

В данный класс соответствия полностью или частично входят следующие функциональные модули:

- Person_organization;
- Requirement_representation;
- Structured_text;
- System_Architecture;
- System_Verification;
- Version_management.

5.22 Класс 15: Среда функциональной архитектуры и требования к ней

Данный класс соответствия распространяется на среду для функциональной архитектуры и требования к ней и сочетает в себе требования классов соответствия 03, 06 и 07, а также поддерживает распределение требований по функциональным характеристикам.

В данный класс соответствия полностью или частично входят следующие функциональные модули:

- Explicit_Functional_reference;

- Functional_Behaviour_basics;
- Functional_Behaviour_Causal_chain;
- Functional_Behaviour_finite_State_Machine;
- Functional_Behaviour_Interaction;
- Functional_Hierarchy;
- Functional_Performance;
- Measurement_Unit;
- Person_organization;
- Relationship_Cardinality;
- Requirement_Allocation;
- Requirement_representation;
- Requirement_representation_Implied_Functionality;
- Requirement_representation_Structured_Formats;
- Structured_text;
- System_Architecture;
- Version_management.

5.23 Класс 16: Среда физической архитектуры и требования к ней

Данный класс соответствия распространяется на среду физической архитектуры и требования к ней; объединяет в себе требования классов соответствия 03 и 08, а также поддерживает распределение требований по элементам физической архитектуры системы.

В данный класс соответствия полностью или частично входят следующие функциональные модули:

- Explicit_Physical_reference;
- Person_organization;
- Physical_Architecture;
- Relationship_Cardinality;
- Requirement_Allocation;
- Requirement_representation;
- Requirement_representation_Implied_Functionality;
- Requirement_representation_Structured_Formats;
- Structured_text;
- System_Architecture;
- Version_management.

5.24 Класс 17: Среда полного инженерного проектирования систем

Данный класс соответствия распространяется на среду полного инженерного проектирования систем и сочетает в себе требования классов соответствия 15 и 16.

В данный класс соответствия полностью или частично входят следующие функциональные модули:

- Explicit_Functional_reference;
- Explicit_Physical_reference;
- Functional_Allocation;
- Functional_Behaviour_basics;
- Functional_Behaviour_Causal_chain;
- Functional_Behaviour_finite_State_Machine;
- Functional_Behaviour_Interaction;
- Functional_Hierarchy;
- Functional_Performance;
- Measurement_Unit;
- Person_organization;
- Physical_Architecture;
- Relationship_Cardinality;
- Requirement_Allocation;
- Requirement_representation;

- Requirement_representation_Implied_Functionality;
- Requirement_representation_Structured_Formats;
- Structured_text;
- System_Architecture;
- System_Verification;
- Version_management.

5.25 Класс 18: Среда объектно-ориентированного инженерного проектирования систем

Данный класс соответствия распространяется на среду объектно-ориентированного инженерного проектирования систем и сочетает в себе требования классов соответствия 01, 09–11.

В данный класс соответствия полностью или частично входят следующие функциональные модули:

- OO_Behaviour;
- OO_Common;
- OO_Implementation;
- OO_Static_Structure;
- OO_Use_Case;
- Person_organization;
- Relationship_Cardinality;
- Requirement_Allocation;
- Requirement_representation;
- Structured_text;
- System_Architecture;
- Version_management.

5.26 Класс 19: Среда расширенного инженерного проектирования систем

Данный класс соответствия распространяется на среду расширенного инженерного проектирования систем и сочетает в себе требования классов соответствия 17 и 18.

В данный класс соответствия полностью или частично входят следующие функциональные модули:

- Explicit_Functional_reference;
- Explicit_Physical_reference;
- Functional_Allocation;
- Functional_Behaviour_basics;
- Functional_Behaviour_Causal_chain;
- Functional_Behaviour_finite_State_Machine;
- Functional_Behaviour_Interaction;
- Functional_Hierarchy;
- Functional_Performance;
- Measurement_Unit;
- OO_Behaviour;
- OO_Common;
- OO_Implementation;
- OO_Static_Structure;
- OO_Use_Case;
- Person_organization;
- Physical_Architecture;
- Relationship_Cardinality;
- Requirement_Allocation;
- Requirement_representation;
- Requirement_representation_Implied_Functionality;
- Requirement_representation_Structured_Formats;
- Structured_text;
- System_Architecture;
- System_Verification;
- Version_management.

Таблица. Соответствия прикладных компонентов классам

Прикладной компонент	Класс соответствия																									
	A	B	C	D	E	F	G	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
abstract_data_type_definition										X	X		X	X								X		X		X
abstract_data_type_member										X	X		X	X								X		X		X
actual_io_port										X	X		X	X								X		X		X
actual_physical_port										X					X								X	X		X
actual_port_position							X																			
address								X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
aggregate_data_type_ definition										X	X		X	X								X		X		X
approval	X																									
approval_assignment	X																									
approval_person_ organization	X																									
approval_relationship	X																									
assessment	X																									
assessment_relationship	X																									
bi_directional_port_indicator										X	X		X	X								X		X		X
binary_data_type_definition										X	X		X	X								X		X		X
boolean_data_type_definition											X															
cardinality_list																	X			X		X	X	X	X	X
cardinality_ranget																	X			X		X	X	X	X	X
causal_block_bound										X		X	X									X		X		X
cb_completion_alternative												X														
cb_completion_alternative_ mapping												X														
cb_firing_condition										X		X	X									X		X		X

Продолжение таблицы

Прикладной компонент	Класс соответствия																								
	A	B	C	D	E	F	G	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
cb_functional_behaviour_ mode										X																
cb_functional_behaviour_ model												X	X									X		X		X
cb_functional_place										X		X	X									X		X		X
cb_functional_transition										X		X	X									X		X		X
cb_initial_marking										X		X	X									X		X		X
cb_input_relationship										X		X	X									X		X		X
cb_output_relationship										X		X	X									X		X		X
cb_place_function_ association										X		X	X									X		X		X
cb_place_reference										X		X	X									X		X		X
cb_transition										X		X	X									X		X		X
cb_transition_relationship										X		X	X									X		X		X
cb_transition_unbounded_ weight										X		X	X									X		X		X
change_order			X																							
change_order_relationship			X																							
change_report			X																							
change_report_element_ assignment			X																							
change_request			X																							
clock										X												X		X		X
clock_assignment_ relationship										X												X		X		X
context_relationship										X														X		X

Продолжение таблицы

Прикладной компонент	Класс соответствия																								
	A	B	C	D	E	F	G	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
complex_data_type_definition										X	X		X	X								X		X		X
complex_value										X	X		X	X								X		X		X
compound_value										X	X		X	X								X		X		X
configuration_element								X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
configuration_element_ relationship								X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
configuration_element_ version								X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
configuration_element_ version_ relationship								X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
context_function_relationship										X										X	X	X	X	X	X	X
context_physical_relationship										X										X	X	X	X	X	X	X
control_io_port										X	X		X	X								X		X		X
coordinate_translation_ information							X																			
critical_issue			X																							
critical_issue_impact			X																							
data_field										X	X		X	X								X		X		X
data_instance										X	X		X	X								X		X		X
data_transfer																						X	X	X		X
date_and_person_ assignment								X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
date_and_person_ organization								X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
date_assignment								X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
date_time								X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Продолжение таблицы

Прикладной компонент	Класс соответствия																									
	A	B	C	D	E	F	G	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
derived_data_type_definition										X	X		X	X								X		X		X
digital_document				X																						
document_assignment				X																						
documentation_reference				X																						
documentation_relationship				X																						
effectiveness_measure								X	X	X											X	X	X	X	X	X
effectiveness_measure_ assignment								X	X	X											X	X	X	X	X	X
effectiveness_measure_ relationship								X	X	X											X	X	X	X	X	X
effectivity		X	X																							
effectivity_assignment		X	X																							
effectivity_relationship		X	X																							
element_ critical_issue_ relationship			X																							
element_identifier								X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
elementary_maths_space										X	X		X	X								X		X		X
engineering_process_activity		X	X																							
engineering_process_ activity_ element_assignment		X	X																							
engineering_process_ activity_ relationship		X	X																							
event_data_type_definition										X	X		X	X								X		X		X
execution_time										X												X		X		X
finite_integer_interval										X	X		X	X								X		X		X
finite_real_interval										X	X		X	X								X		X		X

Продолжение таблицы

Прикладной компонент	Класс соответствия																								
	A	B	C	D	E	F	G	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
finite_space										X	X		X	X								X		X		X
formal_data_interaction_port										X				X								X		X		X
formal_io_port										X	X		X	X								X		X		X
formal_physical_port										X					X								X	X		X
formal_port_position							X																			
fsm_and_state										X				X								X		X		X
fsm_command_interaction_ relationship										X				X								X		X		X
fsm_data_interaction_binding										X				X								X		X		X
fsm_data_interaction_ relationship										X				X								X		X		X
fsm_generic_state										X				X								X		X		X
fsm_initial_state_transition										X				X								X		X		X
fsm_mode										X																
fsm_model														X								X		X		X
fsm_or_state										X				X								X		X		X
fsm_state										X				X								X		X		X
fsm_state_composition_ relationship										X				X								X		X		X
fsm_state_transition										X				X								X		X		X
fsm_transient_state										X				X								X		X		X
fsm_transient_state_ composition_ relationship														X												
function_instance									X	X	X	X	X	X								X		X		X
function_reference										X				X								X		X		X

Продолжение таблицы

Прикладной компонент	Класс соответствия																								
	A	B	C	D	E	F	G	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
functional_behaviour_mode										X																
functional_behaviour_model												X	X	X								X		X		X
functional_behaviour_model_ assignment										X		X	X	X								X		X		X
functional_decomposition_ relationship									X	X	X	X	X	X								X		X		X
functional_link										X	X		X	X								X		X		X
functional_link_allocation_ relationship										X														X		X
functional_link_group										X	X		X	X								X		X		X
functional_link_reference										X												X		X		X
functional_reference_ configuration										X												X		X		X
functional_representation_ relationship										X					X								X	X		X
functional_state_context										X				X								X		X		X
functionality_allocation_ relationship										X														X		X
functionality_instance_ reference										X												X		X		X
functionality_reference_ composition_relationship										X												X		X		X
functionality_reference_ relationship										X												X		X		X
general_function_definition									X	X	X	X	X	X								X		X		X
general_functionality_ instance									X	X	X	X	X	X								X		X		X
general_physical_definition										X					X								X	X		X

Продолжение таблицы

Прикладной компонент	Класс соответствия																								
	A	B	C	D	E	F	G	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
generic_state_context										X				X								X		X		X
graphics_link							X																			
graphics_node							X																			
graphics_point							X																			
graphics_view							X																			
hibound_integer_interval										X	X		X	X								X		X		X
hibound_real_interval										X	X		X	X								X		X		X
implied_external_interaction																						X	X	X		X
infinite_cardinality																	X			X		X	X	X	X	X
initial_ state_transition_ specification_ assignment										X				X								X		X		X
integer_data_type_definition										X	X		X	X								X		X		X
integer_interval										X	X		X	X								X		X		X
io_buffer										X	X		X	X								X		X		X
io_composition_port										X	X		X	X								X		X		X
io_port										X	X		X	X								X		X		X
io_port_binding										X	X		X	X								X		X		X
io_split_join									X	X	X	X	X	X								X		X		X
issue_source_relationship			X																							
issue_system_assignment			X																							
justification	X																									
justification_relationship	X																									
leaf_function_definition									X	X	X	X	X	X								X		X		X
lobound_integer_interval										X	X		X	X								X		X		X

Продолжение таблицы

Прикладной компонент	Класс соответствия																								
	A	B	C	D	E	F	G	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
lobound_real_interval										X	X		X	X								X		X		X
logical_data_type_definition										X	X		X	X								X		X		X
maths_space										X	X		X	X								X		X		X
model_defined_ requirement_ definition										X												X	X	X		X
multi_level_view							X																			
name_binding										X				X								X		X		X
nominal_valuet										X																
non_digital_document				X																						
oo_action																X		X							X	X
oo_action_staten																X		X							X	X
oo_action_staten_transition																X		X							X	X
oo_action_temporal_ relationship																X		X							X	X
oo_actor																X									X	X
oo_argument																	X								X	X
oo_association																	X								X	X
oo_association_class																	X								X	X
oo_association_end																	X								X	X
oo_association_end_ classifier_ relationship																	X								X	X
oo_association_end_ qualifier_ association																	X								X	X
oo_association_end_role																X		X							X	X
oo_association_role																X		X							X	X

Продолжение таблицы

Прикладной компонент	Класс соответствия																								
	A	B	C	D	E	F	G	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
oo_attributen																	X								X	X
oo_attributen_instance																	X								X	X
oo_attributen_link_end_ associatione																X		X							X	X
oo_behavioural_featuree																	X								X	X
oo_call_action																X		X							X	X
oo_class																	X								X	X
oo_classifier_rolen																X		X							X	X
oo_collaboration																X		X							X	X
oo_component																			X						X	X
oo_component_allocation																			X						X	X
oo_constraint																X	X	X	X						X	X
oo_constraint_model_ element_ relationship																X	X	X	X						X	X
oo_create_action																X		X							X	X
oo_dependency																X	X	X	X						X	X
oo_element_import																X	X	X	X						X	X
oo_element_residencen																			X						X	X
oo_extension																X									X	X
oo_extension_point																X									X	X
oo_generalization																X	X	X	X						X	X
oo_generic_association																	X									
oo_generic_association_end																	X									
oo_inclusion																X									X	X

Продолжение таблицы

Прикладной компонент	Класс соответствия																								
	A	B	C	D	E	F	G	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
oo_instance_classifier_ relationship																X	X	X	X						X	X
oo_interaction																X		X							X	X
oo_interfaces																	X								X	X
oo_link																X		X							X	X
oo_link_end																X		X							X	X
oo_message																X		X							X	X
oo_message_temporal_ relationship																X		X							X	X
oo_method																	X								X	X
oo_model_element_ stereotype_ relationship																X	X	X	X						X	X
oo_model_element_tagged_ value_relationship																X	X	X	X						X	X
oo_object																	X								X	X
oo_operation																	X								X	X
oo_operation_interface_ association																	X								X	X
oo_package																	X								X	X
oo_parametere																	X								X	X
oo_reception																X		X							X	X
oo_send_action																X		X							X	X
oo_signal																X		X							X	X
oo_signal_behavioural_ feature_ relationship																X		X							X	X
oo_stereotype																X	X	X	X						X	X

Продолжение таблицы

Прикладной компонент	Класс соответствия																								
	A	B	C	D	E	F	G	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
oo_stimulus																X		X							X	X
oo_stimulus_argument																X		X							X	X
oo_tagged_valuee																X	X	X	X						X	X
oo_use_case																X									X	X
oo_view																X	X	X	X						X	X
oo_view_context_element_ relationship																X	X	X	X						X	X
oo_ view_relationship																X	X	X	X						X	X
oo_view_system_ view_ relationship																X	X	X	X						X	X
organization								X	X		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
organization_relationship								X	X		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
package					X																				
package_classification_ assignment					X																				
package_classification_ system					X																				
package_element_ assignment					X																				
package_hierarchy_ relationship					X																				
partial_ document_ assignment				X																					
partial_ system_view																				X	X	X	X	X	X	X
partial_system_ view_ relationship																				X	X	X	X	X	X	X
persistent_storage									X		X	X	X	X								X		X		X

Продолжение таблицы

Прикладной компонент	Класс соответствия																								
	A	B	C	D	E	F	G	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
persistent_storage_ equivalence_ relationship																						X		X		X
persistent_storage_reference																						X		X		X
person								X	X		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
person_in_organization								X	X		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
person_organization_ assignment								X	X		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
physical_binding															X								X	X		X
physical_composition_ relationship															X								X	X		X
physical_connection															X								X	X		X
physical_instance															X								X	X		X
physical_instance_reference																							X	X		X
physical_link_definition															X								X	X		X
physical_node_definition															X								X	X		X
physical_port															X								X	X		X
physical_reference_ configuration																							X	X		X
physical_reference_ relationship																							X	X		X
project		X	X																							
project_event_reference		X	X																							
project_relationshipe		X	X																							
rank_assignment						X																				
rank_group						X																				
rank_relation						X																				

Продолжение таблицы

Прикладной компонент	Класс соответствия																								
	A	B	C	D	E	F	G	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
ranking_element						X																				
ranking_system						X																				
real_data_type_definition											X		X	X								X		X		X
real_interval											X		X	X								X		X		X
realized_systemp																					X			X		X
realized_ system_ composition_ relationship																					X			X		X
record_data_type_definition											X		X	X								X		X		X
record_data_type_member											X		X	X								X		X		X
recursive_data_type_definition											X		X	X								X		X		X
requirement_allocation_ relationship																						X	X	X	X	X
requirement_class								X	X												X	X	X	X	X	X
requirement_class_ relationships								X	X												X	X	X	X	X	X
requirement_composition_ relationship								X	X												X	X	X	X	X	X
requirement_definition								X	X												X	X	X	X	X	X
requirement_instance								X	X												X	X	X	X	X	X
requirement_occurence								X	X												X	X	X	X	X	X
requirement_relationship								X	X												X	X	X	X	X	X
requirement_relationship_ context_assignment																				X	X	X	X	X	X	X
requirement_relationship_ input_ assignment								X	X												X	X	X	X	X	X
requirement_relationship_ resulting_relationship								X	X												X	X	X	X	X	X

Продолжение таблицы

Прикладной компонент	Класс соответствия																								
	A	B	C	D	E	F	G	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
requirement_ requirement_ class _ assignment								X	X												X	X	X	X	X	X
requirement_ system_view_ assignment																				X	X	X	X	X	X	X
requirement_traces_to_ requirement_ relationship								X	X												X	X	X	X	X	X
root_ requirement_ system_ view_assignment																				X	X	X	X	X	X	X
selection_package					X																					
single_cardinality																	X			X		X	X	X	X	X
specific_ requirement_ allocation_ relationship																						X	X	X	X	X
specification_state_ assignment														X								X		X		X
start_order		X	X																							
start_request		X	X																							
state_context_relationship														X								X		X		X
state_function_interaction_ port														X								X		X		X
state_machine_ functional_ behaviour_model														X								X		X		X
state_transition_ specification_ assignment														X								X		X		X
string_data_type_definition											X		X	X								X		X		X
structured_ requirement_ definition																						X	X	X		X
system_composition_ relationship																				X	X	X	X	X	X	X
system_definition																				X	X	X	X	X	X	X

Продолжение таблицы

Прикладной компонент	Класс соответствия																								
	A	B	C	D	E	F	G	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
system_functional_ configuration																						X		X		X
s system_instance																				X	X	X	X	X	X	X
s system_instance_ relationship																				X	X	X	X	X	X	X
s system_instance_ relationship _ port																				X	X	X	X	X	X	X
s system_instance_ replication_ relationship																				X	X	X	X	X	X	X
system_physical_ configuration																							X	X		X
system_substitution_ relationship																				X	X	X	X	X	X	X
system_view																				X	X	X	X	X	X	X
system_view_assignment																				X	X	X	X	X	X	X
system_view_context																				X	X	X	X	X	X	X
textual_paragraphn								X	X		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
textual_ requirement_ definition								X	X												X	X	X	X	X	X
textual_section								X	X		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
textual_specification								X	X		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
textual_table								X	X		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
triggered_system_ view_ relationship																				X	X	X	X	X	X	X
undefined_data_type_ definition											X		X	X								X		X		X
union_data_type_definition											X		X	X								X		X		X
union_data_type_definition											X		X	X								X		X		X

Окончание таблицы

Прикладной компонент	Класс соответствия																									
	A	B	C	D	E	F	G	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
union_data_type_member											X		X	X								X		X		X
unit											X	X	X	X								X		X		X
user_defined_data_type_ definition											X		X	X								X		X		X
verification_report_for_ verification_specificationIt																					X			X		X
verification_result																					X			X		X
verification_specificationIt																					X			X		X
verification_specificationIt_ allocation																					X			X		X
verification_specificationIt_ system_relationship																				X	X	X	X	X	X	X
view_relationship							X																			
visual_element							X																			
work_ordert		X	X																							
work_request		X	X																							

Приложение А
(обязательное)

Требования, зависящие от метода практической реализации

Метод практической реализации определяет те режимы обмена данными, которые требуются с точки зрения настоящего стандарта, соответствие с которым должно осуществляться в структуре обмена данными. Формат файлов должен кодироваться в соответствии с синтаксисом и преобразованием в языке EXPRESS, определенными в ИСО 10303-21 или ИСО 10303-22, а I-ARM-модель определена в Приложении F настоящего стандарта. Заголовок выражения для обмена данными должен идентифицировать применение настоящего стандарта с помощью имени схемы system_engineering_and_design.

Приложение В
(обязательное)

Формуляр для утверждения протокола соответствия практической реализации системы (PICS)

В данном приложении перечислены дополнительные элементы PAS-спецификации, которые могут выбираться для получения любых сочетаний из них, однако определенные комбинации опций, вероятно, должны применяться совместно (они называются классами соответствия и приведены в данном приложении).

Данное приложение выполнено в виде формуляра, который должен заполняться специалистом по внедрению системы и использоваться испытательной лабораторией при подготовке проверки на совместимость. Заполненный формуляр называется PICS формуляром.

В.1 Идентификатор протокола практической реализации системы

Наименование реализации системы согласно настоящему стандарту	
Текущая версия и дата выпуска	

В.2 Метод практической реализации системы

Укажите выбранный метод реализации системы и поддерживаемые направления перевода.

При использовании нескольких методов реализации необходимо заполнять для каждого из них отдельный PICS-формуляр.

Метод практической реализации	Препроцессор	Постпроцессор
ИСО 10303—21		
ИСО 10303—22		

В.3 Используемые классы соответствия

Классы соответствия	Препроцессор	Постпроцессор
Административная информация (CCA)		
Управление работами (CCB)		
Управление внесением изменений (CCC)		
Ссылки на документацию (CCD)		
Классификация элементов системы (CCE)		
Задание приоритетности элементов системы (CCF)		
Графическое представление информации (CCG)		
Представление требований к текстам (CC01)		
Представление требований к текстам и соответствующим понятиям (CC02)		
Формы представления структурных требований (CC03)		
Диаграммы потоков данных (CC04)		
Функциональные блок-схемы потоков (CC05)		
Диаграммы поведения систем (CC06)		
Структурный анализ систем (CC07)		
Физическая архитектура систем (CC08)		

Классы соответствия	Препроцессор	Постпроцессор
Объектно-ориентированный анализ систем (CC09)		
Объектно-ориентированные статические структуры систем (CC10)		
Объектно-ориентированное поведение систем (CC11)		
Объектно-ориентированная практическая реализация систем (CC12)		
Представление систем (CC13)		
Верификация систем (CC14)		
Среда функциональной архитектуры и требования к ней (CC15)		
Среда физической архитектуры и требования к ней (CC16)		
Среда полного инженерного проектирования систем (CC17)		
Среда объектно-ориентированного инженерного проектирования систем (CC18)		
Среда расширенного инженерного проектирования систем (CC19)		

Приложение С
(обязательное)

Регистрация информационных объектов

С.1 Идентификация документов

Для обеспечения однозначной идентификации информационного объекта в открытой системе настоящему стандарту присваивается следующий объектный идентификатор:

{iso standard 20542 version(1)}

Смысл этого идентификатора определен в ИСО/МЭК 8824-1 и подробно описан в ИСО 10303-1.

С.2 Идентификация схем

Для обеспечения однозначной идентификации спецификаций схем system_engineering_and_design_arm schema (приложение F) в открытой системе настоящему стандарту присваиваются следующие объектные идентификаторы:

{iso standard 20542 version(1) object(1) system-engineering-and-design-arm(1)}

Смысл этих идентификаторов определен в ИСО/МЭК 8824-1 и подробно описан в ИСО 10303-1.

Приложение D
(справочное)

Модель функционирования приложения

Модель функционирования приложения (AAM-модель) предназначена для облегчения понимания области применения данного прикладного протокола и требований к содержащейся в нем информации. Эта модель представлена в виде нескольких рисунков, содержащих диаграммы операций, их описаний и данных. Операции и информационные потоки, которые не рассматриваются в настоящем стандарте, помечены звездочкой (*).

Делается предположение о том, что крупное предприятие может состоять из нескольких организаций и будет отвечать за полный жизненный цикл системы. При этом специалисты по системам управления являются сотрудниками этого предприятия.

D.1 Определения вспомогательных операций (моделей функционирования) приложения

D.1.1 Операция Accept model information (Прием информации о модели)

Позволяет подтверждать пригодность и состояние всей информации, необходимой для проведения анализа компромиссных решений.

D.1.2 Операция accept Additional model information (Прием дополнительной информации о модели)

Позволяет получать информацию, которая доступна из какого-либо внешнего источника информации, с целью повышения качества содержания инженерной модели системы.

Примечание — Подобная информация может подвергаться управлению и поэтому вместе со всей другой информацией будет исходить из соответствующего процесса и становиться частью процесса управления проектом.

D.1.3 Операция agreement feedback (Обратная связь при согласовании операций)

Позволяет обеспечивать обратную связь, переводящую результаты выполнения операций в предприятии в соответствующие процессы согласования.

D.1.4 Операция Analyze context (Анализ контекста)

Позволяет выполнять комплексный процесс для создания набора моделей и рассмотрения контекста объекта в рамках соответствующего класса систем.

D.1.5 Операция Analyze subject (Анализ объекта)

Позволяет выполнять комплексный процесс для создания набора моделей и рассмотрения объекта в рамках соответствующего класса систем.

D.1.6 Операция Assess available information (Оценка доступной информации)

Позволяет подготавливать базовую информацию, необходимую для инженерного моделирования системы.

Примечание — Эта информация может включать в себя:
- требования к тексту;
- информацию о наследовании;
- пользовательскую информацию;
- принципы работы в виде текста и
- исходные модели.

D.1.7 Операция Attribute value (Значение атрибута)

Позволяет определять состояние (величину) атрибута, указывающее на определенную характеристику системы, которая соответствует предлагаемой модели системы.

D.1.8 Операция Authorized engineering change (Внесение разрешенных изменений)

Позволяет определять изменение, имеющее полученное разрешение на применение одного или нескольких технических элементов при создании или эксплуатации системы.

Примечание — Техническими элементами могут быть инженерные модели системы, технические задания на разработку, системы или их элементы.

D.1.9 Операция Authorized project action (Разрешенная проектная операция)

Позволяет определять операцию, которая допускается на основе разрешения, полученного в соответствии с определенными полномочиями в проекте.

D.1.10 Операция Behaviour model (Модель поведения)

Позволяет определять модель, характеризующую все функции контекста или объекта в классе системы, а также распределение и входные/выходные параметры этих функций.

D.1.11 Операция Build solution model (Модель построения решения)

Позволяет идентифицировать показатели эффективности в качестве основы для компромисса и создавать модели поведения и структуры.

D.1.12 Операция Business reality (Реальные условия ведения бизнеса)

Позволяет определять коммерческие факторы, включающие в себя наличие ресурсов, риски влияния и период от начала разработки продукции до выхода его на рынок.

D.1.13 Операция By-product (Промежуточный продукт)

Позволяет определять элемент, который будет прирастать в процессе производства, эксплуатации или технического обслуживания системы и в дальнейшем больше не будет играть полезную роль в жизненном цикле системы.

D.1.14 Операция Calculate system performance (Расчет функциональных характеристик системы)

Позволяет использовать полученные состояния (значения) атрибутов для получения комплексных функциональных характеристик системы с помощью соответствующего набора требований.

D.1.15 Операция Change proposal (Изменения в коммерческом предложении)

Позволяет определять предложение, в котором будут идентифицироваться характеристики изменения какого-либо технического элемента при создании, эксплуатации, техническом обслуживании и утилизации системы.

Примечание — Коллектив разработчиков проекта будет рассматривать нетехнические последствия изменения предложения перед тем, как разрешение этого изменения станет реальным. Объем и формальная сторона этого рассмотрения будет зависеть от ожидаемого влияния этого изменения (в общем случае это влияние будет увеличиваться в процессе жизненного цикла системы). Основные принципы организации, предприятия или проекта будут определять критерии оценки изменения предложения.

D.1.16 Операция Chosen solution (Выбранное решение)

Позволяет определять информацию о модели, которая является оптимальной спецификацией системы.

Примечание — Показатели эффективности являются основой для выбора одного этого решения из набора допустимых вариантов.

D.1.17 Операция compare Effectiveness of feasible solutions (Эффективность допустимых решений)

Позволяет определять категорию всех допустимых решений на основе соответствующих показателей эффективности, а также оптимальное решение.

D.1.18 Операция Component for integration (Компонент для объединения

Позволяет определять компонент, который готов к объединению с разрабатываемой системой.

D.1.19 Операция Component tier model (Модель компонентов в системе)

Позволяет определять набор, содержащий контекстную и объектную модели, характеризующие данный компонент в системе.

D.1.20 Операция Concept tier model (Модель понятий в системе)

Позволяет определять набор разработанных моделей, содержащий контекстную и объектную модели, а также характеризующий представление принципов системы.

D.1.21 Операция Context model set (Набор контекстных моделей)

Позволяет определять набор, содержащий поведенческую и структурную модели, а также план реализации контекста объекта в классе системы.

D.1.22 Операция Control engineering change (Контроль технических изменений)

Позволяет анализировать техническое влияние исследованных вопросов и формировать обратную связь для инициализации процессов контроля проекта с целью оценки влияния на него стоимости, рисков и графика выполнения работ.

Примечание — Операция change proposal* является основой для активации операции manage project с целью просмотра таких нетехнических показателей проекта, как стоимость, риски и график выполнения работ.

D.1.23 Операция Create behaviour model (Формирование модели поведения системы)

Позволяет определять либо стимулирующее воздействие, либо реакцию на него для объекта в представляющей интерес системе.

D.1.24 Операция Create model for component tie (Формирование модели для класса компонента)

Позволяет выполнять процесс общего инженерного моделирования систем для получения спецификации на требования к элементарной части системы или модели класса подсистемы.

D.1.25 Операция Create model for concept tier (Формирование модели для класса компонента)

Позволяет выполнять процесс общего инженерного моделирования систем для получения спецификации на требования, направляемые в организацию заказчика.

Примечание — При этом моделировании создаются требования, которые часто называются требованиями пользователя или заинтересованной стороны (контекстное представление) и системные требования (объектное представление). Указанные требования могут представляться в виде документа, содержащего требования пользователя (URD), и документа, содержащего системные требования (SRD).

D.1.26 Операция Create model for domain tier (Формирование модели для класса области)

Позволяет выполнять процесс общего инженерного моделирования систем для получения спецификации, которая может требоваться многим организациям заказчика в данной отрасли.

Пример — При этом моделировании создается описание проблем, которые могут, возможно, возникать в нескольких отдельно приобретаемых взаимодействующих системах, например при изучении области воздушного транспорта, в которой самолеты взаимодействуют с аэропортами и системами управления воздушным движением.

D.1.27 Операция Create model for sub-system tier (Формирование модели для класса подсистем)

Позволяет выполнять процесс общего инженерного моделирования систем для получения спецификации, которая содержит требования к элементарной части системы или модели класса подсистемы более высокого уровня.

D.1.28 Операция Create model for system tier (Формирование модели для класса систем)

Позволяет выполнять процесс общего инженерного моделирования систем для получения спецификации, которая содержит набор моделей понятий как источник требований.

Примечание — При этом моделировании создается то, что обычно называется проектированием архитектуры системы.

D.1.29 Операция Create plan for build & test (Формирование программы создания и проверки моделей)

Позволяет рассматривать управление техническими вопросами с целью установления графика, пригодного для выработки решения относительно моделей спецификации.

D.1.30 Операция Create structure model (Формирование модели структуры)

Позволяет идентифицировать логическое и физическое содержание спецификации.

D.1.31 Операция Create systems engineering model (Формирование инженерной модели системы)

Позволяет разрабатывать модель для каждого уровня системы в качестве основы для структурированного набора спецификации.

D.1.32 Операция Define effectiveness measure (Определение меры эффективности)

Позволяет устанавливать критерий, с помощью которого можно судить об альтернативных модельных спецификациях.

D.1.33 Операция Define system (Определение системы)

Позволяет создавать спецификации последовательных классов (уровней) систем вплоть до отдельных компонентов.

D.1.34 Операция Demand on system capability (Требования к возможностям системы)

Позволяет изменять то, что происходит с рабочей средой системы и изменять ее реакцию.

D.1.35 Операция Design for system component (Проектирование для компонентов системы)

Позволяет проектировать компоненты, включая структурные и поведенческие аспекты для получения достаточной точности, при которой компонент может производиться для его встраивания в интегрированную систему.

D.1.36 Операция Design system component (Проектирование для компонентов системы)

Позволяет проектировать компоненты, которые будут обеспечивать достаточную детализацию, а также его производство и испытание.

D.1.37 Операция Determine attribute value (Определение значения атрибута)

Позволяет получать на приемлемом уровне точности значение системного атрибута, который значим для анализа компромиссных решений.

Примечание — Процесс определения включает в себя:

- оценку атрибута;
- его имитацию (также известную как моделирование или эмуляция);
- измерение по макетам (требующее изготовления подходящих прототипов программного обеспечения или аппаратной реализации).

D.1.38 Операция Developed model set (Разработка набора моделей)

Позволяет устанавливать набор, который содержит контекстную и объектную модель на уровне системы.

Пример — Разработанная модель может описывать:

- ***требования пользователя;***
- ***системные требования;***
- ***проект архитектуры системы.***

D.1.39 Операция Effectiveness measure (Мера эффективности)

Позволяет определять требования, которые настолько важны, что могут оказывать определяющее влияние на успех или неуспех реализации системы.

Пример — При инженерном проектировании ноутбука двумя стандартными требованиями к нему (критериями оптимизации) являются вес и срок службы аккумулятора. Проектировщик должен стремиться к снижению веса ноутбука и повышению срока службы аккумулятора.

D.1.40 Операция Enabling system

Позволяет создавать систему, которая будет дополнять представляющую интерес систему для процессов с одним или несколькими циклами эксплуатации.

Примечание — Например, разработка enabling system необходима для изготовления и интеграции системы. Каждая допускаемая система обладает индивидуальным жизненным циклом.

D.1.41 Операция Enterprise feedback (Обратная связь на предприятии)

Позволяет осуществлять обратную связь, переводящую результаты работ по проектированию на более широкий уровень на этом предприятии.

D.1.42 Операция Enterprise policy or procedure (Метод или процедура на предприятии)

Позволяет контролировать основу для проведения технических или управленческих работ по проектам в рамках предприятия.

Примечание — Метод или процедура, используемые на предприятии, являются дополняющими все значимые для организации методы, процедуры, социальные, юридические или политические предписания.

D.1.43 Операция Enterprise (Предприятие)

Позволяет определять виртуальную организацию, которая появляется в результате одного или нескольких соглашений между двумя или несколькими организациями и несет ответственность за жизненный цикл системы.

D.1.44 Операция Existing system or element (Существующая система или элемент)

Позволяет определять компонент, который доступен для другого предприятия и может повторно использоваться в другой системе.

D.1.45 Операция Feasible solution (Допустимое решение)

Позволяет получать модельную информацию, которая определяет решение, отвечающее соответствующим требованиям.

D.1.46 Операция Implementation plan (План реализации системы)

Позволяет получать план, который будет определять порядок построения и испытания системы.

D.1.47 Операция Information from external source (Информация из внешнего источника)

Позволяет получать информацию, поступающую из любого источника вне предприятия.

D.1.48 Операция Initial model set (Исходный набор моделей)

Позволяет получать модельную информацию, доступную в начале процесса инженерного проектирования системы.

Примечание — В общем случае процесс инженерного проектирования системы является достаточно гибким. Порядок подобного моделирования может быть нисходящим, восходящим, возвратно-поступательным или от середины процесса. Для простых систем различие между уровнями требует минимального количества формального контроля проекта, причем каждый проектировщих может параллельно работать на нескольких уровнях.

D.1.49 Операция Install system (Инсталляция системы)

Позволяет принимать проверенную систему и устанавливать ее в рабочую среду.

D.1.50 Операция Integrate component into system (Интеграция компонента в систему)

Позволяет принимать компонент и вводить его в соответствующее место системы.

D.1.51 ИСО 15288

Инженерное проектирование систем — Процессы в жизненном цикле системы.

D.1.52 ИСО 15288

Инженерное проектирование систем — Процессы в жизненном цикле системы.

D.1.53 ИСО 9001

Системы управления качеством — Требования.

D.1.54 ИСО 9001

Системы управления качеством — Требования.

D.1.55 Операция Make decision (Принятие решений)

Позволяет выбирать наиболее выигрышное направление для проведения проектировочных работ (при наличии альтернативных вариантов).

Примечание — Проектная обратная связь дает информацию относительно способов выявления существующих проблем и возможностей и приводит к требованию принятия соответствующих проектных решений.

D.1.56 Операция Manage enterprise (Управление предприятием)

Позволяет устанавливать, обеспечивать функционирование и поддерживать работу предприятия согласно концепции, с помощью которой инициализировались, поддерживались и контролировались проекты.

D.1.57 Операция Manage project (Управление проектом)

Позволяет проводить надлежащее планирование и контроль проекта и связанных с ним факторов.

Примечание — Этими факторами являются риски, конфигурация и управление проектом.

D.1.58 Операция Manage risk, configuration & information (Управление рисками, конфигурацией и информацией, связанными с проектом)

Позволяет минимизировать влияние недостоверных событий, устанавливать и поддерживать целостность всех результатов проекта и предоставлять важную своевременную, полностью достоверную и конфиденциальную информацию предприятию и его внешним партнерам.

D.1.59 Операция Map of functions (Отображение функций)

Позволяет устанавливать связи каждого структурного элемента с его поведением.

D.1.60 Операция Model set (Набор моделей)

Позволяет получать информацию, которая будет содержать все инженерные модели предлагаемых систем на каком-либо системном уровне.

D.1.61 Операция Modelling database (Моделирование базы данных)

Позволяет получать информацию, которая будет содержать все находящиеся на сегодняшний день в работе инженерные модели систем.

D.1.62 Операция Organization policy or procedure (Методы и процедуры, используемые организацией)

Позволяет контролировать бизнес-работы, проводимые организацией — членом системного предприятия, и требовать отчета ему.

Примечание — Методы и процедуры, используемые организацией, являются дополнящими все соответствующие социальные, юридические или политические предписания.

D.1.63 Операция Organization (Организация)

Позволяет определять структурную бизнес-единицу, содержащую взаимодействующие между собой персонал, инфраструктуру и объекты.

Примечание — Например, организацией может быть:
- компания;
- государственное учреждение.

D.1.64 Другие стандарты

Определяет стандарт, который не является ИСО 9001 или ИСО 15288 и который можно получить в какой-либо организации по стандартизации.

Пример — Нижеперечисленные стандарты относятся к инженерному проектированию систем:
- EIA 632 «Процессы проектирования системы»;
- IEEE 1220 «Стандарт на применение и управление процессом инженерного проектирования системы».

D.1.65 Операция Perform trade-off analysis (Выполнение анализа компромиссных решений)

Позволяет выбирать в моделях альтернативные решения.

D.1.66 Операция Plan, assess & control project (Планирование, оценка и контроль проекта)

Позволяет устанавливать и развивать программы проектирования, давать оценку реальным достижениям и ходу проектирования относительно программ, а также контролировать выполнение проекта вплоть до его окончания.

D.1.67 Операция Produce system component (Производство компонента системы)

Позволяет реализовывать и испытывать компонент системы в соответствии с применимым проектом.

D.1.68 Операция Produce system (Изготовление системы)

Позволяет реализовывать и последовательно испытывать систему на различных уровнях в соответствии с применимой спецификацией.

D.1.69 Операция Project decision (Принятие проектных решений)

Позволяет выбирать оптимальное проектное решение, установленное согласно заданному набору критериев.

D.1.70 Операция Project definition (Определение проекта)

Позволяет определять, что включает в себя узловое представление проекта для предприятия и является основой для активизации членов предприятия с целью продолжения выполнения проекта.

Примечание — Определение проекта включает в себя:
- подотчетность и полномочия по проекту;
- ожидаемые конечные результаты выполнения проекта;
- распределение проектных ресурсов;
- требования к отчетности по проекту;
- промежуточные отчетные этапы проекта.

D.1.71 Операция Project feedback (Проектная обратная связь)

Позволяет определять проектную обратную связь, которая передает результаты технических работ по проекту его менеджерам.

D.1.72 Операция Project policy or procedure (Метод или процедура проектирования)

Позволяет контролировать основу, на которой выполняются технические процессы по проекту.

Примечание — Метод или процедура проектирования являются дополняющими все значимые для предприятия или организации методы, процедуры, социальные, юридические или политические предписания.

D.1.73 Операция Raw material or consumable (Сырье или расходные материалы)

Позволяет определять объекты, которые производственные процессы будут преобразовывать для создания составной части системы.

D.1.74 Операция Realized system or sub-system (Реализованная система или подсистема)

Позволяет определять систему или подсистему, которые завершены и готовы к проверочным испытаниям.

D.1.75 Операция Recycled raw material (Утилизация материалов)

Позволяет определять элемент, который уже однажды был частью системы или был необходим для работы, поддержания или утилизации системы и готов к превращению в исходное сырье для каких-либо последующих процессов.

D.1.76 Операция Register of risk (Регистрация рисков)

Позволяет определять информацию, которая будет идентифицировать и определять риски, связанные с проектом.

D.1.77 Операция Rejected system or element (Отбраковка системы или элемента)

Позволяет определять элемент, который не отвечает применимым к нему критериям испытаний и требует проведения корректирующих операций или утилизации.

D.1.78 Операция Requirements database (База данных для требований)

Позволяет определять объем информации, которая описывает основу для создания системы.

Пример — Эти требования могут включать в себя следующее:

- ***текстовые описания;***
- ***принципы работы системы;***
- ***информацию о наследовании свойств (признаков);***
- ***информацию о пользователях.***

D.1.79 Операция Satisfied demand on system capability (Удовлетворенное требование к функциональным характеристикам системы)

Позволяет определять изменение, которое происходит в рабочей среде системы в результате эксплуатации системы.

D.1.80 Операция Set of component requirements (Установление требований к компонентам)

Позволяет устанавливать содержание поведения системы и ее структурной модели для компонента этой системы.

D.1.81 Операция Social, legal or political directive (Социальные, юридические или политические директивы)

Позволяет определять директиву, которая обеспечивает контроль проекта в дополнение к непосредственному контролю, осуществляемому системным предприятием или составляющими его организациями.

D.1.82 Операция Society or organization feedback (Обратная связь со стороны общества или организации)

Позволяет определять обратную связь, которая будет объединять предприятие с расширенным промышленным или социальным сообществом.

Примечание — Подобная обратная связь может включать в себя:

- реакции на требования предоставления информации либо организациям — членам предприятия, либо любым другим внешним заинтересованным сторонам;
- запросы на предоставление информации от внешних заинтересованных сторон.

D.1.83 Операция Specification for existing system or element (Спецификация на существующую систему или элемент)

Позволяет определять спецификацию, в которой приводятся характеристики элемента, который будет доступен для других предприятий.

D.1.84 Операция Structure model (Структурная модель)

Позволяет определять модель, которая характеризует статическую структуру контекста или объекта на каком-либо уровне системы.

Примечание — Структурная модель — это не модель для структурного анализа, которая указывает на зависимость деформации от напряжения под нагрузкой на механическую систему или физический компонент.

D.1.85 Операция Sub-system for integration (Подсистема для интеграции)

Позволяет определять подсистему, которая готова для ее интеграции с расширяемой системой.

D.1.86 Операция System assurance evidence (Свидетельство гарантии системы)

Позволяет определять доказательства поддержки и проверки ожидаемых характеристик системы и ее безопасности, которые являются основой, на которой приобретатель системы может принимать ее в соответствии с условиями договора с соответствующим предприятием.

D.1.87 Операция System for installation (Система, подготовленная для ее установки)

Позволяет определять систему, которая полностью проверена и готова к ее установке в производственную среду.

D.1.88 Операция System for validation (Система, подготовленная для ее валидации)

Позволяет определять систему, которая уже установлена в производственную среду и готова к аттестационным испытаниям.

D.1.89 Операция System or element for re-use (Система или элемент, подготовленные для их повторного использования)

Позволяет определять объект, который является результатом инженерного проектирования системы и готов к повторному использованию в системе другого предприятия.

D.1.90 Операция System ready for exploitation (Система, подготовленная для ее эксплуатации)

Позволяет определять систему, которая была проверена и готова к эксплуатации.

D.1.91 Операция Technical issue (Техническая проблема)

Позволяет определять техническую проблему, которая связана с каким-либо техническим элементом при создании, эксплуатации, техническом обслуживании или утилизации системы.

D.1.92 Операция Trade-off result (Результат анализа компромиссных решений)

Позволяет определять результат, который будет указывать основу, на которой будут строиться отдельные возможности системы или решения по выбору элементов системы, отвечающие или не отвечающие установленным показателям эффективности.

D.1.93 Операция Transfer of system responsibility (Передача ответственности за систему)

Позволяет определять обозначение, которое будет указывать на успешную установку системы в производственной среде и передавать ответственность за систему назначенному оператору.

D.1.94 Операция Validate produced system (Проверка созданной системы)

Позволяет предоставлять объективные свидетельства того, что функциональные возможности системы при ее эксплуатации будут отвечать требованиям заинтересованных сторон.

D.1.95 Операция Verify system (Проверка системы)

Позволяет демонстрировать, что характеристики и поведение системы или ее элементов отвечают соответствующей спецификации на проектирование архитектуры системы.

D.1.96 Операция Waste material (Отходы производства)

Позволяет определять объекты, которые не отвечают условиям, необходимым для их полезного применения.

D.1.97 Операция Agreement (Соглашение)

Позволяет производить взаимное оповещение о сроках и условиях осуществления рабочих взаимоотношений между партнерами.

D.1.98 Операция Establish enterprise agreement (Установление соглашения между предприятиями)

Позволяет достигать соглашения между двумя организациями, одна из которых будет выполнять роль приобретателя, а другая — роль поставщика с тем, чтобы он производил снабжение продукцией или предоставление услуг.

D.1.99 Операция Execute project (Выполнение проекта)

Позволяет выполнять технические процессы, характеризующие последовательные этапы жизненного цикла системы.

D.1.100 Операция Exploit system (Эксплуатация системы)

Позволяет эксплуатировать, поддерживать и утилизировать систему таким образом, чтобы удовлетворять требованиям к функциональным характеристикам системы в течение согласованного срока службы системы.

D.1.101 Операция Multi-project management aid (Помощь в управлении несколькими проектами)

Позволяет оказывать помощь в управлении несколькими проектами на предприятии и поддерживать общий подход к методам управления.

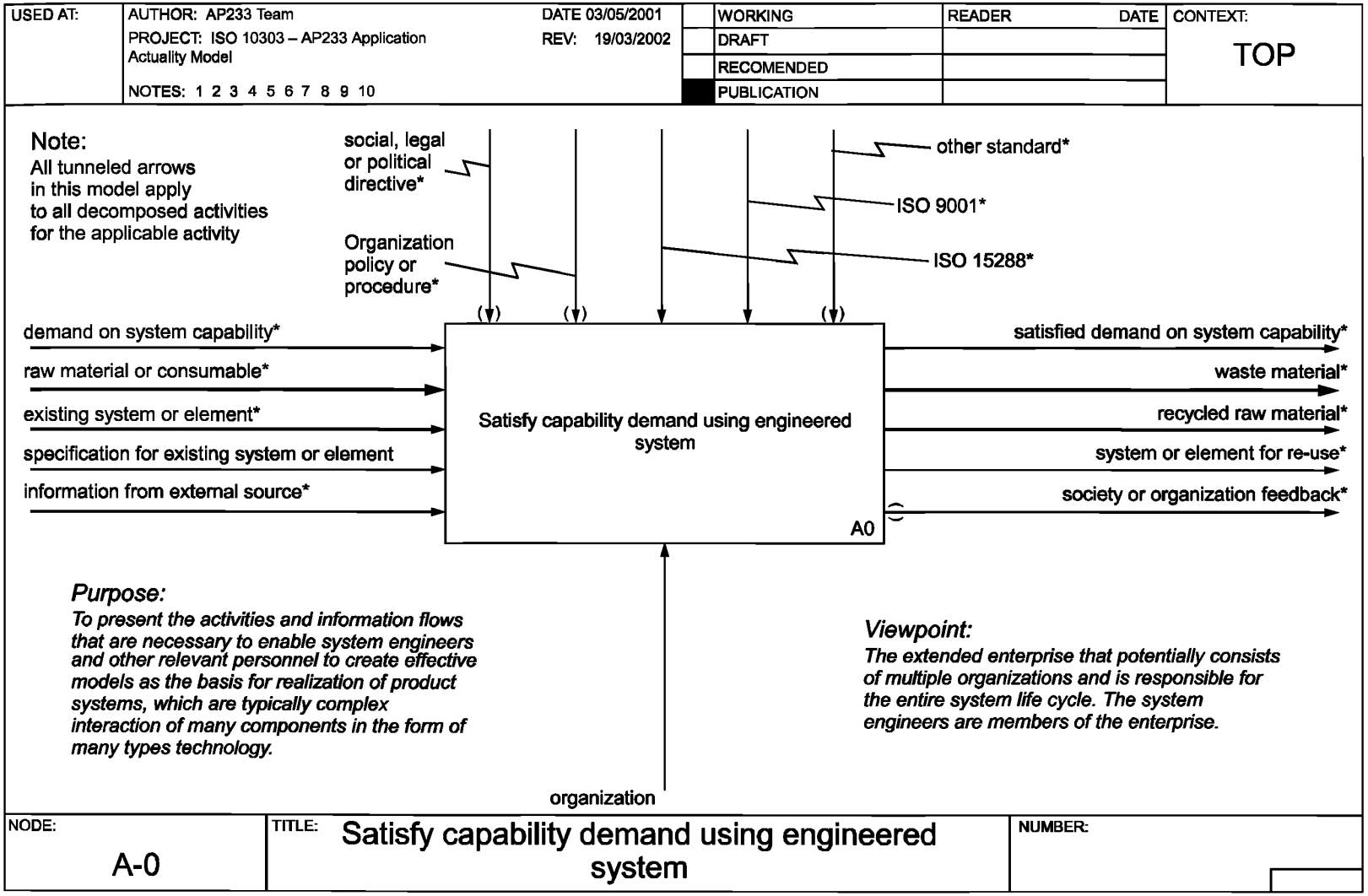
D.1.102 Операция Satisfy capability demand using engineered system (Удовлетворение требований к функциональным характеристикам системы с помощью спроектированной системы)

Позволяет сводить воедино несколько огранизаций для образования укрупненного предприятия с целью создания, эксплуатации и утилизации системы, обеспечивающей заданные функциональные характеристики.

D.2 Операция Application activity model diagrams (Модельные диаграммы прикладных операций)

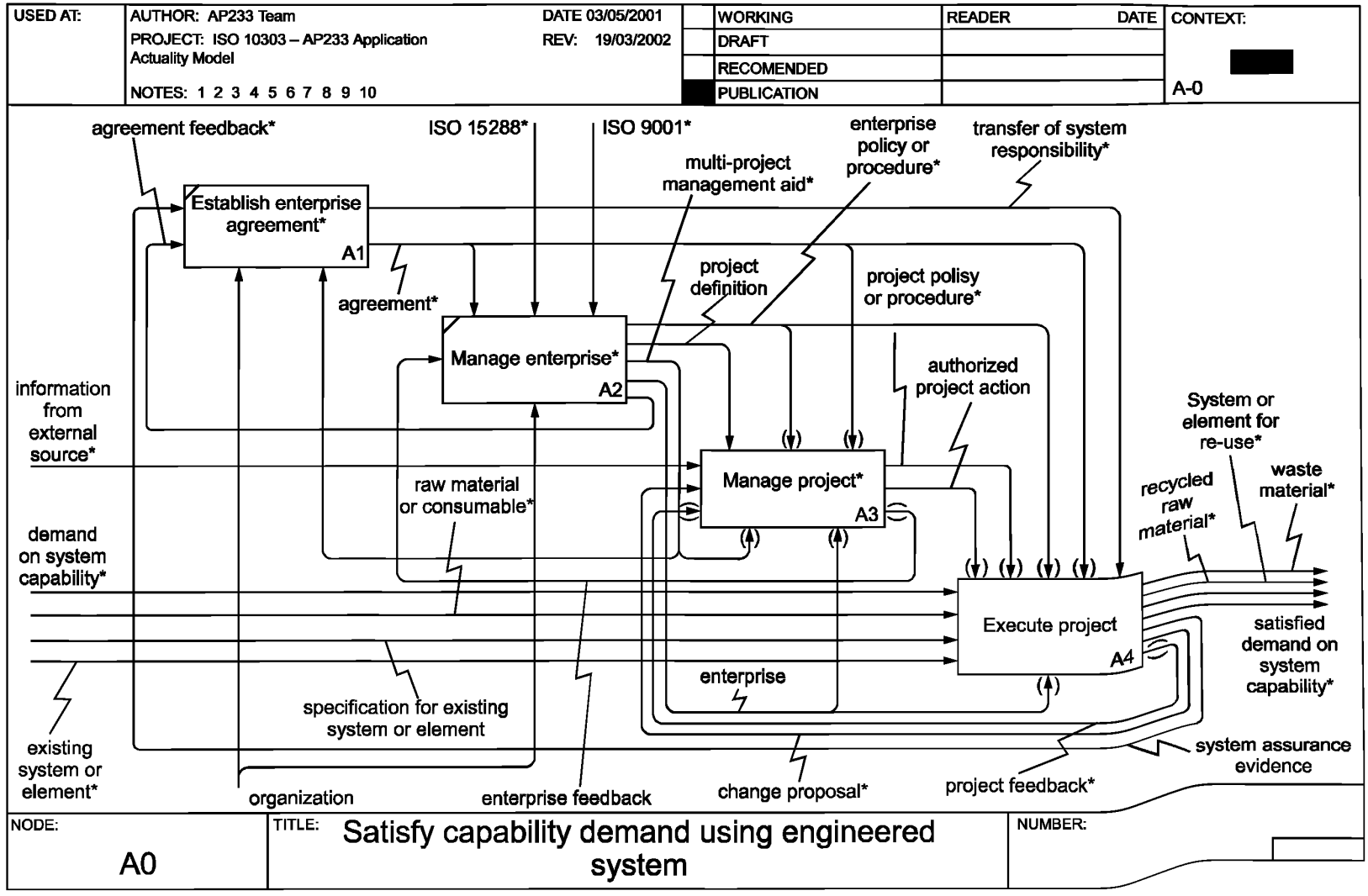
AAM-модель иллюстрируется рисунками D.1–D.11. Операции и потоки данных на этих рисунках, которые в настоящем стандарте не рассматриваются, помечены звездочкой*.

D.1–A-0 Выполнение требований к функциональным характеристикам системы с использованием спроектированной системы



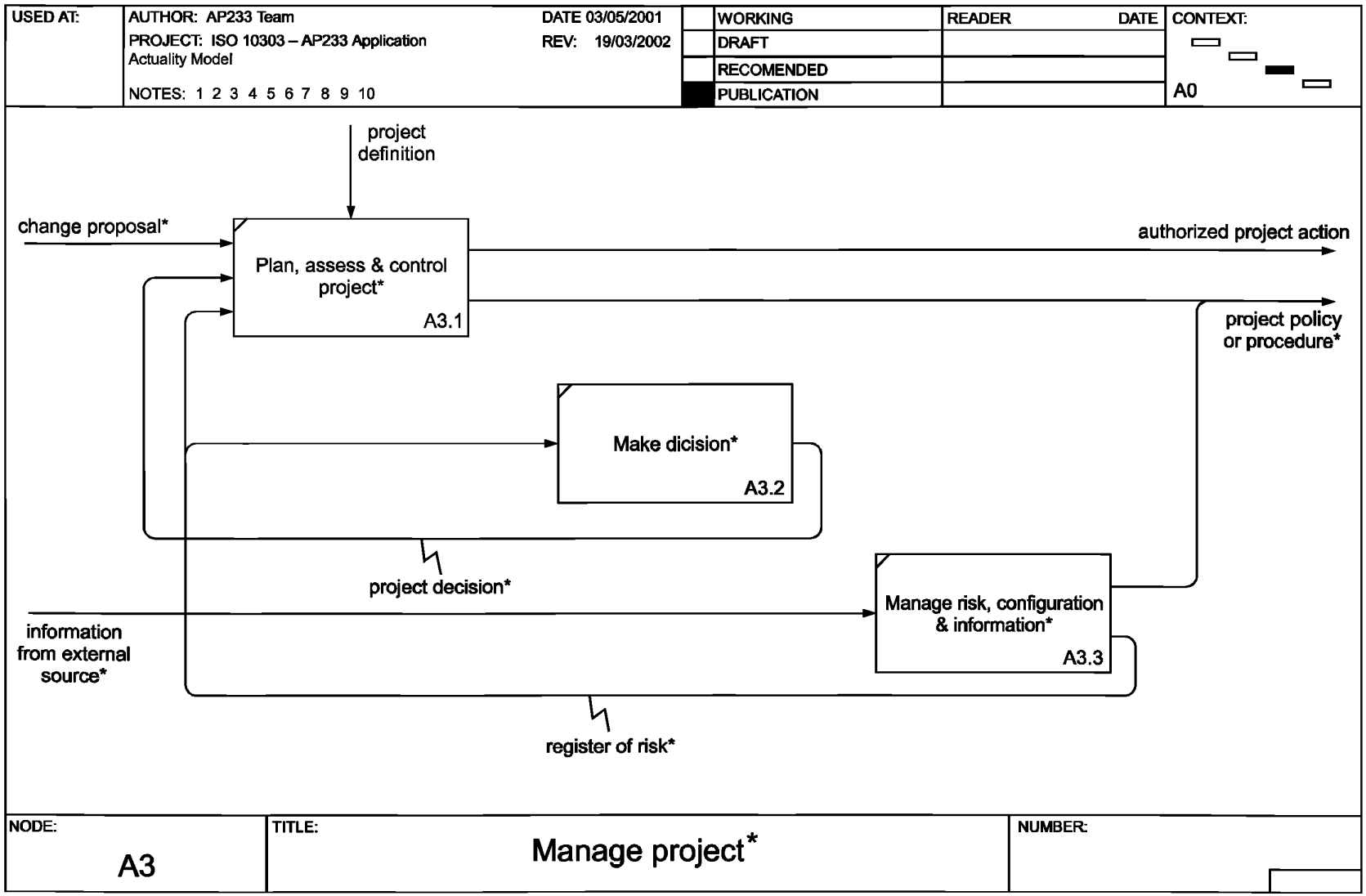
Used at — ИСПОЛЬЗОВАНО ПРИ:; Author — АВТОР:; Project — ПРОЕКТ: Activity model — Модель действия; Notes — ПРИМЕЧАНИЯ:; Date — ДАТА:; Revision — ВАРИАНТ:; Working — Работа; Draft — ЧЕРТЕЖ; Recommended — РЕКОМЕНДОВАНО; Publication — ПУБЛИКАЦИЯ; Reader — ЧИТАТЕЛЬ; Date — ДАТА; Context — КОНТЕКСТ:; TOP — ВЕРХ; Note: All tunneled arrows in this model apply to all decomposed activities for the applicable activity — ПРИМЕЧАНИЕ: Ломаные стрелки на данной модели относятся ко всем анализируемым работам в данном приложении; demand on system capability* — Требование к функциональным возможностям системы; raw material or consumable* — Необработанные и расходные материалы; existing system or element* — Существующая система или элемент; specification for existing system or element — Спецификация на существующую систему или элемент; information from external source* — Информация от внешних источников; Purpose: To present the activities and information flows that are necessary to enable system engineers and other relevant personnel to create effective models as the basis for realization of product systems, which are typically complex interaction of many components in the form of many types technology — НАЗНАЧЕНИЕ: Для предоставления работ и информационных потоков, которые необходимы для системных инженеров и другого привлекаемого персонала с целью создания эффективных моделей в качестве основы для реализации производственных систем, которые обычно имеют сложное взаимодействие между любыми компонентами в виде любого типа технологии; social, legal or political directive* — Социальное, правовое или политическое указание; Organization policy or procedure* — Организационный принцип или процедура; Satisfy capability demand using engineered system — Удовлетворение требований к функциональным возможностям с использованием спроектированной системы; other standard* — Другой стандарт; Satisfied demand on system capability* — Выполненное требование к функциональным возможностям системы; Waste material* — Отходы производства; Recycled raw material* — Повторно используемый материал; System or element for re-use* — Система или материал для повторного использования; Society or organization feedback* — Обратная связь с обществом или организациями*; Viewpoint: The extended enterprise that potentially consists of multiple organizations and is responsible for the entire system life cycle. The system engineers are members of the enterprise — ПОЗИЦИЯ: Расширенное предприятие, которое может состоять из нескольких организаций и отвечать за полный жизненный цикл системы. Системные инженеры являются сотрудниками этого предприятия; Organization — Организация; Node — ЭЛЕМЕНТ; Title — НАИМЕНОВАНИЕ; Number — НОМЕР.

D.2–A0 D.1–A-0 Выполнение требований к D.1 — A-0 Выполнение требований к функциональным характеристикам системы с использованием спроектированной системы



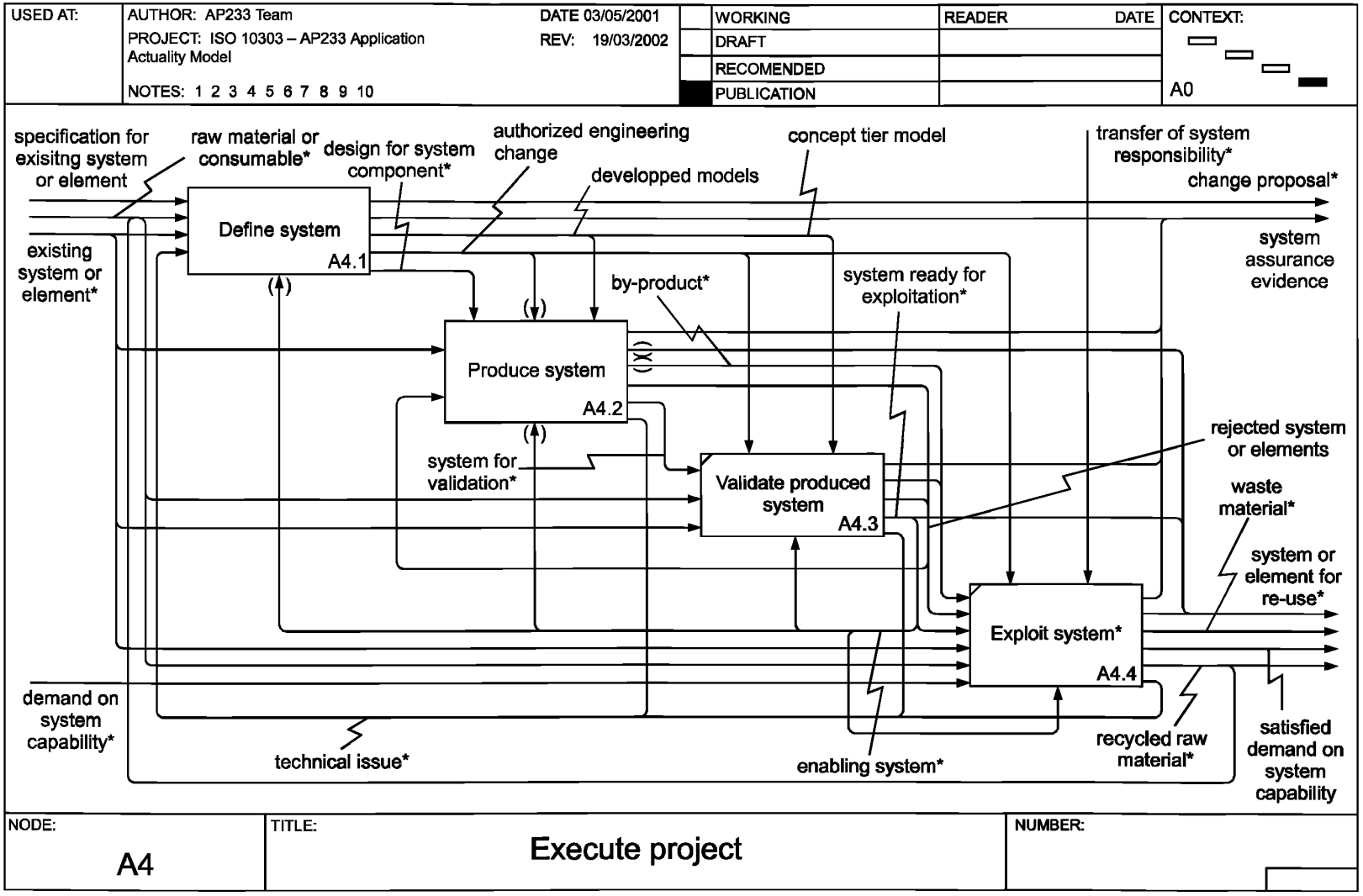
Used at — ИСПОЛЬЗОВАНО ПРИ; Author — АВТОР:; Project — ПРОЕКТ: Activity model — Модель действия; Notes — ПРИМЕЧАНИЯ:; Date — ДАТА:; Revision — ВАРИАНТ:; Working — Работа; Draft — ЧЕРТЕЖ; Recommended — РЕКОМЕНДОВАНО; Publication — ПУБЛИКАЦИЯ; Reader — ЧИТАТЕЛЬ; Date — ДАТА; Context — КОНТЕКСТ:; TOP — ВЕРХ; Agreement feedback* — Договорная обратная связь*; Information from external source — Информация из внешнего источника; Demand on system capability* — Требование к функциональным возможностям системы*; Existing system or element* — Существующая система или элемент*; Establish enterprise agreement* — Установление соглашения с предприятием; Organization — Организация; Agreement* — Соглашение*; Specification for existing system or element — Спецификация на существующую систему или элемент; Manage enterprise* — Управление предприятием*; Raw material or consumable* — Необработанные и расходные материалы*; Enterprise feedback — Обратная связь в предпрятии; Multi-project management aid* — Содействие упрвлению многими проектами; Project definition — Определение проекта; Manage project* — Управление проектом; Enterprise — Предприятие; Change proposal* — Изменение коммерческого предложения; Enterprise policy or procedure* — Организационный принцип или процедура*; Authorized project action — Утвержденная проектная операция; Execute project — Выполнение проекта; Enterprise feedback — Обратная связь в предпрятии; Transfer of system responsibility* — Передача ответственности за систему*; Waste material* - Отходы производства*; System assurance evidence — Подтверждение гарантии системы; System or element for re-use — Система или материал для повторного использования; Satisfied demand on system capability* — Удовлетворенное требование к функциональным возможностям системы*; Organization - Организация; Node — ЭЛЕМЕНТ; Title — НАИМЕНОВАНИЕ; Satisfy capability demand using engineered system - Удовлетворение требований к функциональным возможностям с использованием спроектированной системы; Number — НОМЕР.

D.3–A3 Процесс управления проектом



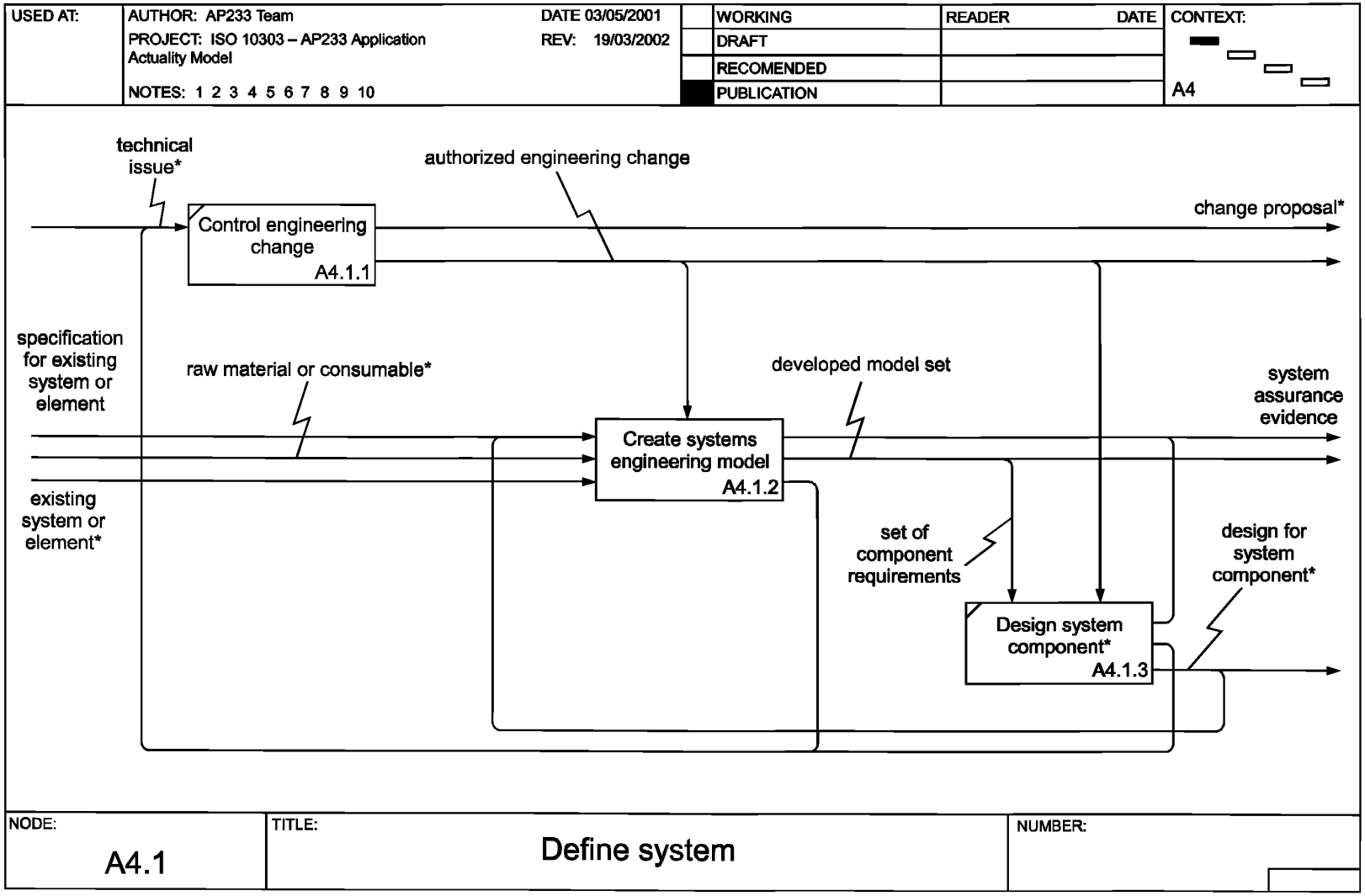
Used at — ИСПОЛЬЗОВАНО ПРИ; Author — АВТОР:; Project — ПРОЕКТ: Activity model — Модель действия; Notes — ПРИМЕЧАНИЯ:; Date — ДАТА:; Revision — ВАРИАНТ:; Working — Работа; Draft — ЧЕРТЕЖ; Recommended — РЕКОМЕНДОВАНО; Publication — ПУБЛИКАЦИЯ; Reader — ЧИТАТЕЛЬ; Date — ДАТА; Context — КОНТЕКСТ; Change proposal* — Предложение об изменении *; Information from external source* — Информация из внешнего источника; Plan, assess & control project* — Планирование, оценка и контроль проекта*; Project decision* — Проектное решение*; Project definition — Определение проекта; Register of risk* — Регистрация риска*; Make dicision* — Принятие решения*; Manage risk, configuration & information* — Управление рисками, конфигурирование и информирование; Project policy or procedure* — Проектный принцип или процедура*; Node - ЭЛЕМЕНТ; Title — НАИМЕНОВАНИЕ; Manage project — Управление проектом; Number — НОМЕР.

D.4–A4 Процесс выполнения проекта



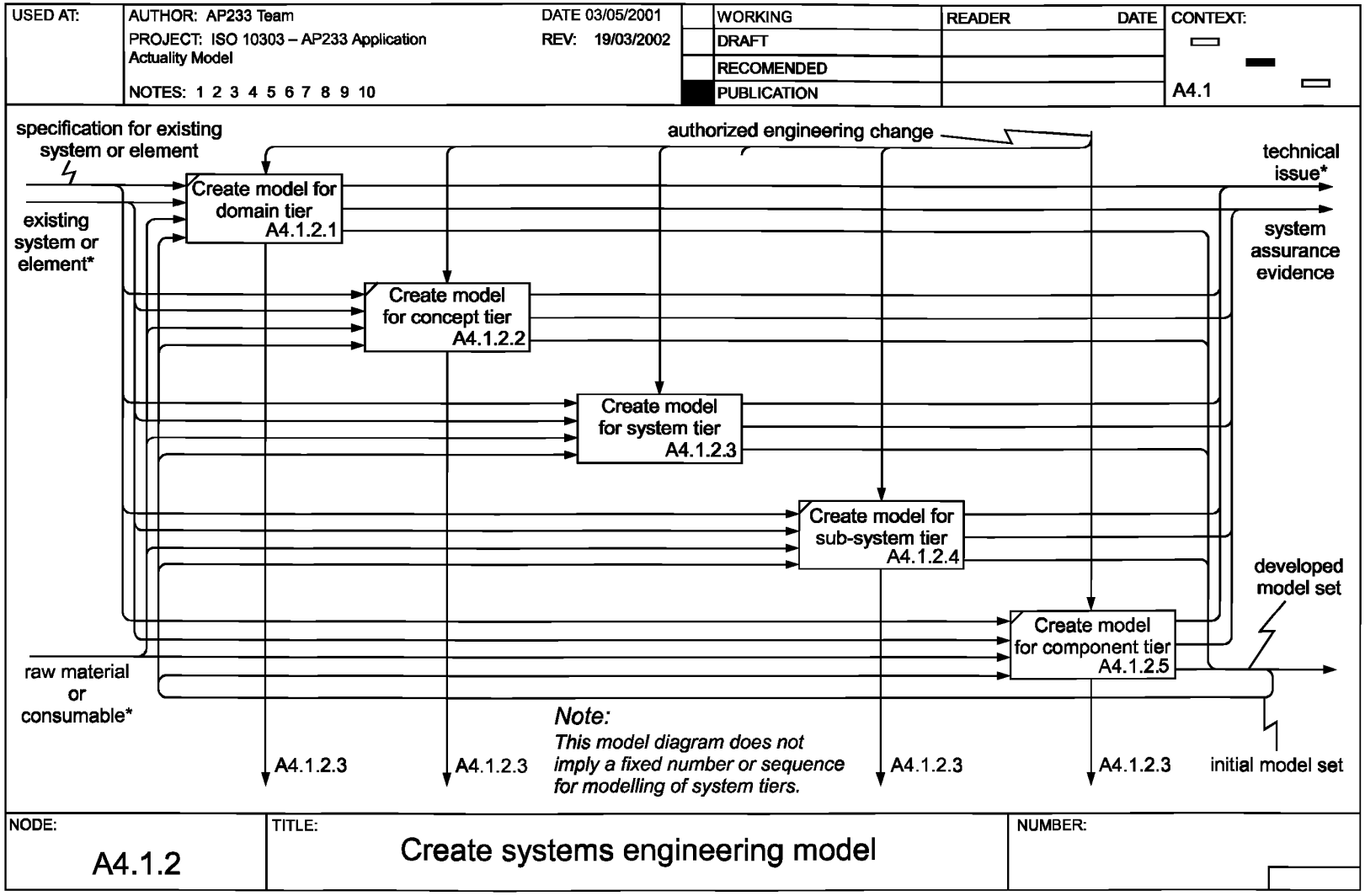
Used at — ИСПОЛЬЗОВАНО ПРИ; Author — АВТОР:; Project — ПРОЕКТ: Activity model — Модель действия; Notes — ПРИМЕЧАНИЯ:; Date — ДАТА:; Revision — ВАРИАНТ:; Working — Работа; Draft — ЧЕРТЕЖ; Recommended — РЕКОМЕНДОВАНО; Publication — ПУБЛИКАЦИЯ; Reader — ЧИТАТЕЛЬ; Date — ДАТА; Context — КОНТЕКСТ; Specification for existing system or element — Спецификация на существующую систему или элемент; Existing system or element* — Существующая система или элемент*; Demand on system capability* — Требование к функциональным возможностям системы*; Raw material or consumable* — Необработанные и расходные материалы*; Define system — Определение системы; Technical issue* — Технический вопрос*; Design for system component* — Проектирование компонта системы*; System for validation* — Система для проверки*; Produce system — Система производства; Authorized engineering change — Утвержденное инженерное изменение; Developped models — Разработанные модели; By-product* — Для продукции*; Validate produced system — Проверка изготовленной системы; Enabling system — Обеспечивающая система*; Concept tier model — Концептуальная модель; System ready for exploitation* — Готовность системы для эксплуатации*; Exploit system* — Эксплуатация системы*; System or element for re-use* — Система или материал для повторного использования; Transfer of system responsibility* — Передача ответственности за систему*; System assurance evidence — Подтверждение гарантии системы; Rejected system or element — Отбракованная система или элемент; Waste material* — Отходы производства*; Satisfied demand on system capability — Удовлетворение требований к функциональным возможностям; Node — ЭЛЕМЕНТ; Title — НАИМЕНОВАНИЕ; Execute project — Выполнение проекта; Number — НОМЕР; Change proposal* — Предложение по изменению*.

D.5–A4.1 Процесс задания системы



Used at — ИСПОЛЬЗОВАНО ПРИ; Author — АВТОР:; Project — ПРОЕКТ: Activity model — Модель действия; Notes — ПРИМЕЧАНИЯ:; Date — ДАТА:; Revision — ВАРИАНТ:; Working — Работа; Draft — ЧЕРТЕЖ; Recommended — РЕКОМЕНДОВАНО; Publication — ПУБЛИКАЦИЯ; Reader — ЧИТАТЕЛЬ; Date — ДАТА; Context — КОНТЕКСТ; Specification for existing system or element — Спецификация на существующую систему или элемент; Existing system or element* — Существующая система или элемент*; Control engineering change — Контроль инженерного изменения; Raw material or consumable* — Необработанные и расходные материалы*; Authorized engineering change — Утвержденное инженерное изменение; Create systems engineering model* — Создание инженерной модели систем*; Developed model set — Разработанный ряд моделей; Set of component requirements — Набор требований к компонентам; Design system component* — Разработка компонента системы*; Change proposal* — Изменение протокола*; System assurance ecidence — Подтверждение гарантии системы; Design for system component* — Проект для компонента системы*; Node — ЭЛЕМЕНТ; Title — НАИМЕНОВАНИЕ; Define system — Определение системы; Number — НОМЕР.

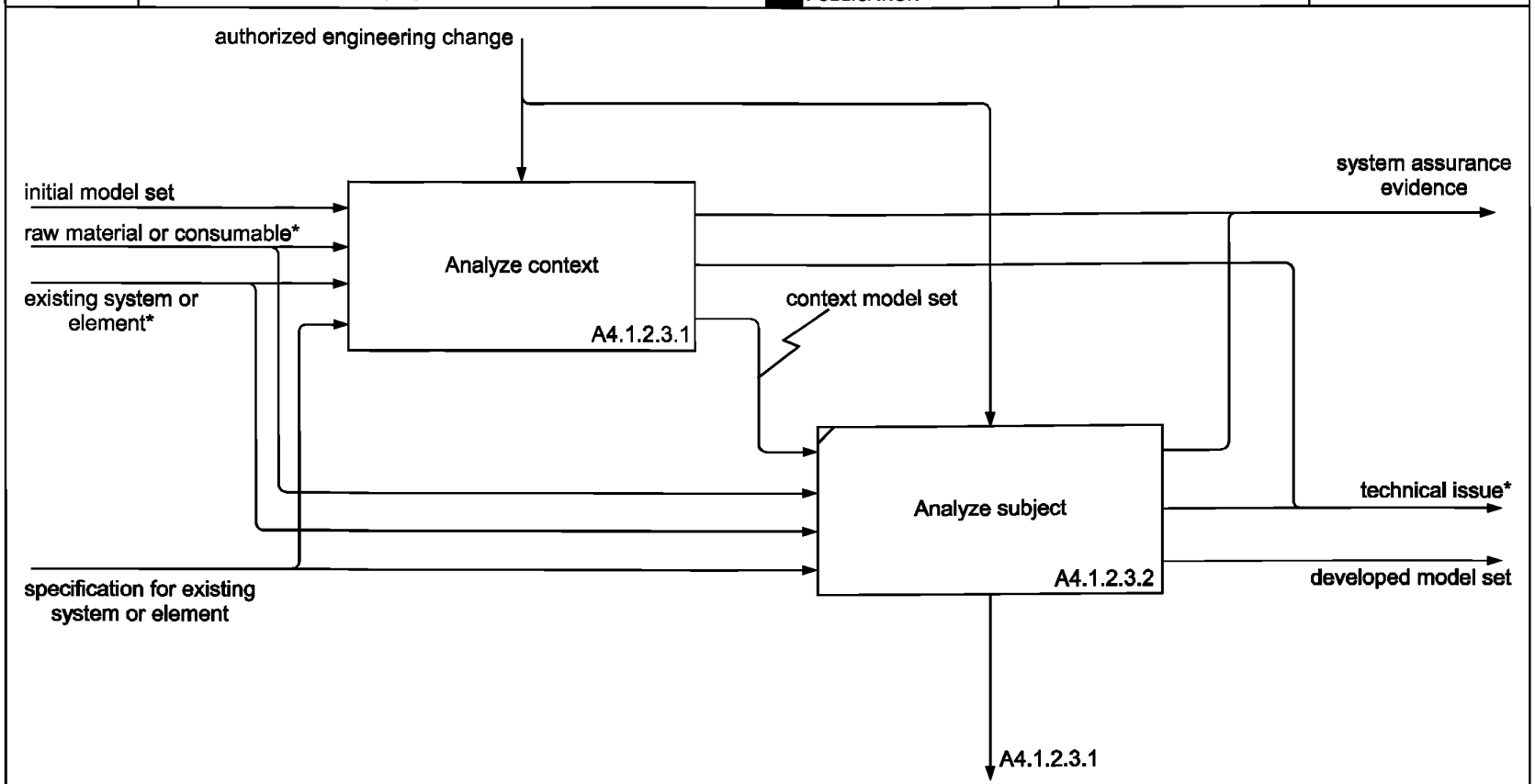
D.6–A4.1.2 Процесс разработки инженерной модели системы



Used at — ИСПОЛЬЗОВАНО ПРИ; Author — АВТОР:; Project — ПРОЕКТ: Activity model — Модель действия; Notes — ПРИМЕЧАНИЯ:; Date — ДАТА:; Revision — ВАРИАНТ:; Working — Работа; Draft — ЧЕРТЕЖ; Recommended — РЕКОМЕНДОВАНО; Publication — ПУБЛИКАЦИЯ; Reader — ЧИТАТЕЛЬ; Date — ДАТА; Context — КОНТЕКСТ; Specification for existing system or element — Спецификация на существующую систему или элемент; Existing system or element* — Существующая система или элемент*; Raw material or consumable* — Необработанные и расходные материалы*; Create model for domain tier — Создание модели для класса области; Create model for concept tier — Создание модели для класса понятий; Note: This model diagram does not imply a fixed number or sequence for modelling of system tiers — ПРИМЕЧАНИЕ: Эта диаграмма модели не предполагает наличия фиксированного числа или последовательности для моделирования классов систем; Create model for system tier — Создание модели для класса системы; Authorized engineering change — Утвержденное инженерное изменение; Create model for sub-system tier — Создание модели для класса подсистем; Create model for component tier — Создание модели для класса компонентов; Technical issue* — Технический вопрос; System assurance evidence — Подтверждение гарантии системы; Developed model set — Разработанный ряд моделей; Initial model set — Исходный ряд моделей; Node — ЭЛЕМЕНТ; Title — НАИМЕНОВАНИЕ; Create systems engineering model — Создание модели инженерных систем; Number — НОМЕР.

D.7–A4.1.2.3 Процесс создания модели для класса систем

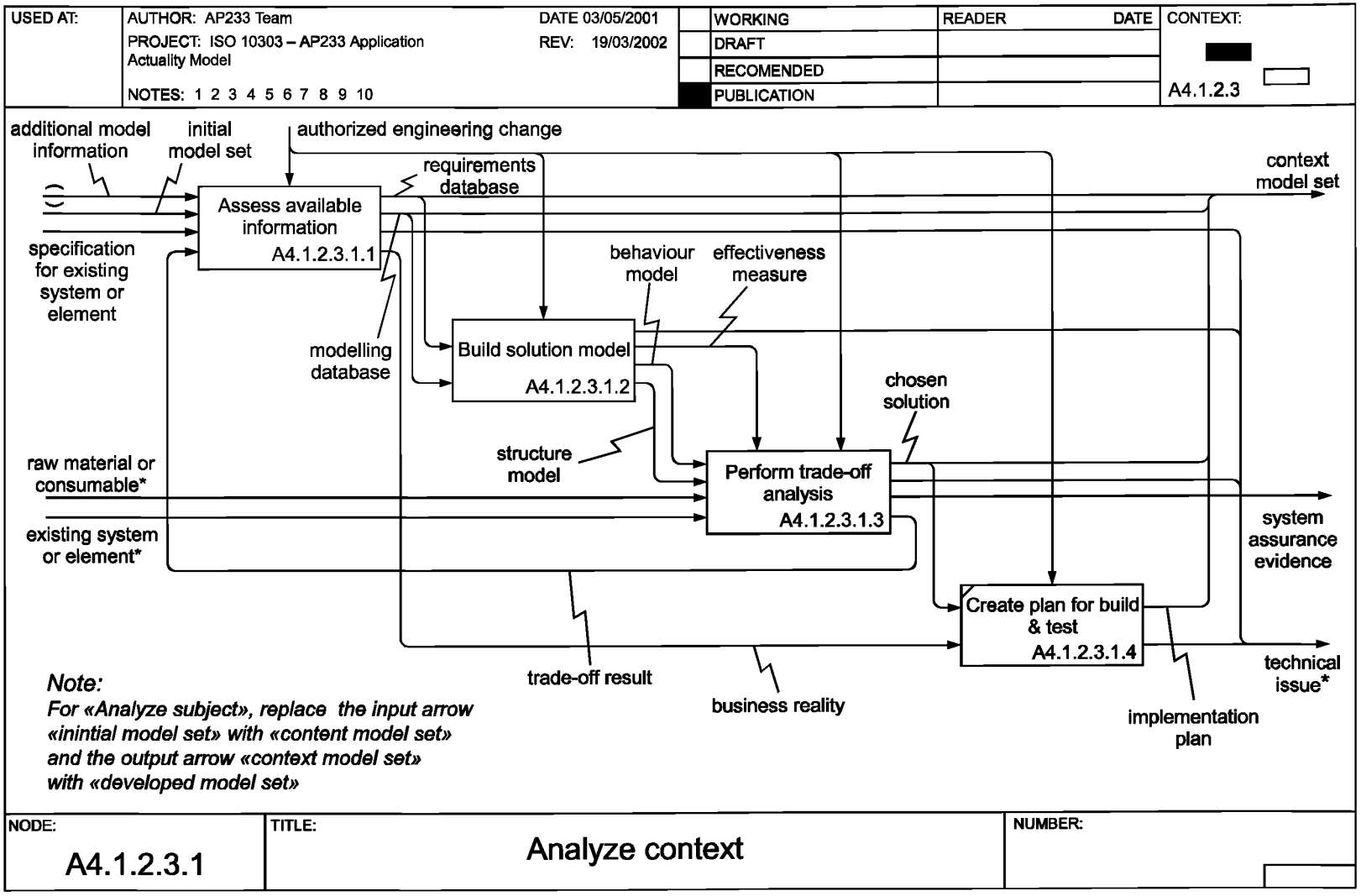
USED AT:	AUTHOR: AP233 Team	DATE 03/05/2001		WORKING	READER DATE	CONTEXT:
	PROJECT: ISO 10303 – AP233 Application Actuality Model	REV: 19/03/2002		DRAFT		
				RECOMENDED		
	NOTES: 1 2 3 4 5 6 7 8 9 10		■	PUBLICATION		A4.1.2



NODE:	TITLE:	NUMBER:
A4.1.2.3	Create model for system tier	

Used at — ИСПОЛЬЗОВАНО ПРИ; Author — АВТОР:; Project — ПРОЕКТ: Activity model — Модель действия; Notes — ПРИМЕЧАНИЯ:; Date — ДАТА:; Revision — ВАРИАНТ:; Working — Работа; Draft — ЧЕРТЕЖ; Recommended — РЕКОМЕНДОВАНО; Publication — ПУБЛИКАЦИЯ; Reader — ЧИТАТЕЛЬ; Date — ДАТА; Context — КОНТЕКСТ; Initial model set — Исходный ряд моделей; Raw material or consumable* — Необработанные и расходные материалы*; Existing system or element* — Существующая система или элемент*; Specification for existing system or element — Спецификация на существующую систему или элемент; Authorized engineering change — Утвержденное инженерное изменение; Analyze context — Анализ контекста; Context model set — Ряд контекстных моделей; Analyze subject — Анализ субъекта; System assurance evidence — Подтверждение гарантии системы; Technical issue — Технический вопрос; Developed model set — Ряд разработанных моделей; Node — ЭЛЕМЕНТ; Title — НАИМЕНОВАНИЕ; Create model for system tier — Создание модели для класса систем; Number — НОМЕР.

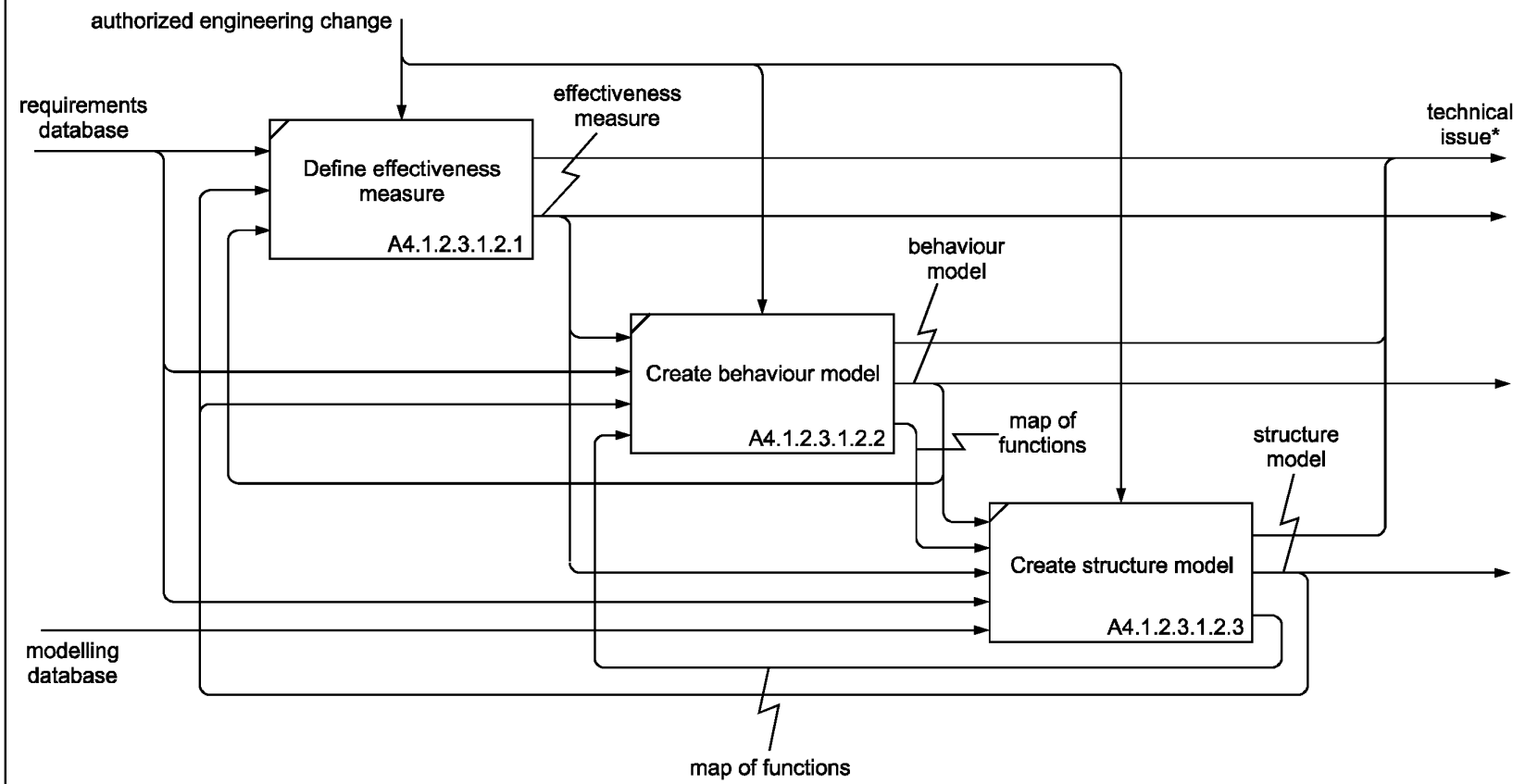
D.8–A4.1.2.3.1 Процесс анализа контекста системы



Used at — ИСПОЛЬЗОВАНО ПРИ; Author — АВТОР:; Project — ПРОЕКТ: Activity model — Модель действия; Notes — ПРИМЕЧАНИЯ:; Date — ДАТА:; Revision — ВАРИАНТ:; Working — Работа; Draft — ЧЕРТЕЖ; Recommended — РЕКОМЕНДОВАНО; Publication — ПУБЛИКАЦИЯ; Reader — ЧИТАТЕЛЬ; Date — ДАТА; Context — КОНТЕКСТ; Additional model information — Дополнительная информация о модели; Specification for existing system or element — Спецификация на существующую систему или элемент; Raw material or consumable* — Необработанные и расходные материалы*; Existing system or element* — Существующая система или элемент*; Initial model set — Исходные модели; Assess available information — Оценка доступной информации; Modelling database — Моделирование базы данных; Authorized engineering change — Утвержденное инженерное изменение; Requirements database — Требования к базе данных; Build solution model — Построение модели решения; Structure model — Структурная модель; Trade-off result — Компромиссный результат; Note: For «Analyze subject», replace the input arrow «initial model set» with «content model set» and the output arrow «context model set» with «developed model set» — ПРИМЕЧАНИЕ: Для элемента «Анализ субъекта» замените входящую стрелку «исходный ряд моделей» на «ряд контекстных моделей», а выходящую стрелку «ряд контекстных моделей» — на «ряд разработанных моделей»; Behaviour model — Модель поведения; Perform trade-off analysis — Проведение компромиссного анализа; Business reality — Бизнес-реальность; Effectiveness measure — Мера эффективности; Chosen solution — Выбранное решение; Create plan for build & test — Создание плана для построения и испытаний; Implementation plan — План внедрения; Context model set — Контекстные модели; System assurance evidence — Подтверждение гарантии системы; Technical issue — Технический вопрос; Node — ЭЛЕМЕНТ; Title — НАИМЕНОВАНИЕ; Analyze context — Анализ контекста; Number — НОМЕР.

D.9–A4.1.2.3.1.2 Процесс построения модели решения

USED AT:	AUTHOR: AP233 Team	DATE 03/05/2001		WORKING	READER DATE	CONTEXT:
	PROJECT: ISO 10303 – AP233 Application Actuality Model	REV: 19/03/2002		DRAFT		
				RECOMENDED		
	NOTES: 1 2 3 4 5 6 7 8 9 10		■	PUBLICATION		A4.1.2.3.1

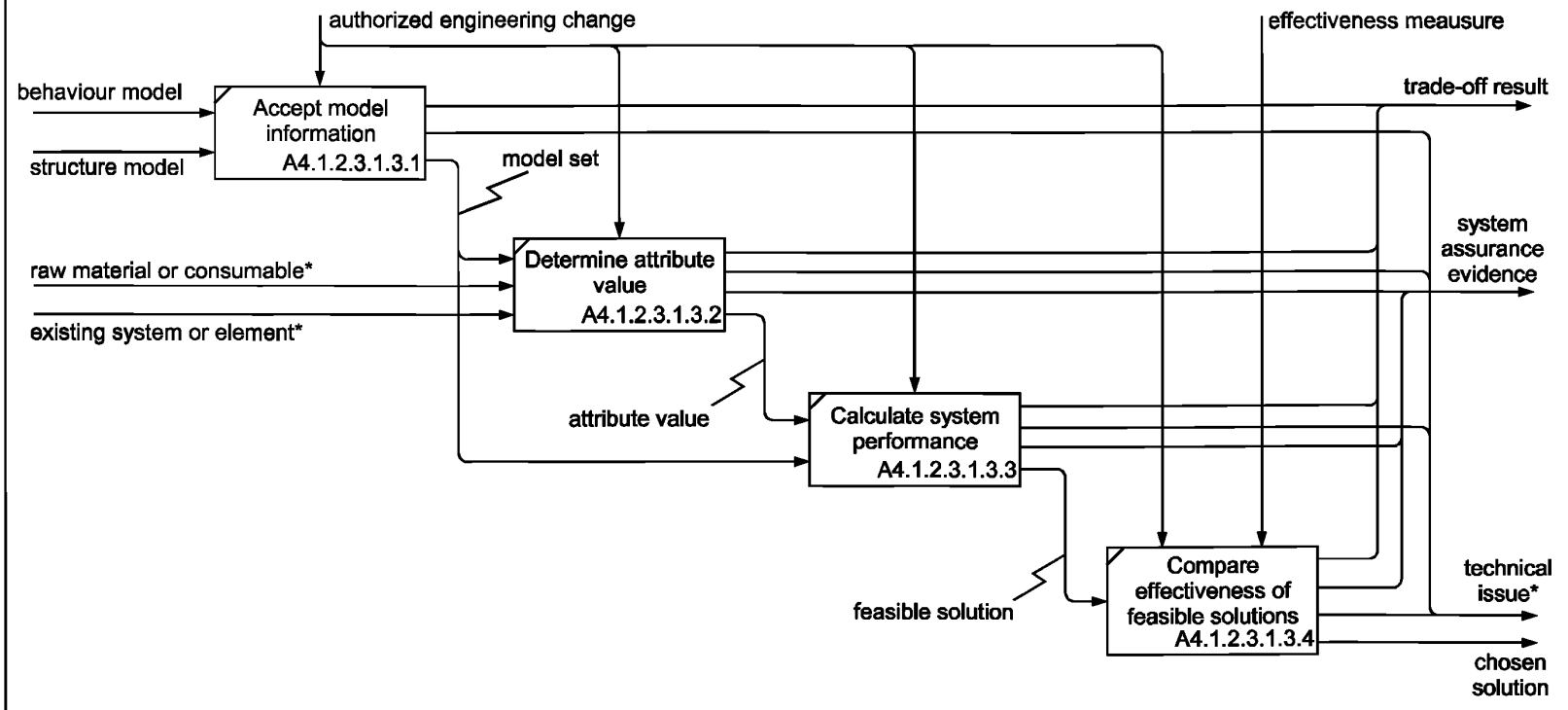


NODE:	TITLE:	NUMBER:
A4.1.2.3.1.2	Build solution model	

Used at — ИСПОЛЬЗОВАНО ПРИ; Author — АВТОР:; Project — ПРОЕКТ: Activity model — Модель действия; Notes — ПРИМЕЧАНИЯ:; Date — ДАТА:; Revision — ВАРИАНТ:; Working — Работа; Draft — ЧЕРТЕЖ; Recommended — РЕКОМЕНДОВАНО; Publication — ПУБЛИКАЦИЯ; Reader — ЧИТАТЕЛЬ; Date — ДАТА; Context — КОНТЕКСТ; Requirements database — Требования к базе данных; Modelling database — Моделирование базы данных; Authorized engineering change — Утвержденное инженерное изменение; Define effectiveness measure — Определение меры эффективности; Effectiveness measure — Мера эффективности; Create behaviour model — Создание модели поведения; Map of functions — Отображение функций; Behaviour model — Модель поведения; Create structure model — Создание модели структуры; Structure model — Модель структуры; Technical issue* — Технический вопрос*; Node — ЭЛЕМЕНТ; Title — НАИМЕНОВАНИЕ; Build solution model — Построение модели решения; Number — НОМЕР.

D.10–A4.1.2.3.1.3 Процесс сопоставительного анализа решений

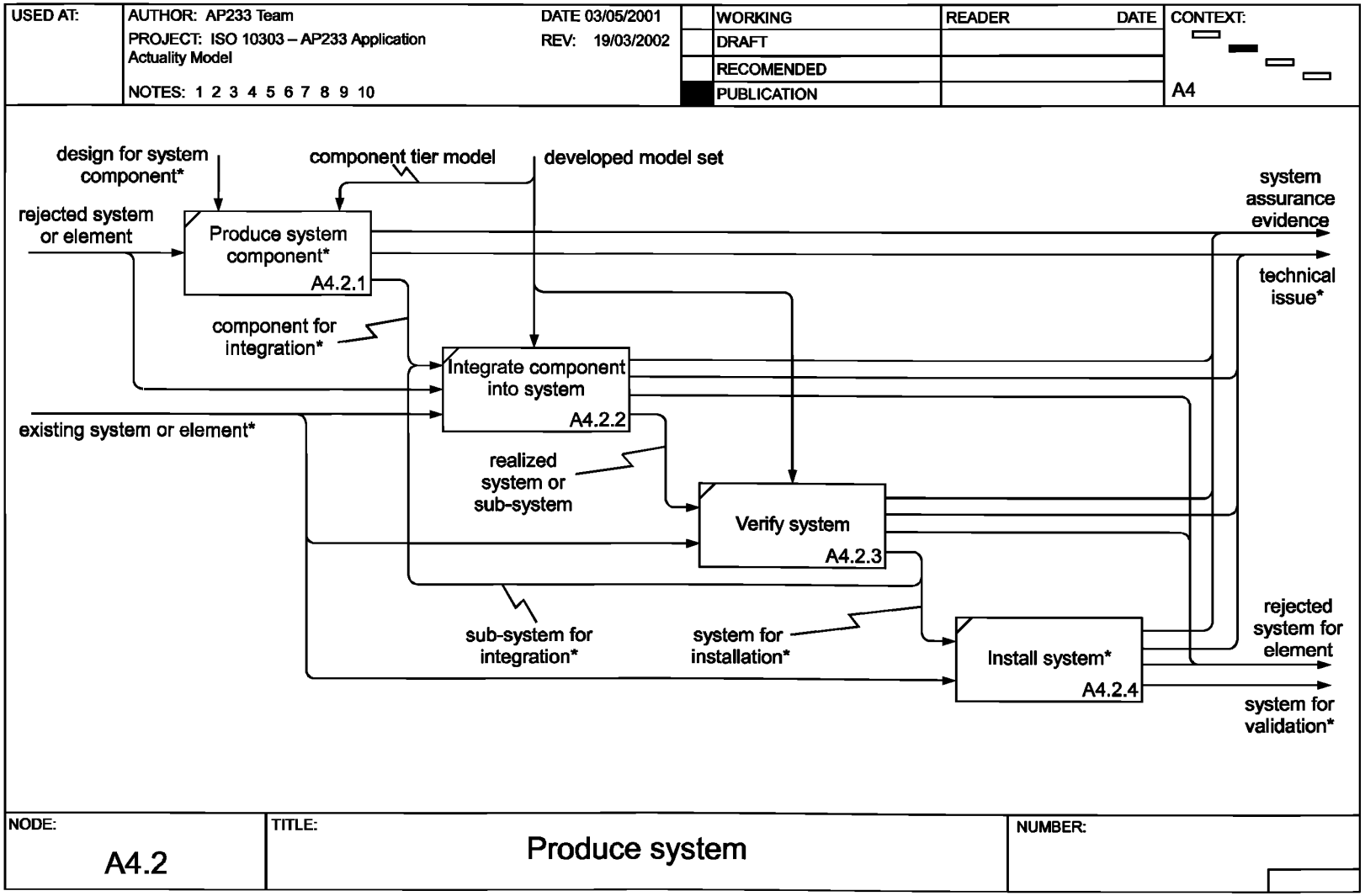
USED AT:	AUTHOR: AP233 Team	DATE 03/05/2001		WORKING	READER DATE	CONTEXT:
	PROJECT: ISO 10303 – AP233 Application Actuality Model	REV: 19/03/2002		DRAFT		
				RECOMENDED		
	NOTES: 1 2 3 4 5 6 7 8 9 10		■	PUBLICATION		A4.1.2.3.1



NODE:	TITLE:	NUMBER:
A4.1.2.3.1.3	Perform trade-off analysis	

Used at — ИСПОЛЬЗОВАНО ПРИ; Author — АВТОР:; Project — ПРОЕКТ: Activity model — Модель действия; Notes — ПРИМЕЧАНИЯ:; Date — ДАТА:; Revision — ВАРИАНТ:; Working — Работа; Draft — ЧЕРТЕЖ; Recommended — РЕКОМЕНДОВАНО; Publication — ПУБЛИКАЦИЯ; Reader — ЧИТАТЕЛЬ; Date — ДАТА; Context — КОНТЕКСТ; Behaviour model — Модель поведения; Structure model — Модель структуры; Raw material or consumable* — Необработанные и расходные материалы*; Existing system or element* — Существующая система или элемент*; Accept model information — Прием информации о модели; Authorized engineering change — Утвержденное инженерное изменение; Model set — Ряд моделей; Determine attribute value — Определение значения атрибута; Attribute value — Значение атрибута; Calculate system performance — Расчет функциональных возможностей системы; Feasible solution — Допустимое решение; Compare effectiveness of feasible solutions — Сравнение эффективности допустимых решений; Effectiveness meausure — Мера эффективности; Trade-off result — Компромиссный результат; System assurance evidence — Подтверждение гарантии системы; Technical issue — Технический вопрос; Chosen solution — Выбранное решение; Node — ЭЛЕМЕНТ; Title — НАИМЕНОВАНИЕ; Perform trade-off analysis — Проведение компромиссного анализа; Number — НОМЕР.

D.11–A4.2 Процесс изготовления системы



Used at — ИСПОЛЬЗОВАНО ПРИ; Author — АВТОР:; Project — ПРОЕКТ: Activity model — Модель действия; Notes — ПРИМЕЧАНИЯ:; Date — ДАТА:; Revision — ВАРИАНТ:; Working — Работа; Draft — ЧЕРТЕЖ; Recommended — РЕКОМЕНДОВАНО; Publication — ПУБЛИКАЦИЯ; Reader — ЧИТАТЕЛЬ; Date — ДАТА; Context — КОНТЕКСТ; Rejected system or element — Отбракованная система или элемент; Existing system or element* — Существующая система или элемент*; Design for system component* — Проектирование компонента системы*; Produce system component* — Изготовление компонента системы*; Component for integration* — Компонент для интеграции*; Sub-system for integration* — Подсистема для интеграции*; Integrate component into system — Интеграция компонента в систему; Realized system or sub-system — Реализация системы или подсистемы; Developed model set — Разработанные модели; Verify system — Проверка системы; System for installation* — Система для монтажа*; Install system* — Монтаж системы*; System assurance evidence — Подтверждение гарантии системы; Technical issue — Технический вопрос; System for validation — Система для проверки; Node — ЭЛЕМЕНТ; Title — НАИМЕНОВАНИЕ; Produce system — Изготовление системы; Number — НОМЕР

Приложение Е
(справочное)

Ссылочная модель приложения EXPRESS-G

Диаграммы, приведенные ниже, соответствуют ссылочной модели приложения EXPRESS, описанной в разделе 4. Диаграммы выполнены с использованием графической нотации EXPRESS-G языка EXPRESS. Графическая нотация EXPRESS-G приведена в приложении D ИСО 10303-11.

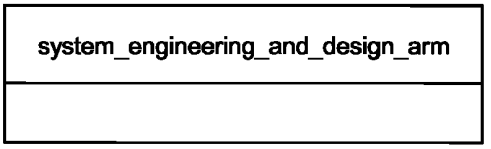


Рисунок Е.1 — Диаграмма ссылочной модели приложения на уровне схемы в нотации EXPRESS-G

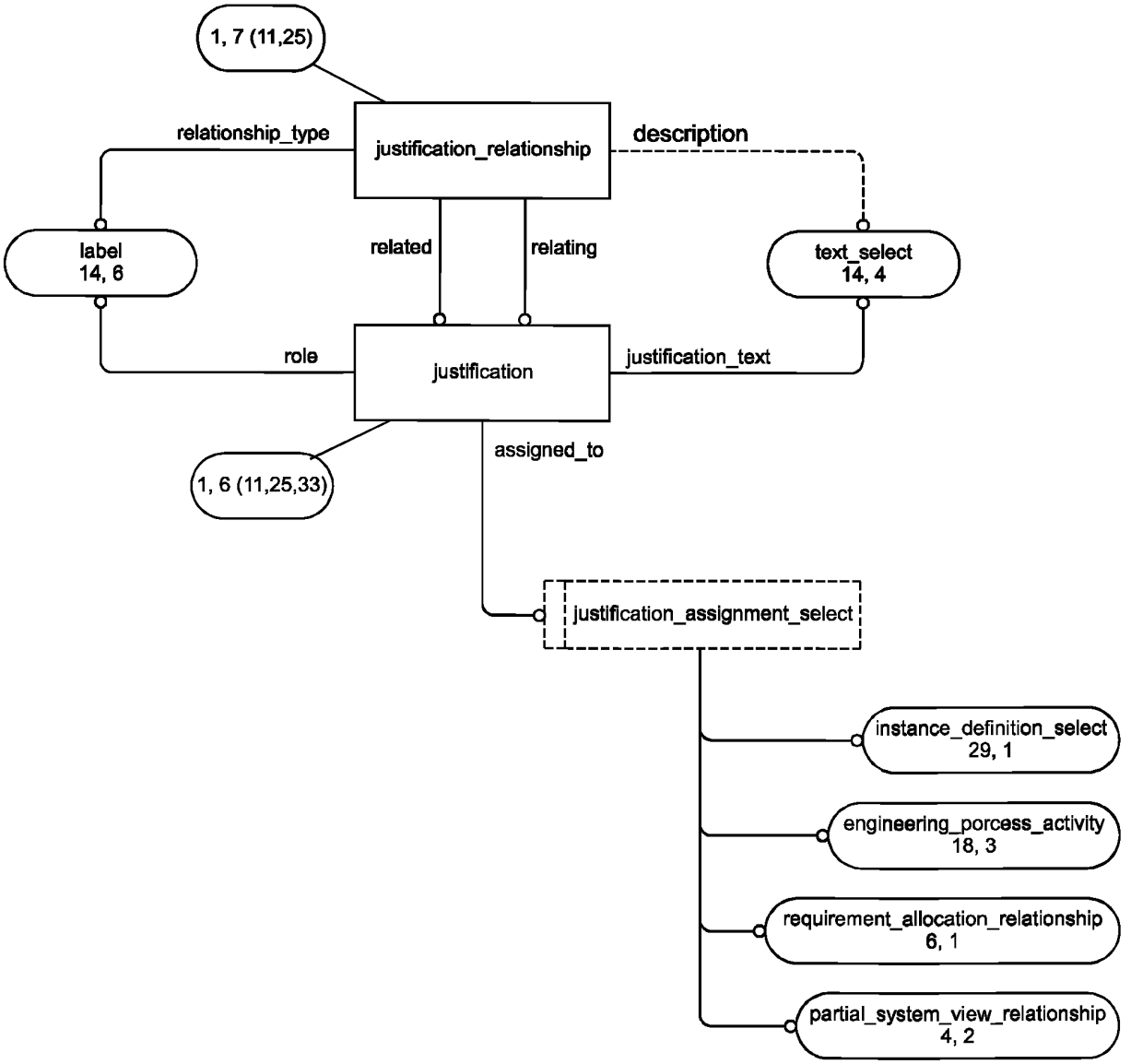


Рисунок Е.2 — Диаграмма ссылочной модели приложения на уровне сущности в нотации EXPRESS-G (1 из 41)

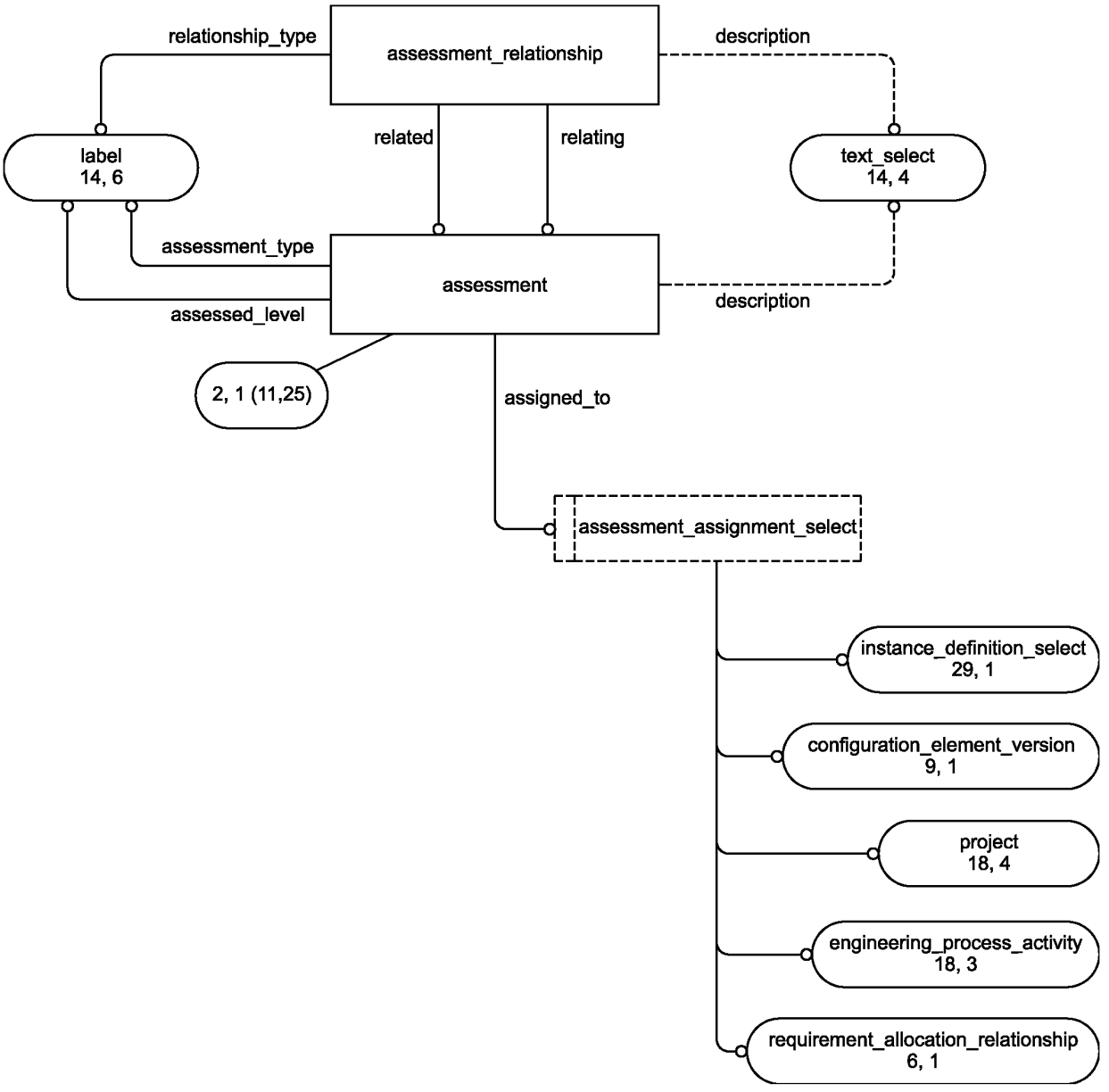


Рисунок Е.3 — Диаграмма ссылочной модели приложения
на уровне сущности в нотации EXPRESS-G (2 из 41)

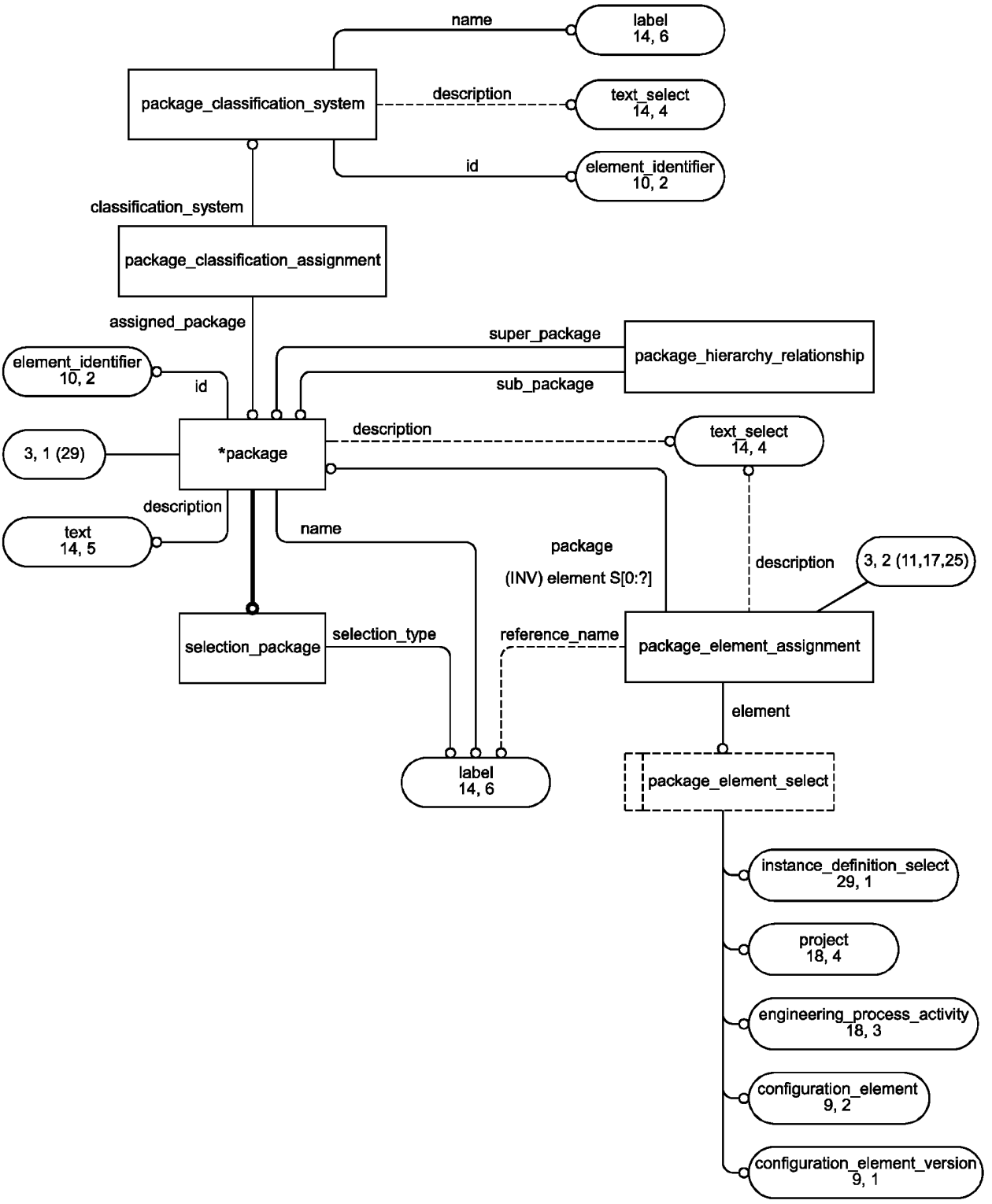


Рисунок Е.4 — Диаграмма ссылочной модели приложения
на уровне сущности в нотации EXPRESS-G (3 из 41)

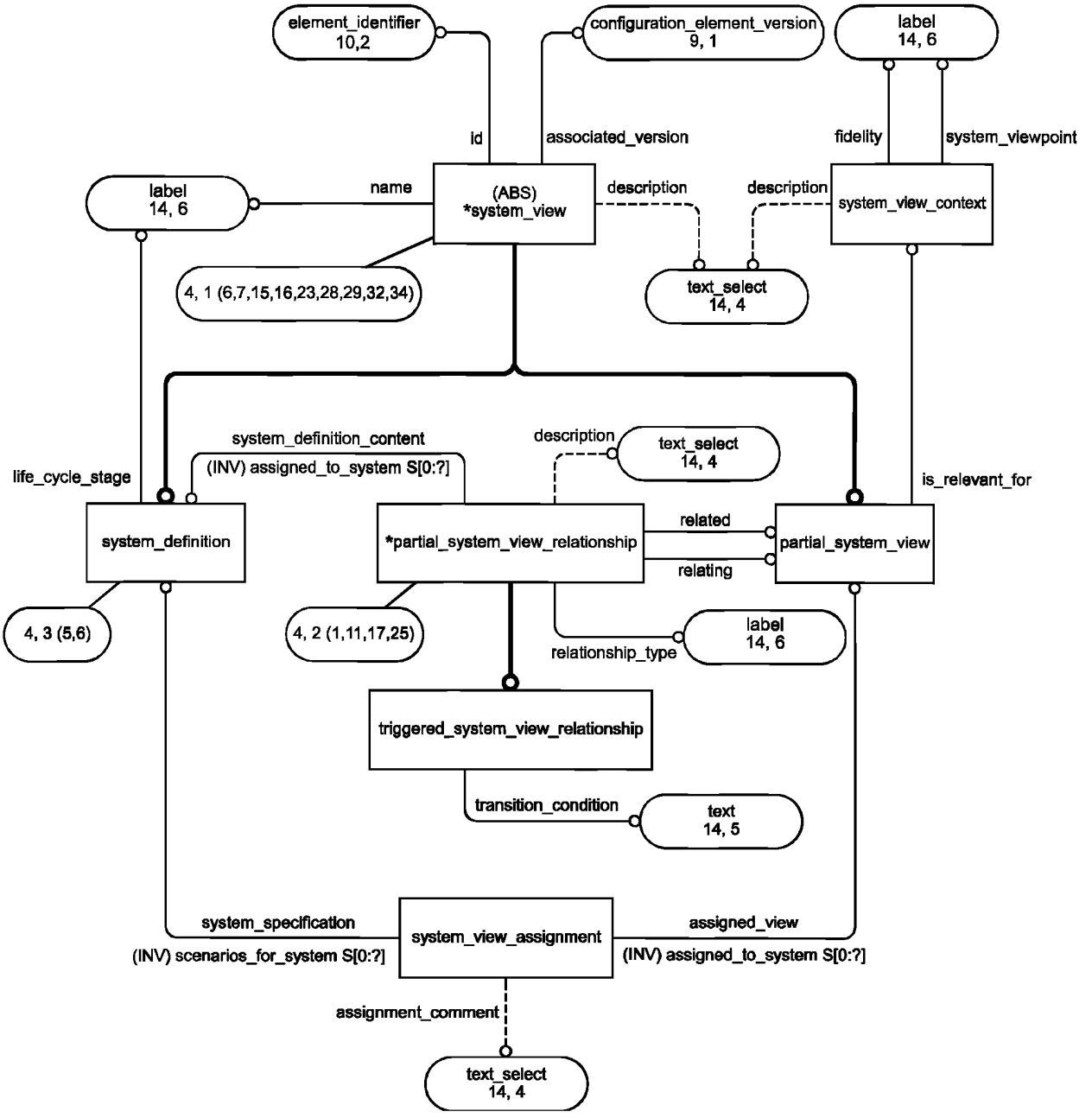


Рисунок Е.5 — Диаграмма ссылочной модели приложения
на уровне сущности в нотации EXPRESS-G (4 из 41)

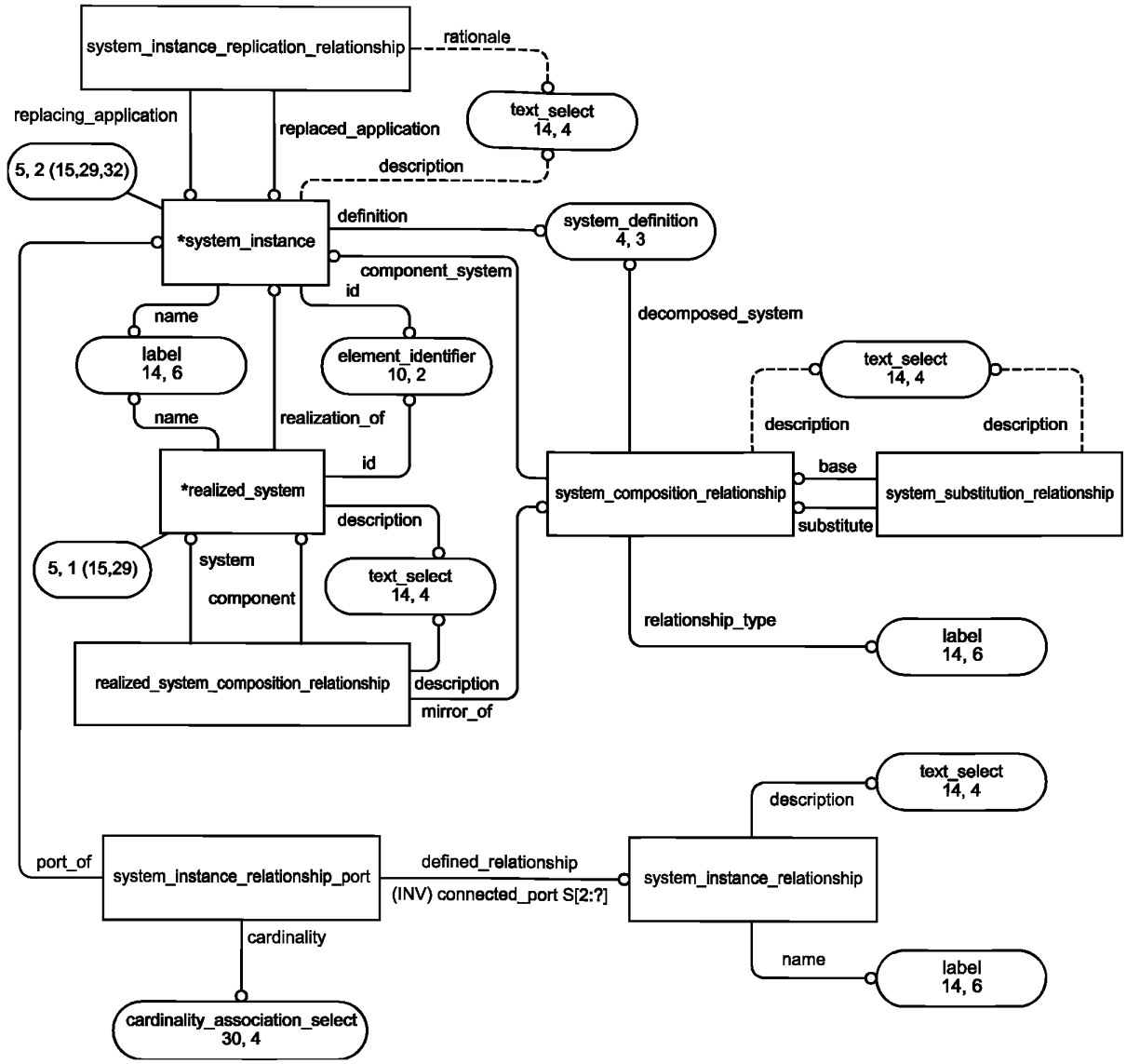


Рисунок Е.6 — Диаграмма ссылочной модели приложения
на уровне сущности в нотации EXPRESS-G (5 из 41)

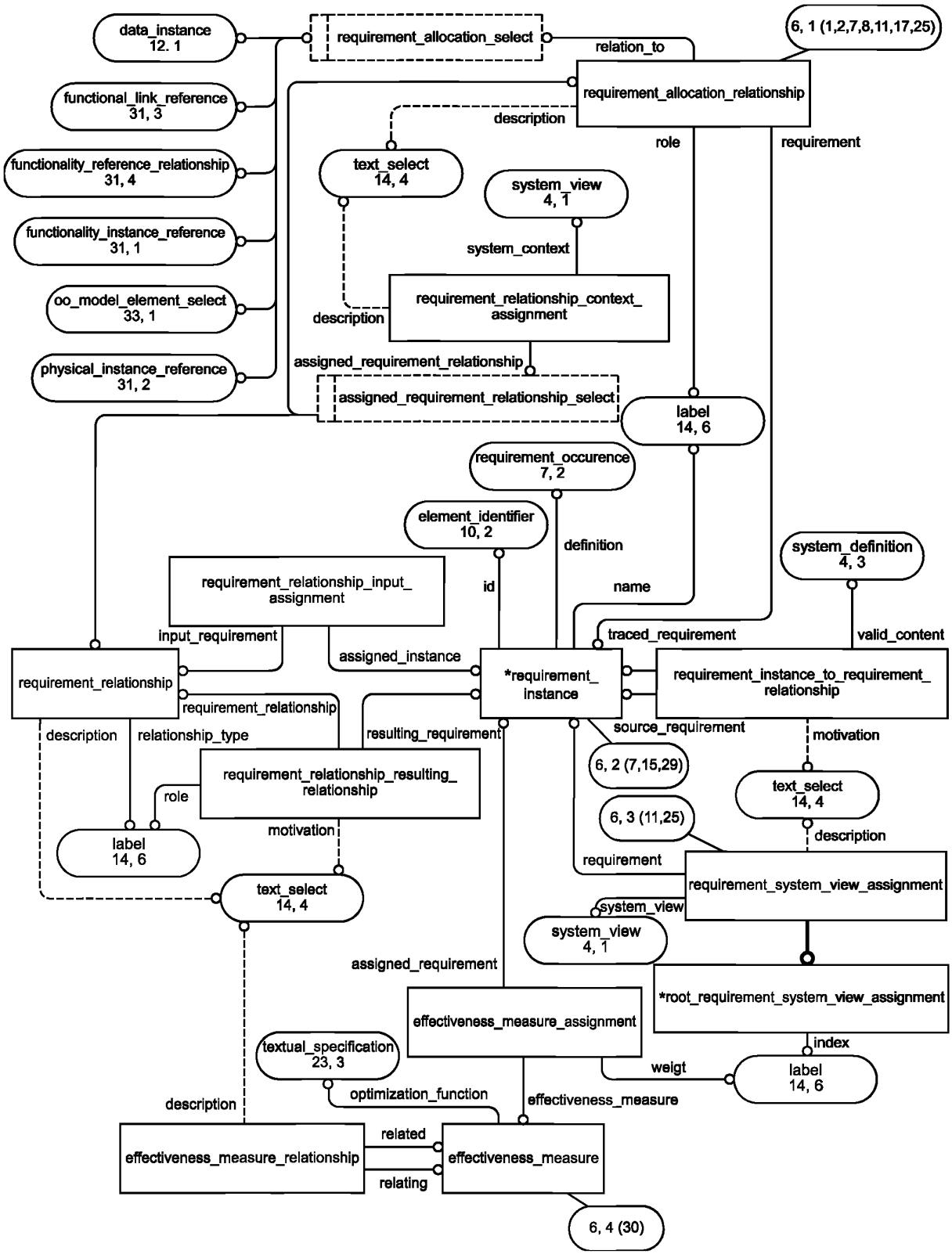


Рисунок Е.7 — Диаграмма ссылочной модели приложения на уровне сущности в нотации EXPRESS-G (6 из 41)

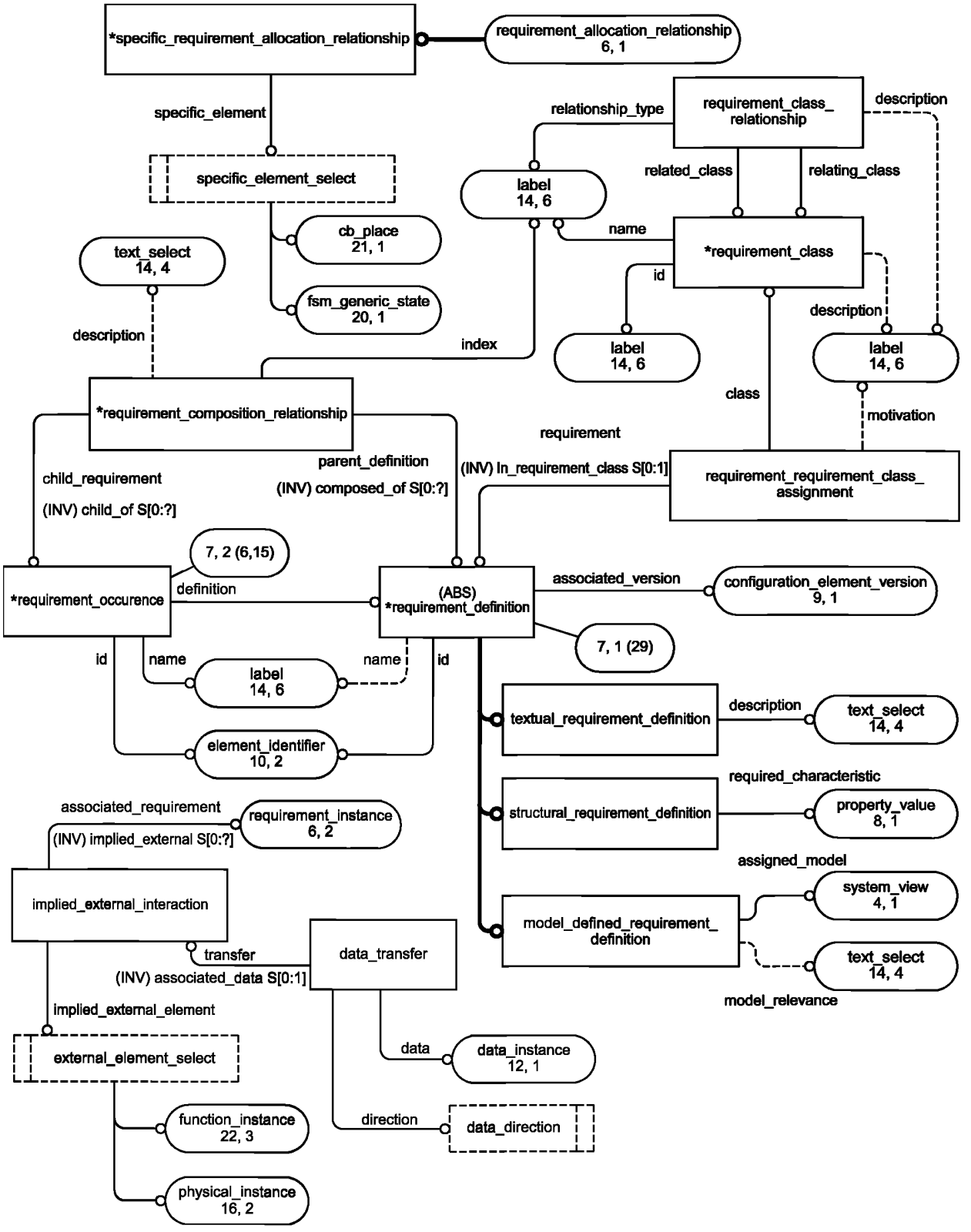


Рисунок Е.8 — Диаграмма ссылочной модели приложения на уровне сущности в нотации EXPRESS-G (7 из 41)

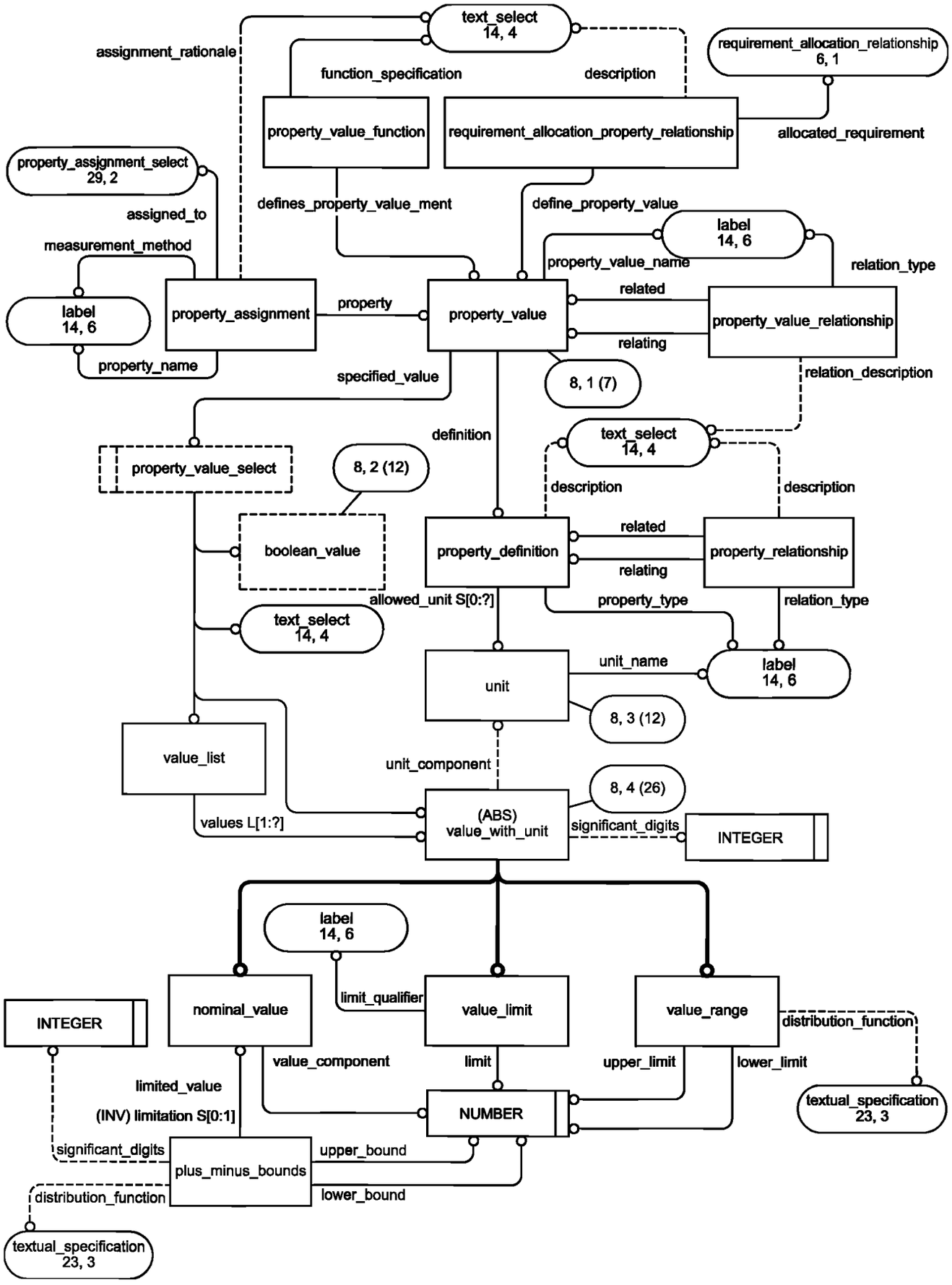


Рисунок Е.9 — Диаграмма ссылочной модели приложения на уровне сущности в нотации EXPRESS-G (8 из 41)

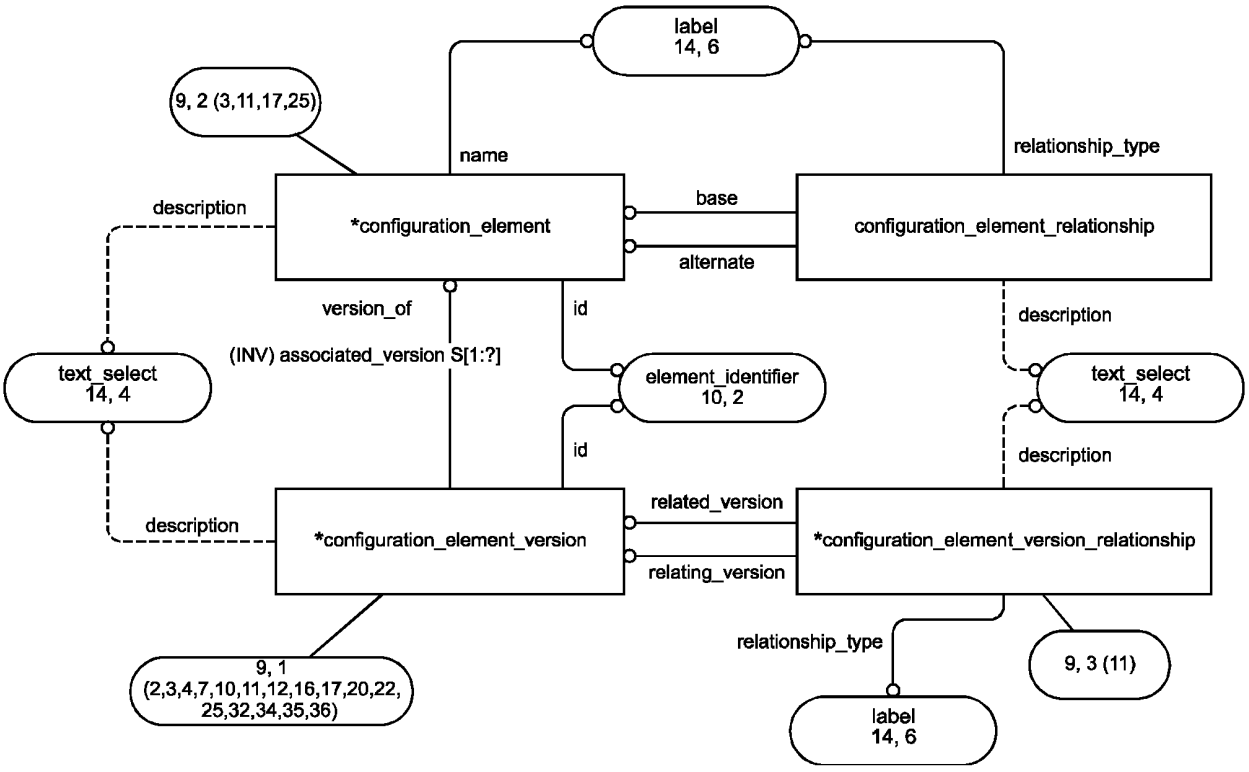


Рисунок Е.10 — Диаграмма ссылочной модели приложения
на уровне сущности в нотации EXPRESS-G (9 из 41)

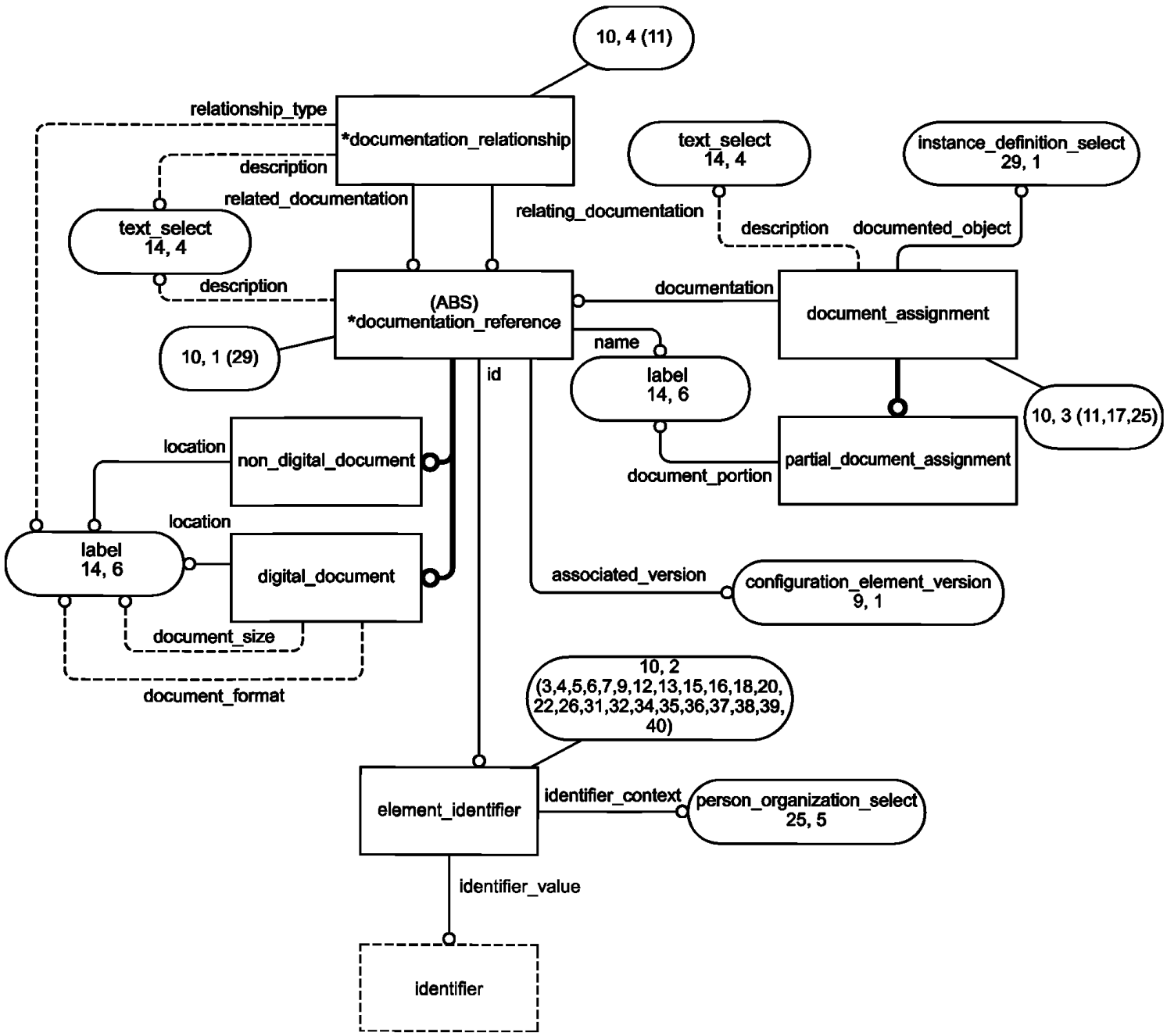


Рисунок Е.11 — Диаграмма ссылочной модели приложения на уровне сущности в нотации EXPRESS-G (10 из 41)

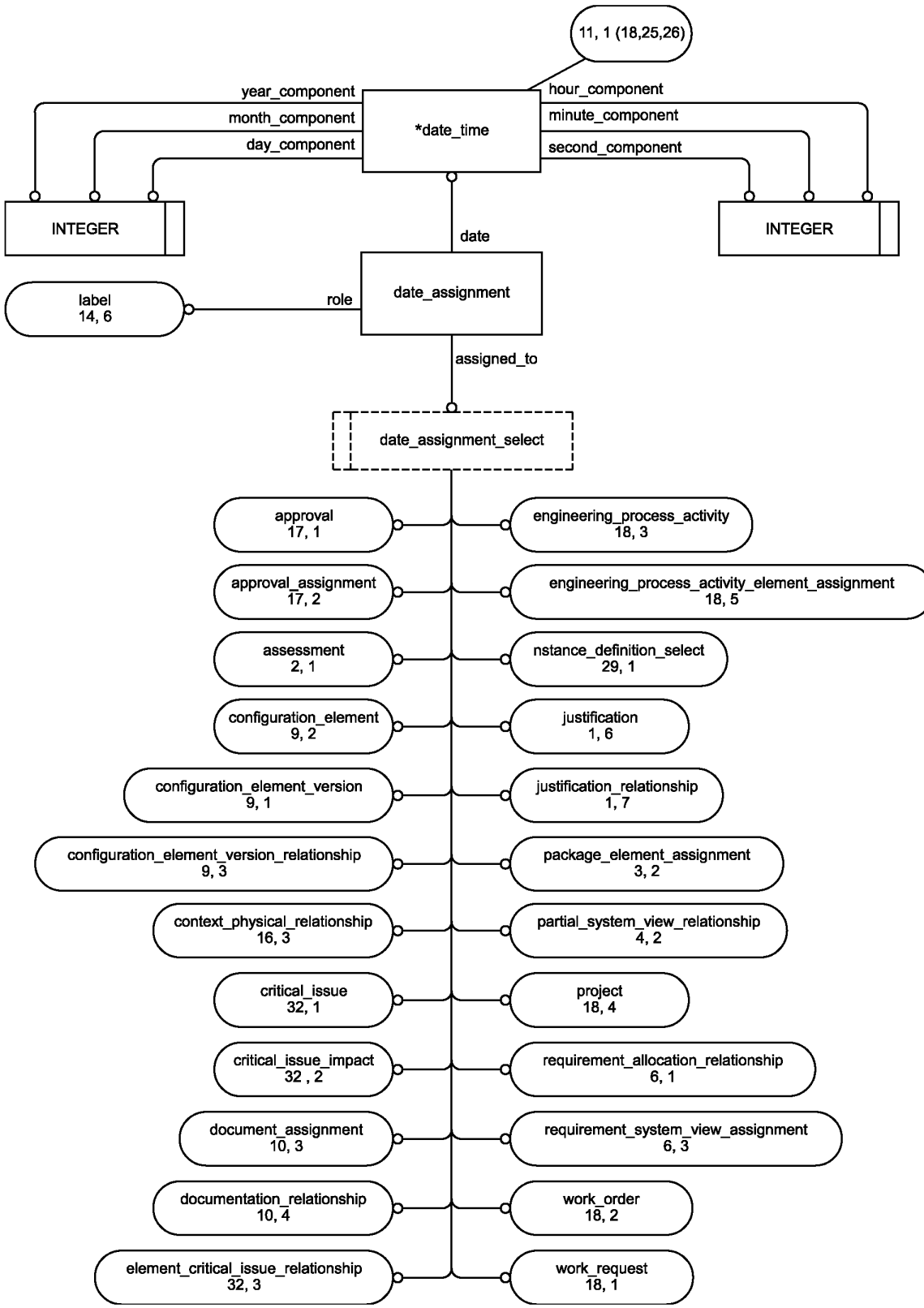


Рисунок Е.12 — Диаграмма ссылочной модели приложения на уровне сущности в нотации EXPRESS-G (11 из 41)

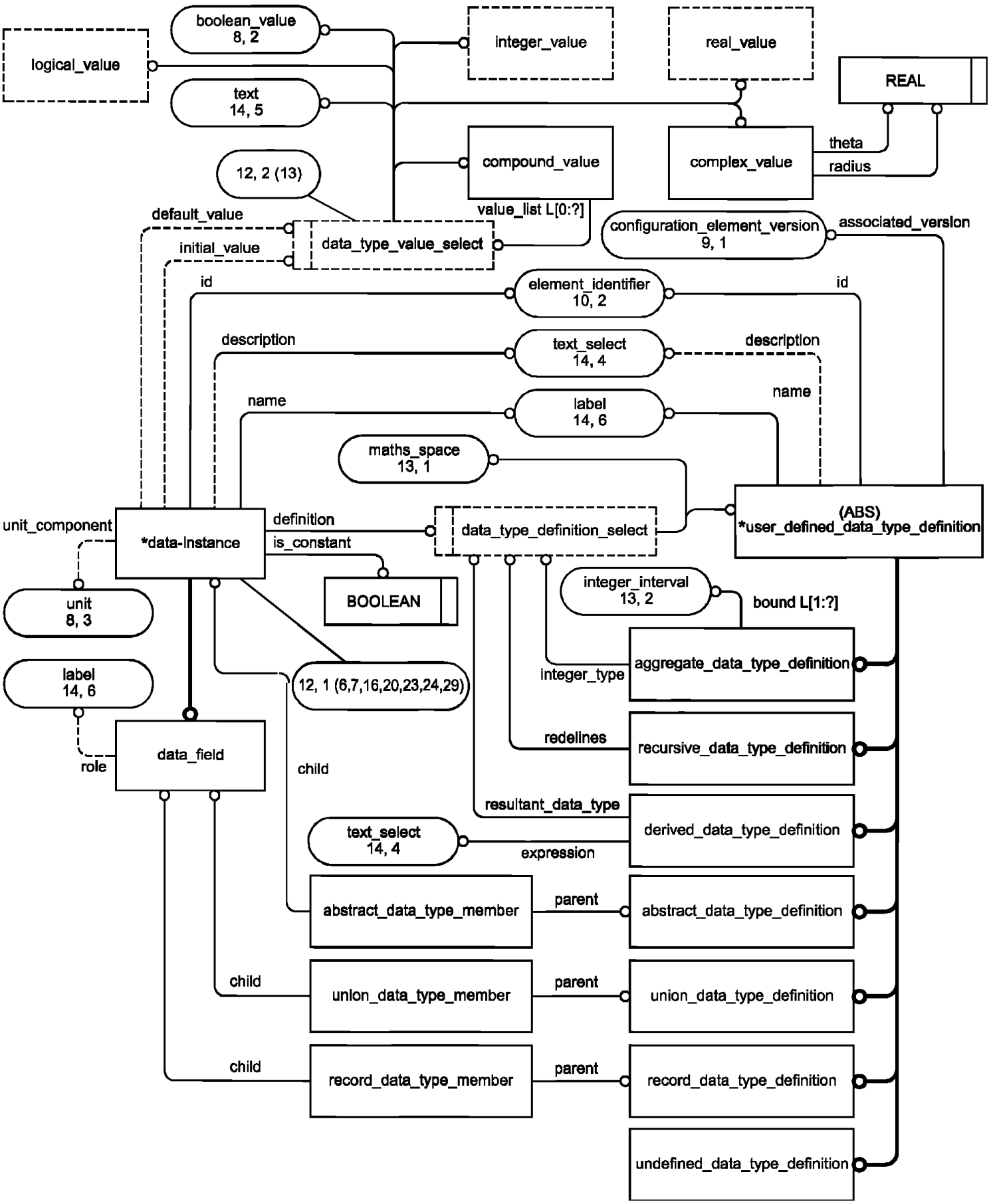


Рисунок Е.13 — Диаграмма ссылочной модели приложения
на уровне сущности в нотации EXPRESS-G (12 из 41)

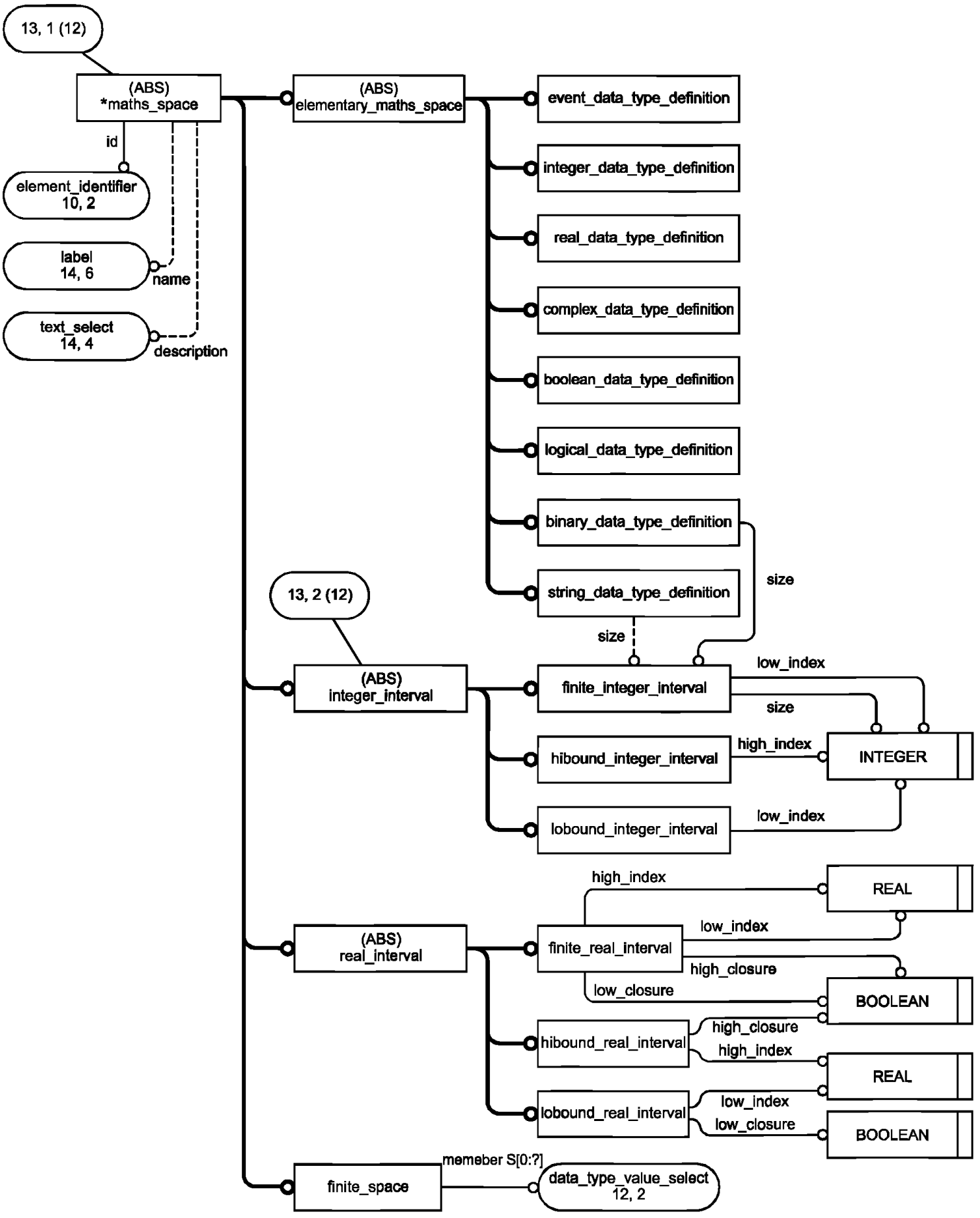


Рисунок Е.14 — Диаграмма ссылочной модели приложения на уровне сущности в нотации EXPRESS-G (13 из 41)

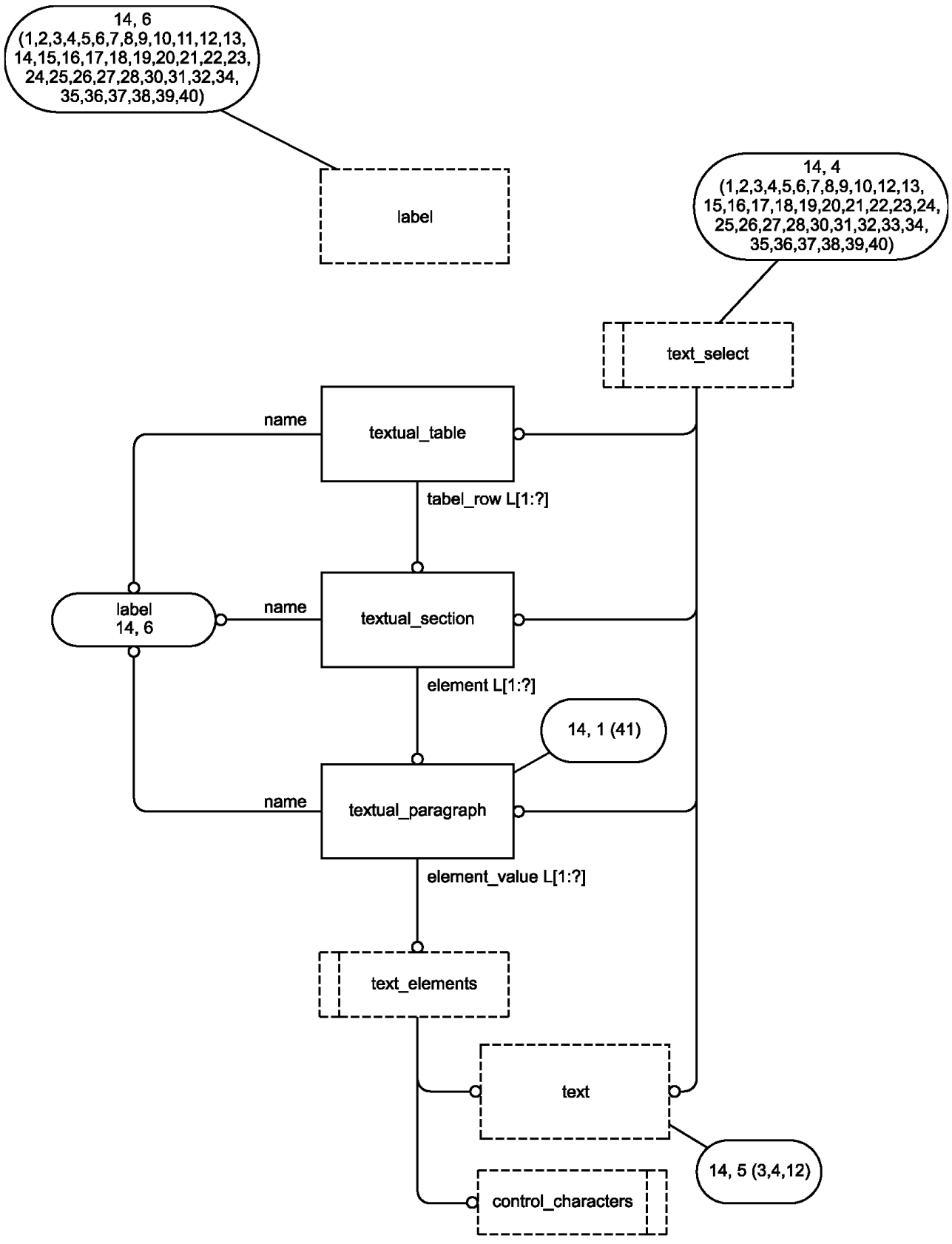


Рисунок Е.15 — Диаграмма ссылочной модели приложения
на уровне сущности в нотации EXPRESS-G (14 из 41)

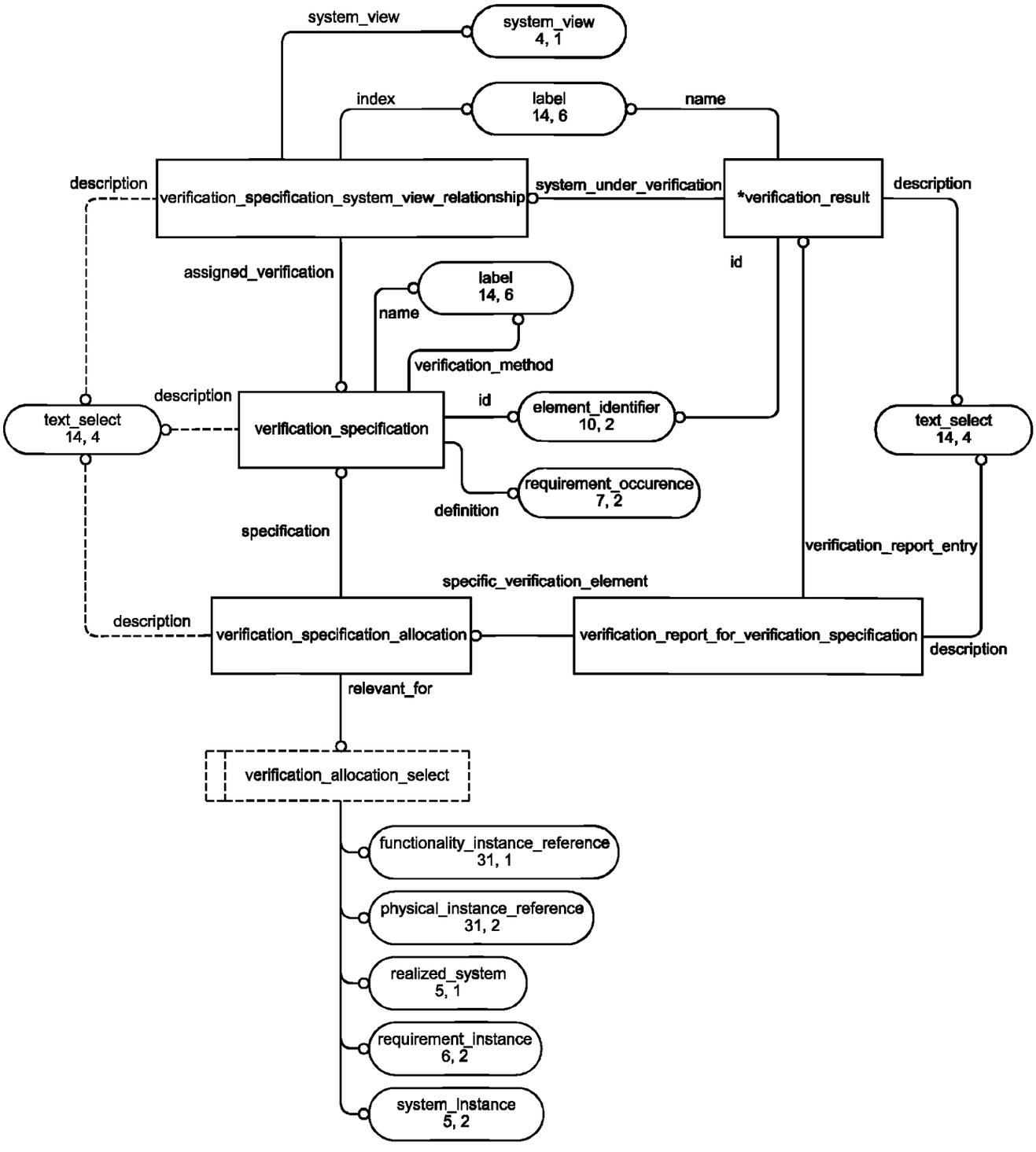


Рисунок Е.16 — Диаграмма ссылочной модели приложения на уровне сущности в нотации EXPRESS-G (15 из 41)

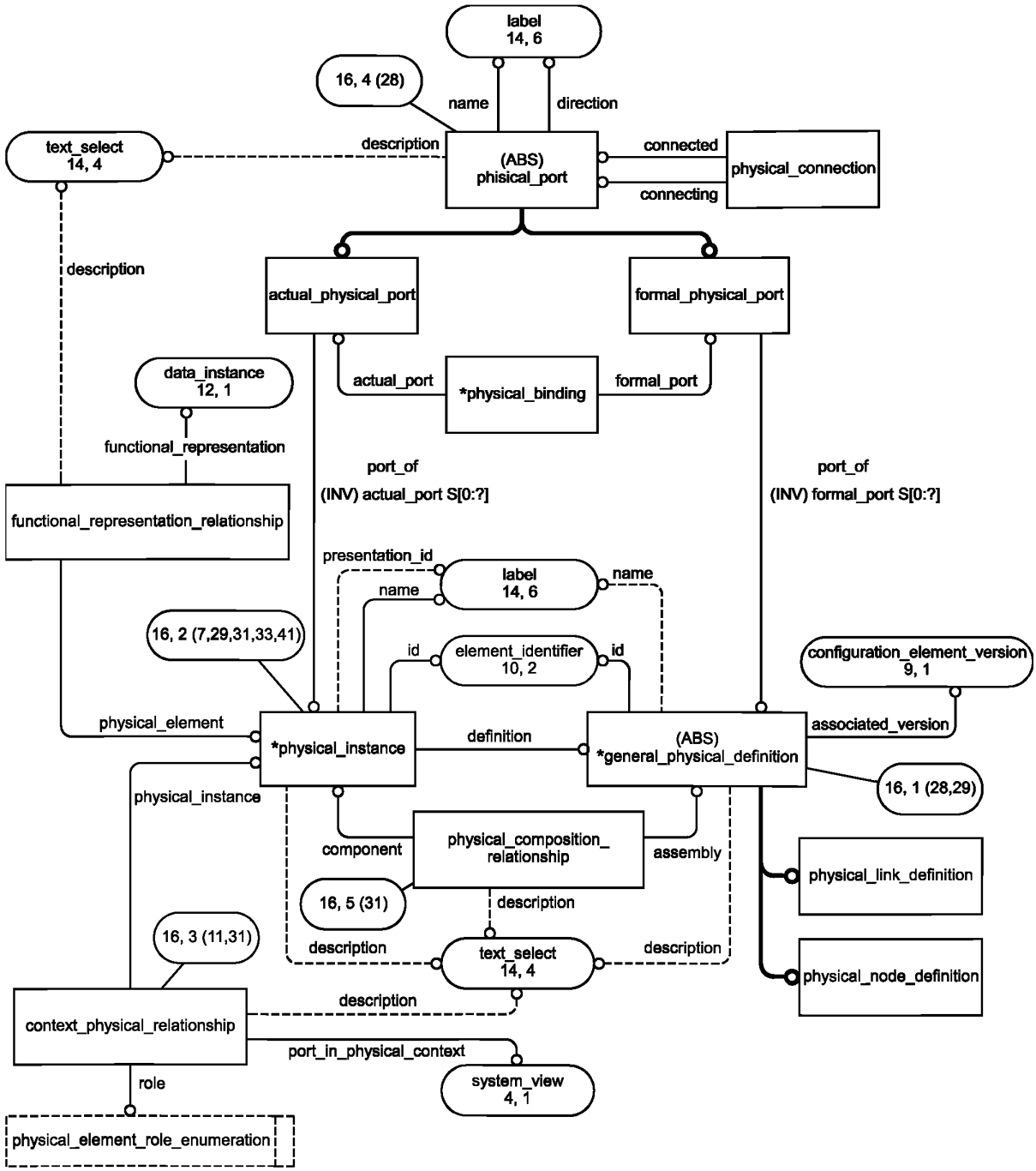


Рисунок Е.17 — Диаграмма ссылочной модели приложения
на уровне сущности в нотации EXPRESS-G (16 из 41)

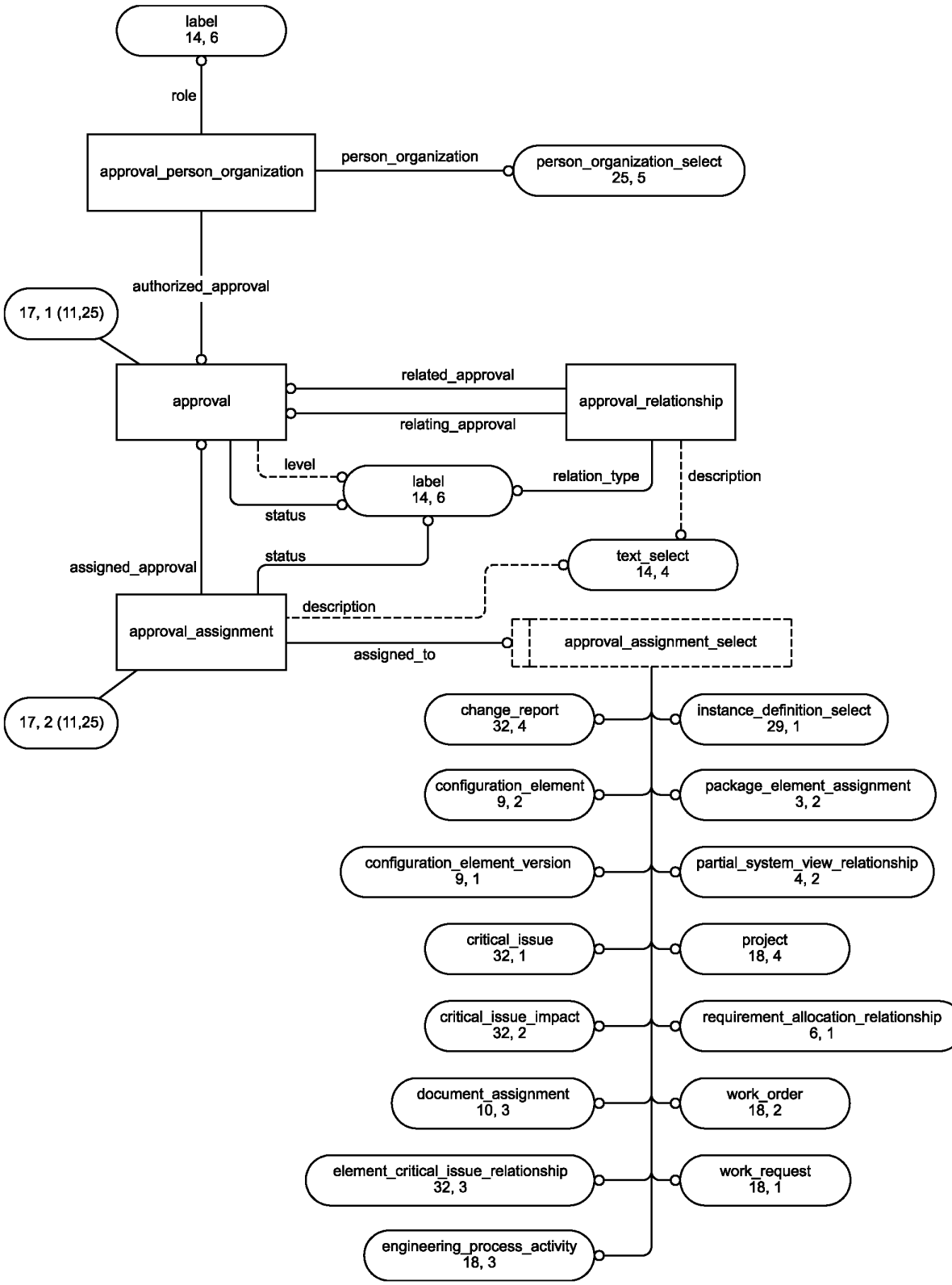


Рисунок Е.18 — Диаграмма ссылочной модели приложения
на уровне сущности в нотации EXPRESS-G (17 из 41)

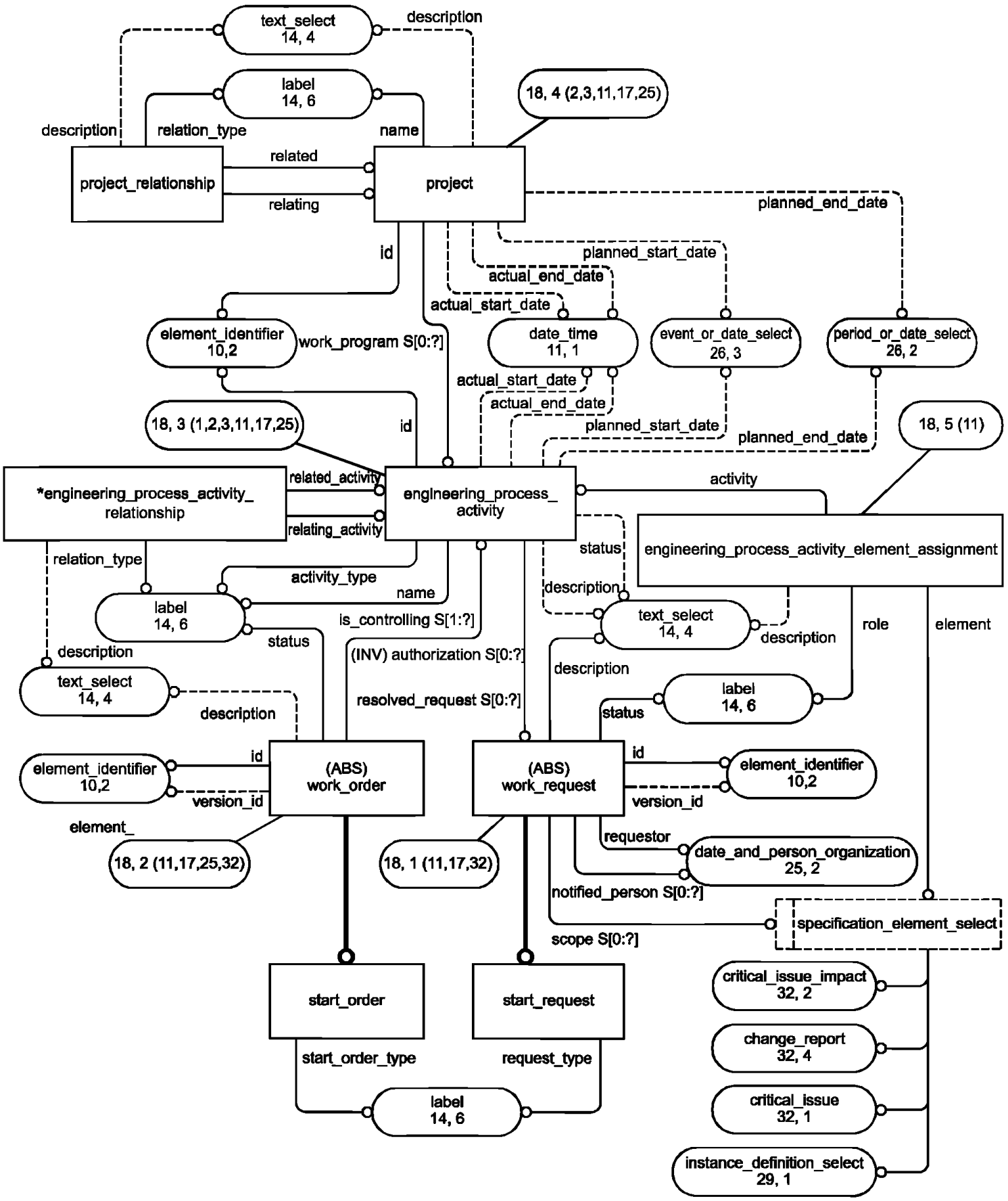
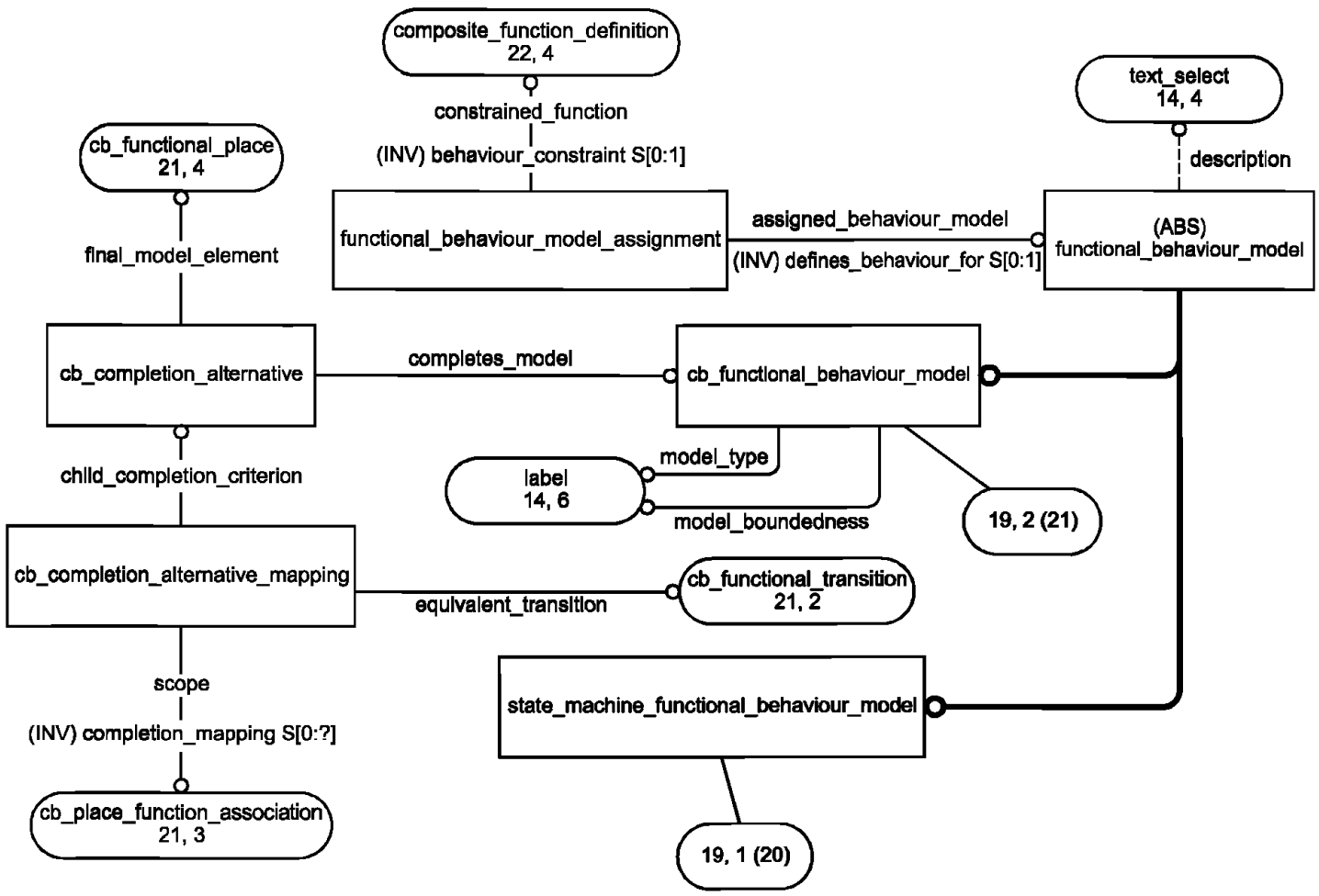


Рисунок Е.19 — Диаграмма ссылочной модели приложения
на уровне сущности в нотации EXPRESS-G (18 из 41)



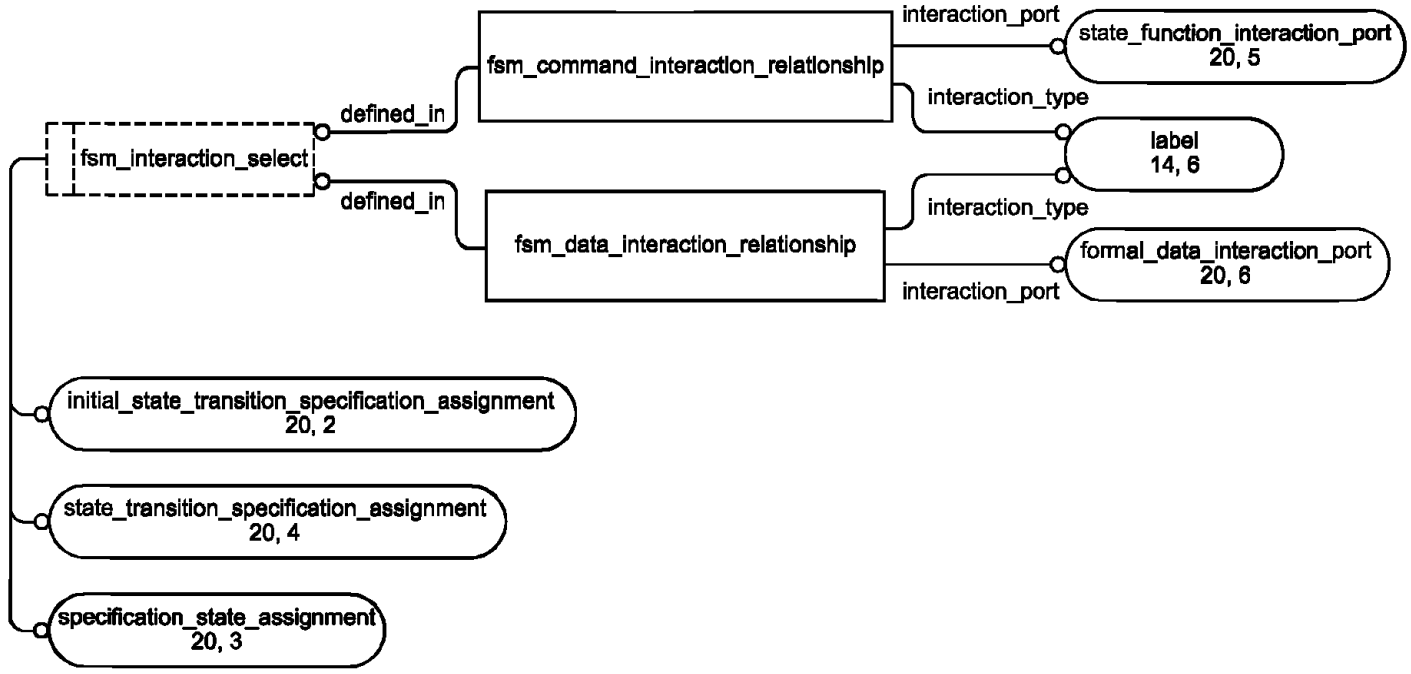


Рисунок Е.20 — Диаграмма ссылочной модели приложения
на уровне сущности в нотации EXPRESS-G (19 из 41)

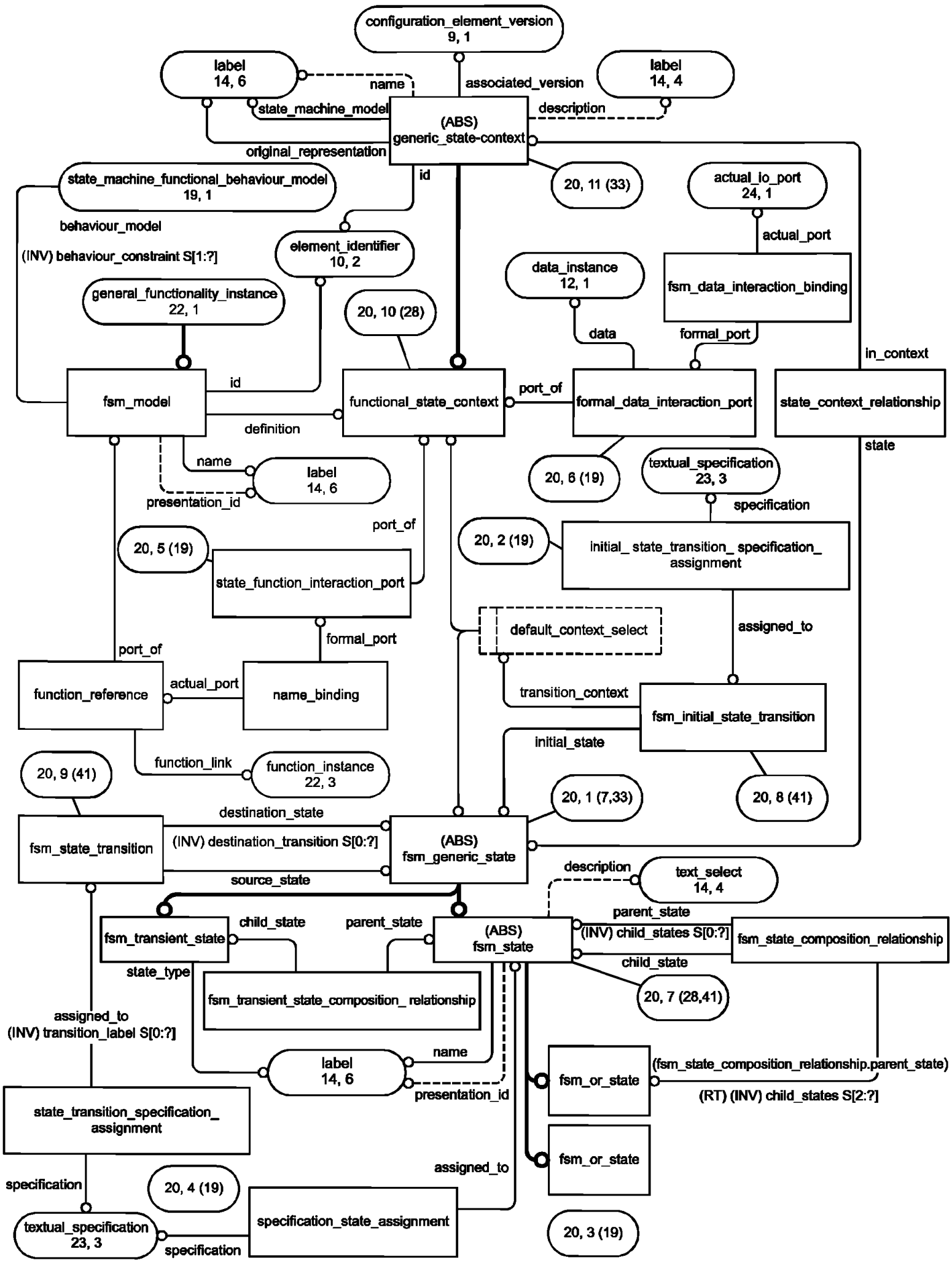


Рисунок Е.21 — Диаграмма ссылочной модели приложения на уровне сущности в нотации EXPRESS-G (20 из 41)

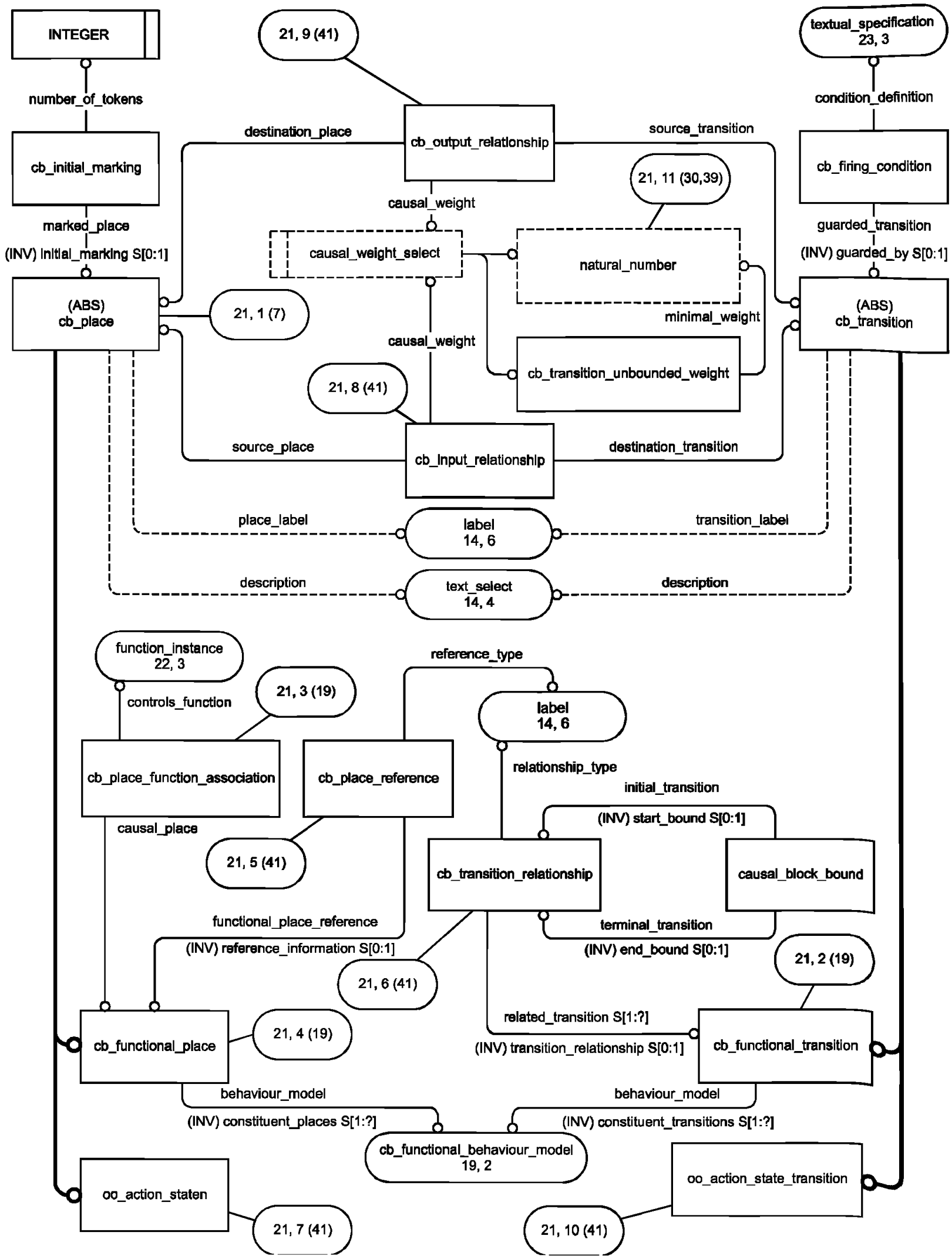


Рисунок Е.22 — Диаграмма ссылочной модели приложения на уровне сущности в нотации EXPRESS-G (21 из 41)

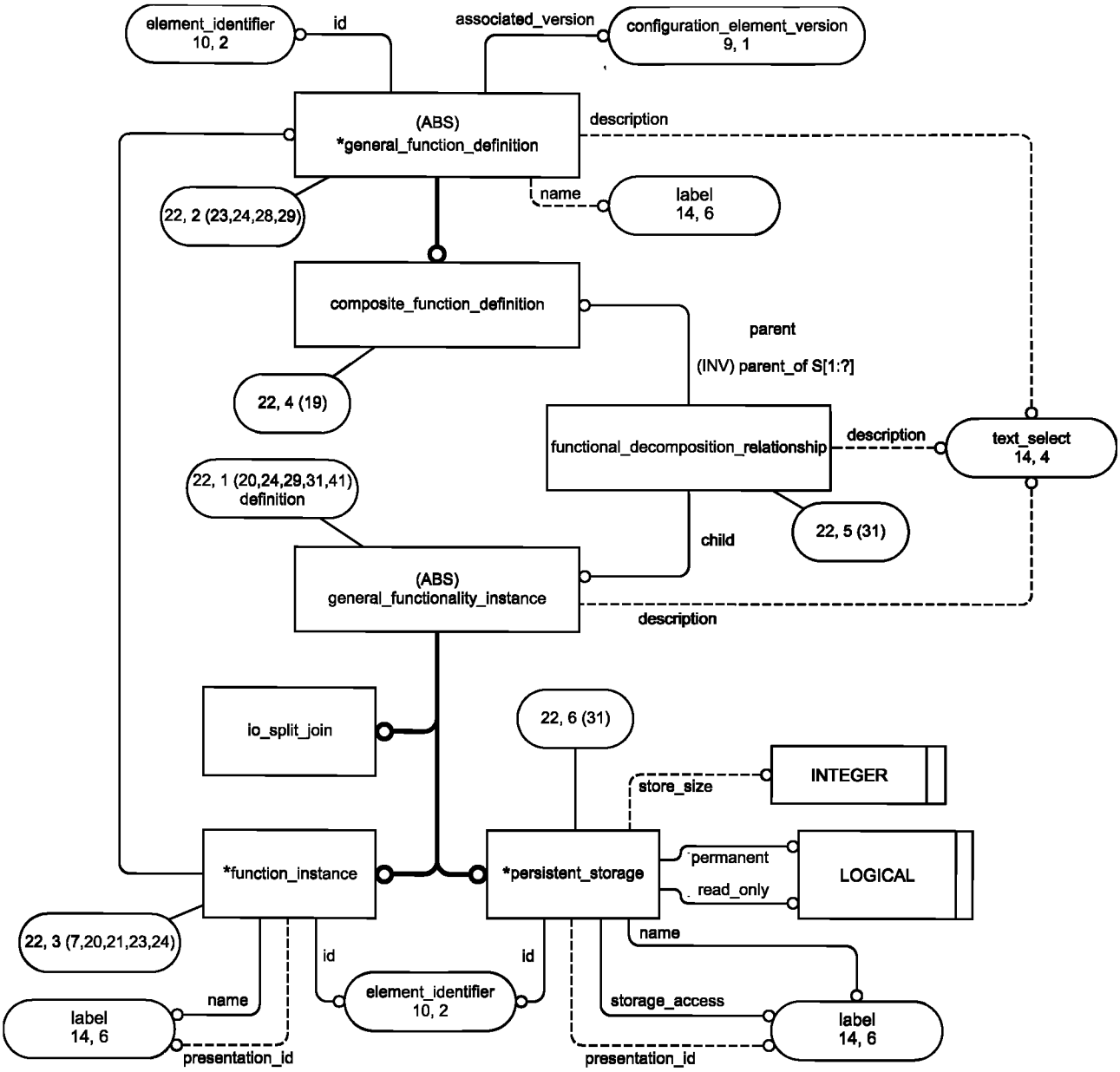


Рисунок Е.23 — Диаграмма ссылочной модели приложения
на уровне сущности в нотации EXPRESS-G (22 из 41)

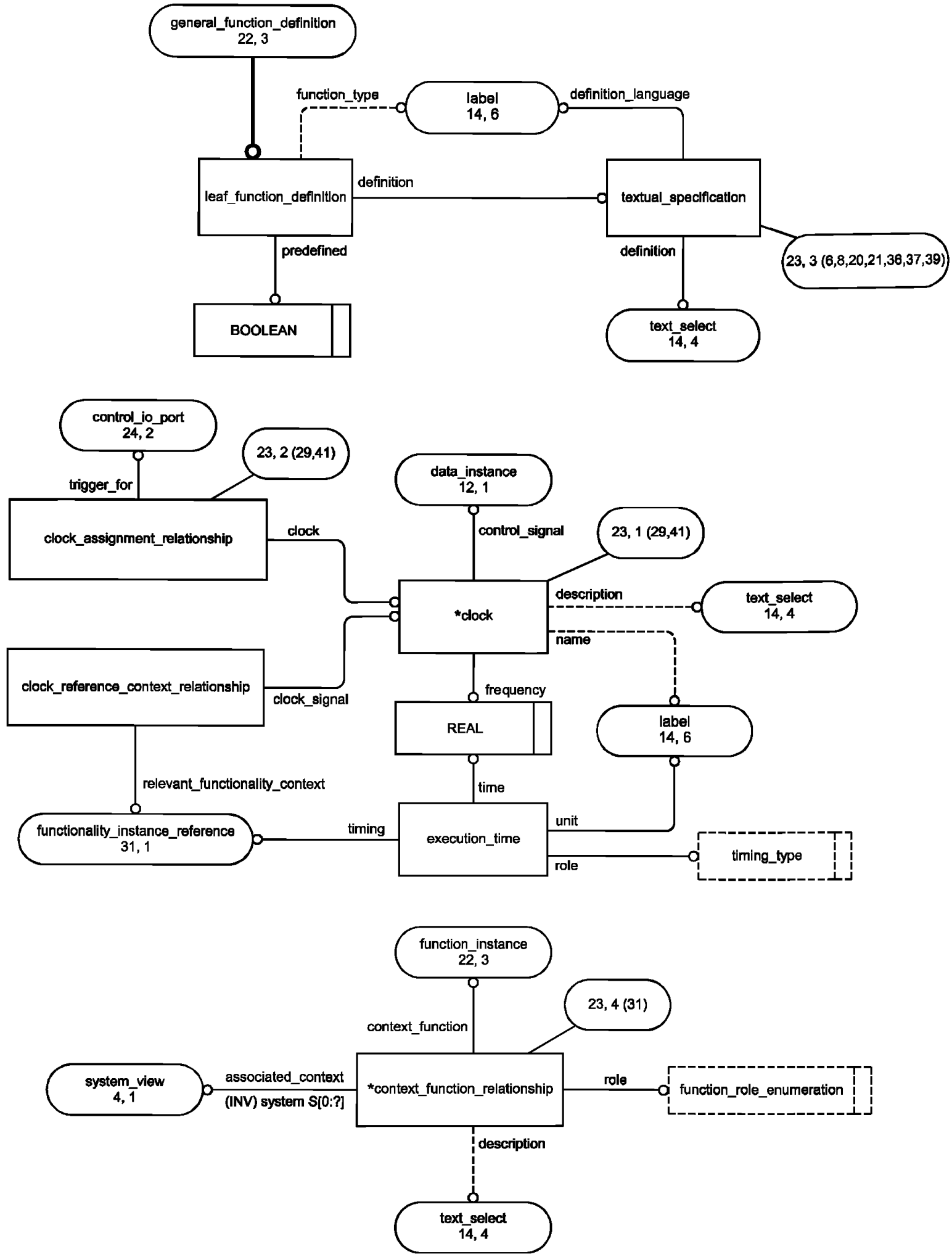


Рисунок Е.24 — Диаграмма ссылочной модели приложения
на уровне сущности в нотации EXPRESS-G (23 из 41)

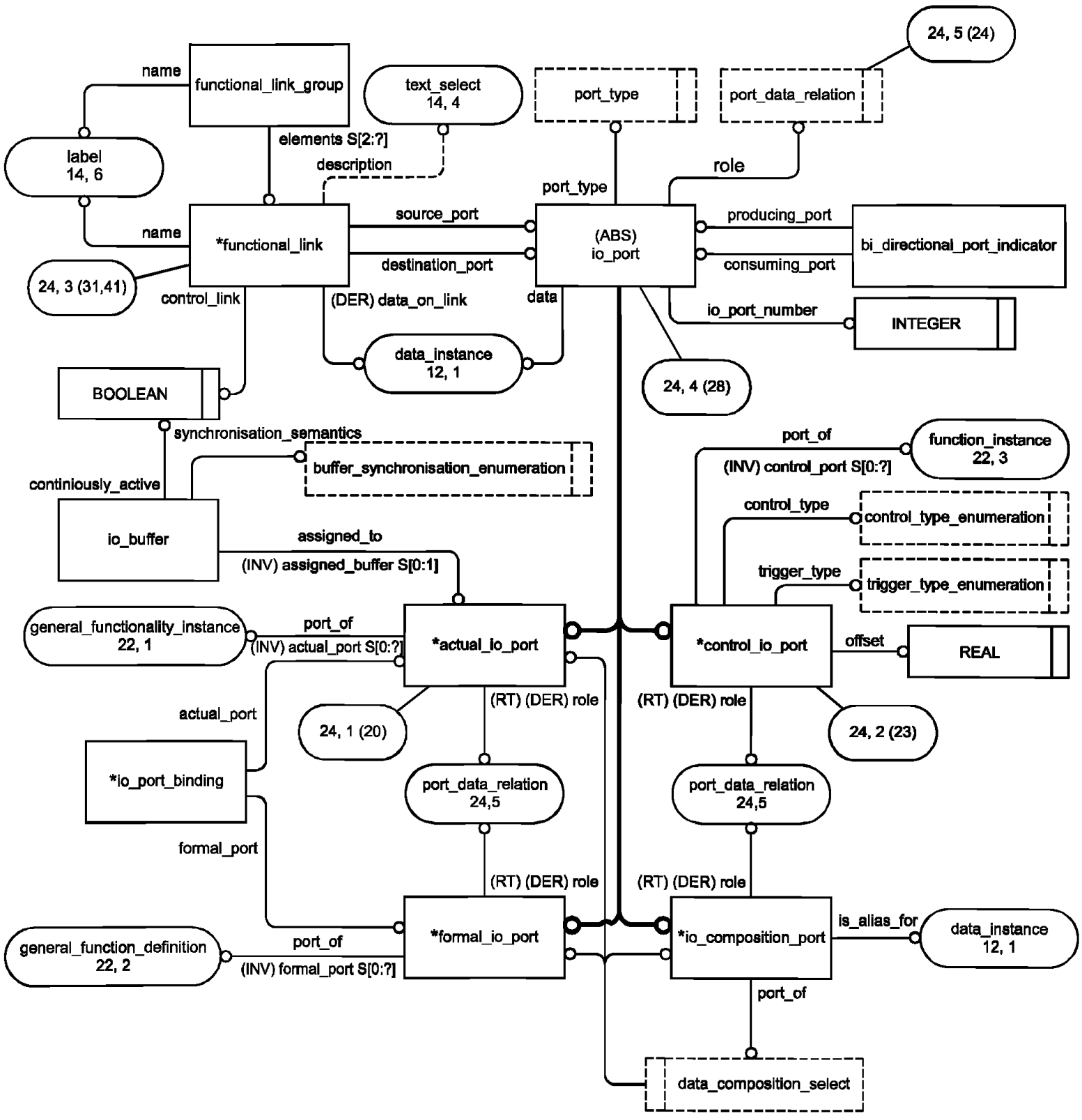


Рисунок Е.25 — Диаграмма ссылочной модели приложения на уровне сущности в нотации EXPRESS-G (24 из 41)

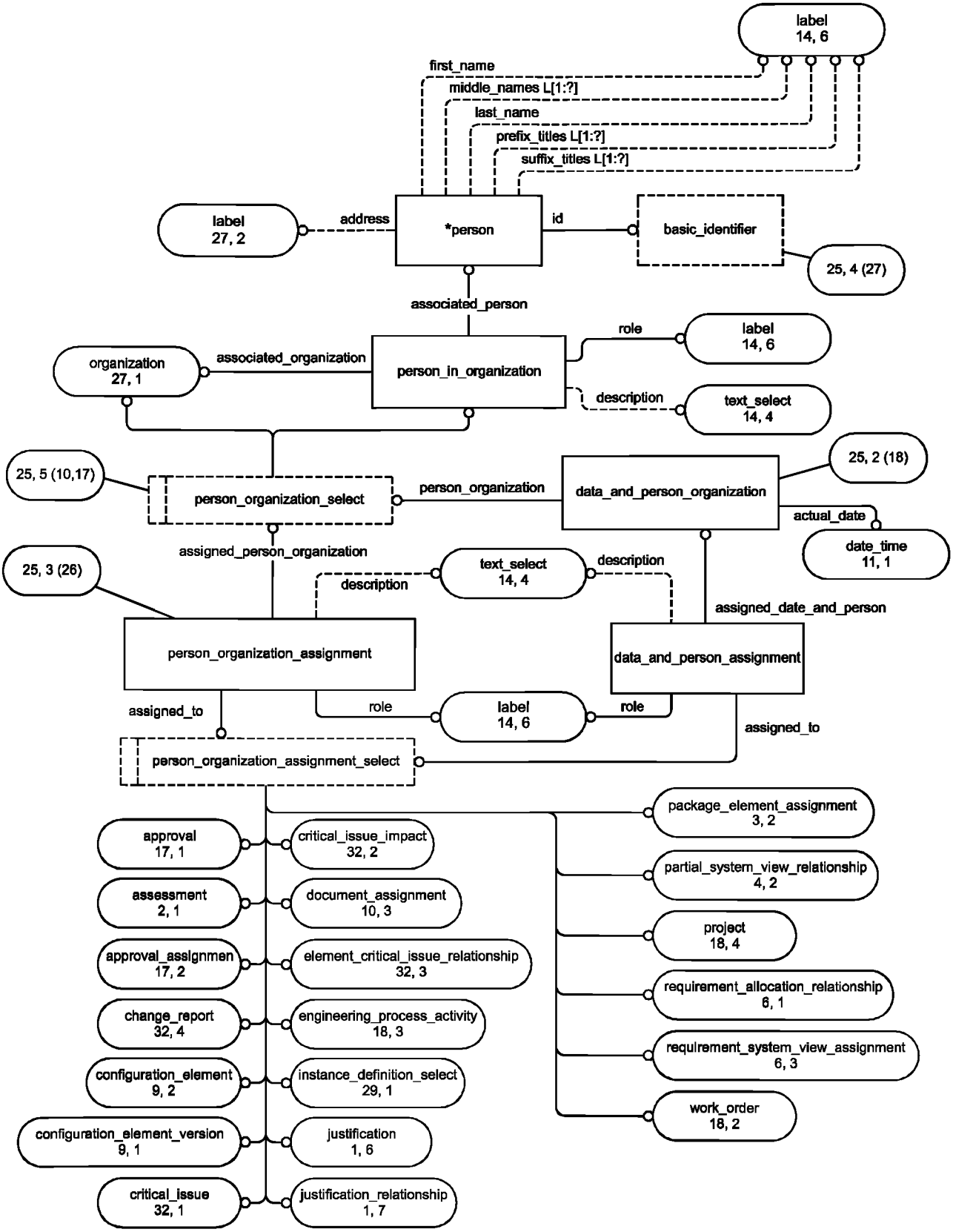


Рисунок Е.26 — Диаграмма ссылочной модели приложения
на уровне сущности в нотации EXPRESS-G (25 из 41)

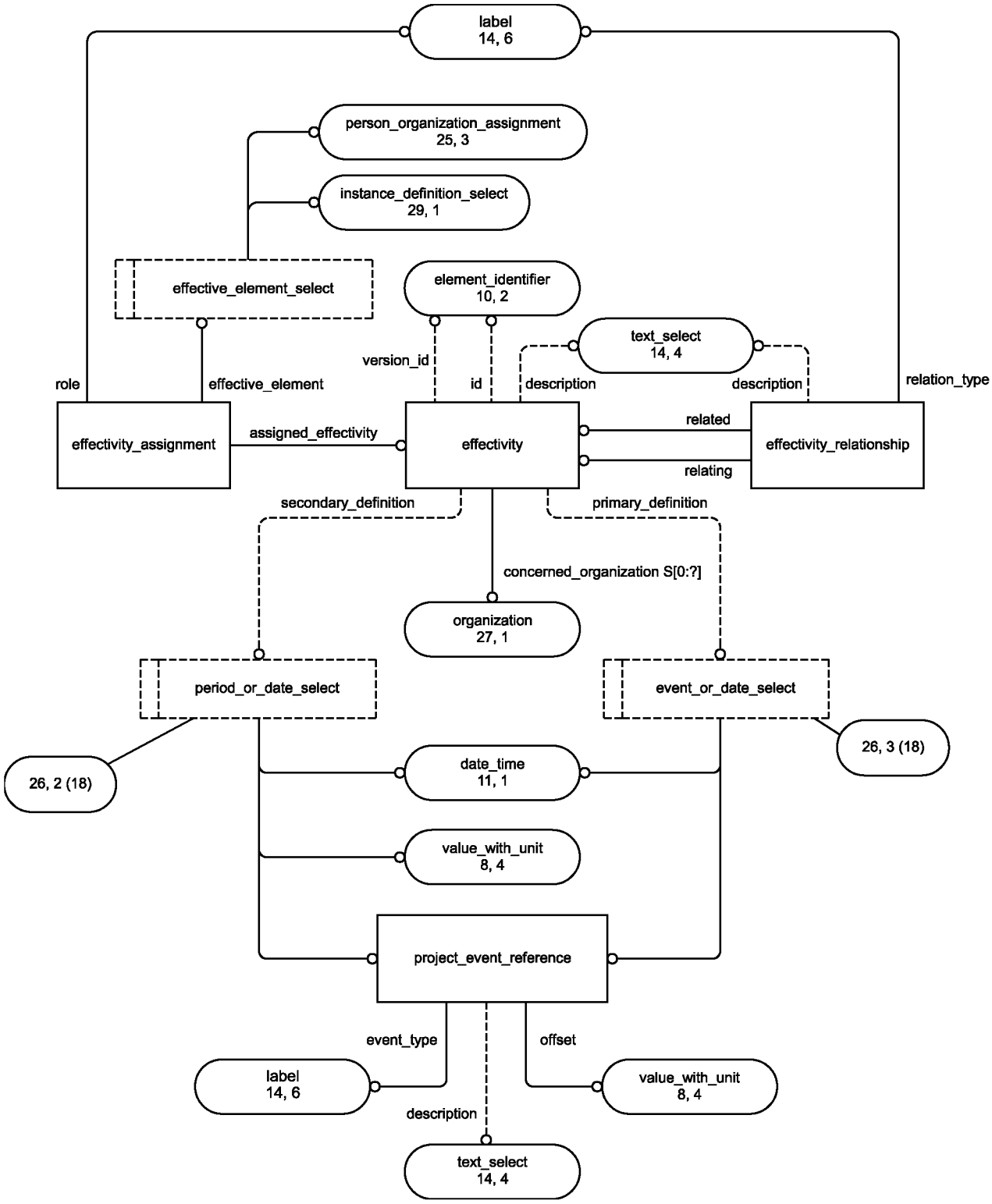


Рисунок Е.27 — Диаграмма ссылочной модели приложения
на уровне сущности в нотации EXPRESS-G (26 из 41)

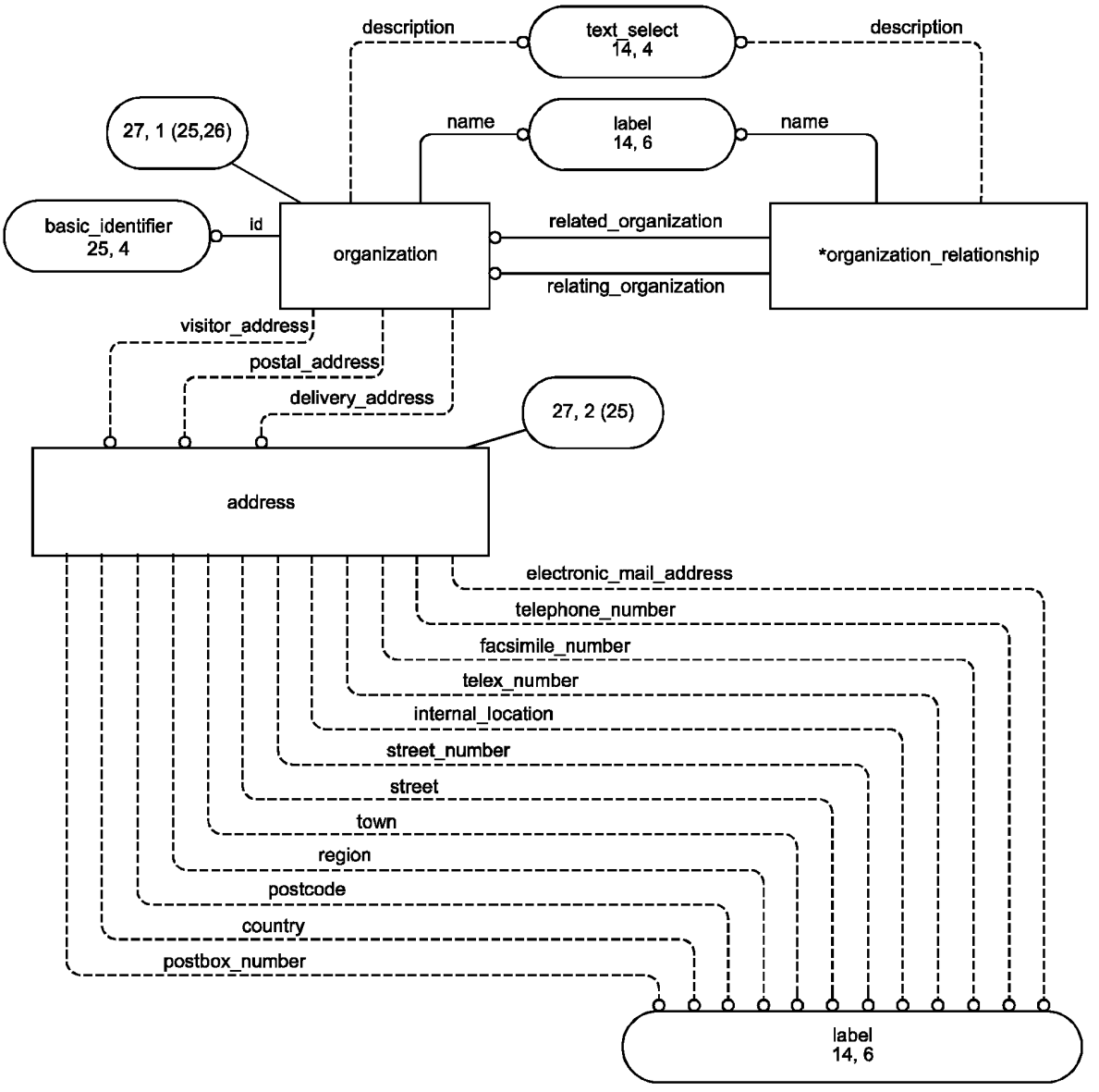


Рисунок Е.28 — Диаграмма ссылочной модели приложения
на уровне сущности в нотации EXPRESS-G (27 из 41)

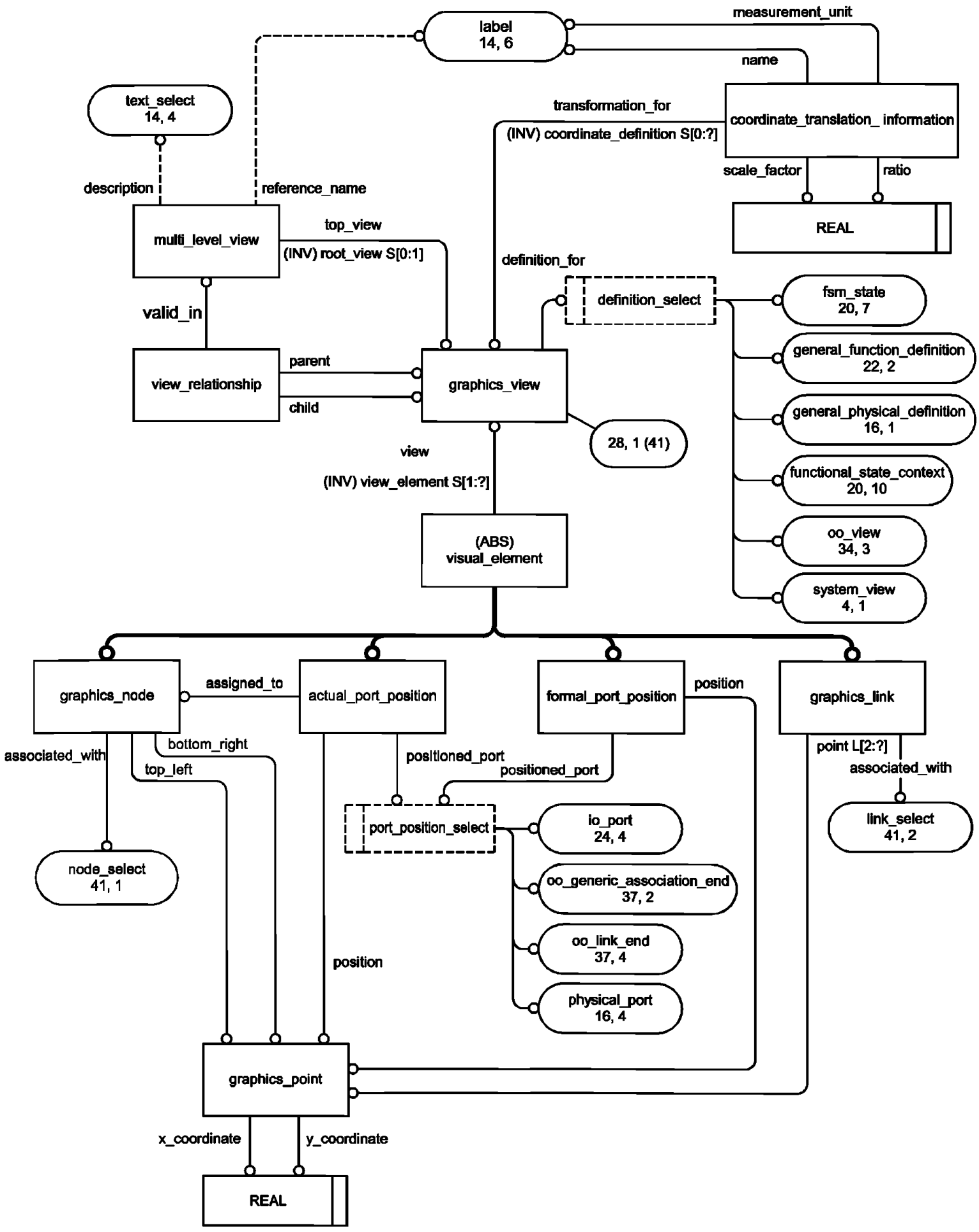


Рисунок Е.29 — Диаграмма ссылочной модели приложения
на уровне сущности в нотации EXPRESS-G (28 из 41)

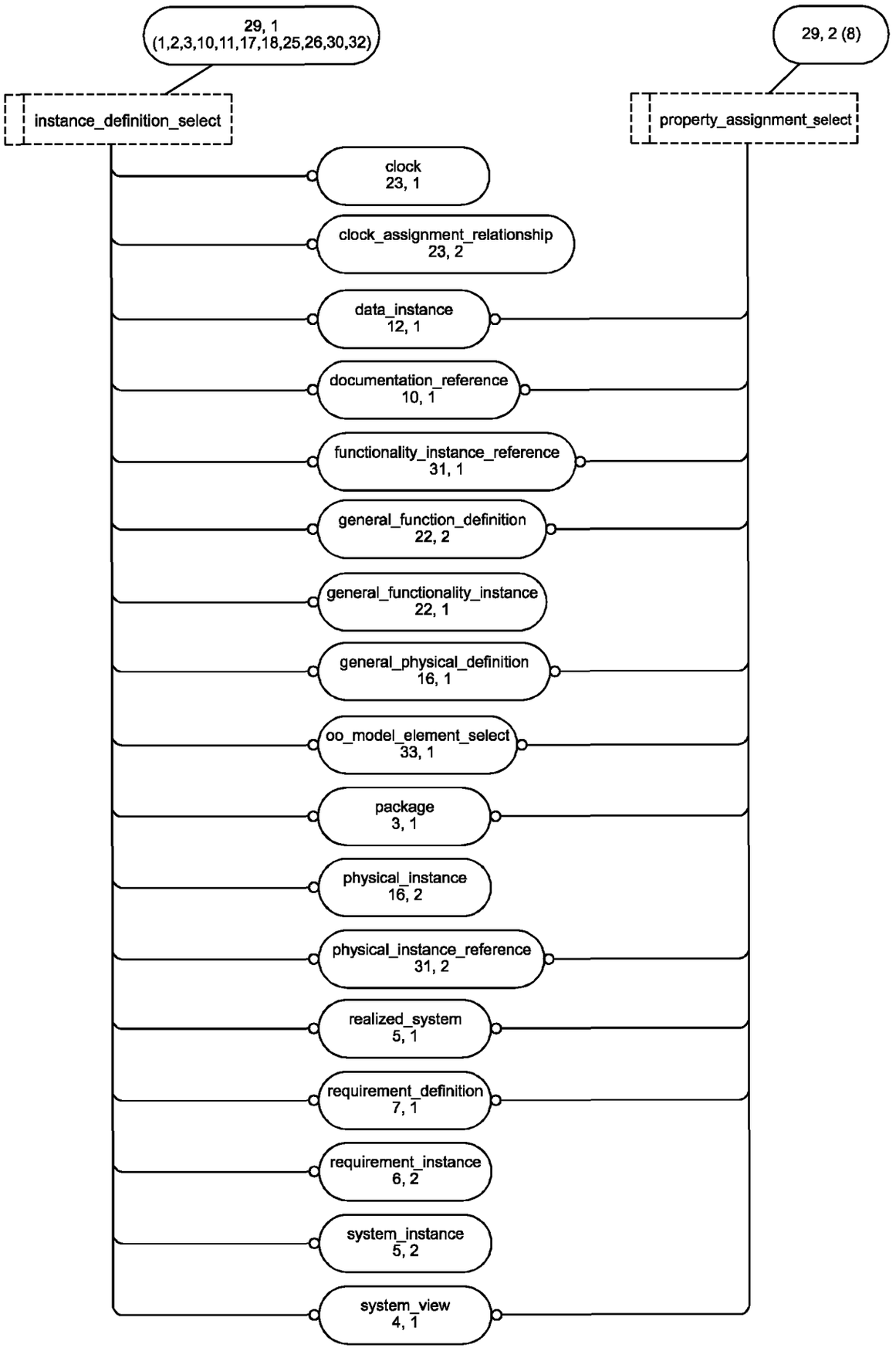


Рисунок Е.30 — Диаграмма ссылочной модели приложения на уровне сущности в нотации EXPRESS-G (29 из 41)

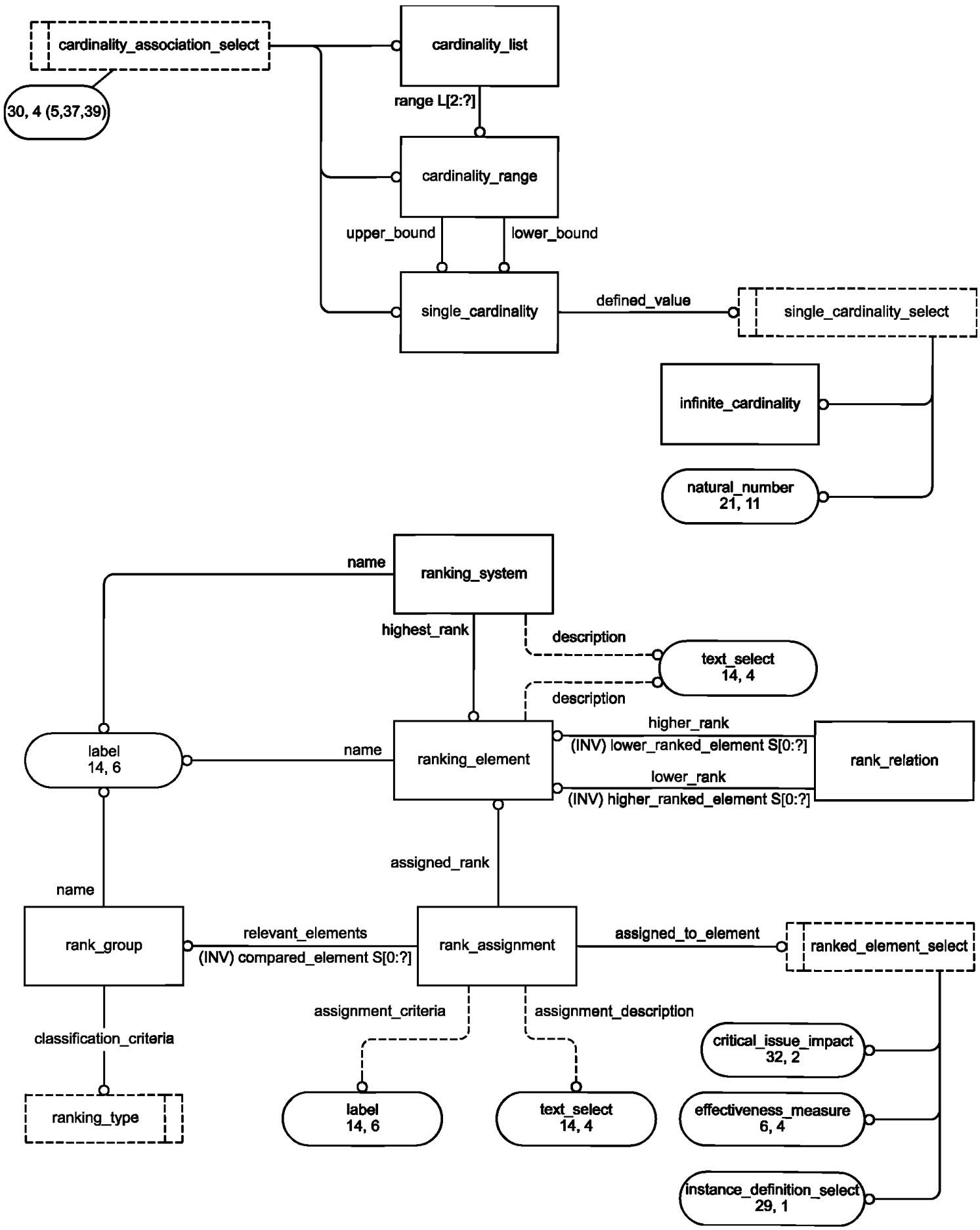


Рисунок Е.31 — Диаграмма ссылочной модели приложения
на уровне сущности в нотации EXPRESS-G (30 из 41)

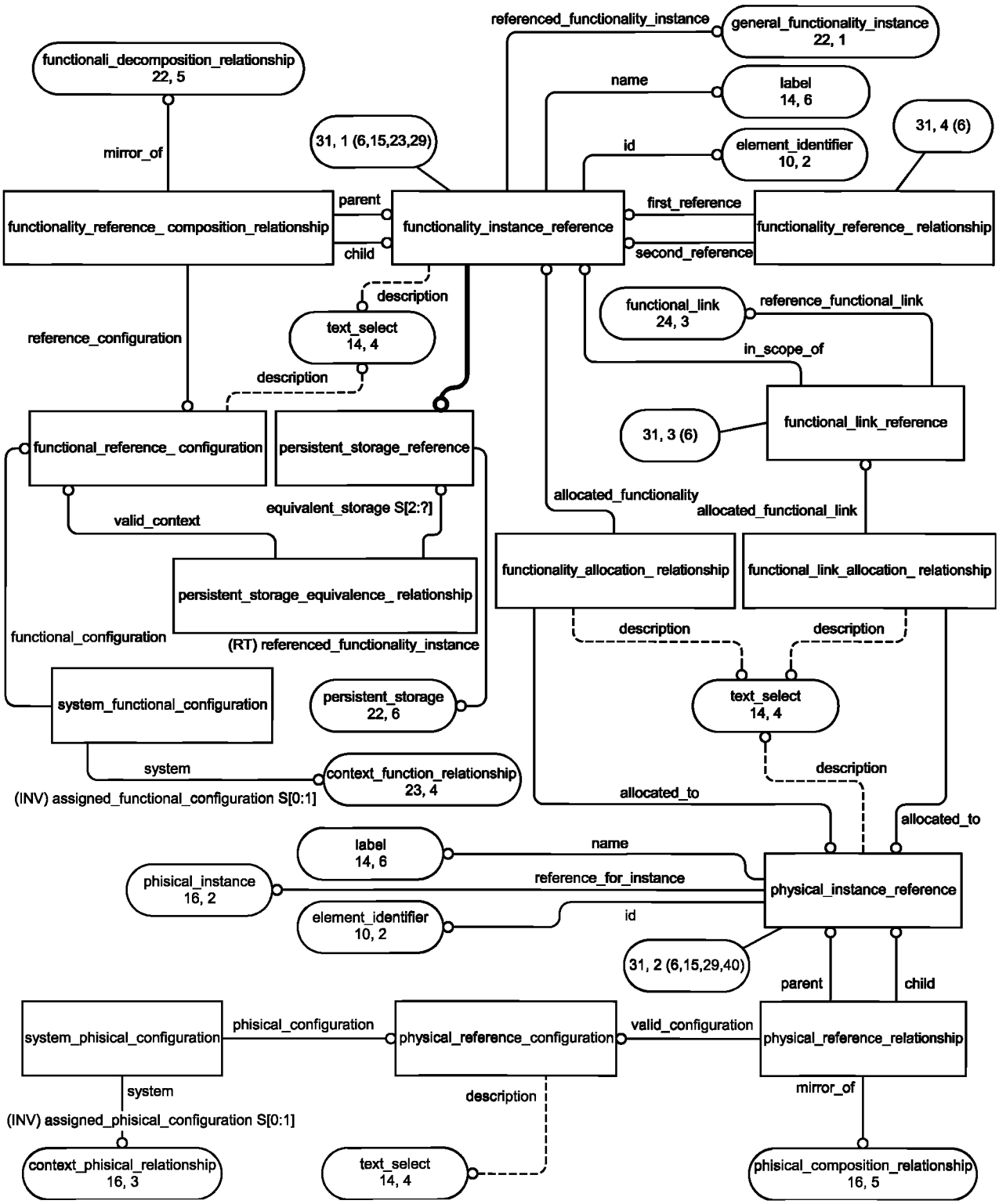


Рисунок Е.32 — Диаграмма ссылочной модели приложения
на уровне сущности в нотации EXPRESS-G (31 из 41)

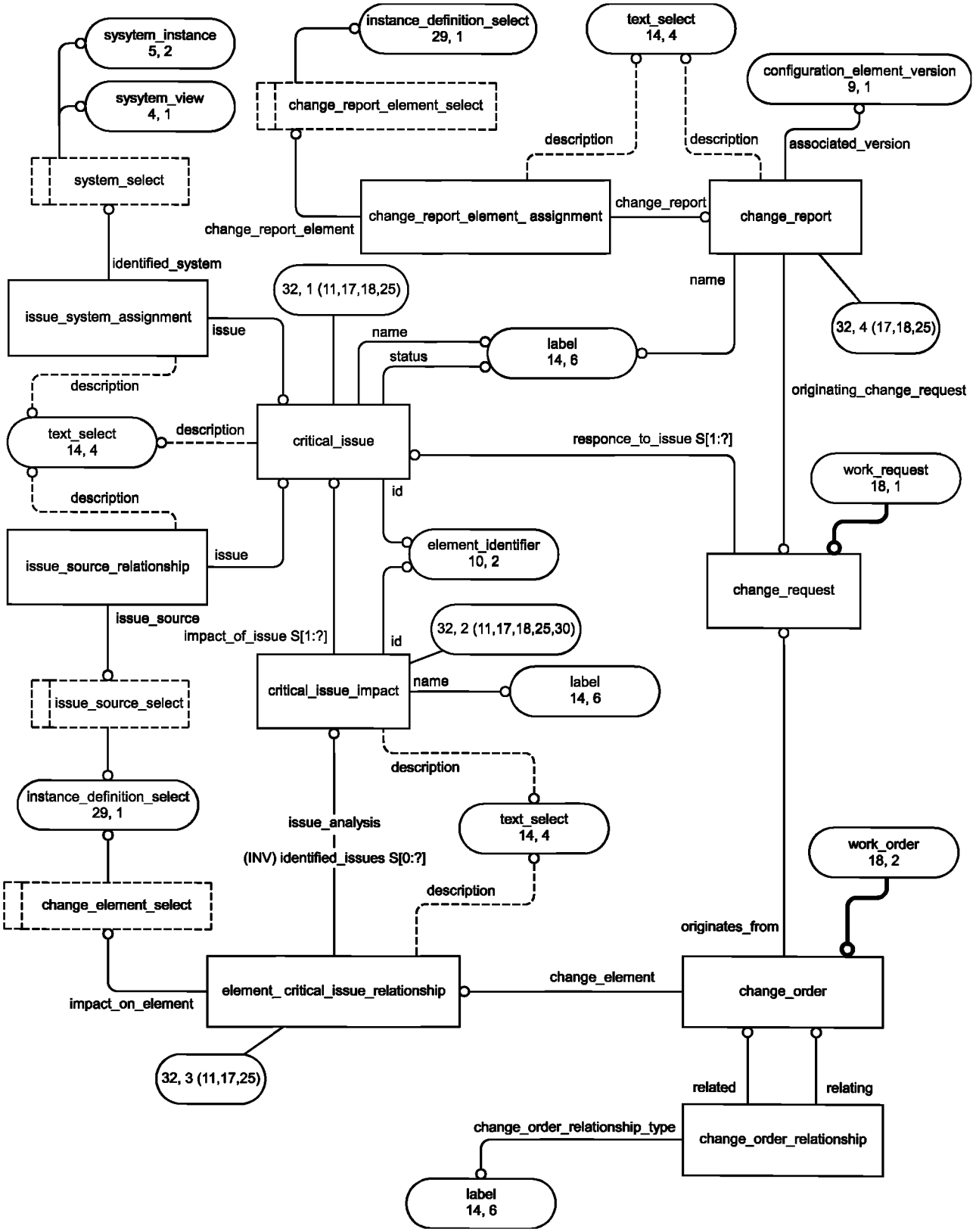


Рисунок Е.33 — Диаграмма ссылочной модели приложения
на уровне сущности в нотации EXPRESS-G (32 из 41)

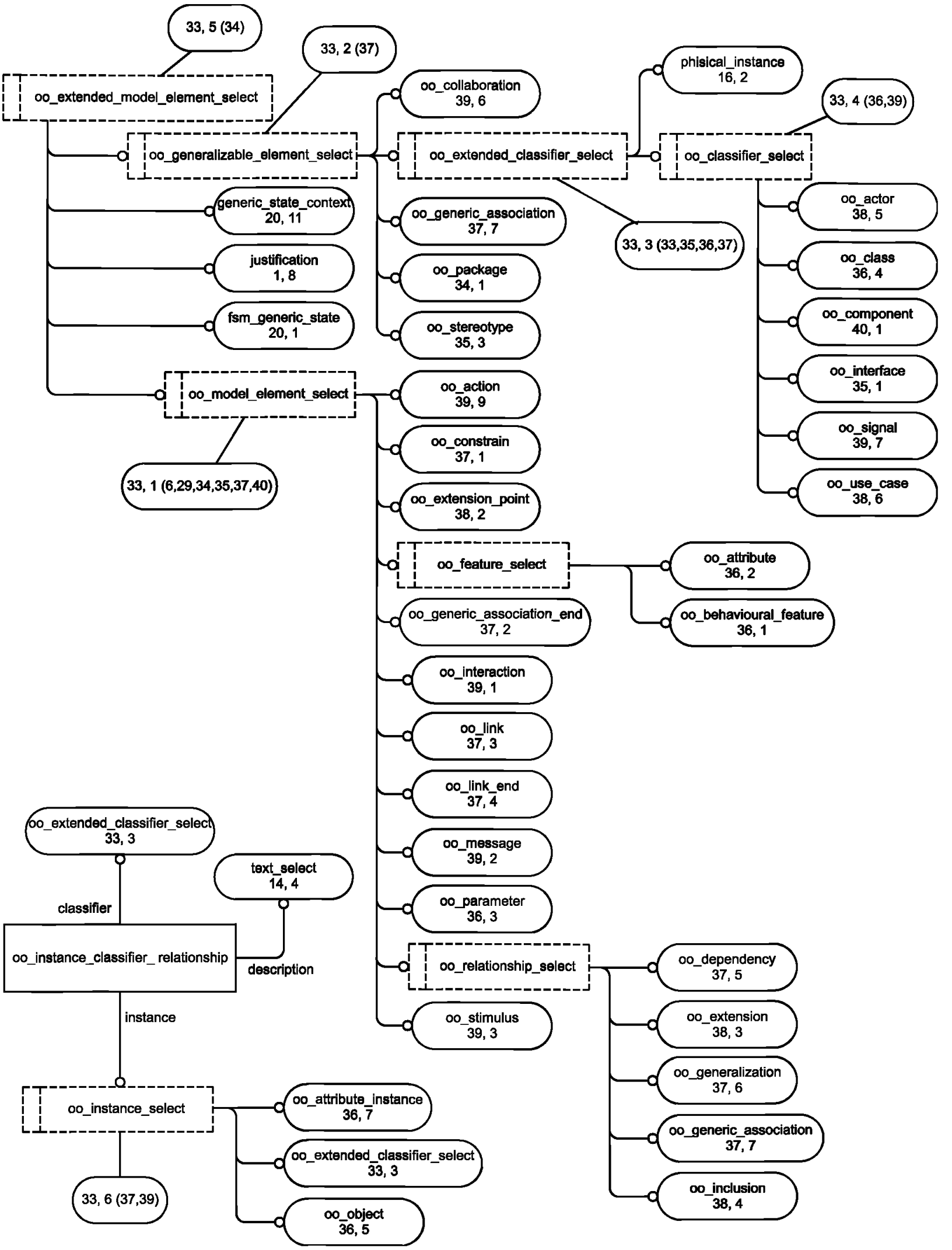


Рисунок Е.34 — Диаграмма ссылочной модели приложения на уровне сущности в нотации EXPRESS-G (33 из 41)

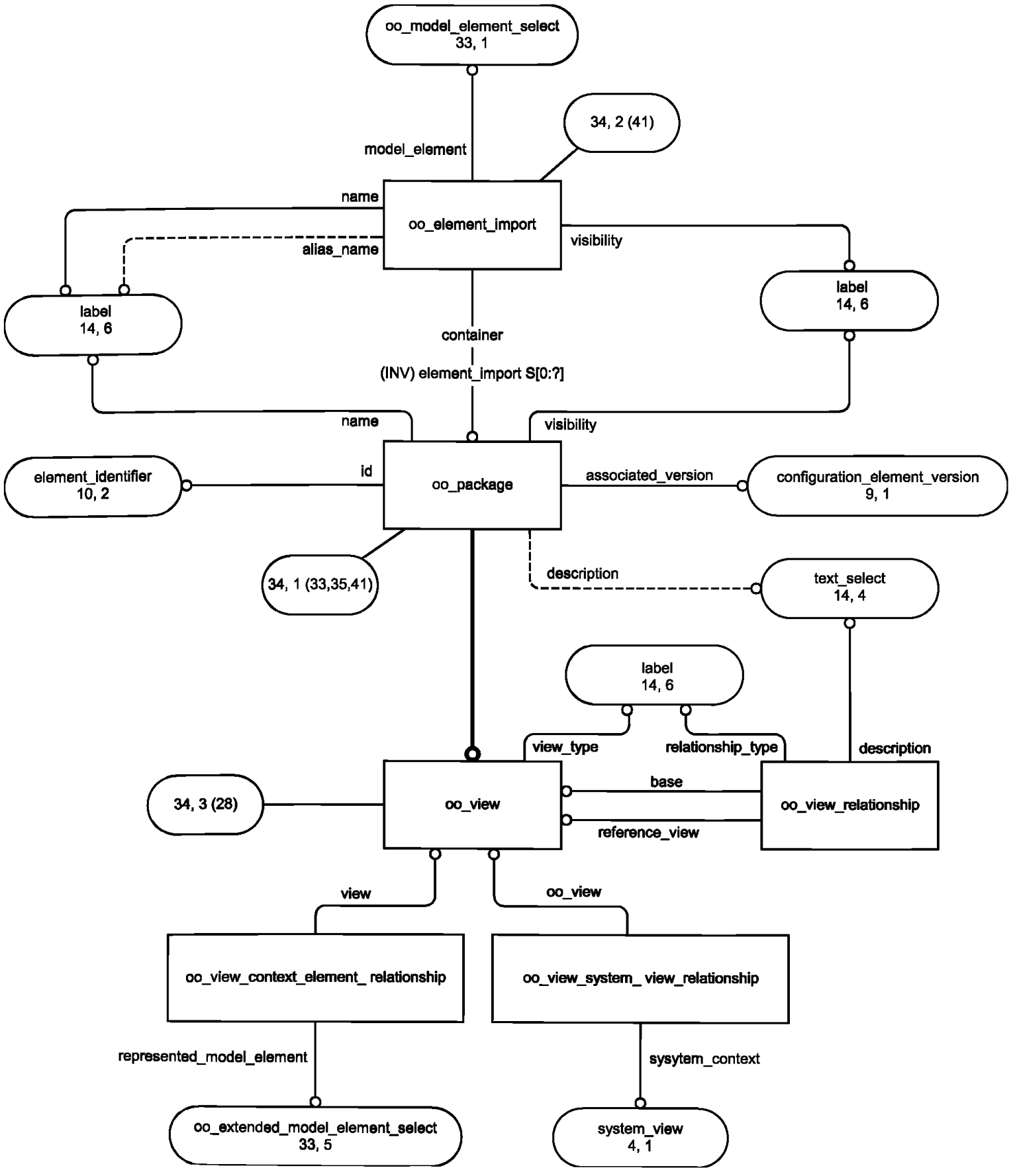


Рисунок Е.35 — Диаграмма ссылочной модели приложения на уровне сущности в нотации EXPRESS-G (34 из 41)

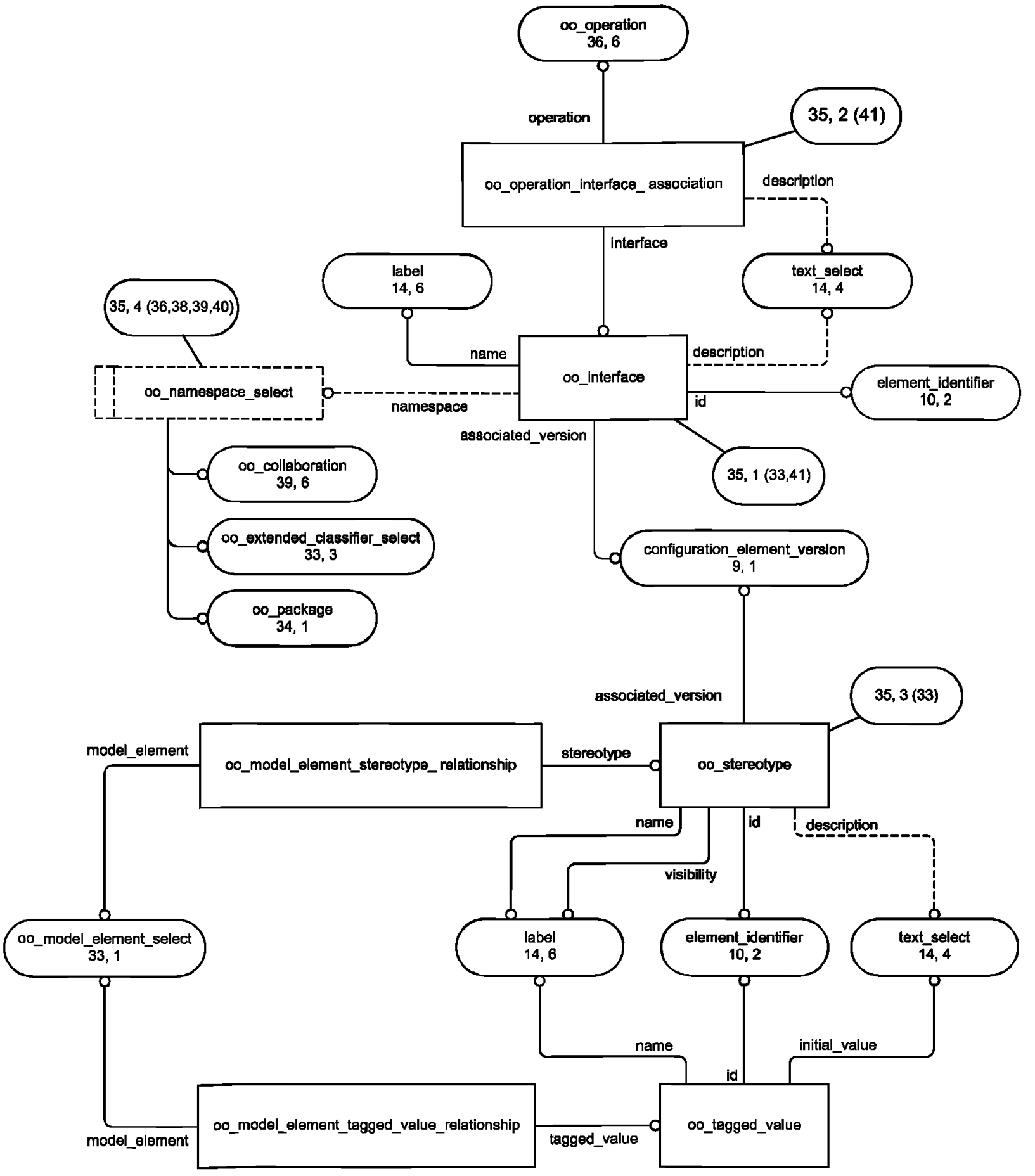


Рисунок Е.36 — Диаграмма ссылочной модели приложения
на уровне сущности в нотации EXPRESS-G (35 из 41)

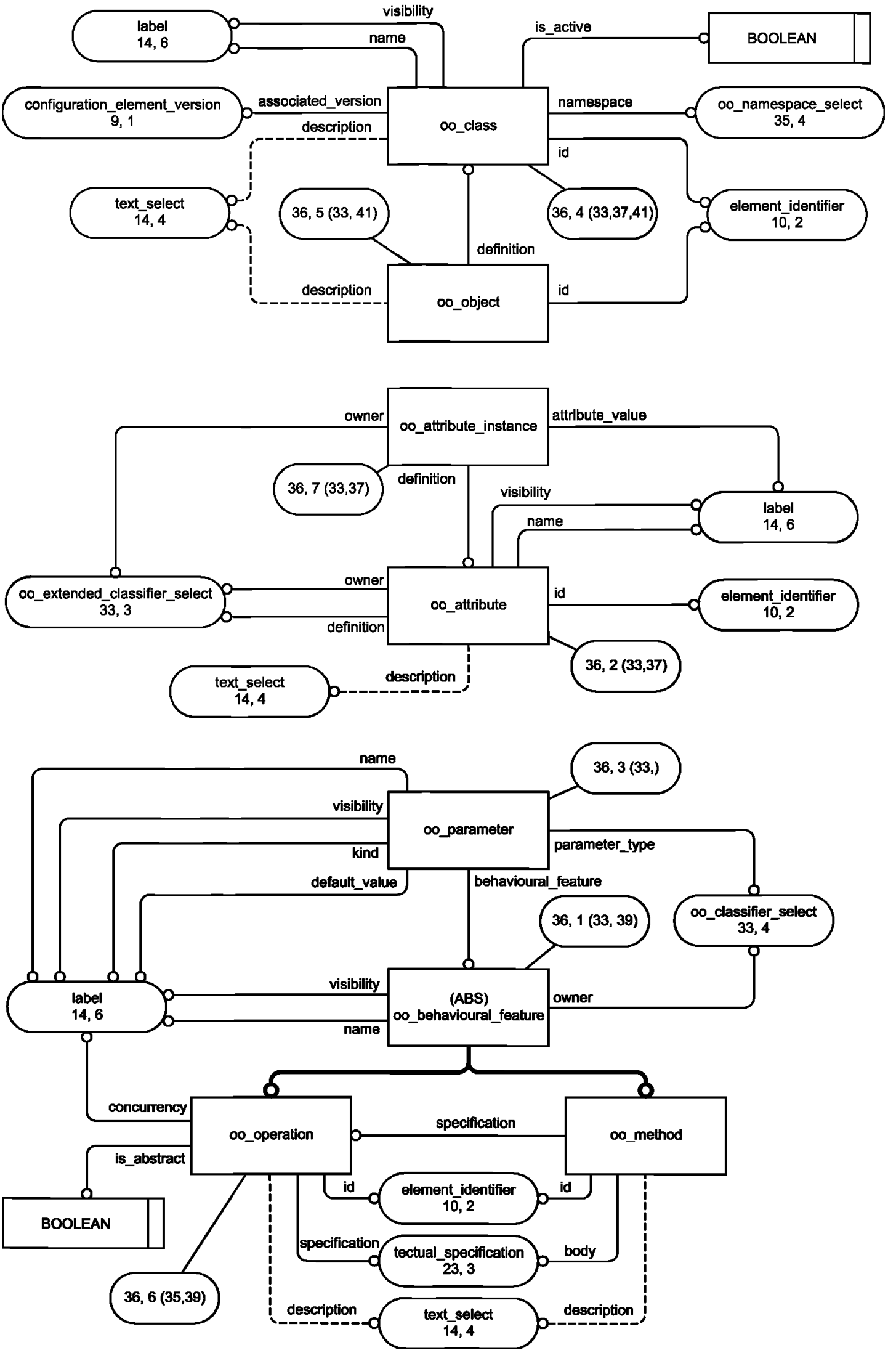


Рисунок Е.37 — Диаграмма ссылочной модели приложения на уровне сущности в нотации EXPRESS-G (36 из 41)

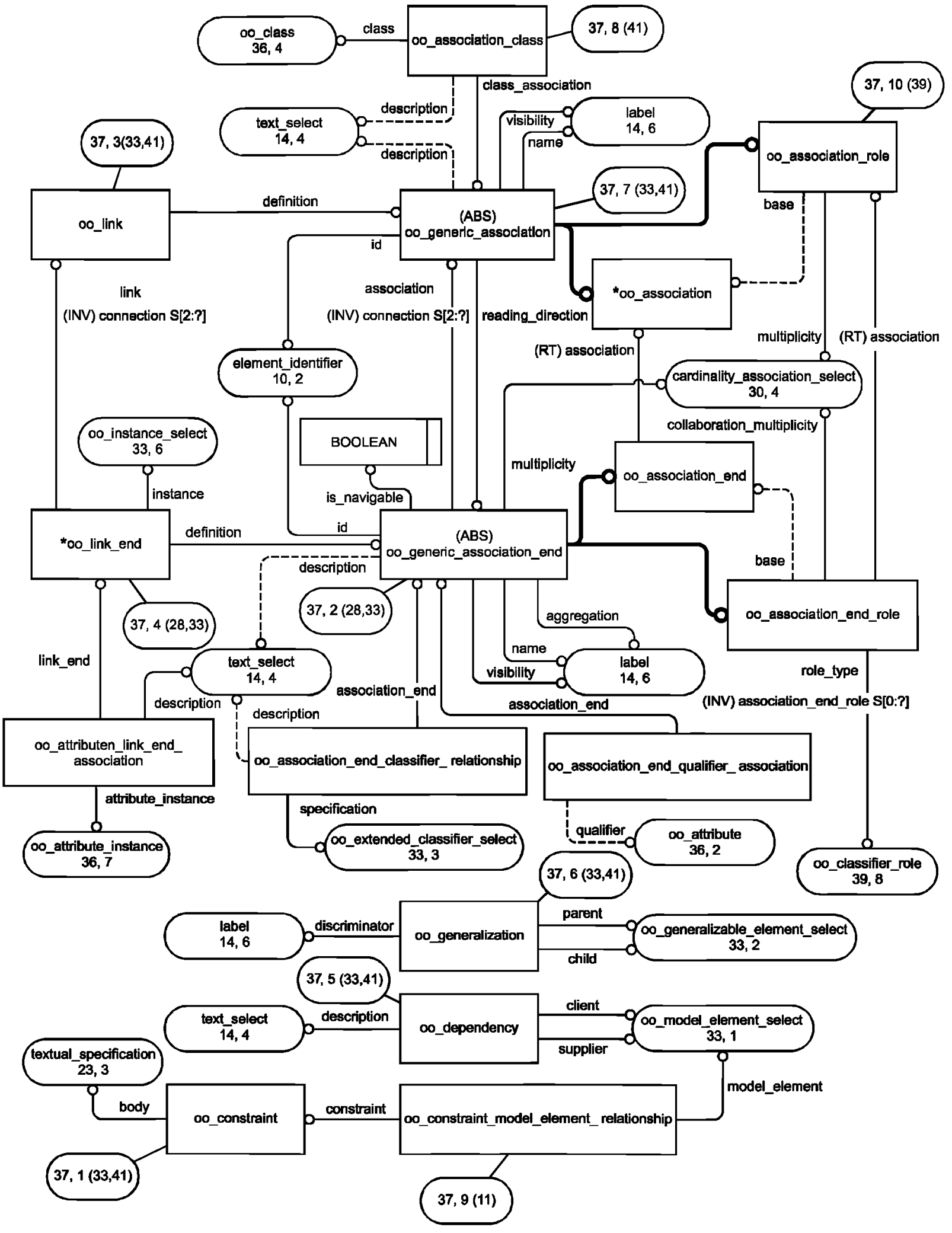


Рисунок Е.38 — Диаграмма ссылочной модели приложения
на уровне сущности в нотации EXPRESS-G (37 из 41)

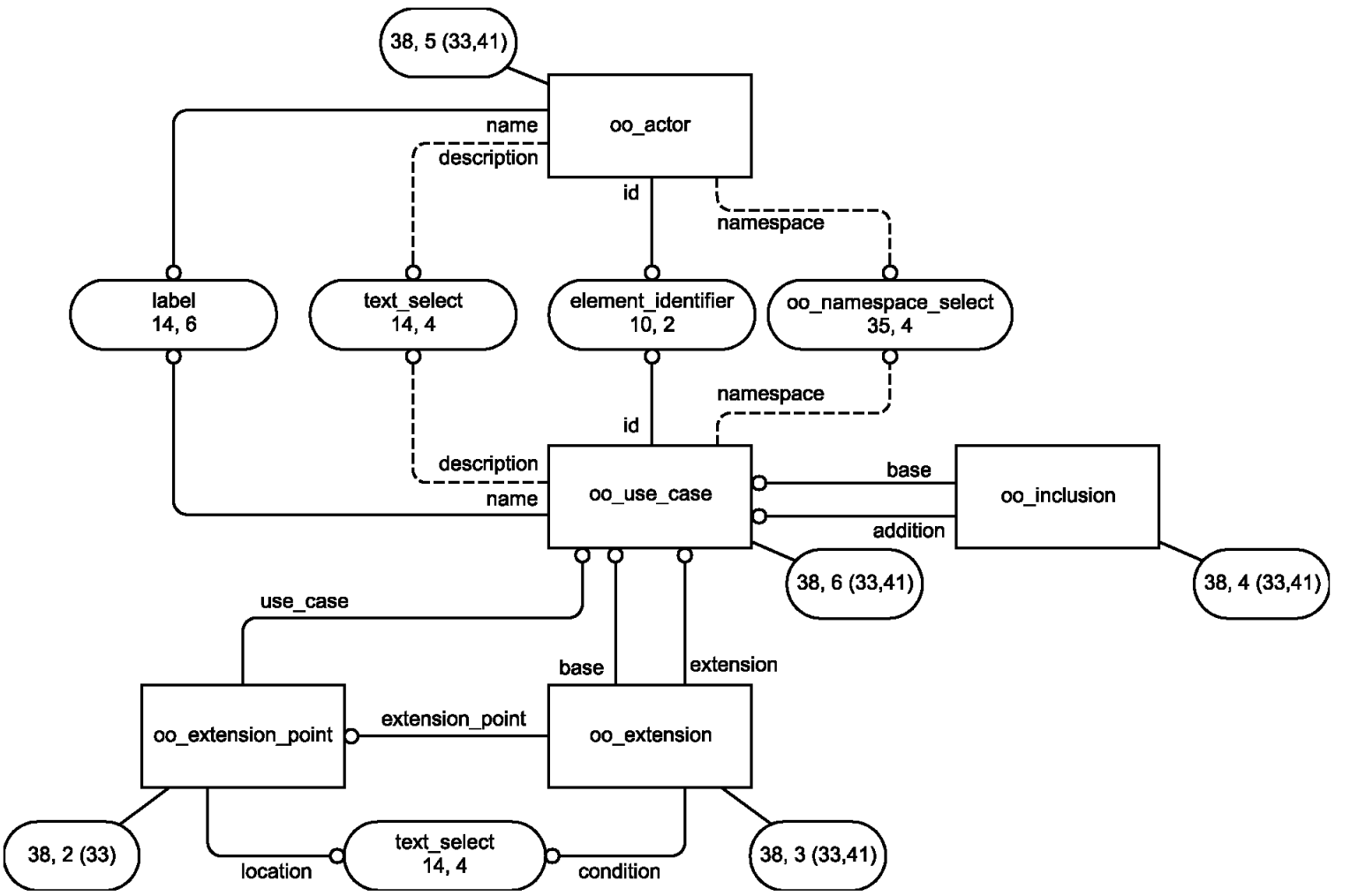


Рисунок Е.39 — Диаграмма ссылочной модели приложения
на уровне сущности в нотации EXPRESS-G (38 из 41)

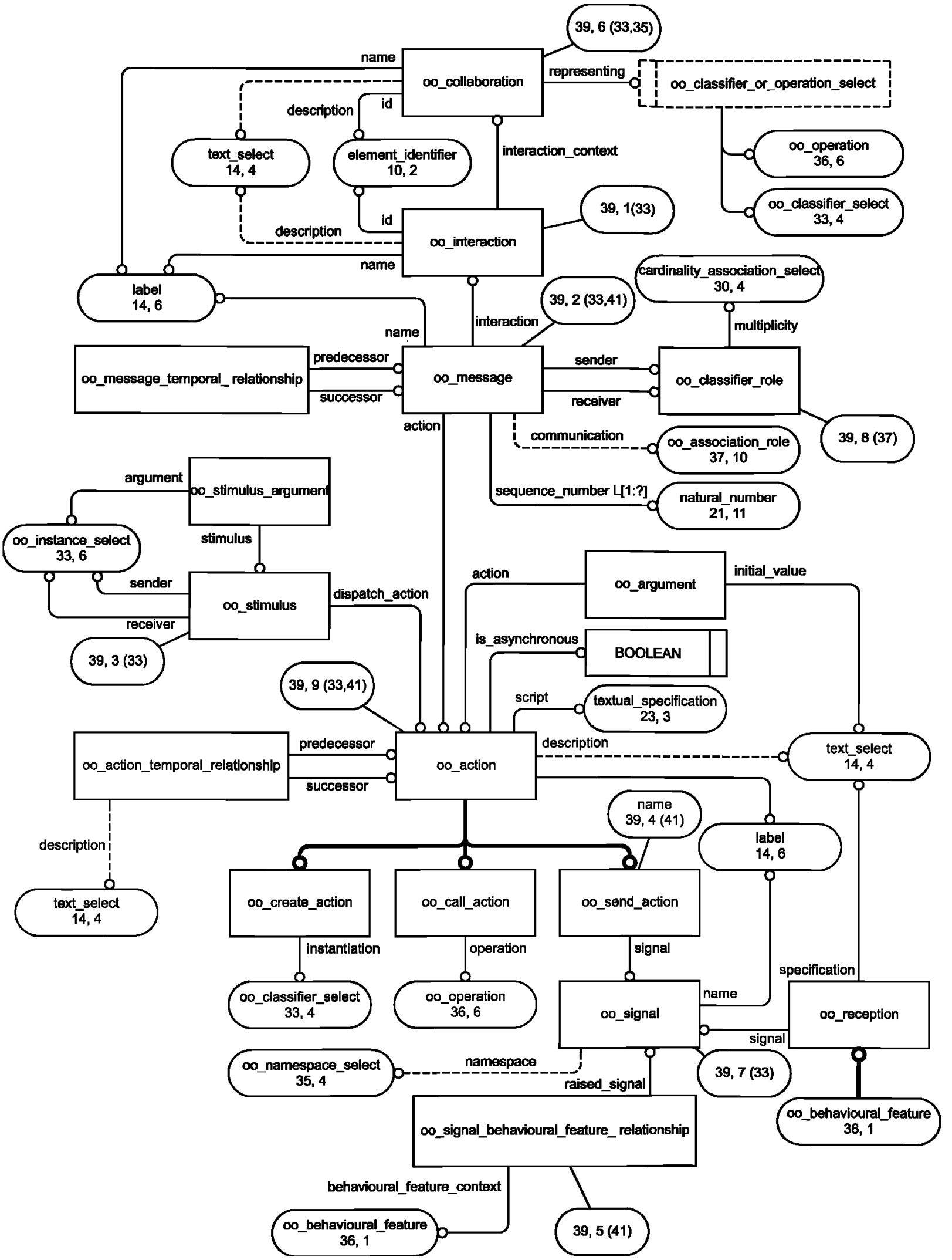


Рисунок Е.40 — Диаграмма ссылочной модели приложения на уровне сущности в нотации EXPRESS-G (39 из 41)

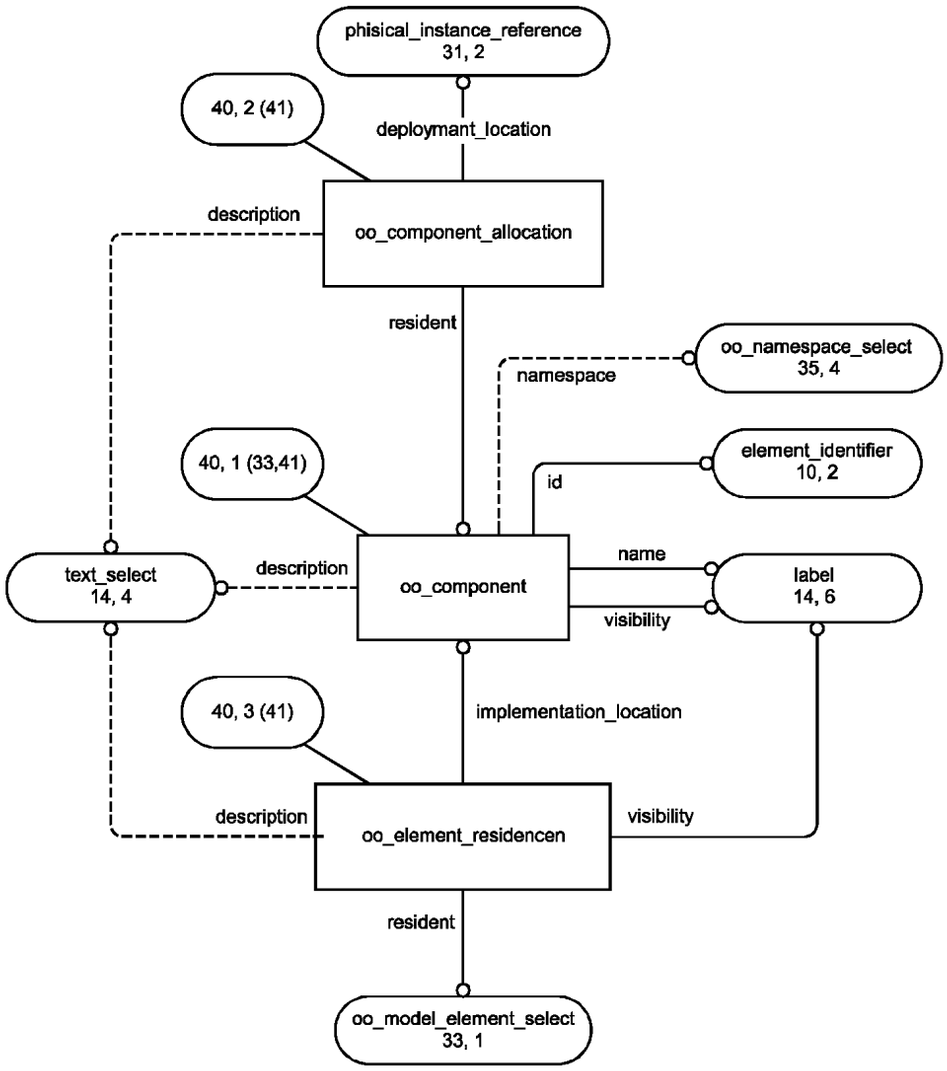


Рисунок Е.41 — Диаграмма ссылочной модели приложения на уровне сущности в нотации EXPRESS-G (40 из 41)

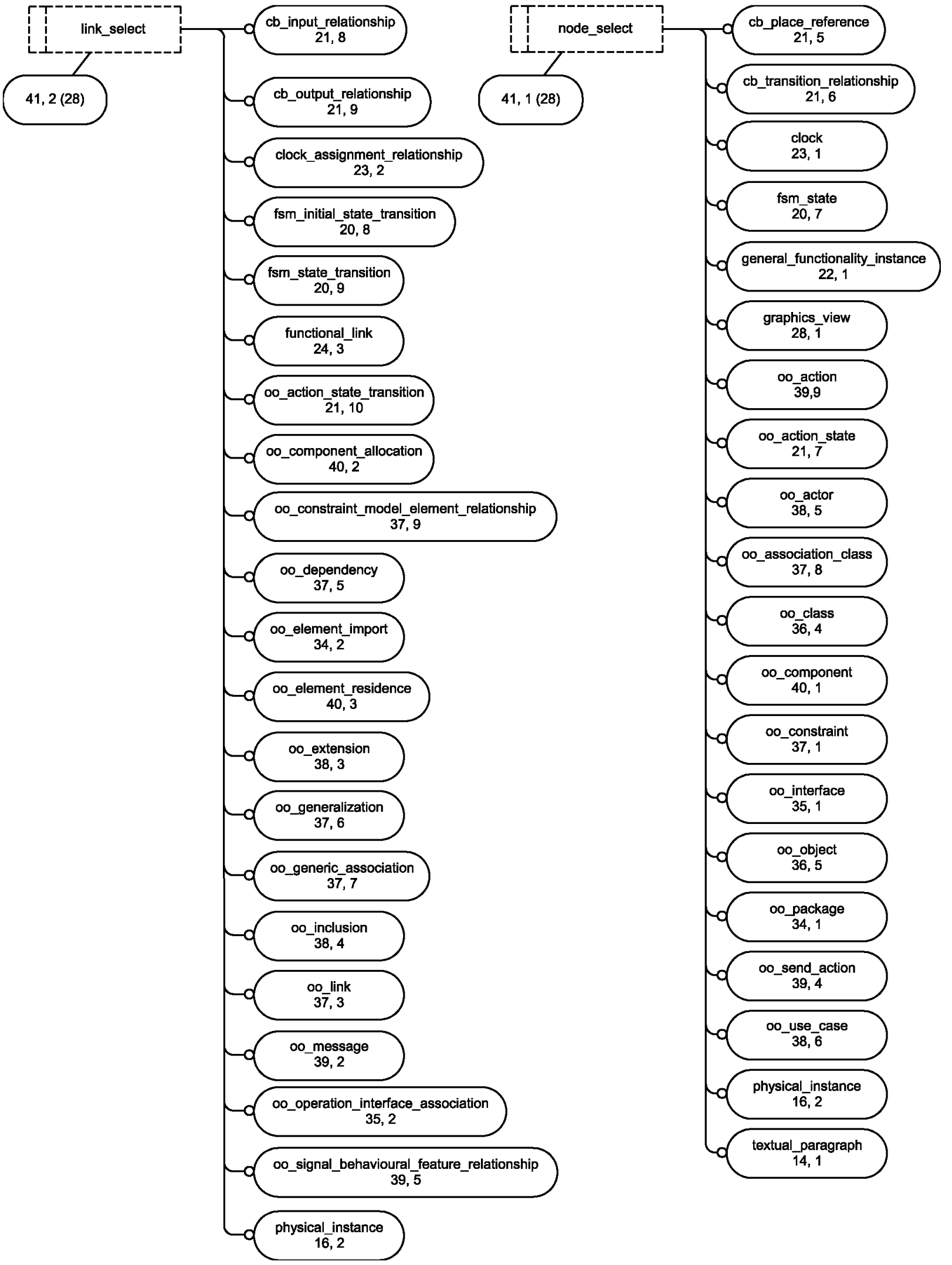


Рисунок Е.42 — Диаграмма ссылочной модели приложения на уровне сущности в нотации EXPRESS-G (41 из 41)

Приложение F
(обязательное)

Листинг для интерпретации на компьютере

	ARM
HTML version	ARM EXPRESS
Text version	ARM EXPRESS text

Листинг ARM EXPRESS

```
(*
ISO TC184/SC4/WG3 N1355 - ISO/PAS 20542 Reference model for systems engineering - EXPRESS ARM
*)

SCHEMA system_engineering_and_design_arm;

TYPE approval_assignment_select = SELECT WITH (change_report, configuration_element, configuration_element_
version, critical_issue, critical_issue_impact,document_assignment, element_critical_issue_relationship, engineering_
process_activity, instance_definition_select, package_element_assignment,partial_system_view_relationship, project, re-
quirement_allocation_relationship, work_order, work_request);
END_TYPE;

TYPE assessment_assignment_select = SELECT WITH (configuration_element_version, engineering_process_activi-
ty, instance_definition_select, project,requirement_allocation_relationship);
END_TYPE;

TYPE assigned_requirement_relationship_select = SELECT WITH (requirement_allocation_relationship, require-
ment_relationship);
END_TYPE;

TYPE basic_identifier = STRING;
END_TYPE;

TYPE boolean_value = BOOLEAN;
END_TYPE;

TYPE buffer_synchronisation_enumeration = ENUMERATION OF
  (asynchronous,synchronouos);
END_TYPE;

TYPE cardinality_association_select = SELECT WITH (cardinality_list, cardinality_range, single_cardinality);
END_TYPE;

TYPE causal_weight_select = SELECT WITH (cb_transition_unbounded_weight, natural_number);
END_TYPE;

TYPE change_element_select = SELECT WITH (instance_definition_select);
END_TYPE;

TYPE change_report_element_select = SELECT WITH (instance_definition_select);
END_TYPE;

TYPE control_characters = ENUMERATION OF
  (cr,tab);
END_TYPE;
```

TYPE **control_type_enumeration** = ENUMERATION OF
  (activate,activate_deactivate);
END_TYPE;

TYPE **data_composition_select** = SELECT WITH (actual_io_port, formal_io_port, io_composition_port);
END_TYPE;

TYPE **data_direction** = ENUMERATION OF
  (from_system,to_system);
END_TYPE;

TYPE **data_type_definition_select** = SELECT WITH (maths_space, user_defined_data_type_definition);
END_TYPE;

TYPE **data_type_value_select** = SELECT WITH (boolean_value, complex_value, compound_value, integer_value, logical_value, real_value, text);
END_TYPE;

TYPE **date_assignment_select** = SELECT WITH (approval, approval_assignment, assessment, configuration_element, configuration_element_version,configuration_element_version_relationship, context_physical_relationship, critical_issue, critical_issue_impact, document_assignment,documentation_relationship, element_critical_issue_relationship, engineering_process_activity, engineering_process_activity_element_assignment,instance_definition_select, justification, justification_relationship, package_element_assignment, partial_system_view_relationship, project,requirement_allocation_relationship, requirement_system_view_assignment, work_order, work_request);
END_TYPE;

TYPE **default_context_select** = SELECT WITH (fsm_generic_state, functional_state_context);
END_TYPE;

TYPE **definition_select** = SELECT WITH (fsm_state, functional_state_context, general_function_definition, general_physical_definition, oo_view, system_view);
END_TYPE;

TYPE **effective_element_select** = SELECT WITH (instance_definition_select, person_organization_assignment);
END_TYPE;

TYPE **event_or_date_select** = SELECT WITH (date_time, project_event_reference);
END_TYPE;

TYPE **external_element_select** = SELECT WITH (function_instance, physical_instance);
END_TYPE;

TYPE **fsm_interaction_select** = SELECT WITH (initial_state_transition_specification_assignment, specification_state_assignment,state_transition_specification_assignment);
END_TYPE;

TYPE **function_role_enumeration** = ENUMERATION OF
  (external_element,system_function);
END_TYPE;

TYPE **identifier** = STRING;
END_TYPE;

TYPE **instance_definition_select** = SELECT WITH (clock, clock_assignment_relationship, data_instance, documentation_reference,functionality_instance_reference, general_function_definition, general_functionality_instance, general_physical_definition, oo_model_element_select,physical_instance, physical_instance_reference, realized_system, requirement_definition, requirement_instance, system_instance, system_view);
END_TYPE;

TYPE **integer_value** = INTEGER;
END_TYPE;

TYPE **issue_source_select** = SELECT WITH (instance_definition_select);
END_TYPE;

TYPE **justification_assignment_select** = SELECT WITH (engineering_process_activity, instance_definition_select, partial_system_view_relationship,requirement_allocation_relationship);
END_TYPE;

TYPE **label** = STRING;
END_TYPE;

TYPE **link_select** = SELECT WITH (cb_input_relationship, cb_output_relationship, clock_assignment_relationship, fsm_initial_state_transition,fsm_state_transition, functional_link, oo_action_state_transition, oo_component_allocation, oo_constraint_model_element_relationship, oo_dependency,oo_element_import, oo_element_residence, oo_extension, oo_generalization, oo_generic_association, oo_inclusion, oo_link, oo_message,oo_operation_interface_association, oo_signal_behavioural_feature_relationship, physical_instance);
END_TYPE;

TYPE **logical_value** = LOGICAL;
END_TYPE;

TYPE **natural_number** = INTEGER;
END_TYPE;

TYPE **node_select** = SELECT WITH (cb_place_reference, cb_transition_relationship, clock, fsm_state, general_functionality_instance, graphics_view, oo_action,oo_action_state, oo_actor, oo_association_class, oo_class, oo_component, oo_constraint, oo_interface, oo_object, oo_package, oo_send_action, oo_use_case,physical_instance, textual_paragraph);
END_TYPE;

TYPE **oo_classifier_or_operation_select** = SELECT WITH (oo_classifier_select, oo_operation);
END_TYPE;

TYPE **oo_classifier_select** = SELECT WITH (oo_actor, oo_class, oo_component, oo_interface, oo_signal, oo_use_case);
END_TYPE;

TYPE **oo_extended_classifier_select** = SELECT WITH (oo_classifier_select, physical_instance);
END_TYPE;

TYPE **oo_extended_model_element_select** = SELECT WITH (fsm_generic_state, generic_state_context, justification, oo_generalizable_element_select,oo_model_element_select);
END_TYPE;

TYPE **oo_feature_select** = SELECT WITH (oo_attribute, oo_behavioural_feature);
END_TYPE;

TYPE **oo_generalizable_element_select** = SELECT WITH (oo_collaboration, oo_extended_classifier_select, oo_generic_association, oo_package, oo_stereotype);
END_TYPE;

TYPE **oo_instance_select** = SELECT WITH (oo_attribute_instance, oo_extended_classifier_select, oo_object);
END_TYPE;

TYPE **oo_model_element_select** = SELECT WITH (oo_action, oo_constraint, oo_extension_point, oo_feature_select, oo_generic_association_end, oo_interaction,oo_link, oo_link_end, oo_message, oo_parameter, oo_relationship_select, oo_stimulus);
END_TYPE;

TYPE **oo_namespace_select** = SELECT WITH (oo_collaboration, oo_extended_classifier_select, oo_package);
END_TYPE;
TYPE **oo_relationship_select** = SELECT WITH (oo_dependency, oo_extension, oo_generalization, oo_generic_associ-

ation, oo_inclusion);
END_TYPE;

TYPE **package_element_select** = SELECT WITH (configuration_element, configuration_element_version, engineering_process_activity, instance_definition_select,project);
END_TYPE;

TYPE **period_or_date_select** = SELECT WITH (date_time, project_event_reference, value_with_unit);
END_TYPE;

TYPE **person_organization_assignment_select** = SELECT WITH (approval, approval_assignment, assessment, change_report, configuration_element,configuration_element_version, critical_issue, critical_issue_impact, document_assignment, element_critical_issue_relationship,engineering_process_activity, instance_definition_select, justification, justification_relationship, package_element_assignment,partial_system_view_relationship, project, requirement_allocation_relationship, requirement_system_view_assignment, work_order);
END_TYPE;

TYPE **person_organization_select** = SELECT WITH (organization, person_in_organization);
END_TYPE;

TYPE **physical_element_role_enumeration** = ENUMERATION OF
  (external_element,system_element);
END_TYPE;

TYPE **port_data_relation** = ENUMERATION OF
  (consumer,producer);
END_TYPE;

TYPE **port_position_select** = SELECT WITH (io_port, oo_generic_association_end, oo_link_end, physical_port);
END_TYPE;

TYPE **port_type** = ENUMERATION OF
  (control,input,mechanism,output);
END_TYPE;

TYPE **property_assignment_select** = SELECT WITH (data_instance, documentation_reference, functionality_instance_reference, general_function_definition,general_physical_definition, oo_model_element_select, package, physical_instance_reference, realized_system, requirement_definition, system_view);
END_TYPE;

TYPE **property_value_select** = SELECT WITH (boolean_value, text_select, value_list, value_with_unit);
END_TYPE;

TYPE **ranked_element_select** = SELECT WITH (critical_issue_impact, effectiveness_measure, instance_definition_select);
END_TYPE;

TYPE **ranking_type** = ENUMERATION OF
  (cost,criticality,priority,project_criticality);
END_TYPE;

TYPE **real_value** = REAL;
END_TYPE;

TYPE **requirement_allocation_select** = SELECT WITH (data_instance, functional_link_reference, functionality_instance_reference,functionality_reference_relationship, oo_model_element_select, physical_instance_reference);
END_TYPE;

TYPE **single_cardinality_select** = SELECT WITH (infinite_cardinality, natural_number);
END_TYPE;
TYPE **specific_element_select** = SELECT WITH (cb_place, fsm_generic_state);
END_TYPE;

```
TYPE specification_element_select = SELECT WITH (change_report, critical_issue, critical_issue_impact, instance_
definition_select);
END_TYPE;

TYPE system_select = SELECT WITH (system_instance, system_view);
END_TYPE;

TYPE text = STRING;
END_TYPE;

TYPE text_elements = SELECT WITH (control_characters, text);
END_TYPE;

TYPE text_select = SELECT WITH (text, textual_paragraph, textual_section, textual_table);
END_TYPE;

TYPE timing_type = ENUMERATION OF
  (best_case,nominal_case,worst_case);
END_TYPE;

TYPE trigger_type_enumeration = ENUMERATION OF
  (flank,level);
END_TYPE;

TYPE verification_allocation_select = SELECT WITH (functionality_instance_reference, physical_instance_refer-
ence, realized_system, requirement_instance,system_instance);
END_TYPE;

ENTITY abstract_data_type_definition
 SUBTYPE OF (user_defined_data_type_definition) ;
END_ENTITY;

ENTITY abstract_data_type_member;
  child : data_instance;
  parent : abstract_data_type_definition;
END_ENTITY;

ENTITY actual_io_port
 SUBTYPE OF (io_port) ;
  port_of : general_functionality_instance;
 DERIVE
 SELF\io_port. RENAMED     role : port_data_relation : determineformalportrole(SELF);
 INVERSE
   assigned_buffer : SET[0:1] OF io_buffer FOR assigned_to;
  UNIQUE
   UR1: port_of, io_port_number, port_type;
END_ENTITY;

ENTITY actual_physical_port
 SUBTYPE OF (physical_port) ;
  port_of : physical_instance;
END_ENTITY;

ENTITY actual_port_position
 SUBTYPE OF (visual_element) ;
  assigned_to : graphics_node;
  position : graphics_point;
  positioned_port : port_position_select;
END_ENTITY;
```

```
ENTITY address;
  country : OPTIONAL label;
  electronic_mail_address : OPTIONAL label;
  facsimile_number : OPTIONAL label;
  internal_location : OPTIONAL label;
  postbox_number : OPTIONAL label;
  postcode : OPTIONAL label;
  region : OPTIONAL label;
  street : OPTIONAL label;
  street_number : OPTIONAL label;
  telephone_number : OPTIONAL label;
  telex_number : OPTIONAL label;
  town : OPTIONAL label;
END_ENTITY;

ENTITY aggregate_data_type_definition
 SUBTYPE OF (user_defined_data_type_definition) ;
  aggregate_type : data_type_definition_select;
  bound : LIST[1:?] OF integer_interval;
END_ENTITY;

ENTITY approval;
  level : OPTIONAL label;
  status : label;
END_ENTITY;

ENTITY approval_assignment;
  assigned_approval : approval;
  assigned_to : approval_assignment_select;
  description : OPTIONAL text_select;
  status : label;
END_ENTITY;

ENTITY approval_person_organization;
  authorized_approval : approval;
  person_organization : person_organization_select;
  role : label;
END_ENTITY;

ENTITY approval_relationship;
  description : OPTIONAL text_select;
  related_approval : approval;
  relating_approval : approval;
  relation_type : label;
END_ENTITY;

ENTITY assessment;
  assessed_level : label;
  assessment_type : label;
  assigned_to : assessment_assignment_select;
  description : OPTIONAL text_select;
END_ENTITY;

ENTITY assessment_relationship;
  description : OPTIONAL text_select;
  related : assessment;
  relating : assessment;
  relationship_type : label;
END_ENTITY;

ENTITY bi_directional_port_indicator;
```

```
  consuming_port : io_port;
  producing_port : io_port;
END_ENTITY;

ENTITY binary_data_type_definition
 SUBTYPE OF (elementary_maths_space) ;
  size : finite_integer_interval;
END_ENTITY;

ENTITY boolean_data_type_definition
 SUBTYPE OF (elementary_maths_space) ;
END_ENTITY;

ENTITY cardinality_list;
  range : LIST[2:?] OF cardinality_range;
END_ENTITY;

ENTITY cardinality_range;
  lower_bound : single_cardinality;
  upper_bound : single_cardinality;
END_ENTITY;

ENTITY causal_block_bound;
  initial_transition : cb_transition_relationship;
  terminal_transition : cb_transition_relationship;
END_ENTITY;

ENTITY cb_completion_alternative;
  completes_model : cb_functional_behaviour_model;
  final_model_element : cb_functional_place;
END_ENTITY;

ENTITY cb_completion_alternative_mapping;
  child_completion_criterion : cb_completion_alternative;
  equivalent_transition : cb_functional_transition;
  scope : cb_place_function_association;
END_ENTITY;

ENTITY cb_firing_condition;
  condition_definition : textual_specification;
  guarded_transition : cb_transition;
END_ENTITY;

ENTITY cb_functional_behaviour_model
 SUBTYPE OF (functional_behaviour_model) ;
  model_boundedness : label;
  model_type : label;
 INVERSE
   constituent_places : SET[1:?] OF cb_functional_place FOR behaviour_model;
   constituent_transitions : SET[2:?] OF cb_functional_transition FOR behaviour_model;
END_ENTITY;

ENTITY cb_functional_place
 SUBTYPE OF (cb_place) ;
  behaviour_model : cb_functional_behaviour_model;
 INVERSE
   reference_information : SET[0:1] OF cb_place_reference FOR functional_place_reference;
END_ENTITY;

ENTITY cb_functional_transition
 SUBTYPE OF (cb_transition) ;
```

```
  behaviour_model : cb_functional_behaviour_model;
 INVERSE
  transition_relationship : SET[0:1] OF cb_transition_relationship FOR related_transition;
END_ENTITY;

ENTITY cb_initial_marking;
  marked_place : cb_place;
  number_of_tokens : INTEGER;
END_ENTITY;

ENTITY cb_input_relationship;
  causal_weight : causal_weight_select;
  destination_transition : cb_transition;
  source_place : cb_place;
END_ENTITY;

ENTITY cb_output_relationship;
  causal_weight : causal_weight_select;
  destination_place : cb_place;
  source_transition : cb_transition;
END_ENTITY;

ENTITY cb_place
 ABSTRACT SUPERTYPE  OF ( ONEOF(cb_functional_place, oo_action_state) ) ;
  description : OPTIONAL text_select;
  place_label : OPTIONAL label;
 INVERSE
  initial_marking : SET[0:1] OF cb_initial_marking FOR marked_place;
END_ENTITY;

ENTITY cb_place_function_association;
  causal_place : cb_functional_place;
  controls_function : function_instance;
 INVERSE
  completion_mapping : SET[0:?] OF cb_completion_alternative_mapping FOR scope;
END_ENTITY;

ENTITY cb_place_reference;
  functional_place_reference : cb_functional_place;
  reference_type : label;
END_ENTITY;

ENTITY cb_transition
 ABSTRACT SUPERTYPE  OF ( ONEOF(cb_functional_transition, oo_action_state_transition) ) ;
  description : OPTIONAL text_select;
  transition_label : OPTIONAL label;
 INVERSE
  guarded_by : SET[0:1] OF cb_firing_condition FOR guarded_transition;
END_ENTITY;

ENTITY cb_transition_relationship;
  related_transition : SET[1:?] OF cb_functional_transition;
  relationship_type : label;
 INVERSE
  end_bound : SET[0:1] OF causal_block_bound FOR terminal_transition;
  start_bound : SET[0:1] OF causal_block_bound FOR initial_transition;
END_ENTITY;

ENTITY cb_transition_unbounded_weight;
  minimal_weight : natural_number;
END_ENTITY;
```

```
ENTITY change_order
  SUBTYPE OF (work_order) ;
   change_element : element_critical_issue_relationship;
   originates_from : change_request;
END_ENTITY;

ENTITY change_order_relationship;
   change_order_relationship_type : label;
   related : change_order;
   relating : change_order;
END_ENTITY;

ENTITY change_report;
   associated_version : configuration_element_version;
   description : OPTIONAL text_select;
   name : label;
   originating_change_request : change_request;
END_ENTITY;

ENTITY change_report_element_assignment;
   change_report : change_report;
   change_report_element : change_report_element_select;
   description : OPTIONAL text_select;
END_ENTITY;

ENTITY change_request
  SUBTYPE OF (work_request) ;
   response_to_issue : SET[1:?] OF critical_issue;
END_ENTITY;

ENTITY clock;
   control_signal : data_instance;
   description : OPTIONAL text_select;
   frequency : REAL;
   name : OPTIONAL label;
  WHERE
    correct_data_definition: 'SYSTEMS_ENGINEERING_DATA_REPRESENTATION.EVENT_DATA_TYPE_DEFINITION'
IN TYPEOF(control_signal.definition);
END_ENTITY;

ENTITY clock_assignment_relationship;
   clock : clock;
   trigger_for : control_io_port;
END_ENTITY;

ENTITY clock_reference_context_relationship;
   clock_signal : clock;
   relevant_functionality_context : functionality_instance_reference;
END_ENTITY;

ENTITY complex_data_type_definition
  SUBTYPE OF (elementary_maths_space) ;
END_ENTITY;

ENTITY complex_value;
   radius : REAL;
   theta : REAL;
END_ENTITY;

ENTITY composite_function_definition
  SUBTYPE OF (general_function_definition) ;
```

```
  INVERSE    behaviour_constraint : SET[0:1] OF functional_behaviour_model_assignment FOR constrained_function;
    parent_of : SET[1:?] OF functional_decomposition_relationship FOR parent;
END_ENTITY;

ENTITY compound_value;
  value_list : LIST[0:?] OF data_type_value_select;
END_ENTITY;

ENTITY configuration_element;
  description : OPTIONAL text_select;
  id : element_identifier;
  name : label;
 INVERSE
   associated_version : SET[1:?] OF configuration_element_version FOR version_of;
  UNIQUE
   UR1: id;
END_ENTITY;

ENTITY configuration_element_relationship;
  alternate : configuration_element;
  base : configuration_element;
  description : OPTIONAL text_select;
  relationship_type : label;
END_ENTITY;

ENTITY configuration_element_version;
  description : OPTIONAL text_select;
  id : element_identifier;
  version_of : configuration_element;
  UNIQUE
   UR1: id;
END_ENTITY;

ENTITY configuration_element_version_relationship;
  description : OPTIONAL text_select;
  related_version : configuration_element_version;
  relating_version : configuration_element_version;
  relationship_type : label;
 WHERE
   WR1: related_version :<>: relating_version;
END_ENTITY;

ENTITY context_function_relationship;
  associated_context : system_view;
  context_function : function_instance;
  description : OPTIONAL text_select;
  role : function_role_enumeration;
 INVERSE
   assigned_functional_configuration : SET[0:1] OF system_functional_configuration FOR system;
  UNIQUE
   UR1: associated_context, context_function;
END_ENTITY;

ENTITY context_physical_relationship;
  description : OPTIONAL text_select;
  part_in_physical_context : system_view;
  physical_instance : physical_instance;
  role : physical_element_role_enumeration;
 INVERSE
   assigned_physical_configuration : SET[0:1] OF system_physical_configuration FOR system;
END_ENTITY;
```

```
ENTITY control_io_port
  SUBTYPE OF (io_port) ;
   control_type : control_type_enumeration;
   offset : REAL;
   port_of : function_instance;
   trigger_type : trigger_type_enumeration;
  DERIVE
  SELF\io_port. RENAMED     role : port_data_relation : consumer;
   UNIQUE
    UR1: port_of, io_port_number, port_type;
  WHERE
    good_offset: offset >= 0.0;
    port_data_direction: (SELF\io_port.port_type <> output);
    WR1: ('SYSTEMS_ENGINEERING_DATA_REPRESENTATION.EVENT_DATA_TYPE_DEFINITION' IN TYPEOF
(data.definition)) / ('SYSTEMS_ENGINEERING_DATA_REPRESENTATION.LOGICAL_DATA_TYPE_DEFINITION' IN
TYPEOF(data.definition));
END_ENTITY;

ENTITY coordinate_translation_information;
   measurement_unit : label;
   name : label;
   ratio : REAL;
   scale_factor : REAL;
   transformation_for : graphics_view;
END_ENTITY;

ENTITY critical_issue;
   description : OPTIONAL text_select;
   id : element_identifier;
   name : label;
   status : label;
END_ENTITY;

ENTITY critical_issue_impact;
   description : OPTIONAL text_select;
   id : element_identifier;
   impact_of_issue : SET[1:?] OF critical_issue;
   name : label;
  INVERSE
    identified_issues : SET[0:?] OF element_critical_issue_relationship FOR issue_analysis;
END_ENTITY;

ENTITY data_field
  SUBTYPE OF (data_instance) ;
   role : OPTIONAL label;
END_ENTITY;

ENTITY data_instance
  SUPERTYPE OF ( data_field ) ;
   default_value : OPTIONAL data_type_value_select;
   definition : data_type_definition_select;
   description : OPTIONAL text_select;
   id : element_identifier;
   initial_value : OPTIONAL data_type_value_select;
   is_constant : BOOLEAN;
   name : label;
   unit_component : OPTIONAL unit;
   UNIQUE
    UR1: definition, id;
END_ENTITY;
```

```
ENTITY data_transfer;
  data : data_instance;
  direction : data_direction;
  transfer : implied_external_interaction;
END_ENTITY;

ENTITY date_and_person_assignment;
  assigned_date_and_person : date_and_person_organization;
  assigned_to : person_organization_assignment_select;
  description : OPTIONAL text_select;
  role : label;
END_ENTITY;

ENTITY date_and_person_organization;
  actual_date : date_time;
  person_organization : person_organization_select;
END_ENTITY;

ENTITY date_assignment;
  assigned_to : date_assignment_select;
  date : date_time;
  role : label;
END_ENTITY;

ENTITY date_time;
  day_component : INTEGER;
  hour_component : INTEGER;
  minute_component : INTEGER;
  month_component : INTEGER;
  second_component : INTEGER;
  year_component : INTEGER;
 WHERE
  WR1: {0 <= hour_component <= 23};
  WR2: {0 <= minute_component <= 59};
  WR3: {0 <= second_component <= 59};
  WR4: {1 <= month_component <= 12};
  WR5: {1 <= day_component <= 31};
  WR6: year_component >= 0;
END_ENTITY;

ENTITY derived_data_type_definition
 SUBTYPE OF (user_defined_data_type_definition) ;
  expression : text_select;
  resultant_data_type : data_type_definition_select;
END_ENTITY;

ENTITY digital_document
 SUBTYPE OF (documentation_reference) ;
  document_format : OPTIONAL label;
  document_size : OPTIONAL label;
  location : label;
END_ENTITY;

ENTITY document_assignment
 SUPERTYPE OF ( partial_document_assignment ) ;
  description : OPTIONAL text_select;
  documentation : documentation_reference;
  documented_object : instance_definition_select;
END_ENTITY;
```

```
ENTITY documentation_reference
 ABSTRACT SUPERTYPE   OF ( ONEOF(digital_document, non_digital_document) ) ;
  associated_version : configuration_element_version;
  description : OPTIONAL text_select;
  id : element_identifier;
  name : label;
  UNIQUE
   UR1: id;
END_ENTITY;

ENTITY documentation_relationship;
  description : OPTIONAL text_select;
  related_documentation : documentation_reference;
  relating_documentation : documentation_reference;
  relationship_type : OPTIONAL label;
 WHERE
   correct_relation: relating_documentation :<>: related_documentation;
END_ENTITY;

ENTITY effectiveness_measure;
  optimization_function : textual_specification;
END_ENTITY;

ENTITY effectiveness_measure_assignment;
  assigned_requirement : requirement_instance;
  effectiveness_measure : effectiveness_measure;
  weight : label;
END_ENTITY;

ENTITY effectiveness_measure_relationship;
  description : OPTIONAL text_select;
  related : effectiveness_measure;
  relating : effectiveness_measure;
END_ENTITY;

ENTITY effectivity;
  concerned_organization : SET[0:?] OF organization;
  description : OPTIONAL text_select;
  id : OPTIONAL element_identifier;
  primary_definition : OPTIONAL event_or_date_select;
  secondary_definition : OPTIONAL period_or_date_select;
  version_id : OPTIONAL element_identifier;
END_ENTITY;

ENTITY effectivity_assignment;
  assigned_effectivity : effectivity;
  effective_element : effective_element_select;
  role : label;
END_ENTITY;

ENTITY effectivity_relationship;
  description : OPTIONAL text_select;
  related : effectivity;
  relating : effectivity;
  relation_type : label;
END_ENTITY;

ENTITY element_critical_issue_relationship;
  description : OPTIONAL text_select;
  impact_on_element : change_element_select;
  issue_analysis : critical_issue_impact;
END_ENTITY;
```

```
ENTITY element_identifier;
  identifier_context : person_organization_select;
  identifier_value : identifier;
END_ENTITY;

ENTITY elementary_maths_space
 ABSTRACT SUPERTYPE  OF ( ONEOF(binary_data_type_definition, boolean_data_type_definition, complex_data_
type_definition, event_data_type_definition, integer_data_type_definition, logical_data_type_definition, real_data_type_
definition, string_data_type_definition) )
 SUBTYPE OF (maths_space) ;
END_ENTITY;

ENTITY engineering_process_activity;
  activity_type : label;
  actual_end_date : OPTIONAL date_time;
  actual_start_date : OPTIONAL date_time;
  description : OPTIONAL text_select;
  id : element_identifier;
  name : label;
  planned_end_date : OPTIONAL period_or_date_select;
  planned_start_date : OPTIONAL event_or_date_select;
  resolved_request : SET[0:?] OF work_request;
  status : OPTIONAL text_select;
 INVERSE
   authorization : SET[0:1] OF work_order FOR is_controlling;
END_ENTITY;

ENTITY engineering_process_activity_element_assignment;
  activity : engineering_process_activity;
  description : OPTIONAL text_select;
  element : specification_element_select;
  role : label;
END_ENTITY;

ENTITY engineering_process_activity_relationship;
  description : OPTIONAL text_select;
  related_activity : engineering_process_activity;
  relating_activity : engineering_process_activity;
  relation_type : label;
 WHERE
   WR1: relating_activity :<>: related_activity;
END_ENTITY;

ENTITY event_data_type_definition
 SUBTYPE OF (elementary_maths_space) ;
END_ENTITY;

ENTITY execution_time;
  role : timing_type;
  time : REAL;
  timing : functionality_instance_reference;
  unit : label;
END_ENTITY;

ENTITY finite_integer_interval
 SUBTYPE OF (integer_interval) ;
  low_index : INTEGER;
  size : INTEGER;
END_ENTITY;
```

```
ENTITY finite_real_interval
  SUBTYPE OF (real_interval) ;
   high_closure : BOOLEAN;
   high_index : REAL;
   low_closure : BOOLEAN;
   low_index : REAL;
END_ENTITY;

ENTITY finite_space
  SUBTYPE OF (maths_space) ;
   member : SET[0:?] OF data_type_value_select;
END_ENTITY;

ENTITY formal_data_interaction_port;
   data : data_instance;
   port_of : functional_state_context;
END_ENTITY;

ENTITY formal_io_port
  SUBTYPE OF (io_port) ;
   port_of : general_function_definition;
  DERIVE
  SELF\io_port. RENAMED     role : port_data_relation : determineformalportrole(SELF);
   UNIQUE
    UR1: port_of, io_port_number, port_type;
END_ENTITY;

ENTITY formal_physical_port
  SUBTYPE OF (physical_port) ;
   port_of : general_physical_definition;
END_ENTITY;

ENTITY formal_port_position
  SUBTYPE OF (visual_element) ;
   position : graphics_point;
   positioned_port : port_position_select;
END_ENTITY;

ENTITY fsm_and_state
  SUBTYPE OF (fsm_state) ;
  INVERSE
  SELF\fsm_state. RENAMED     child_states : SET[2:?] OF fsm_state_composition_relationship FOR parent_state;
END_ENTITY;

ENTITY fsm_command_interaction_relationship;
   defined_in : fsm_interaction_select;
   interaction_port : state_function_interaction_port;
   interaction_type : label;
END_ENTITY;

ENTITY fsm_data_interaction_binding;
   actual_port : actual_io_port;
   formal_port : formal_data_interaction_port;
END_ENTITY;

ENTITY fsm_data_interaction_relationship;
   defined_in : fsm_interaction_select;
   interaction_port : formal_data_interaction_port;
   interaction_type : label;
END_ENTITY;
```

```
ENTITY fsm_generic_state
 ABSTRACT SUPERTYPE   OF ( ONEOF(fsm_state, fsm_transient_state) ) ;
 INVERSE
   destination_transition : SET[0:?] OF fsm_state_transition FOR destination_state;
END_ENTITY;

ENTITY fsm_initial_state_transition;
  initial_state : fsm_generic_state;
  transition_context : default_context_select;
END_ENTITY;

ENTITY fsm_model
 SUBTYPE OF (general_functionality_instance) ;
  behaviour_model : state_machine_functional_behaviour_model;
  definition : functional_state_context;
  id : element_identifier;
  name : label;
  presentation_id : OPTIONAL label;
END_ENTITY;

ENTITY fsm_or_state
 SUBTYPE OF (fsm_state) ;
END_ENTITY;

ENTITY fsm_state
 ABSTRACT SUPERTYPE   OF ( ONEOF(fsm_and_state, fsm_or_state) )
 SUBTYPE OF (fsm_generic_state) ;
  description : OPTIONAL text_select;
  name : label;
  presentation_id : OPTIONAL label;
 INVERSE
   child_states : SET[0:?] OF fsm_state_composition_relationship FOR parent_state;
END_ENTITY;

ENTITY fsm_state_composition_relationship;
  child_state : fsm_state;
  parent_state : fsm_state;
END_ENTITY;

ENTITY fsm_state_transition;
  destination_state : fsm_generic_state;
  source_state : fsm_generic_state;
 INVERSE
   transition_label : SET[0:1] OF state_transition_specification_assignment FOR assigned_to;
END_ENTITY;

ENTITY fsm_transient_state
 SUBTYPE OF (fsm_generic_state) ;
  state_type : label;
END_ENTITY;

ENTITY fsm_transient_state_composition_relationship;
  child_state : fsm_transient_state;
  parent_state : fsm_state;
END_ENTITY;

ENTITY function_instance
 SUBTYPE OF (general_functionality_instance) ;
  definition : general_function_definition;
  id : element_identifier;
  name : label;
```

```
  presentation_id : OPTIONAL label;
 INVERSE
  control_port : SET[0:?] OF control_io_port FOR port_of;
 UNIQUE
  UR1: definition, id;
END_ENTITY;

ENTITY function_reference;
  function_link : function_instance;
  port_of : fsm_model;
END_ENTITY;

ENTITY functional_behaviour_model
 ABSTRACT SUPERTYPE  OF ( ONEOF(cb_functional_behaviour_model, state_machine_functional_behaviour_mod-
el) ) ;
  description : OPTIONAL text_select;
 INVERSE
  defines_behaviour_for : SET[0:1] OF functional_behaviour_model_assignment FOR assigned_behaviour_model;
END_ENTITY;

ENTITY functional_behaviour_model_assignment;
  assigned_behaviour_model : functional_behaviour_model;
  constrained_function : composite_function_definition;
END_ENTITY;

ENTITY functional_decomposition_relationship;
  child : general_functionality_instance;
  description : OPTIONAL text_select;
  parent : composite_function_definition;
END_ENTITY;

ENTITY functional_link;
  control_link : BOOLEAN;
  description : OPTIONAL text_select;
  destination_port : io_port;
  name : label;
  source_port : io_port;
 DERIVE
  data_on_link : data_instance : source_port.data;
 WHERE
  correct_ports: ((destination_port.role = consumer) AND (source_port.role = producer));
END_ENTITY;

ENTITY functional_link_allocation_relationship;
  allocated_functional_link : functional_link_reference;
  allocated_to : physical_instance_reference;
  description : OPTIONAL text_select;
END_ENTITY;

ENTITY functional_link_group;
  elements : SET[2:?] OF functional_link;
  name : label;
END_ENTITY;

ENTITY functional_link_reference;
  in_scope_of : functionality_instance_reference;
  reference_functional_link : functional_link;
END_ENTITY;

ENTITY functional_reference_configuration;
  description : OPTIONAL text_select;
```

```
END_ENTITY;

ENTITY functional_representation_relationship;
  description : OPTIONAL text_select;
  functional_representation : data_instance;
  physical_element : physical_instance;
END_ENTITY;

ENTITY functional_state_context
 SUBTYPE OF (generic_state_context) ;
END_ENTITY;

ENTITY functionality_allocation_relationship;
  allocated_functionality : functionality_instance_reference;
  allocated_to : physical_instance_reference;
  description : OPTIONAL text_select;
END_ENTITY;

ENTITY functionality_instance_reference
 SUPERTYPE OF ( persistent_storage_reference ) ;
  description : OPTIONAL text_select;
  id : element_identifier;
  name : label;
  referenced_functionality_instance : general_functionality_instance;
END_ENTITY;

ENTITY functionality_reference_composition_relationship;
  child : functionality_instance_reference;
  mirror_of : functional_decomposition_relationship;
  parent : functionality_instance_reference;
  reference_configuration : functional_reference_configuration;
END_ENTITY;

ENTITY functionality_reference_relationship;
  first_reference : functionality_instance_reference;
  second_reference : functionality_instance_reference;
END_ENTITY;

ENTITY general_function_definition
 ABSTRACT SUPERTYPE   OF ( ONEOF(composite_function_definition, leaf_function_definition) ) ;
  associated_version : configuration_element_version;
  description : OPTIONAL text_select;
  id : element_identifier;
  name : OPTIONAL label;
 INVERSE
   formal_port : SET[0:?] OF formal_io_port FOR port_of;
  UNIQUE
   UR1: id;
END_ENTITY;

ENTITY general_functionality_instance
 ABSTRACT SUPERTYPE   OF ( ONEOF(fsm_model, function_instance, io_split_join, persistent_storage) ) ;
  description : OPTIONAL text_select;
 INVERSE
   actual_port : SET[0:?] OF actual_io_port FOR port_of;
END_ENTITY;

ENTITY general_physical_definition
 ABSTRACT SUPERTYPE   OF ( ONEOF(physical_link_definition, physical_node_definition) ) ;
  associated_version : configuration_element_version;
  description : OPTIONAL text_select;
```

```
  id : element_identifier;
  name : OPTIONAL label;
 INVERSE
  formal_port : SET[0:?] OF formal_physical_port FOR port_of;
 UNIQUE
  UR1: id;
END_ENTITY;

ENTITY generic_state_context
 ABSTRACT SUPERTYPE ;
  associated_version : configuration_element_version;
  description : OPTIONAL text_select;
  id : element_identifier;
  name : OPTIONAL label;
  original_representation : label;
  state_machine_model : label;
END_ENTITY;

ENTITY graphics_link
 SUBTYPE OF (visual_element) ;
  associated_with : link_select;
  point : LIST[2:?] OF graphics_point;
END_ENTITY;

ENTITY graphics_node
 SUBTYPE OF (visual_element) ;
  associated_with : node_select;
  bottom_right : graphics_point;
  top_left : graphics_point;
END_ENTITY;

ENTITY graphics_point;
  x_coordinate : REAL;
  y_coordinate : REAL;
END_ENTITY;

ENTITY graphics_view;
  definition_for : definition_select;
 INVERSE
  coordinate_definition : SET[0:?] OF coordinate_translation_information FOR transformation_for;
  root_view : SET[0:1] OF multi_level_view FOR top_view;
  view_element : SET[1:?] OF visual_element FOR view;
END_ENTITY;

ENTITY hibound_integer_interval
 SUBTYPE OF (integer_interval) ;
  high_index : INTEGER;
END_ENTITY;

ENTITY hibound_real_interval
 SUBTYPE OF (real_interval) ;
  high_closure : BOOLEAN;
  high_index : REAL;
END_ENTITY;

ENTITY implied_external_interaction;
  associated_requirement : requirement_instance;
  implied_external_element : external_element_select;
 INVERSE
  associated_data : SET[0:1] OF data_transfer FOR transfer;
END_ENTITY;
```

```
ENTITY infinite_cardinality;
END_ENTITY;

ENTITY initial_state_transition_specification_assignment;
  assigned_to : fsm_initial_state_transition;
  specification : textual_specification;
END_ENTITY;

ENTITY integer_data_type_definition
 SUBTYPE OF (elementary_maths_space) ;
END_ENTITY;

ENTITY integer_interval
 ABSTRACT SUPERTYPE   OF ( ONEOF(finite_integer_interval, hibound_integer_interval, lobound_integer_interval) )
 SUBTYPE OF (maths_space) ;
END_ENTITY;

ENTITY io_buffer;
  assigned_to : actual_io_port;
  continuously_active : BOOLEAN;
  synchronisation_semantics : buffer_synchronisation_enumeration;
END_ENTITY;

ENTITY io_composition_port
 SUBTYPE OF (io_port) ;
  is_alias_for : data_instance;
  port_of : data_composition_select;
 DERIVE
 SELF\io_port. RENAMED     role : port_data_relation : ;
  UNIQUE
   UR1: port_of, io_port_number;
END_ENTITY;

ENTITY io_port
 ABSTRACT SUPERTYPE   OF ( ONEOF(actual_io_port, control_io_port, formal_io_port, io_composition_port) ) ;
  data : data_instance;
  io_port_number : INTEGER;
  port_type : port_type;
  role : port_data_relation;
END_ENTITY;

ENTITY io_port_binding;
  actual_port : actual_io_port;
  formal_port : formal_io_port;
 WHERE
   WR1: (SELF.formal_port.data :<>: SELF.actual_port.data);
   WR2: formal_port.role <> actual_port.role;
   WR3: correct_binding(SELF);
END_ENTITY;

ENTITY io_split_join
 SUBTYPE OF (general_functionality_instance) ;
END_ENTITY;

ENTITY issue_source_relationship;
  description : OPTIONAL text_select;
  issue : critical_issue;
  issue_source : issue_source_select;
END_ENTITY;
```

```
ENTITY issue_system_assignment;
  description : OPTIONAL text_select;
  identified_system : system_select;
  issue : critical_issue;
END_ENTITY;

ENTITY justification;
  assigned_to : justification_assignment_select;
  justification_text : text_select;
  role : label;
END_ENTITY;

ENTITY justification_relationship;
  description : OPTIONAL text_select;
  related : justification;
  relating : justification;
  relationship_type : label;
END_ENTITY;

ENTITY leaf_function_definition
 SUBTYPE OF (general_function_definition) ;
  definition : textual_specification;
  function_type : OPTIONAL label;
  predefined : BOOLEAN;
END_ENTITY;

ENTITY lobound_integer_interval
 SUBTYPE OF (integer_interval) ;
  low_index : INTEGER;
END_ENTITY;

ENTITY lobound_real_interval
 SUBTYPE OF (real_interval) ;
  low_closure : BOOLEAN;
  low_index : REAL;
END_ENTITY;

ENTITY logical_data_type_definition
 SUBTYPE OF (elementary_maths_space) ;
END_ENTITY;

ENTITY maths_space
 ABSTRACT SUPERTYPE   OF ( ONEOF(elementary_maths_space, finite_space, integer_interval, real_interval) ) ;
  description : OPTIONAL text_select;
  id : element_identifier;
  name : OPTIONAL label;
  UNIQUE
   UR1: id;
END_ENTITY;

ENTITY model_defined_requirement_definition
 SUBTYPE OF (requirement_definition) ;
  assigned_model : system_view;
  model_relevance : OPTIONAL text_select;
END_ENTITY;

ENTITY multi_level_view;
  description : OPTIONAL text_select;
  reference_name : OPTIONAL label;
  top_view : graphics_view;
END_ENTITY;
```

```
ENTITY name_binding;
  actual_port : function_reference;
  formal_port : state_function_interaction_port;
END_ENTITY;

ENTITY nominal_value
 SUBTYPE OF (value_with_unit) ;
  value_component : NUMBER;
 INVERSE
   limitation : SET[0:1] OF plus_minus_bounds FOR limited_value;
END_ENTITY;

ENTITY non_digital_document
 SUBTYPE OF (documentation_reference) ;
  location : label;
END_ENTITY;

ENTITY oo_action
 SUPERTYPE OF ( ONEOF(oo_call_action, oo_create_action, oo_send_action) ) ;
  description : OPTIONAL text_select;
  is_asynchronous : BOOLEAN;
  name : label;
  script : textual_specification;
END_ENTITY;

ENTITY oo_action_state
 SUBTYPE OF (cb_place) ;
END_ENTITY;

ENTITY oo_action_state_transition
 SUBTYPE OF (cb_transition) ;
END_ENTITY;

ENTITY oo_action_temporal_relationship;
  description : OPTIONAL text_select;
  predecessor : oo_action;
  successor : oo_action;
END_ENTITY;

ENTITY oo_actor;
  description : OPTIONAL text_select;
  id : element_identifier;
  name : label;
  namespace : OPTIONAL oo_namespace_select;
END_ENTITY;

ENTITY oo_argument;
  action : oo_action;
  initial_value : text_select;
END_ENTITY;

ENTITY oo_association
 SUBTYPE OF (oo_generic_association) ;
 WHERE
  WR1: SELF\oo_generic_association.reading_direction IN SELF\oo_generic_association.connection;
END_ENTITY;

ENTITY oo_association_class;
  class : oo_class;
  class_association : oo_generic_association;
  description : OPTIONAL text_select;
```

```
END_ENTITY;

ENTITY oo_association_end
  SUBTYPE OF (oo_generic_association_end) ;
  SELF\oo_generic_association_end. RENAMED    association : oo_association;
END_ENTITY;

ENTITY oo_association_end_classifier_relationship;
   association_end : oo_generic_association_end;
   description : OPTIONAL text_select;
   specification : oo_extended_classifier_select;
END_ENTITY;

ENTITY oo_association_end_qualifier_association;
   association_end : oo_generic_association_end;
   qualifier : OPTIONAL oo_attribute;
END_ENTITY;

ENTITY oo_association_end_role
  SUBTYPE OF (oo_generic_association_end) ;
   base : OPTIONAL oo_association_end;
   collaboration_multiplicity : cardinality_association_select;
   role_type : oo_classifier_role;
  SELF\oo_generic_association_end. RENAMED    association : oo_association_role;
END_ENTITY;

ENTITY oo_association_role
  SUBTYPE OF (oo_generic_association) ;
   base : OPTIONAL oo_association;
   multiplicity : cardinality_association_select;
  WHERE
    WR1: SELF\oo_generic_association.reading_direction IN SELF\oo_generic_association.connection;
END_ENTITY;

ENTITY oo_attribute;
   definition : oo_extended_classifier_select;
   description : OPTIONAL text_select;
   id : element_identifier;
   name : label;
   owner : oo_extended_classifier_select;
   visibility : label;
END_ENTITY;

ENTITY oo_attribute_instance;
   attribute_value : label;
   definition : oo_attribute;
   owner : oo_extended_classifier_select;
END_ENTITY;

ENTITY oo_attribute_link_end_association;
   attribute_instance : oo_attribute_instance;
   description : text_select;
   link_end : oo_link_end;
END_ENTITY;

ENTITY oo_behavioural_feature
  ABSTRACT SUPERTYPE   OF ( ONEOF(oo_method, oo_operation, oo_reception) ) ;
   name : label;
   owner : oo_classifier_select;
   visibility : label;
END_ENTITY;
```

```
ENTITY oo_call_action
  SUBTYPE OF (oo_action) ;
   operation : oo_operation;
END_ENTITY;

ENTITY oo_class;
   associated_version : configuration_element_version;
   description : OPTIONAL text_select;
   id : element_identifier;
   is_active : BOOLEAN;
   name : label;
   namespace : OPTIONAL oo_namespace_select;
   visibility : label;
END_ENTITY;

ENTITY oo_classifier_role;
   multiplicity : cardinality_association_select;
  INVERSE
    association_end_role : SET[0:?] OF oo_association_end_role FOR role_type;
END_ENTITY;

ENTITY oo_collaboration;
   description : OPTIONAL text_select;
   id : element_identifier;
   name : label;
   representing : oo_classifier_or_operation_select;
END_ENTITY;

ENTITY oo_component;
   description : OPTIONAL text_select;
   id : element_identifier;
   name : label;
   namespace : OPTIONAL oo_namespace_select;
   visibility : label;
END_ENTITY;

ENTITY oo_component_allocation;
   deployment_location : physical_instance_reference;
   description : OPTIONAL text_select;
   resident : oo_component;
END_ENTITY;

ENTITY oo_constraint;
   body : textual_specification;
END_ENTITY;

ENTITY oo_constraint_model_element_relationship;
   constraint : oo_constraint;
   model_element : oo_model_element_select;
END_ENTITY;

ENTITY oo_create_action
  SUBTYPE OF (oo_action) ;
   instantiation : oo_classifier_select;
END_ENTITY;

ENTITY oo_dependency;
   client : oo_model_element_select;
   description : text_select;
   supplier : oo_model_element_select;
END_ENTITY;
```

```
ENTITY oo_element_import;
  alias_name : OPTIONAL label;
  container : oo_package;
  model_element : oo_model_element_select;
  name : label;
  visibility : label;
END_ENTITY;

ENTITY oo_element_residence;
  description : OPTIONAL text_select;
  implementation_location : oo_component;
  resident : oo_model_element_select;
  visibility : label;
END_ENTITY;

ENTITY oo_extension;
  base : oo_use_case;
  condition : text_select;
  extension : oo_use_case;
  extension_point : oo_extension_point;
END_ENTITY;

ENTITY oo_extension_point;
  location : text_select;
  use_case : oo_use_case;
END_ENTITY;

ENTITY oo_generalization;
  child : oo_generalizable_element_select;
  discriminator : label;
  parent : oo_generalizable_element_select;
END_ENTITY;

ENTITY oo_generic_association
 ABSTRACT SUPERTYPE  OF ( ONEOF(oo_association, oo_association_role) ) ;
  description : OPTIONAL text_select;
  id : element_identifier;
  name : label;
  reading_direction : oo_generic_association_end;
  visibility : label;
 INVERSE
   connection : SET[2:?] OF oo_generic_association_end FOR association;
END_ENTITY;

ENTITY oo_generic_association_end
 ABSTRACT SUPERTYPE  OF ( ONEOF(oo_association_end, oo_association_end_role) ) ;
  aggregation : label;
  association : oo_generic_association;
  description : OPTIONAL text_select;
  id : element_identifier;
  is_navigable : BOOLEAN;
  multiplicity : cardinality_association_select;
  name : label;
  visibility : label;
END_ENTITY;

ENTITY oo_inclusion;
  addition : oo_use_case;
  base : oo_use_case;
END_ENTITY;
```

```
ENTITY oo_instance_classifier_relationship;
  classifier : oo_extended_classifier_select;
  description : text_select;
  instance : oo_instance_select;
END_ENTITY;

ENTITY oo_interaction;
  description : OPTIONAL text_select;
  id : element_identifier;
  interaction_context : oo_collaboration;
  name : label;
END_ENTITY;

ENTITY oo_interface;
  associated_version : configuration_element_version;
  description : OPTIONAL text_select;
  id : element_identifier;
  name : label;
  namespace : OPTIONAL oo_namespace_select;
END_ENTITY;

ENTITY oo_link;
  definition : oo_generic_association;
 INVERSE
   connection : SET[2:?] OF oo_link_end FOR link;
END_ENTITY;

ENTITY oo_link_end;
  definition : oo_generic_association_end;
  instance : oo_instance_select;
  link : oo_link;
 WHERE
   WR1: definition.association :=: link.definition;
END_ENTITY;

ENTITY oo_message;
  action : oo_action;
  communication : OPTIONAL oo_association_role;
  interaction : oo_interaction;
  name : label;
  receiver : oo_classifier_role;
  sender : oo_classifier_role;
  sequence_number : LIST[1:?] OF natural_number;
END_ENTITY;

ENTITY oo_message_temporal_relationship;
  predecessor : oo_message;
  successor : oo_message;
END_ENTITY;

ENTITY oo_method
 SUBTYPE OF (oo_behavioural_feature) ;
  body : textual_specification;
  description : OPTIONAL text_select;
  id : element_identifier;
  specification : oo_operation;
END_ENTITY;

ENTITY oo_model_element_stereotype_relationship;
  model_element : oo_model_element_select;
  stereotype : oo_stereotype;
END_ENTITY;
```

```
ENTITY oo_model_element_tagged_value_relationship;
  model_element : oo_model_element_select;
  tagged_value : oo_tagged_value;
END_ENTITY;

ENTITY oo_object;
  definition : oo_class;
  description : OPTIONAL text_select;
  id : element_identifier;
END_ENTITY;

ENTITY oo_operation
 SUBTYPE OF (oo_behavioural_feature) ;
  concurrency : label;
  description : OPTIONAL text_select;
  id : element_identifier;
  is_abstract : BOOLEAN;
  specification : textual_specification;
END_ENTITY;

ENTITY oo_operation_interface_association;
  description : OPTIONAL text_select;
  interface : oo_interface;
  operation : oo_operation;
END_ENTITY;

ENTITY oo_package
 SUPERTYPE OF ( oo_view ) ;
  associated_version : configuration_element_version;
  description : OPTIONAL text_select;
  id : element_identifier;
  name : label;
  visibility : label;
 INVERSE
   element_import : SET[0:?] OF oo_element_import FOR container;
END_ENTITY;

ENTITY oo_parameter;
  behavioural_feature : oo_behavioural_feature;
  default_value : label;
  kind : label;
  name : label;
  parameter_type : oo_classifier_select;
  visibility : label;
END_ENTITY;

ENTITY oo_reception
 SUBTYPE OF (oo_behavioural_feature) ;
  signal : oo_signal;
  specification : text_select;
END_ENTITY;

ENTITY oo_send_action
 SUBTYPE OF (oo_action) ;
  signal : oo_signal;
END_ENTITY;

ENTITY oo_signal;
  name : label;
  namespace : OPTIONAL oo_namespace_select;
END_ENTITY;
```

```
ENTITY oo_signal_behavioural_feature_relationship;
  behavioural_feature_context : oo_behavioural_feature;
  raised_signal : oo_signal;
END_ENTITY;

ENTITY oo_stereotype;
  associated_version : configuration_element_version;
  description : OPTIONAL text_select;
  id : element_identifier;
  name : label;
  visibility : label;
END_ENTITY;

ENTITY oo_stimulus;
  dispatch_action : oo_action;
  receiver : oo_instance_select;
  sender : oo_instance_select;
END_ENTITY;

ENTITY oo_stimulus_argument;
  argument : oo_instance_select;
  stimulus : oo_stimulus;
END_ENTITY;

ENTITY oo_tagged_value;
  id : element_identifier;
  initial_value : text_select;
  name : label;
END_ENTITY;

ENTITY oo_use_case;
  description : OPTIONAL text_select;
  id : element_identifier;
  name : label;
  namespace : OPTIONAL oo_namespace_select;
END_ENTITY;

ENTITY oo_view
 SUBTYPE OF (oo_package) ;
  view_type : label;
END_ENTITY;

ENTITY oo_view_context_element_relationship;
  represented_model_element : oo_extended_model_element_select;
  view : oo_view;
END_ENTITY;

ENTITY oo_view_relationship;
  base : oo_view;
  description : text_select;
  reference_view : oo_view;
  relationship_type : label;
END_ENTITY;

ENTITY oo_view_system_view_relationship;
  oo_view : oo_view;
  system_context : system_view;
END_ENTITY;

ENTITY organization;
  delivery_address : OPTIONAL address;
```

```
  description : OPTIONAL text_select;
  id : basic_identifier;
  name : label;
  postal_address : OPTIONAL address;
  visitor_address : OPTIONAL address;
END_ENTITY;

ENTITY organization_relationship;
  description : OPTIONAL text_select;
  name : label;
  related_organization : organization;
  relating_organization : organization;
 WHERE
   WR1: related_organization :<>: relating_organization;
END_ENTITY;

ENTITY package
 SUPERTYPE OF ( selection_package ) ;
  description : OPTIONAL text_select;
  discriminator : text;
  id : element_identifier;
  name : label;
 INVERSE
   element : SET[0:?] OF package_element_assignment FOR package;
  UNIQUE
   UR1: id;
END_ENTITY;

ENTITY package_classification_assignment;
  assigned_package : package;
  classification_system : package_classification_system;
END_ENTITY;

ENTITY package_classification_system;
  description : OPTIONAL text_select;
  id : element_identifier;
  name : label;
END_ENTITY;

ENTITY package_element_assignment;
  description : OPTIONAL text_select;
  element : package_element_select;
  package : package;
  reference_name : OPTIONAL label;
END_ENTITY;

ENTITY package_hierarchy_relationship;
  sub_package : package;
  super_package : package;
END_ENTITY;

ENTITY partial_document_assignment
 SUBTYPE OF (document_assignment) ;
  document_portion : label;
END_ENTITY;

ENTITY partial_system_view
 SUBTYPE OF (system_view) ;
  is_relevant_for : system_view_context;
 INVERSE
   assigned_to_systems : SET[1:?] OF system_view_assignment FOR assigned_view;
END_ENTITY;
```

```
ENTITY partial_system_view_relationship
  SUPERTYPE OF ( triggered_system_view_relationship ) ;
   description : OPTIONAL text_select;
   related : partial_system_view;
   relating : partial_system_view;
   relationship_type : label;
   system_definition_context : system_definition;
  WHERE
    correct_relationship: related :<>: relating;
END_ENTITY;

ENTITY persistent_storage
  SUBTYPE OF (general_functionality_instance) ;
   id : element_identifier;
   name : label;
   permanent : LOGICAL;
   presentation_id : OPTIONAL label;
   read_only : LOGICAL;
   storage_access : label;
   store_size : OPTIONAL INTEGER;
   UNIQUE
    UR1: id;
END_ENTITY;

ENTITY persistent_storage_equivalence_relationship;
   equivalent_storage : SET[2:?] OF persistent_storage_reference;
   valid_context : functional_reference_configuration;
END_ENTITY;

ENTITY persistent_storage_reference
  SUBTYPE OF (functionality_instance_reference) ;
  SELF\functionality_instance_reference. RENAMED    referenced_functionality_instance : persistent_storage;
END_ENTITY;

ENTITY person;
   address : OPTIONAL address;
   first_name : OPTIONAL label;
   id : basic_identifier;
   last_name : OPTIONAL label;
   middle_names : OPTIONAL LIST[1:?] OF label;
   prefix_titles : OPTIONAL LIST[1:?] OF label;
   suffix_titles : OPTIONAL LIST[1:?] OF label;
   UNIQUE
    UR1: id;
END_ENTITY;

ENTITY person_in_organization;
   associated_organization : organization;
   associated_person : person;
   description : OPTIONAL text_select;
   role : label;
END_ENTITY;

ENTITY person_organization_assignment;
   assigned_person_organization : person_organization_select;
   assigned_to : person_organization_assignment_select;
   description : OPTIONAL text_select;
   role : label;
END_ENTITY;
```

```
ENTITY physical_binding;
  actual_port : actual_physical_port;
  formal_port : formal_physical_port;
 WHERE
   WR1: formal_port.port_of :=: actual_port.port_of.definition;
END_ENTITY;

ENTITY physical_composition_relationship;
  assembly : general_physical_definition;
  component : physical_instance;
  description : OPTIONAL text_select;
END_ENTITY;

ENTITY physical_connection;
  connected : physical_port;
  connecting : physical_port;
END_ENTITY;

ENTITY physical_instance;
  definition : general_physical_definition;
  description : OPTIONAL text_select;
  id : element_identifier;
  name : label;
  presentation_id : OPTIONAL label;
 INVERSE
   actual_port : SET[0:?] OF actual_physical_port FOR port_of;
  UNIQUE
   UR1: definition, id;
END_ENTITY;

ENTITY physical_instance_reference;
  description : OPTIONAL text_select;
  id : element_identifier;
  name : label;
  reference_for_instance : physical_instance;
END_ENTITY;

ENTITY physical_link_definition
 SUBTYPE OF (general_physical_definition) ;
END_ENTITY;

ENTITY physical_node_definition
 SUBTYPE OF (general_physical_definition) ;
END_ENTITY;

ENTITY physical_port
 ABSTRACT SUPERTYPE   OF ( ONEOF(actual_physical_port, formal_physical_port) ) ;
  description : OPTIONAL text_select;
  direction : label;
  name : label;
END_ENTITY;

ENTITY physical_reference_configuration;
  description : OPTIONAL text_select;
END_ENTITY;

ENTITY physical_reference_relationship;
  child : physical_instance_reference;
  mirror_of : physical_composition_relationship;
  parent : physical_instance_reference;
  valid_configuration : physical_reference_configuration;
END_ENTITY;
```

```
ENTITY plus_minus_bounds;
  distribution_function : OPTIONAL textual_specification;
  limited_value : nominal_value;
  lower_bound : NUMBER;
  significant_digits : OPTIONAL INTEGER;
  upper_bound : NUMBER;
END_ENTITY;

ENTITY project;
  actual_end_date : OPTIONAL date_time;
  actual_start_date : OPTIONAL date_time;
  description : OPTIONAL text_select;
  id : element_identifier;
  name : label;
  planned_end_date : OPTIONAL period_or_date_select;
  planned_start_date : OPTIONAL event_or_date_select;
  work_program : SET[0:?] OF engineering_process_activity;
END_ENTITY;

ENTITY project_event_reference;
  description : OPTIONAL text_select;
  event_type : label;
  offset : value_with_unit;
END_ENTITY;

ENTITY project_relationship;
  description : OPTIONAL text_select;
  related : project;
  relating : project;
  relation_type : label;
END_ENTITY;

ENTITY property_assignment;
  assigned_to : property_assignment_select;
  assignment_rationale : OPTIONAL text_select;
  measurement_method : label;
  property : property_value;
  property_name : label;
END_ENTITY;

ENTITY property_definition;
  allowed_unit : SET[0:?] OF unit;
  description : OPTIONAL text_select;
  property_type : label;
END_ENTITY;

ENTITY property_relationship;
  description : OPTIONAL text_select;
  related : property_definition;
  relating : property_definition;
  relation_type : label;
END_ENTITY;

ENTITY property_value;
  definition : property_definition;
  property_value_name : label;
  specified_value : property_value_select;
END_ENTITY;

ENTITY property_value_function;
  defines_property_value_merit : property_value;
```

```
  function_specification : text_select;
END_ENTITY;

ENTITY property_value_relationship;
  related : property_value;
  relating : property_value;
  relation_description : OPTIONAL text_select;
  relation_type : label;
END_ENTITY;

ENTITY rank_assignment;
  assigned_rank : ranking_element;
  assigned_to_element : ranked_element_select;
  assignment_criteria : OPTIONAL label;
  assignment_description : OPTIONAL text_select;
  relevant_elements : rank_group;
END_ENTITY;

ENTITY rank_group;
  classification_criteria : ranking_type;
  name : label;
 INVERSE
   compared_element : SET[0:?] OF rank_assignment FOR relevant_elements;
END_ENTITY;

ENTITY rank_relation;
  higher_rank : ranking_element;
  lower_rank : ranking_element;
END_ENTITY;

ENTITY ranking_element;
  description : OPTIONAL text_select;
  name : label;
 INVERSE
   higher_ranked_element : SET[0:?] OF rank_relation FOR lower_rank;
   lower_ranked_element : SET[0:?] OF rank_relation FOR higher_rank;
END_ENTITY;

ENTITY ranking_system;
  description : OPTIONAL text_select;
  highest_rank : ranking_element;
  name : label;
END_ENTITY;

ENTITY real_data_type_definition
 SUBTYPE OF (elementary_maths_space) ;
END_ENTITY;

ENTITY real_interval
 ABSTRACT SUPERTYPE   OF ( ONEOF(finite_real_interval, hibound_real_interval, lobound_real_interval) )
 SUBTYPE OF (maths_space) ;
END_ENTITY;

ENTITY realized_system;
  description : text_select;
  id : element_identifier;
  name : label;
  realization_of : system_instance;
  UNIQUE
   UR1: id, realisation_of;
END_ENTITY;
```

ENTITY **realized_system_composition_relationship**;
  component : realized_system;
  description : text_select;
  mirror_of : system_composition_relationship;
  system : realized_system;
END_ENTITY;

ENTITY **record_data_type_definition**
 SUBTYPE OF (user_defined_data_type_definition) ;
END_ENTITY;

ENTITY **record_data_type_member**;
  child : data_field;
  parent : record_data_type_definition;
END_ENTITY;

ENTITY **recursive_data_type_definition**
 SUBTYPE OF (user_defined_data_type_definition) ;
  redefines : data_type_definition_select;
END_ENTITY;

ENTITY **requirement_allocation_property_relationship**;
  allocated_requirement : requirement_allocation_relationship;
  define_property_value : property_value;
  description : OPTIONAL text_select;
END_ENTITY;

ENTITY **requirement_allocation_relationship**
 SUPERTYPE OF (specific_requirement_allocation_relationship) ;
  description : OPTIONAL text_select;
  relation_to : requirement_allocation_select;
  requirement : requirement_instance;
  role : label;
END_ENTITY;

ENTITY **requirement_class**;
  description : OPTIONAL text_select;
  id : element_identifier;
  name : label;
  UNIQUE
   UR1: id;
END_ENTITY;

ENTITY **requirement_class_relationship**;
  description : OPTIONAL text_select;
  related_class : requirement_class;
  relating_class : requirement_class;
  relationship_type : label;
END_ENTITY;

ENTITY **requirement_composition_relationship**;
  child_requirement : requirement_occurence;
  description : OPTIONAL text_select;
  index : label;
  parent_definition : requirement_definition;
  UNIQUE
   UR1: index, parent_definition;
END_ENTITY;

ENTITY **requirement_definition**
 ABSTRACT SUPERTYPE OF (ONEOF(model_defined_requirement_definition, structured_requirement_definition, textual_requirement_definition)) ;

```
  associated_version : configuration_element_version;
  id : element_identifier;
  name : OPTIONAL label;
 INVERSE
  composed_of : SET[0:?] OF requirement_composition_relationship FOR parent_definition;
  in_requirement_class : SET[0:1] OF requirement_requirement_class_assignment FOR requirement;
 UNIQUE
  UR1: id;
END_ENTITY;

ENTITY requirement_instance;
  definition : requirement_occurence;
  id : element_identifier;
  name : label;
 INVERSE
  implied_external : SET[0:?] OF implied_external_interaction FOR associated_requirement;
 UNIQUE
  UR1: definition, id;
END_ENTITY;

ENTITY requirement_occurence;
  definition : requirement_definition;
  id : element_identifier;
  name : label;
 INVERSE
  child_of : SET[0:?] OF requirement_composition_relationship FOR child_requirement;
 UNIQUE
  UR1: definition, id;
END_ENTITY;

ENTITY requirement_relationship;
  description : OPTIONAL text_select;
  relationship_type : label;
END_ENTITY;

ENTITY requirement_relationship_context_assignment;
  assigned_requirement_relationship : assigned_requirement_relationship_select;
  description : OPTIONAL text_select;
  system_context : system_view;
END_ENTITY;

ENTITY requirement_relationship_input_assignment;
  assigned_instance : requirement_instance;
  input_requirement : requirement_relationship;
END_ENTITY;

ENTITY requirement_relationship_resulting_relationship;
  motivation : OPTIONAL text_select;
  requirement_relationship : requirement_relationship;
  resulting_requirement : requirement_instance;
  role : label;
END_ENTITY;

ENTITY requirement_requirement_class_assignment;
  class : requirement_class;
  motivation : OPTIONAL text_select;
  requirement : requirement_definition;
END_ENTITY;

ENTITY requirement_system_view_assignment
 SUPERTYPE OF ( root_requirement_system_view_assignment ) ;
  description : OPTIONAL text_select;
```

```
  requirement : requirement_instance;
  system_view : system_view;
END_ENTITY;

ENTITY requirement_traces_to_requirement_relationship;
  motivation : OPTIONAL text_select;
  source_requirement : requirement_instance;
  traced_requirement : requirement_instance;
  valid_context : system_definition;
END_ENTITY;

ENTITY root_requirement_system_view_assignment
 SUBTYPE OF (requirement_system_view_assignment) ;
  index : label;
  UNIQUE
   UR1: index, system_view;
END_ENTITY;

ENTITY selection_package
 SUBTYPE OF (package) ;
  selection_type : label;
END_ENTITY;

ENTITY single_cardinality;
  defined_value : single_cardinality_select;
END_ENTITY;

ENTITY specific_requirement_allocation_relationship
 SUBTYPE OF (requirement_allocation_relationship) ;
  specific_element : specific_element_select;
 WHERE
  WR1: 'SYSTEMS_ENGINEERING_DATA_REPRESENTATION.FUNCTIONALITY_INSTANCE_REFERENCE' IN
TYPEOF(SELF\requirement_allocation_relationship.relation_to);
END_ENTITY;

ENTITY specification_state_assignment;
  assigned_to : fsm_state;
  specification : textual_specification;
END_ENTITY;

ENTITY start_order
 SUBTYPE OF (work_order) ;
  start_order_type : label;
END_ENTITY;

ENTITY start_request
 SUBTYPE OF (work_request) ;
  request_type : label;
END_ENTITY;

ENTITY state_context_relationship;
  in_context : generic_state_context;
  state : fsm_generic_state;
END_ENTITY;

ENTITY state_function_interaction_port;
  port_of : functional_state_context;
END_ENTITY;

ENTITY state_machine_functional_behaviour_model
 SUBTYPE OF (functional_behaviour_model) ;
```

```
  INVERSE
    behaviour_constraint : SET[1:?] OF fsm_model FOR behaviour_model;
END_ENTITY;

ENTITY state_transition_specification_assignment;
   assigned_to : fsm_state_transition;
   specification : textual_specification;
END_ENTITY;

ENTITY string_data_type_definition
  SUBTYPE OF (elementary_maths_space) ;
   size : OPTIONAL finite_integer_interval;
END_ENTITY;

ENTITY structured_requirement_definition
  SUBTYPE OF (requirement_definition) ;
   required_characteristic : property_value;
END_ENTITY;

ENTITY system_composition_relationship;
   component_system : system_instance;
   decomposed_system : system_definition;
   description : OPTIONAL text_select;
   relationship_type : label;
END_ENTITY;

ENTITY system_definition
  SUBTYPE OF (system_view) ;
   life_cycle_stage : label;
  INVERSE
    assigned_to_system : SET[0:?] OF partial_system_view_relationship FOR system_definition_context;
    scenarios_for_system : SET[0:?] OF system_view_assignment FOR system_specification;
END_ENTITY;

ENTITY system_functional_configuration;
   functional_configuration : functional_reference_configuration;
   system : context_function_relationship;
END_ENTITY;

ENTITY system_instance;
   definition : system_definition;
   description : OPTIONAL text_select;
   id : element_identifier;
   name : label;
   UNIQUE
    UR1: definition, id;
END_ENTITY;

ENTITY system_instance_relationship;
   description : text_select;
   name : label;
  INVERSE
    connected_port : SET[2:?] OF system_instance_relationship_port FOR defined_relationship;
END_ENTITY;

ENTITY system_instance_relationship_port;
   cardinality : cardinality_association_select;
   defined_relationship : system_instance_relationship;
   port_of : system_instance;
END_ENTITY;
```

```
ENTITY system_instance_replication_relationship;
  rationale : OPTIONAL text_select;
  replaced_application : system_instance;
  replacing_application : system_instance;
END_ENTITY;

ENTITY system_physical_configuration;
  physical_configuration : physical_reference_configuration;
  system : context_physical_relationship;
END_ENTITY;

ENTITY system_substitution_relationship;
  base : system_composition_relationship;
  description : OPTIONAL text_select;
  substitute : system_composition_relationship;
END_ENTITY;

ENTITY system_view
 ABSTRACT SUPERTYPE   OF ( ONEOF(partial_system_view, system_definition) ) ;
  associated_version : configuration_element_version;
  description : OPTIONAL text_select;
  id : element_identifier;
  name : label;
 INVERSE
   system : SET[0:?] OF context_function_relationship FOR associated_context;
  UNIQUE
   UR1: id;
 WHERE
   WR1: at_most_one_system_function_assigned(SELF);
END_ENTITY;

ENTITY system_view_assignment;
  assigned_view : partial_system_view;
  assignment_comment : OPTIONAL text_select;
  system_specification : system_definition;
END_ENTITY;

ENTITY system_view_context;
  description : OPTIONAL text_select;
  fidelity : label;
  system_viewpoint : label;
END_ENTITY;

ENTITY textual_paragraph;
  element_value : LIST[1:?] OF text_elements;
  name : label;
END_ENTITY;

ENTITY textual_requirement_definition
 SUBTYPE OF (requirement_definition) ;
  description : text_select;
END_ENTITY;

ENTITY textual_section;
  element : LIST[1:?] OF textual_paragraph;
  name : label;
END_ENTITY;

ENTITY textual_specification;
  definition : text_select;
  definition_language : label;
END_ENTITY;
```

```
ENTITY textual_table;
  name : label;
  table_row : LIST[1:?] OF textual_section;
END_ENTITY;

ENTITY triggered_system_view_relationship
 SUBTYPE OF (partial_system_view_relationship) ;
  transition_condition : text;
END_ENTITY;

ENTITY undefined_data_type_definition
 SUBTYPE OF (user_defined_data_type_definition) ;
END_ENTITY;

ENTITY union_data_type_definition
 SUBTYPE OF (user_defined_data_type_definition) ;
END_ENTITY;

ENTITY union_data_type_member;
  child : data_field;
  parent : union_data_type_definition;
END_ENTITY;

ENTITY unit;
  unit_name : label;
END_ENTITY;

ENTITY user_defined_data_type_definition
 ABSTRACT SUPERTYPE  OF ( ONEOF(abstract_data_type_definition, aggregate_data_type_definition, derived_
data_type_definition, record_data_type_definition, recursive_data_type_definition, undefined_data_type_definition,
union_data_type_definition) ) ;
  associated_version : configuration_element_version;
  description : OPTIONAL text_select;
  id : element_identifier;
  name : label;
  UNIQUE
   UR1: id;
END_ENTITY;

ENTITY value_limit
 SUBTYPE OF (value_with_unit) ;
  limit : NUMBER;
  limit_qualifier : label;
END_ENTITY;

ENTITY value_list;
  values : LIST[1:?] OF value_with_unit;
END_ENTITY;

ENTITY value_range
 SUBTYPE OF (value_with_unit) ;
  distribution_function : OPTIONAL textual_specification;
  lower_limit : NUMBER;
  upper_limit : NUMBER;
END_ENTITY;

ENTITY value_with_unit
 ABSTRACT SUPERTYPE  OF ( ONEOF(nominal_value, value_limit, value_range) ) ;
  significant_digits : OPTIONAL INTEGER;
  unit_component : OPTIONAL unit;
END_ENTITY;
```

ENTITY **verification_report_for_verification_specification**;
  description : text_select;
  specific_verification_element : verification_specification_allocation;
  verification_report_entry : verification_result;
END_ENTITY;

ENTITY **verification_result**;
  description : text_select;
  id : element_identifier;
  name : label;
  system_under_verification : verification_specification_system_view_relationship;
  UNIQUE
   UR1: id, system_under_verification;
END_ENTITY;

ENTITY **verification_specification**;
  definition : requirement_occurence;
  description : OPTIONAL text_select;
  id : element_identifier;
  name : label;
  verification_method : label;
END_ENTITY;

ENTITY **verification_specification_allocation**;
  description : OPTIONAL text_select;
  relevant_for : verification_allocation_select;
  specification : verification_specification;
END_ENTITY;

ENTITY **verification_specification_system_view_relationship**;
  assigned_verification : verification_specification;
  description : OPTIONAL text_select;
  index : label;
  system_view : system_view;
END_ENTITY;

ENTITY **view_relationship**;
  child : graphics_view;
  parent : graphics_view;
  valid_in : multi_level_view;
END_ENTITY;

ENTITY **visual_element**
 ABSTRACT SUPERTYPE OF (ONEOF(actual_port_position, formal_port_position, graphics_link, graphics_node)) ;
  view : graphics_view;
END_ENTITY;

ENTITY **work_order**
 ABSTRACT SUPERTYPE OF (ONEOF(change_order, start_order)) ;
  description : OPTIONAL text_select;
  id : element_identifier;
  is_controlling : SET[1:?] OF engineering_process_activity;
  status : label;
  version_id : OPTIONAL element_identifier;
END_ENTITY;

ENTITY **work_request**
 ABSTRACT SUPERTYPE OF (ONEOF(change_request, start_request)) ;
  description : text_select;
  id : element_identifier;
  notified_person : SET[0:?] OF date_and_person_organization;

```
  requestor : date_and_person_organization;
  scope : SET[0:?] OF specification_element_select;
  status : label;
  version_id : OPTIONAL element_identifier;
END_ENTITY;

END_SCHEMA;
```

ARM EXPRESS text (text version)

```
(*
ISO TC184/SC4/WG3 N1355 - ISO/PAS 20542 system engineering and design - EXPRESS ARM
*)

SCHEMA system_engineering_and_design_arm;

TYPE approval_assignment_select = SELECT
 (change_report,
  configuration_element,
  configuration_element_version,
  critical_issue,
  critical_issue_impact,
  document_assignment,
  element_critical_issue_relationship,
  engineering_process_activity,
  instance_definition_select,
  package_element_assignment,
  partial_system_view_relationship,
  project,
  requirement_allocation_relationship,
  work_order,
  work_request);
END_TYPE;

TYPE assessment_assignment_select = SELECT
 (configuration_element_version,
  engineering_process_activity,
  instance_definition_select,
  project,
  requirement_allocation_relationship);
END_TYPE;

TYPE assigned_requirement_relationship_select = SELECT
 (requirement_allocation_relationship,
  requirement_relationship);
END_TYPE;

TYPE basic_identifier = STRING;
END_TYPE;

TYPE boolean_value = BOOLEAN;
END_TYPE;

TYPE buffer_synchronisation_enumeration = ENUMERATION OF
 (asynchronous,
  synchronouos);
END_TYPE;

TYPE cardinality_association_select = SELECT
 (cardinality_list,
  cardinality_range,
```

```
   single_cardinality);
END_TYPE;

TYPE causal_weight_select = SELECT
  (cb_transition_unbounded_weight,
   natural_number);
END_TYPE;

TYPE change_element_select = SELECT
  (instance_definition_select);
END_TYPE;

TYPE change_report_element_select = SELECT
  (instance_definition_select);
END_TYPE;

TYPE control_characters = ENUMERATION OF
  (cr,
   tab);
END_TYPE;

TYPE control_type_enumeration = ENUMERATION OF
  (activate,
   activate_deactivate);
END_TYPE;

TYPE data_composition_select = SELECT
  (actual_io_port,
   formal_io_port,
   io_composition_port);
END_TYPE;

TYPE data_direction = ENUMERATION OF
  (from_system,
   to_system);
END_TYPE;

TYPE data_type_definition_select = SELECT
  (maths_space,
   user_defined_data_type_definition);
END_TYPE;

TYPE data_type_value_select = SELECT
  (boolean_value,
   complex_value,
   compound_value,
   integer_value,
   logical_value,
   real_value,
   text);
END_TYPE;

TYPE date_assignment_select = SELECT
  (approval,
   approval_assignment,
   assessment,
   configuration_element,
   configuration_element_version,
   configuration_element_version_relationship,
   context_physical_relationship,
   critical_issue,
```

```
   critical_issue_impact,
   document_assignment,
   documentation_relationship,
   element_critical_issue_relationship,
   engineering_process_activity,
   engineering_process_activity_element_assignment,
   instance_definition_select,
   justification,
   justification_relationship,
   package_element_assignment,
   partial_system_view_relationship,
   project,
   requirement_allocation_relationship,
   requirement_system_view_assignment,
   work_order,
   work_request);
END_TYPE;

TYPE default_context_select = SELECT
  (fsm_generic_state,
   functional_state_context);
END_TYPE;

TYPE definition_select = SELECT
  (fsm_state,
   functional_state_context,
   general_function_definition,
   general_physical_definition,
   oo_view,
   system_view);
END_TYPE;

TYPE effective_element_select = SELECT
  (instance_definition_select,
   person_organization_assignment);
END_TYPE;

TYPE event_or_date_select = SELECT
  (date_time,
   project_event_reference);
END_TYPE;

TYPE external_element_select = SELECT
  (function_instance,
   physical_instance);
END_TYPE;

TYPE fsm_interaction_select = SELECT
  (initial_state_transition_specification_assignment,
   specification_state_assignment,
   state_transition_specification_assignment);
END_TYPE;

TYPE function_role_enumeration = ENUMERATION OF
  (external_element,
   system_function);
END_TYPE;

TYPE identifier = STRING;
END_TYPE;
```

```
TYPE instance_definition_select = SELECT
  (clock,
   clock_assignment_relationship,
   data_instance,
   documentation_reference,
   functionality_instance_reference,
   general_function_definition,
   general_functionality_instance,
   general_physical_definition,
   oo_model_element_select,
   physical_instance,
   physical_instance_reference,
   realized_system,
   requirement_definition,
   requirement_instance,
   system_instance,
   system_view);
END_TYPE;

TYPE integer_value = INTEGER;
END_TYPE;

TYPE issue_source_select = SELECT
  (instance_definition_select);
END_TYPE;

TYPE justification_assignment_select = SELECT
  (engineering_process_activity,
   instance_definition_select,
   partial_system_view_relationship,
   requirement_allocation_relationship);
END_TYPE;

TYPE label = STRING;
END_TYPE;

TYPE link_select = SELECT
  (cb_input_relationship,
   cb_output_relationship,
   clock_assignment_relationship,
   fsm_initial_state_transition,
   fsm_state_transition,
   functional_link,
   oo_action_state_transition,
   oo_component_allocation,
   oo_constraint_model_element_relationship,
   oo_dependency,
   oo_element_import,
   oo_element_residence,
   oo_extension,
   oo_generalization,
   oo_generic_association,
   oo_inclusion,
   oo_link,
   oo_message,
   oo_operation_interface_association,
   oo_signal_behavioural_feature_relationship,
   physical_instance);
END_TYPE;

TYPE logical_value = LOGICAL;
END_TYPE;
```

```
TYPE natural_number = INTEGER;
WHERE
  WR1: SELF > 0;
END_TYPE;

TYPE node_select = SELECT
  (cb_place_reference,
   cb_transition_relationship,
   clock,
   fsm_state,
   general_functionality_instance,
   graphics_view,
   oo_action,
   oo_action_state,
   oo_actor,
   oo_association_class,
   oo_class,
   oo_component,
   oo_constraint,
   oo_interface,
   oo_object,
   oo_package,
   oo_send_action,
   oo_use_case,
   physical_instance,
   textual_paragraph);
END_TYPE;

TYPE oo_classifier_or_operation_select = SELECT
  (oo_classifier_select,
   oo_operation);
END_TYPE;

TYPE oo_classifier_select = SELECT
  (oo_actor,
   oo_class,
   oo_component,
   oo_interface,
   oo_signal,
   oo_use_case);
END_TYPE;

TYPE oo_extended_classifier_select = SELECT
  (oo_classifier_select,
   physical_instance);
END_TYPE;

TYPE oo_extended_model_element_select = SELECT
  (fsm_generic_state,
   generic_state_context,
   justification,
   oo_generalizable_element_select,
   oo_model_element_select);
END_TYPE;

TYPE oo_feature_select = SELECT
  (oo_attribute,
   oo_behavioural_feature);
END_TYPE;

TYPE oo_generalizable_element_select = SELECT
  (oo_collaboration,
```

```
   oo_extended_classifier_select,
   oo_generic_association,
   oo_package,
   oo_stereotype);
END_TYPE;

TYPE oo_instance_select = SELECT
  (oo_attribute_instance,
   oo_extended_classifier_select,
   oo_object);
END_TYPE;

TYPE oo_model_element_select = SELECT
  (oo_action,
   oo_constraint,
   oo_extension_point,
   oo_feature_select,
   oo_generic_association_end,
   oo_interaction,
   oo_link,
   oo_link_end,
   oo_message,
   oo_parameter,
   oo_relationship_select,
   oo_stimulus);
END_TYPE;

TYPE oo_namespace_select = SELECT
  (oo_collaboration,
   oo_extended_classifier_select,
   oo_package);
END_TYPE;

TYPE oo_relationship_select = SELECT
  (oo_dependency,
   oo_extension,
   oo_generalization,
   oo_generic_association,
   oo_inclusion);
END_TYPE;

TYPE package_element_select = SELECT
  (configuration_element,
   configuration_element_version,
   engineering_process_activity,
   instance_definition_select,
   project);
END_TYPE;

TYPE period_or_date_select = SELECT
  (date_time,
   project_event_reference,
   value_with_unit);
END_TYPE;

TYPE person_organization_assignment_select = SELECT
  (approval,
   approval_assignment,
   assessment,
   change_report,
   configuration_element,
```

```
    configuration_element_version,
    critical_issue,
    critical_issue_impact,
    document_assignment,
    element_critical_issue_relationship,
    engineering_process_activity,
    instance_definition_select,
    justification,
    justification_relationship,
    package_element_assignment,
    partial_system_view_relationship,
    project,
    requirement_allocation_relationship,
    requirement_system_view_assignment,
    work_order);
END_TYPE;

TYPE person_organization_select = SELECT
  (organization,
    person_in_organization);
END_TYPE;

TYPE physical_element_role_enumeration = ENUMERATION OF
  (external_element,
    system_element);
END_TYPE;

TYPE port_data_relation = ENUMERATION OF
  (consumer,
    producer);
END_TYPE;

TYPE port_position_select = SELECT
  (io_port,
    oo_generic_association_end,
    oo_link_end,
    physical_port);
END_TYPE;

TYPE port_type = ENUMERATION OF
  (control,
    input,
    mechanism,
    output);
END_TYPE;

TYPE property_assignment_select = SELECT
  (data_instance,
    documentation_reference,
    functionality_instance_reference,
    general_function_definition,
    general_physical_definition,
    oo_model_element_select,
    package,
    physical_instance_reference,
    realized_system,
    requirement_definition,
    system_view);
END_TYPE;

TYPE property_value_select = SELECT
  (boolean_value,
```

```
   text_select,
   value_list,
   value_with_unit);
END_TYPE;

TYPE ranked_element_select = SELECT
  (critical_issue_impact,
   effectiveness_measure,
   instance_definition_select);
END_TYPE;

TYPE ranking_type = ENUMERATION OF
  (cost,
   criticality,
   priority,
   project_criticality);
END_TYPE;

TYPE real_value = REAL;
END_TYPE;

TYPE requirement_allocation_select = SELECT
  (data_instance,
   functional_link_reference,
   functionality_instance_reference,
   functionality_reference_relationship,
   oo_model_element_select,
   physical_instance_reference);
END_TYPE;

TYPE single_cardinality_select = SELECT
  (infinite_cardinality,
   natural_number);
END_TYPE;

TYPE specific_element_select = SELECT
  (cb_place,
   fsm_generic_state);
END_TYPE;

TYPE specification_element_select = SELECT
  (change_report,
   critical_issue,
   critical_issue_impact,
   instance_definition_select);
END_TYPE;

TYPE system_select = SELECT
  (system_instance,
   system_view);
END_TYPE;

TYPE text = STRING;
END_TYPE;

TYPE text_elements = SELECT
  (control_characters,
   text);
END_TYPE;

TYPE text_select = SELECT
  (text,
```

```
  textual_paragraph,
  textual_section,
  textual_table);
END_TYPE;

TYPE timing_type = ENUMERATION OF
 (best_case,
  nominal_case,
  worst_case);
END_TYPE;

TYPE trigger_type_enumeration = ENUMERATION OF
 (flank,
  level);
END_TYPE;

TYPE verification_allocation_select = SELECT
 (functionality_instance_reference,
  physical_instance_reference,
  realized_system,
  requirement_instance,
  system_instance);
END_TYPE;

(* An Abstract_data_type_definition is type of User_defined_data_type_definition that may have the value of any, all or
none of its members. *)
ENTITY abstract_data_type_definition
SUBTYPE OF (user_defined_data_type_definition);
END_ENTITY;

(* An Abstract_data_type_member is a relationship between an Abstract_data_type_definition and a Data_instance it
contains. The members of an Abstract_data_type_definition are independent of each other. *)
ENTITY abstract_data_type_member;
 (* The child specifies the Data_instance object in the relationship. *)
 child : data_instance;
 (* The parent specifies the Abstract_data_type_definition object in the relationship. *)
 parent : abstract_data_type_definition;
END_ENTITY;

(* An Actual_io_port is a type of Io_port and an element of the interface of a General_functionality_instance object. An
Actual_io_port defines an input or an output parameter to the General_functionality_instance object it is referring to via
the port_of attribute. <note>Flow ports are classified according to three criteria in the data model:<ol><li>Whether the
port is formal (attached to a General_function_definition type object) or actual (attached to a General_functionality_in-
stance type object).</li><li>Whether the port is an input or output port.</li><li>Whether the port is a flow or control port (a
flow port carries data whereas control port carries triggering information such as start, stop, suspend, resume). This kind
of port is further defined in control_io_port</li><p>From this classification, the role a port plays with regard to the Flow
objects connected to the port can be derived. A port can “consume” io data (the data on the flow is delivered to the port)
or it can “produce” io data (the data on the flow is delivered from the port).</p><p>For instance, a Formal_io_port whose
direction attribute is “input” produces data, while an Actual_io_port whose direction attribute is “input” consumes data. In
the data model this information is captured in the derived attribute role.</p><p>This attribute is used to ensure that Io ob-
jects are connected correctly (an io object must have one producer and one consumer), and that Io_port_binding objects
applied only to ports where the actual_port in the binding is a producer of information and the formal_port in the binding
is a consumer or vice versa.</p></note> *)
ENTITY actual_io_port
SUBTYPE OF (io_port);
 (* The port_of specifies the General_functionality_instance object of which the port_of is part.
 *)
 port_of : general_functionality_instance;
DERIVE
 (* The role specifies one of ‘producer’, or ‘consumer’. *)
 SELF\io_port.role : port_data_relation := determineactualportrole(SELF);
```

```
INVERSE
  (* The assigned_to specifies the Actual_io_port object to which the Io_buffer is assigned. *)
  assigned_buffer : SET [0 : 1] OF io_buffer FOR assigned_to;
UNIQUE
  UR1: port_of, SELF\io_port.io_port_number, SELF\io_port.port_type;
END_ENTITY;

(* An Actual_physical_port is a type of Physical_port that represents an element in the interface of a Physical_instance
object. The direction of the io objects using the interface is not specified by the Actual_physical_port. *)
ENTITY actual_physical_port
SUBTYPE OF (physical_port);
  (* The port_of specifies the Physical_instance for which the port_of provides part of the interface. *)
  port_of : physical_instance;
END_ENTITY;

(* An Actual_port_position is a type of Visual_element and the representation of the graphical position of an Actual_
io_port  or an Actual_physical_port object. The position is represented by a Graphics_point object. Because the actual
port may have many different positions (the object the port is a port_of can be a physically extended object, such as a
rectangle, rather than a single point) the Actual_port_position shall point to a position that is on the border of the Graph-
ics_node of the object the port is associated with. <note>This rule is not formalized in EXPRESS.</note> *)
ENTITY actual_port_position
SUBTYPE OF (visual_element);
  (* The assigned_to specifies the graphics_node that provides visual presentation information for the Actual_port_posi-
tion. *)
  assigned_to : graphics_node;
  (* The position specifies the visual placement of the port. *)
  position : graphics_point;
  (* The positioned_port specifies the port for which the visual placement information is valid. *)
  positioned_port : port_position_select;
END_ENTITY;

(* An Address is an address of either a Person or an Organization. All attributes of an Address are optional. However an
Address object must have at least one non empty attribute. <note>This entity is taken directly from the PDM Schema.
Should probably be abstracted away from ARM level.</note> *)
ENTITY address;
  (* The country specifies the country of the Address. *)
  country : OPTIONAL label;
  (* The electronic_mail_address specifies an electronic address associated with this Address. *)
  electronic_mail_address : OPTIONAL label;
  (* The facsimile_number specifies a fax associated with this Address. *)
  facsimile_number : OPTIONAL label;
  (* The internal_location specifies a specific location within the Address. <example number="1">An internal_location may
be a room number or a floor of a building.</example> *)
  internal_location : OPTIONAL label;
  (* The postbox_number specifies a postbox associated with the Address. *)
  postbox_number : OPTIONAL label;
  (* The postcode specifies a postcode associated with the Address. *)
  postcode : OPTIONAL label;
  (* The region specifies a region associated with the Address. *)
  region : OPTIONAL label;
  (* The street specifies a street associated with the Address. *)
  street : OPTIONAL label;
  (* The street_number specifies a street number associated with the Address.. *)
  street_number : OPTIONAL label;
  (* The telephone_number specifies a telephone number associated with the Address. *)
  telephone_number : OPTIONAL label;
  (* The telex_number specifies a telex number associated with the Address. *)
  telex_number : OPTIONAL label;
  (* The town specifies a town associated with the Address. *)
  town : OPTIONAL label;
END_ENTITY;
```

```
(* An Aggregate_data_type_definition is is a type of User_defined_data_type_definition that comprises a list (or lists) of values. *)
ENTITY aggregate_data_type_definition
SUBTYPE OF (user_defined_data_type_definition);
  (* The aggregate_type attribute specifies the values that each element in the list can take. *)
  aggregate_type : data_type_definition_select;
  (* The bound specifies the ordered set of elements of the Aggregate_data_type_definition. It is assumed, for the purposes of assigning compound values, that the aggregate is stored in row order. *)
  bound : LIST [1 : ?] OF integer_interval;
END_ENTITY;

(* An Approval is a judgement concerning the quality of those product data that are subject to approval. An Approval represents a formal statement made by technical personnel or management personnel regarding whether or not certain requirements are perceived to be met by the reviewing body. <note number="1">This needs to be reviewed for applicability of text</note> <note number="2">This entity is from the PDM schema.</note> *)
ENTITY approval;
  (* The level indicates the kind of activity that may be performed and granted by the Approval. Where applicable, the following values shall be used: <ul><li>disposition: The approved object is approved for series production;</li><li>equipment order: The approved object has reached a status in which changes are subject to a defined change process and tools and other equipment required for production may be ordered;</li><li>planning: The approved object is technically complete and has reached a status sufficiently stable so that other designs may be based on it.</li></ul> *)
  level : OPTIONAL label;
  (* The status specifies the judgement made about the product data that is the subject of this Approval. *)
  status : label;
END_ENTITY;

(* An Approval_assignment is the assignment of an Approval to an element of a system specification. *)
ENTITY approval_assignment;
  (* The assigned_approval specifies the Approval that is assigned. *)
  assigned_approval : approval;
  (* The assigned_to specifies the system engineering specification element for which the Approval_assignment is valid. *)
  assigned_to : approval_assignment_select;
  (* The description specifies additional information about the Approval_assignment. *)
  description : OPTIONAL text_select;
  (* The status specifies the judgement made about the system engineering specification element that is the subject of this Approval_assignment. *)
  status : label;
END_ENTITY;

(* An Approval_person_organization is the relation between an approval and the juridical person involved in the approval. The semantics of the relation is further defined by the role attribute. *)
ENTITY approval_person_organization;
  (* The authorized_approval specifies the approval for the Approval_person_organization. *)
  authorized_approval : approval;
  (* The person_organization specifies the juridical authority involved in the Approval_person_organization. *)
  person_organization : person_organization_select;
  (* The role specifies the semantics of the Approval_person_organization.<p>Where applicable one of the following values shall be used:</p><ul><li>project responsible: The person approving is doing so in the role of being responsible for the project to which the system specification element belongs;</li><li>verification responsible: The person approving is doing so in the role of being responsible for verification of the specification element.</li></ul> *)
  role : label;
END_ENTITY;

(* An Approval_relationship is a relationship between two Approval objects. <note>Attribute for defining the semantics of the relation shall be added.</note> *)
ENTITY approval_relationship;
  (* The description specifies additional information about the Approval_relationship. *)
  description : OPTIONAL text_select;
  (* The related_approval specifies the second Approval in the relationship. *)
  related_approval : approval;
```

```
  (* The relating_approval specifies the first Approval in the relationship. *)
  relating_approval : approval;
  (* The relation_type specifies the nature of the Approval_relationship. *)
  relation_type : label;
END_ENTITY;

(* An Assessment is an evaluation of the status of an element of a systems engineering specification during its development. The semantics of the estimation is given by the assessment_type attribute. *)
ENTITY assessment;
  (* The assessed_level specifies the assessed value. *)
  assessed_level : label;
  (* The assessment_type specifies the type and purpose of estimation performed. Where applicable the following values shall be used:<ul><li>maturity: The Assessment is an estimation of the maturity of the element;</li><li>completeness: The Assessment is an estimation of the level of completeness of the assigned element;</li><li>risk: The Assessment is an estimation of the project risk associated with the assigned element;</li> <li>stability: The Assessment is an estimation of the stability of the assigned element.</ul> *)
  assessment_type : label;
  (* The assigned_to specifies the system engineering specification element for which the Assessment is valid. *)
  assigned_to : assessment_assignment_select;
  (* The description specifies additional information about the Assessment. *)
  description : OPTIONAL text_select;
END_ENTITY;

(* An Assessment_relationship is a relationship between two Assessment objects. The semantics of the relationship are defined by the relationship_type attribute. *)
ENTITY assessment_relationship;
  (* The description specifies additional information about the Assessment_relationship. *)
  description : OPTIONAL text_select;
  (* The related specifies the second of the two Assessment objects involved in the relationship. *)
  related : assessment;
  (* The relating specifies the first of the two Assessment objects involved in the relationship. *)
  relating : assessment;
  (* The relationship_type specifies the semantics of the Assessment_relationship. Where applicable one of the following values shall be used: <ul> <li>replacement: The relating Assessment is replacing the related Assessment;</li><li>-conflict: The relating Assessment identified as being in conflict with the related Assessment;</li> <li>complement: The relating Assessment identified as complementary to the related Assessment.</li></ul> *)
  relationship_type : label;
END_ENTITY;

(* A Bi_directional_port_indicator is a mechanism for relating two Io_port objects that consume and produce the same data_instance object. <note number="1">The Bi_directional_port_indicator is the mechanism for representing bi-directional ports using pairs of Io_port objects. The consuming_port and the producing_port shall identify the same Data_instance via the data attribute. Moreover, the Io_port defined by the producing_port attribute shall produce Data_instance objects, and the Io_port defined by the consuming_port attribute shall produce Data_instance objects.</note> <note number="2">The port_of attribute of both Io_port objects in the Bi_directional_port_indicator must connect to the same functional object, that is either a General_functionality_instance or a General_function_definition.</note> *)
ENTITY bi_directional_port_indicator;
  (* The consuming_port specifies the Io_port in the Bi_directional_port_indicator that identifies the port that consumes the data. *)
  consuming_port : io_port;
  (* The producing_port specifies the Io_port in the Bi_directional_port_indicator that identifies the port that produces the data. *)
  producing_port : io_port;
END_ENTITY;

(* A Binary_data_type_definition is a type of Elementary_maths_space that contains the values from 0 to 2n -1, where n is the size (or number of bits) of the data type. <note>It is often the case with data bus signals that many pieces of information are packed into a single word.</note><example number="2">An eight bit word is subdivided to carry four pieces of information about a pump.<ul><li>bits 1-3 identify the pump (system contains 8 pumps numbered 0 - 7)</li><li>bits 4-6 carry the status of the pump (pump can be in one of 8 states numbered 0 - 7).</li><li>bit 7 indicates the direction the pump is pumping in (0 - forewards, 1 - backwards),</li><li>bit 8 indicates whether the pump is switched on or off (0 - off,
```

1 - on).
</example>A group of bits is not an aggregate. The bits form a single value and they have a position within a word. Typically the producer and receiver of the data use bit masks to manipulate parts of the word data. *)
ENTITY binary_data_type_definition
SUBTYPE OF (elementary_maths_space);
 (* The size specifies a finite interval that defines the maximum length of a Binary_data_type_definition. *)
 size : finite_integer_interval;
END_ENTITY;

(* A Boolean_data_type_definition is a type of Elementary_maths_space whose value range is TRUE or FALSE. *)
ENTITY boolean_data_type_definition
SUBTYPE OF (elementary_maths_space);
END_ENTITY;

(* A Cardinality_list is a definition of a valid cardinality range. *)
ENTITY cardinality_list;
 (* The range specifies a strictly increasing non overlapping list of Cardinality_range objects that specifies valid intervals for the Cardinality_list. *)
 range : LIST [2 : ?] OF cardinality_range;
END_ENTITY;

(* A Cardinality_range is a pair of values defining the valid lower and upper bounds of an interval. The value of the lower_bound shall be less than or equal to the higher_bound value. The value domain of the attributes is that of the natural numbers. *)
ENTITY cardinality_range;
 (* The lower_bound specifies the lower bound in the interval. *)
 lower_bound : single_cardinality;
 (* The upper_bound specifies the upper bound in the interval. *)
 upper_bound : single_cardinality;
END_ENTITY;

(* A Causal_block_bound is the relationship between two Cb_transition_relationship objects indicating the scope of a branch in a Cb_functional_behaviour_model.<note>This entity is an extension to the Petri-nets formalism and is required for correct representation of parallel and conditional branches in a causal chain model.</note> *)
ENTITY causal_block_bound;
 (* The initial_transition specifies the initial Cb_transition_relationship of the block. *)
 initial_transition : cb_transition_relationship;
 (* The terminal_transition specifies final Cb_transition_relationship of the block. *)
 terminal_transition : cb_transition_relationship;
END_ENTITY;

(* A cb_completion_alternative is the representation of the final Cb_functional_place for an execution thread in a Cb_functional_behaviour_model. The final Cb_functional_place in the behaviour model is identified by the final model element attribute. In cases where there are two or more final Cb_functional_place objects (where each is the final Cb_functional_place object of an execution thread) there shall be one cb_completion_alternative instantiated for each thread.<note>The cb_completion_alternative has been introduced to allow mappping of multiple exit conditions from a decomposed function to the appropriate exit conditions in the composition.</note> *)
ENTITY cb_completion_alternative;
 (* The completes_model specifies the Cb_functional_behaviour_model for which the completes_model specifies the exit statement. *)
 completes_model : cb_functional_behaviour_model;
 (* The final_model_element specifies the Cb_functional_place which is the last in an execution thread for a Cb_functional_behaviour_model. *)
 final_model_element : cb_functional_place;
END_ENTITY;

(* A Cb_completion_alternative_mapping is the mechanism for associating a Cb_completion_alternative of a decomposed function with the equivalent Cb_functional_place at a higher level of completion. <note>A Cb_completion_alternative_mapping specifies the mapping between a Cb_completion_alternative of a Cb_functional_behaviour_model and a Cb_functional_transition. The semantics are that the selection of a completion alternative in a composed function shall be interpreted as the selection of the Cb_functional_transition at one level higher in the composition.</note> *)
ENTITY cb_completion_alternative_mapping;

```
  (* The child_completion_criterion specifies the Cb_completion_alternative in the Cb_completion_alternative_mapping. *)
  child_completion_criterion : cb_completion_alternative;
  (* The equivalent_transition is the Cb_functional_transition to which the child_completion_criterion is mapped to by the
Cb_completion_alternative_mapping. *)
  equivalent_transition : cb_functional_transition;
  (* The scope specifies the Cb_place_function_association of the Cb_completion_alternative_mapping. *)
  scope : cb_place_function_association;
END_ENTITY;

(* A Cb_firing_condition is a specification of a condition that must be fulfilled to allow firing a transition in a Cb_functional_
behaviour_model. <note>A cb_firing_condition is associated with the Cb_output_relationship preceding the Cb_transition
object that shall be fired when the condition is fulfilled.</note> *)
ENTITY cb_firing_condition;
  (* The condition_definition specifies the condition that shall be fulfilled for firing a transition. *)
  condition_definition : textual_specification;
  (* The guarded_transition specifies the Cb_output_relationship to which the Cb_transition applies. *)
  guarded_transition : cb_transition;
END_ENTITY;

(* A Cb_functional_behaviour_model is a type of Functional_behaviour_model and a behaviour model that defines partial
function ordering, such as function sequence, concurrency or branching among functions.<note number="1">When
assigned to a Composite_function_definition, functional behaviour is defined by the functional interaction model of the
Composite_function_definition and the cb_functional_behaviour_model in combination.</note> <note number="2">The
building blocks of a cb_functional_behaviour_model equal that of a Petri-nets. It is built from Cb_place, Cb_transition,
Cb_input_relationship and Cb_output_relationship.</note> *)
ENTITY cb_functional_behaviour_model
SUBTYPE OF (functional_behaviour_model);
  (* The model_boundedness specifies the maximum number of tokens allowed per Cb_place in the Cb_functional_be-
haviour_model. <note number="3">For causal formalisms popular among systems engineers - FFDB and Behaviour
diagrams - model boundedness is 1, except if the replication concept is used in a model, in this case the model_bound-
edness is the maximum number of threads in the replicated structure.</note> *)
  model_boundedness : label;
  (* The model_type specifies the type of behaviour model that is realized by the Cb_functional_behaviour_model and the
Composite_function_definition in combination. *)
  model_type : label;
INVERSE
  (* The behaviour_model specifies the Cb_functional_behaviour_model of which the Cb_functional_place is a part. *)
  constituent_places : SET [1 : ?] OF cb_functional_place FOR behaviour_model;
  (* The behaviour_model specifies the Cb_functional_behaviour_model of which the Cb_functional_transition is a part. *)
  constituent_transitions : SET [2 : ?] OF cb_functional_transition FOR behaviour_model;
END_ENTITY;

(* A Cb_functional_place is a type of Cb_place specialized to capture static states of a functional_behaviour_model.
<note><p>A causal behaviour specification is built by connecting Cb_functional_place and Cb_functional_transition
objects using Cb_input_relationship and Cb_output_relationship objects. If more than one Cb_output_relationship
objects are connected to a Cb_functional_transition this means that this is the first element in a parallel branch in the
behaviour specification.</p><p>If more than one Cb_input_relationship object is connected to a Cb_functional_transi-
tion, this means that this is the termination of a parallel branch. Alternative threads can be specified by assigning several
Cb_output_relationship objects to a single cb_functional_place. Similarly, termination of alternate threads is specified by
connecting several Cb_input_relationship objects to a Cb_functional_place object.</p></note> *)
ENTITY cb_functional_place
SUBTYPE OF (cb_place);
  (* The behaviour_model specifies the Cb_functional_behaviour_model of which the Cb_functional_place is a part. *)
  behaviour_model : cb_functional_behaviour_model;
INVERSE
  (* The functional_place_reference specifies the Cb_functional_place  for which the Cb_place_reference is defined. *)
  reference_information : SET [0 : 1] OF cb_place_reference FOR functional_place_reference;
END_ENTITY;

(* A Cb_functional_transition  a type of Cb_transition specialized to define a transient node in a Cb_functional_be-
haviour_model. <note><p>A causal behaviour specification is built by connecting Cb_functional_place and Cb_functional_
```

transition objects using Cb_input_relationship and Cb_output_relationship objects. If more than one Cb_output_relationship objects are connected to a Cb_functional_transition, this means that this is the first element in a parallel branch in the behaviour specification. If more than one Cb_input_relationship object is connected to a Cb_functional_transition, this means that this is the termination of a parallel branch. Alternative threads can be specified by assigning several Cb_output_relationship objects to a single Cb_functional_place. Similarly, termination of alternate threads is specified by connecting several Cb_input_relationship objects to a Cb_place. </note> *)
ENTITY cb_functional_transition
SUBTYPE OF (cb_transition);
(* The behaviour_model specifies the Cb_functional_behaviour_model of which the Cb_functional_transition is a part. *)
behaviour_model : cb_functional_behaviour_model;
INVERSE
(* The related_transition specifies the set of Cb_transition objects that is involved in the Cb_functional_transition. *)
transition_relationship : SET [0 : 1] OF cb_transition_relationship FOR related_transition;
END_ENTITY;

(* A Cb_initial_marking is an indication that a Cb_place in a Cb_functional_behaviour_model is part of the initial condition of the behaviour model. *)
ENTITY cb_initial_marking;
(* The marked_place specifies the Cb_place that is part of the initial condition of the Cb_functional_behaviour_model. *)
marked_place : cb_place;
(* The number_of_tokens specifies the number of tokens initially assigned to the marked_place.
*)
number_of_tokens : INTEGER;
END_ENTITY;

(* A Cb_input_relationship is the unidirectional relationship from a Cb_place to a Cb_transition indicating a causality constraint between the Cb_place and the Cb_transition. <note number="1">The causal_weight attribute specifies the number of tokens required to fire the transition.</note> <note number="2">In established Petri-nets terminology, the Cb_input_relationship corresponds to an input arc.</note> *)
ENTITY cb_input_relationship;
(* The causal_weight specifies the number of tokens in the Cb_place that are required to fire the transition. Causal_weight is a natural number.
*)
causal_weight : causal_weight_select;
(* The destination_transition specifies the Cb_transition in the relationship.
*)
destination_transition : cb_transition;
(* The source_place specifies the Cb_place in the relationship.
*)
source_place : cb_place;
END_ENTITY;

(* An Cb_output_relationship is the unidirectional relationship from a Cb_transition to a Cb_place indicating a causality constraint between the Cb_transition and the Cb_place. <note number="1">In established Petri-nets terminology the cb_output_relationship corresponds to an output arc. If the transition is fired the causal_weight attribute defines how many tokens will be generated in the destination place.</note> *)
ENTITY cb_output_relationship;
(* The causal_weight specifies the number of tokens that is generated in the destination_place.
<note
number="2">
For all non-Petri-nets applications the causal_weight attribute shall be set to 1. The same holds for safe Petri-nets.
</note> *)
causal_weight : causal_weight_select;
(* The destination_place specifies the Cb_place that is entered as a consequence of firing the Cb_transition defined by the source_transition.
*)
destination_place : cb_place;
(* The source_transition specifies the Cb_transition from which the source_transition originates.
*)
source_transition : cb_transition;
END_ENTITY;

(* A Cb_place is the definition of a static state in a causal functional behaviour model.<note>In Petri-nets terminology a cb_place is a place.</note> *)
ENTITY cb_place
ABSTRACT SUPERTYPE OF (ONEOF(cb_functional_place, oo_action_state));
 (* The description specifies additional information about the Cb_place. *)
 description : OPTIONAL text_select;
 (* The place_label specifies the word, or words, which are used to refer to the Cb_place. *)
 place_label : OPTIONAL label;
INVERSE
 (* The marked_place specifies the Cb_place that is part of the initial condition of the Cb_functional_behaviour_model. *)
 initial_marking : SET [0 : 1] OF cb_initial_marking FOR marked_place;
END_ENTITY;

(* A Cb_place_function_association is the association of a Function_instance with a Cb_place such that the function activation is controlled by the Cb_place. <note>There need not be a one to one correspondence between Cb_place and Function_instance objects. A Cb_place object with no Function_instance object assigned indicate a placeholder for thread synchronisation.</note> *)
ENTITY cb_place_function_association;
 (* The causal_place specifies the Cb_place in the relationship.
*)
 causal_place : cb_functional_place;
 (* The controls_function specifies the Function_instance in the relationship.
*)
 controls_function : function_instance;
INVERSE
 (* The scope specifies the Cb_place_function_association of the Cb_completion_alternative_mapping. *)
 completion_mapping : SET [0 : ?] OF cb_completion_alternative_mapping FOR scope;
END_ENTITY;

(* A Cb_place_reference is the mechanism for associating additional information with a Cb_place. The semantics are further defined by the reference_type attribute. <note number="1">The Cb_place_reference is the mechanism for indicating that a particular Cb_place shall be annotated with a specific syntax symbol. It is primarily inserted to support FFBD and Behaviour diagram concepts.</note> *)
ENTITY cb_place_reference;
 (* The functional_place_reference specifies the Cb_functional_place for which the Cb_place_reference is defined. *)
 functional_place_reference : cb_functional_place;
 (* The reference_type specifies the sematics of the reference_type. Where applicable one of the following values shall be used:<note number="2">The preferred values listed are included to allow for a correct representation of FFBD and Behaviour diagrams. The mani purpose is to indicate the exits of loops or the terminiation of an execution thread.</note>
thread_terminiation: The Cb_place indicated by the functional_place_reference attribute is the final element in an execution thread;
loop_exit: The Cb_place indicated by the functional_place_reference attribute is the final element in a loop. *)
 reference_type : label;
END_ENTITY;

(* A Cb_transition is a transient node in a causal behaviour model.<note>In Petri-net terminology, a Cb_transition is a transition.</note> *)
ENTITY cb_transition
ABSTRACT SUPERTYPE OF (ONEOF(cb_functional_transition, oo_action_state_transition));
 (* The description specifies additional textual information for the Cb_transition. *)
 description : OPTIONAL text_select;
 (* The transition_label specifies the word, or words, that are used to refer to the Cb_transition. *)
 transition_label : OPTIONAL label;
INVERSE
 (* The guarded_transition specifies the Cb_output_relationship to which the Cb_transition applies. *)
 guarded_by : SET [0 : 1] OF cb_firing_condition FOR guarded_transition;
END_ENTITY;

(* A Cb_transition_relationship is the relationship between two Cb_transition objects. <note number="1">The intention with this entity is to group selective branches in a causal chain. The semantics can also be captured by extending the entity Causal_block_bound slightly.</note> <note number="2">This entity shall probably be removed from the model.</note> *)

```
ENTITY cb_transition_relationship;
  (* The related_transition specifies the set of Cb_transition objects that is involved in the Cb_functional_transition. *)
  related_transition : SET [1 : ?] OF cb_functional_transition;
  (* The relationship_type specifies the nature of the relationship. Where applicable the relationship_type shall be one of
the following:<ul> <li>or: The Cb_transition_relationship represents a set of Cb_functional_transition (see ISO/WD PAS
20542 clause 4.2.25) objects that indicate the start or an end of a selective structure.</li> </ul> <note number="3">This
corresponds to an IF .. THEN .. ELSE statement in programming languages</note> <ul> <li>and: The Cb_transition_re-
lationship represents a set of Cb_functional_transition objects that indicate the start or an end of a concurrent structure;</
li> <li>loop: The Cb_transition_relationship represents a set of Cb_functional_transition objects that indicate the start
or an end of a fixed loop structure.</li> </ul> <note number="4">For the loop keyword it is assumed that the number of
iterations in the loop is hardcoded.</note> <ul><li>iteration: The Cb_transition_relationship represents a set of Cb_func-
tional_transition objects that indicate the start or an end of a iterative structure where the exit condition is evaluated
dynamically;</li><li>replication: The Cb_transition_relationship represents a set of Cb_functional_transition objects that
indicate the start or an end of a replicated structure.</li></ul><note number="5">Replication is the representation of a
multiple functional execution threads using a single functional structure.</note> *)
  relationship_type : label;
INVERSE
  (* The terminal_transition specifies final Cb_transition_relationship of the block. *)
  end_bound : SET [0 : 1] OF causal_block_bound FOR terminal_transition;
  (* The initial_transition specifies the initial Cb_transition_relationship of the block. *)
  start_bound : SET [0 : 1] OF causal_block_bound FOR initial_transition;
END_ENTITY;

(* A Cb_transition_unbounded_weight is the mechanism to indicate that the boundedness of a Cb_input_relationship
or Cb_output_relationship is defined at run-time. The boundedness defined by Cb_transition_unbounded_weight is a
natural number larger or equal to the value defined by the minimal_weight attribute. <note>The boundedness defines
the number of tokens generated by a Cb_input_relationship or the number of tokens consumed by a Cb_output_relation-
ship.</note> *)
ENTITY cb_transition_unbounded_weight;
  (* The minimal_weight specifies the least number of tokens produced or consumed. *)
  minimal_weight : natural_number;
END_ENTITY;

(* A Change_order is a type of Work_order that authorizes the implementation of one or more changes. <note>A
Change_order is normally preceeded by a Change_request object.</note> *)
ENTITY change_order
SUBTYPE OF (work_order);
  (* The change_element specifies the elements that are subject to change as a result of the Change_order. *)
  change_element : element_critical_issue_relationship;
  (* The originates_from specifies the change_request that provided the motivation for the Change_order. *)
  originates_from : change_request;
END_ENTITY;

(* A Change_order_relationship is a relationship between two Change_order objects. The semantics of the relationship
are defined by the Change_order_relationship_type attribute. *)
ENTITY change_order_relationship;
  (* The change_order_relationship_type specifies the semantics of the change_order_relationship_type. Where appli-
cable one of the following values shall be used:<ul> <li>temporal_order: the relating Change_order shall be completed
before the implementation of the related Change_order;</li> <li>alternative: the relating Change_order is an alternative
to related Change_order. Only one of the two identified Change_order objects shall be effectuated.</li> </ul> *)
  change_order_relationship_type : label;
  (* The related specifies the second change_order in the relationship. *)
  related : change_order;
  (* The related specifies the first change_order in the relationship. *)
  relating : change_order;
END_ENTITY;

(* A Change_report is a collection of elements reporting on changes performed as a result of a change_request.
<note>The report of what has been changed can either described in text in the description attribute or by assigning other
specification elements to the Change_report by the use of Change_report_element_assignment objects.</note> *)
ENTITY change_report;
```

```
  (* The associated_version specifies the Configuration_element_version for the Change_report. *)
  associated_version : configuration_element_version;
  (* The description specifies additional information on the Change_report. *)
  description : OPTIONAL text_select;
  (* The name specifies the word, or words, that are used to refer to the Change_report. *)
  name : label;
  (* The originating_change_request specifies the Change_order that led to the creation of the Change_report. *)
  originating_change_request : change_request;
END_ENTITY;

(* A Change_report_element_assignment is the mechanism for assigning descriptions of changes implemented to a
Change_report. *)
ENTITY change_report_element_assignment;
  (* The change_report specifies the Change_report involved in the Change_report_element_assignment. *)
  change_report : change_report;
  (* The change_report_element specifies an element that is part of a Change_report. *)
  change_report_element : change_report_element_select;
  (* The description specifies additional textual information on the Change_report_element_assignment. *)
  description : OPTIONAL text_select;
END_ENTITY;

(* A Change_request is a type of Work_request that solicits the implementation of a change. <note>The elements affect-
ed by the Change_request are identified by the attribute scope inherited from the Work_order.</note> *)
ENTITY change_request
SUBTYPE OF (work_request);
  (* The response_to_issue specifies the critical_issue that led to the creation of the Change_request. *)
  response_to_issue : SET [1 : ?] OF critical_issue;
END_ENTITY;

(* A Clock is a device that emits a control signal at regular intervals. <note number="1">A clock can define the synchro-
nous behaviour of one or more Function_instance objects.</note> <note number="2">The subclass of systems engineer-
ing tools supporting synchronous behaviour use different methods to represent the clock that provides the synchronous
activations. Some represent the clock explicitly as a kind of function, some represent it as a property of the function that
is controlled by the clock. The approach taken in PAS 20542 supports both variants in the sense that the Clock can be
presented as a function-like block or be a viewed as a property of the function it controls.</note> *)
ENTITY clock;
  (* The control_signal specifies the Data_instance (whose definition shall be of type Event_data_type_definition) that the
Clock produces at periodic intervals (given by the frequency attribute). *)
  control_signal : data_instance;
  (* The description specifies additional information about the Clock. *)
  description : OPTIONAL text_select;
  (* The frequency specifies the number of pulses emitted during a given time period. *)
  frequency : REAL;
  (* The name specifies the word, or words, that are used to refer to the Clock. *)
  name : OPTIONAL label;
WHERE
  correct_data_definition:
'SYSTEMS_ENGINEERING_DATA_REPRESENTATION.EVENT_DATA_TYPE_DEFINITION' IN TYPEOF(control_sig-
nal.definition);
END_ENTITY;

(* A Clock_assignment_relationship is a relationship between a Clock and a Control_io_port such that the periodic signal
from the Clock is conveyed to the Control_io_port. <note>The Clock_assignment_relationship allows the use of a single
Clock to control several function objects (via the corresponding Control_io_port objects).</note> *)
ENTITY clock_assignment_relationship;
  (* The clock specifies the Clock object in the relationship.
 *)
  clock : clock;
  (* The trigger_for specifies the Control_io_port object in the relationship.
 *)
  trigger_for : control_io_port;
END_ENTITY;
```

```
(* A clock_reference_context_relationship is the mechanism for relating a clock to the functionality_instance_reference
that defines the context of the clock. Through the usage of clock_reference_context_relationship it is possible to define
that a clock object is valid for specific functional_reference_configuration objects only.
 *)
ENTITY clock_reference_context_relationship;
  (* The clock_signal specifies the Clock in the relationship. *)
  clock_signal : clock;
  (* The relevant_functionality_context specifies the Functionality_instance_reference that the Clock is associated with
through the Clock_assignment_relationship. *)
  relevant_functionality_context : functionality_instance_reference;
END_ENTITY;

(* A Complex_data_type_definition is a type of Elementary_maths_space that includes all complex values. *)
ENTITY complex_data_type_definition
SUBTYPE OF (elementary_maths_space);
END_ENTITY;

(* A Complex_value is a means of specifying a complex value. <note>The model uses polar co-ordinates to identify the
complex point, rather than cartesian co-ordinates.</note> *)
ENTITY complex_value;
  (* The radius specifies the distance from the origin (0,0) to the complex point. *)
  radius : REAL;
  (* The theta specifies the angular component of a Complex_value. *)
  theta : REAL;
END_ENTITY;

(* A Composite_function_definition is a type of General_function_definition and represents a function definition that is
decomposed further. It is composed of at least one General_functionality_instance object. *)
ENTITY composite_function_definition
SUBTYPE OF (general_function_definition);
INVERSE
  (* The constrained_function specifies the Composite_function_definition object whose behaviour is constrained by the
assigned Functional_behaviour_model. *)
  behaviour_constraint : SET [0 : 1] OF functional_behaviour_model_assignment FOR constrained_function;
  (* The parent specifies the Composite_function_definition of which the child is a part of the decomposition. *)
  parent_of : SET [1 : ?] OF functional_decomposition_relationship FOR parent;
END_ENTITY;

(* A Compound_value is a value that consists of a list of other values. The list may contain values of different types and
other lists.  If there are fewer entries in the target aggregate than there are in the list then the extra values are not used. If
there are more entries in the target aggregate than there are in the list then the list is applied repeatedly until the aggre-
gate is full. *)
ENTITY compound_value;
  (* The value_list specifies the list of values contained within the Compound_value. *)
  value_list : LIST [0 : ?] OF data_type_value_select;
END_ENTITY;

(* A Configuration_element is either a single object or a Unit in a group of objects. It collects the information that is com-
mon to all versions of the object. <note>Effectively a configuration_element is the logical container for all versions of the
same thing. For example three revisions of the same functional model could be collected together by a configuration_ele-
ment with a name and description of the functional model.</note> *)
ENTITY configuration_element;
  (* The description specifies additional textual information on the Configuration_element. *)
  description : OPTIONAL text_select;
  (* The id specifies the identifier of the Configuration_element. *)
  id : element_identifier;
  (* The name specifies the word, or words, which are used to refer to the Configuration_element. *)
  name : label;
INVERSE
  (* The version_of specifies the Configuration_element object for which the Configuration_element represents a version.
*)
```

```
  associated_version : SET [1 : ?] OF configuration_element_version FOR version_of;
UNIQUE
  UR1: id;
END_ENTITY;

(* A Configuration_element_relationship is the mechanism for relating two Configuration_element objects. The nature of
the relationship is defined by the relationship_type attribute. *)
ENTITY configuration_element_relationship;
  (* The alternate specifies the second configuration_element in the alternate.
 *)
  alternate : configuration_element;
  (* The base specifies the first configuration_element in the base.
 *)
  base : configuration_element;
  (* The description specifies additional information on the description.
 *)
  description : OPTIONAL text_select;
  (* The relationship_type specifies the semantics of the Configuration_element_relationship. Where applicable one
of the following values shall be used: <ul> <li>alternative: the base and alternate Configuration_element objects are
interchangeable;</li> <li>variant: the alternate is a variant of the base Configuration_element.<li> </ul> <note>A variant
implies that the base and alternative Configuration_element objects share substantial amount of information.</note> *)
  relationship_type : label;
END_ENTITY;

(* A Configuration_element_version is a snapshot of any design item for which a change history is maintained. <note
number="1">A change history is kept from an initial item release and is recorded by configuration_element_version.
Therefore, the picture is composed of a Configuration_element and a set of Configuration_element_version objects that
record the evolution of the given Configuration_element. In this PAS, design elements can come from the requirement
UoF, function UoF, behaviour UoF, documentation and data type.<p>This then leads to a situation where a design is
made up of Configuration_element instances referring to a particular version of the Configuration_element represented.
Therefore, a design can be made up of a set of Configuration_element standing in different version. Then a new version
of the whole design can be made of a single change in the design.</p></note><example number="3">For instance,
design version 1 may comprise version 1 of function 1, version 3 of function 2, and version1 of function 3. Then the next
design release (version 2) may comprise version 2 of function 1, version 3 of function 2, and version 1 of function 3. This
example illustrates that there may be 2 levels of version management. One party might wish to manage the whole design
version. Another party might wish to manage versions of single items.</example><note number="2">The standard
assumes that consistency is maintained by either the sending tool, the receiving tool or a third party tool more focussed
on the configuration management items.</note> <note number="3">The set of configuration_element_version objects of
a Configuration_element represents the history of a Configuration_element within a particular life-cycle stage or possibly
over the whole life-cycle.</note> *)
ENTITY configuration_element_version;
  (* The description specifies additional textual information on the Configuration_element_version. *)
  description : OPTIONAL text_select;
  (* The id specifies the identifier of the Configuration_element_version. *)
  id : element_identifier;
  (* The version_of specifies the Configuration_element object for which the Configuration_element represents a version.
*)
  version_of : configuration_element;
UNIQUE
  UR1: id;
END_ENTITY;

(* A Configuration_element_version_relationship is the relationship between two Configuration_element_version objects
establishing the version graph for the objects. *)
ENTITY configuration_element_version_relationship;
  (* The description specifies the nature of the change between the related_version and the relating_version. *)
  description : OPTIONAL text_select;
  (* The related_version specifies the resulting Configuration_element_version object related by the Configuration_ele-
ment_version_relationship. *)
  related_version : configuration_element_version;
  (* The related_version specifies the initial Configuration_element_version related by the Configuration_element_ver-
sion_relationship. *)
```

```
  relating_version : configuration_element_version;
  (* The relationship_type specifies the semantics of the configuration_element_version_relationship. Where applicable,
one of the following values shall be used:<il><li>revision;</li> <li>workspace_revision;</li><li>alternative.</li></ul> *)
  relationship_type : label;
WHERE
  WR1: related_version :<>: relating_version;
END_ENTITY;

(* A Context_function_relationship is the relationship between a Function_instance object and a System_view object
such that the Function_instance object is part of the functional description of the System_view. <note number="1">The
Function_instance may describe functional aspects of the System_view or functional aspects of the environment to the
System_view. In the first case the role attribute shall be set to "system_function", in the second case the role attribute
shall be set to "external_element".</note> *)
ENTITY context_function_relationship;
  (* The associated_context specifies the System_view that is valid for the relationship. *)
  associated_context : system_view;
  (* The Context_function specifies the Function_instance object that is valid for the relationship. *)
  context_function : function_instance;
  (* The description specifies additional textual information on the relationship. *)
  description : OPTIONAL text_select;
  (* The role specifies what the Function_instance describes in the context of a particular System_view. <note num-
ber="2">There is a constraint that there shall only be exactly one function_instance assigned to a System_view as a
"system_function".</note> *)
  role : function_role_enumeration;
INVERSE
  (* The system specifies the context_function_relationship to which the functional_reference_configuration is assigned.
 *)
  assigned_functional_configuration : SET [0 : 1] OF system_functional_configuration FOR system;
UNIQUE
  UR1: associated_context, context_function;
END_ENTITY;

(* A Context_physical_relationship is the relationship between a Physical_instance object and a System_view object
such that the Physical_instance object is part of the top-level physical architecture description of the System_view.
<note>The Physical_instance may describe physical aspects of the System_view or physical aspects of the environment
to the System_view. In the first case the role attribute shall be set to "system_element", in the second case the role attri-
bute shall be set to "external_element".</note> *)
ENTITY context_physical_relationship;
  (* The description specifies additional textual information on the relationship. *)
  description : OPTIONAL text_select;
  (* The part_in_physical_context specifies the System_view object for which the relationship is relevant. *)
  part_in_physical_context : system_view;
  (* The physical_instance specifies the Physical_instance for which the relationship is valid. *)
  physical_instance : physical_instance;
  (* The role specifies the nature of the Physical_instance in the System_view. *)
  role : physical_element_role_enumeration;
INVERSE
  (* The system specifies the Context_physical_relationship to which the Physical_reference_configuration is assigned. *)
  assigned_physical_configuration : SET [0 : 1] OF system_physical_configuration FOR system;
END_ENTITY;

(* A Control_io_port is a type of Io_port and an element in the behavioural interface to a Function_instance object. A Con-
trol_io_port is always an input port and consumes data from Functional_link objects. <note number="1">The semantics
is such that the Control_io_port is enabled or disabled by the value of the data associated with the Control_io_port. If the
Control_io_port is enabled, it may activate the Function_instance.</note> <note number="2">If the data_instance defined
by the data attribute is of type event_data_type_definition then the Control_io_port is enabled by the arrival of an event
in the data_instance.</note> <note number="3">If the data_instance defined by the data attribute is of type logical_data_
type_definition then the Control_io_port is enabled when the value of the data_instance is TRUE. Otherwise the control_
io_port is disabled.</note> <note number="4">If more than one control_io_port is attached to a Function_instance object,
then all must be enabled to activate the Function_instance object.</note> *)
ENTITY control_io_port
```

```
SUBTYPE OF (io_port);
  (* The control_type specifies the role of the Control_io_port. *)
  control_type : control_type_enumeration;
  (* The offset specifies the time delay from reception of the control signal until the Control_io_port is enabled. *)
  offset : REAL;
  (* The port_of specifies the Function_instance object to which the Control_io_port is attached. *)
  port_of : function_instance;
  (* The trigger_type specifies how the Control_io_port is enabled. <note number="5">The exact semantics for triggering
are determined by the data type associated with the control port.</note> *)
  trigger_type : trigger_type_enumeration;
DERIVE
  SELF\io_port.role : port_data_relation := consumer;
UNIQUE
  UR1: port_of, SELF\io_port.io_port_number, SELF\io_port.port_type;
WHERE
  good_offset: offset >= 0.0;
  port_data_direction: (SELF\io_port.port_type <> output);
  WR1: ('SYSTEMS_ENGINEERING_DATA_REPRESENTATION.EVENT_DATA_TYPE_DEFINITION' IN TYPEOF(da-
ta.definition)) OR ('SYSTEMS_ENGINEERING_DATA_REPRESENTATION.LOGICAL_DATA_TYPE_DEFINITION' IN
TYPEOF(data.definition));
END_ENTITY;

(* A Coordinate_translation_information is the mechanism for mapping co-ordinate information from a normalized co-or-
dinate system to the actual presentation co-ordinate system used in a tool. The Coordinate_translation_information  is
assigned as modifier to provide actual presentation information for Graphics_view objects.<p>The size of a view and
the position of individual items in the view is calculated according to the following formula:</p> <ul> <li>y-position:=nor-
malized_x-position*scale_factor;</li> <li>x-position:= normalized_x-position*aspect_ration*scale_factor;</li> </ul>
<note>PAS 20542 uses a normalized co-ordinate system where all position information is given in the range {0..1} for the
x and y co-ordinates. The co-ordinate {0,0} specifies the top-left of a view and the co-ordinate {1,1} specifies the bot-
tom-right of a view.</note> *)
ENTITY coordinate_translation_information;
  (* The measurement_unit specifies the unit of measure that shall be used to decode the size of a view defined by a
Coordinate_translation_information. *)
  measurement_unit : label;
  (* The name specifies the word or words that are used to refer to a  Coordinate_translation_information. *)
  name : label;
  (* The ratio specifies the relation between the x and y axes in a co-ordinate system defined by a  Coordinate_transla-
tion_information. Values larger than 1 indicates that the extension of a view is greater along the x-axis than along the
y-axis. *)
  ratio : REAL;
  (* The scale_factor specifies the multiplier that defines the size of a view. *)
  scale_factor : REAL;
  (* The transformation_for specifies graphics_view for which the  Coordinate_translation_information provides informa-
tion. *)
  transformation_for : graphics_view;
END_ENTITY;

(* A critical_issue is an identification of a percieved or real problem concerning an element of a system specifica-
tion.<note>The issue identified by a Critical_issue need not acknowledged by all stakeholders in a system development
process. It is mere an indentification of a potential problem or issue.</note> *)
ENTITY critical_issue;
  (* The description specifies additional information on the closed: No further actions will be taken to address the Criti-
cal_issue. *)
  description : OPTIONAL text_select;
  (* The id specifies the identifier of the id. *)
  id : element_identifier;
  (* The name specifies the word or words that are used to refer to a Critical_issue. *)
  name : label;
  (* The status specifies the condition of a Critical_issue. Where applicable one of the following values shall be used: <ul>
<li>open: The validity of the Critical_issue has not yet been evaluated;</li><li>acknowledged: The validity of the Critical_
issue has been confirmed;</li><li>closed: No further actions will be taken to address the Critical_issue.</li> *)
```

```
  status : label;
END_ENTITY;

(* A Critical_issue_impact is the mechanism for relating the elements identified to be influenced by a Critical_issue to the
issue itself. <note>Any number of elements can be assigned to a critical_issue_impact. The critical_issue_impact is a
representation of all elements assigned collectively.</note> *)
ENTITY critical_issue_impact;
  (* The description specifies additional information on the Critical_issue_impact. *)
  description : OPTIONAL text_select;
  (* The id specifies the identifier of the Critical_issue_impact. *)
  id : element_identifier;
  (* The impact_of_issue specifies the set of Critical_issue objects that the Critical_issue_impact is valid for. *)
  impact_of_issue : SET [1 : ?] OF critical_issue;
  (* The name specifies the word or words that are used to refer to the Critical_issue_impact. *)
  name : label;
INVERSE
  (* The issue_analysis specifies the Critical_issue_impact for which the element_critical_issue_relationship is valid. *)
  identified_issues : SET [0 : ?] OF element_critical_issue_relationship FOR issue_analysis;
END_ENTITY;

(* A Data_field is a type of Data_instance that and the usage of a data type within a Record_data_type_definition or a
Union_data_type_definition. *)
ENTITY data_field
SUBTYPE OF (data_instance);
  (* The role specifies additional information on the Data_field. *)
  role : OPTIONAL label;
END_ENTITY;

(* A Data_instance is an instantiation of a User_defined_data_type_definition or Maths_space. <note number="1">In the
terminology used in computer science, a Data_instance corresponds to a variable. In the systems engineering domain,
the Data_instance is interpreted in a wider sense.</note> *)
ENTITY data_instance
SUPERTYPE OF (data_field);
  (* The default_value specifies the value to be assigned to the Data_instance when an invalid value is provided, for in-
stance, when the value is out of the bounded range, or of the wrong type.<note number="2">The event that valid values
are absent for a default_value may be due to the fact that the element supplying the value is out of order.</note> <note
number="3">The value of the default_value shall be within the valid value range if one is defined.</note> *)
  default_value : OPTIONAL data_type_value_select;
  (* The definition specifies the user_defined_data_type_definition or maths_space that provides the type definition for
the definition.
 *)
  definition : data_type_definition_select;
  (* The description specifies additional textual information on the Data_instance. *)
  description : OPTIONAL text_select;
  (* The id specifies the identifier of the Data_instance. *)
  id : element_identifier;
  (* The initial_value specifies the value to be assigned to the Data_instance when it is created. <note number="4">The
value of the initial_value shall be the valid value range of the initial_value, if one is defined.</note> *)
  initial_value : OPTIONAL data_type_value_select;
  (* The is_constant specifies whether the Data_instance is a constant or not. If the value is TRUE, the is_constant may
not be updated. In this case the value of the is_constant is specified by the initial_value attribute. If no initial value is
provided then the value of the is_constant is undefined. *)
  is_constant : BOOLEAN;
  (* The name specifies the word, or words, that are used to refer to the Data_instance. *)
  name : label;
  (* The unit_component specifies the units of measurement that shall be associated with the Data_instance. <example
number="4">For a particular Data_instance, the unit_component may be Volts.</example> *)
  unit_component : OPTIONAL unit;
UNIQUE
  UR1: definition, id;
END_ENTITY;
```

```
(* A Data_transfer is the definition of the nature of interaction identified by an Implied_external_interaction object.
<note>The scope of a Data_transfer need not be restricted to data, it may be used to indicate interaction of physical
elements as well.</note> *)
ENTITY data_transfer;
  (* The data is the Data_instance that is part of the interaction between the functional representation of an external entity
and the functional representation of a system. *)
  data : data_instance;
  (* The direction specifies the direction of the interaction. *)
  direction : data_direction;
  (* The transfer specifies the Implied_external_interaction object for which the Data_transfer is valid. *)
  transfer : implied_external_interaction;
END_ENTITY;

(* A Date_and_person_assignment is an object that associates a Date_and_person_organization with an object. <note
number="1">This assignment provides additional information for the associated object. The provision of such data
through this assignment has an organizational character whereas some objects mandate the same kind of data in order
to be semantically complete. This assignment shall not be used to associate the corresponding organizational data
with an object whose attributes are referencing the organizational data directly.</note> <note number="2">Taken from
AP214.</note> *)
ENTITY date_and_person_assignment;
  (* The assigned_date_and_person specifies the Date_and_person_organization. *)
  assigned_date_and_person : date_and_person_organization;
  (* The assigned_to specifies the object to which the Date_and_person_assignment is applied. *)
  assigned_to : person_organization_assignment_select;
  (* The description specifies additional textual information on the Date_and_person_assignment. *)
  description : OPTIONAL text_select;
  (* The role specifies the relationship between the date or time and the Person or Organization in the role. Where appli-
cable, the following values shall be used:<ul><li>creation: The assignment specifies that the referenced object has been
created by the given person or organisation at the given date and time;</li><li>update: The assignment specifies that the
referenced object has been altered by the given person.</li></ul> *)
  role : label;
END_ENTITY;

(* A Date_and_person_organization is a Person_in_organization or an Organization associated with a Date_time. *)
ENTITY date_and_person_organization;
  (* The actual_date specifies the Date_time component of the Date_and_person_organization. *)
  actual_date : date_time;
  (* The Person_organization specifies the Person_in_organization or an Organization  component of the Date_and_per-
son_organization. *)
  person_organization : person_organization_select;
END_ENTITY;

(* A date_assignment is the assignment of Date_time and time information. *)
ENTITY date_assignment;
  (* The assigned_to specifies the object the Date_time information is assigned to. *)
  assigned_to : date_assignment_select;
  (* The date specifies the date component in the assignment. *)
  date : date_time;
  (* The role specifies the semantics of the Date_time assigned for the item. Where applicable the following values shall
be used:<ul><li>created: The assignment specifies that the referenced object was created at the given Date;</li><li>-
modified: The assignment specifies that the referenced object was altered at the given Date.</li></ul><note>The current
value set can be extended to suit further needs. However, in such cases both the sending and the receiving tools shall
agree upon the valid value set.</note> *)
  role : label;
END_ENTITY;

(* A Date_time is the specification of a date and time of day. *)
ENTITY date_time;
  (* The day_component specifies the day number of a month. The value of the day_component must not exceed the
number of days in the current month. *)
  day_component : INTEGER;
```

```
  (* The hour_component specifies the time in a day in hours.<p>The value of the hour_component shall be in the range
0-23.</p> *)
  hour_component : INTEGER;
  (* The minute_component specifies the time within an hour in minutes.<p>The value of the minute_component shall be
in the range 0-59.</p> *)
  minute_component : INTEGER;
  (* The month_component specifies the Date_time within a year in months.<p>The value of the month_component shall
be in the range 1-12.</p> *)
  month_component : INTEGER;
  (* The second_component specifies the time within a minute in seconds.<p>The value of the second_component shall
be in the range 0-59.</p> *)
  second_component : INTEGER;
  (* The year_component specifies the year part of a Date_time.<p>The year_component shall be given with all valid
digits.</p><example number="5">The year 1999 shall be represented as the integer "1999".</example> *)
  year_component : INTEGER;
WHERE
  WR1: {0 <= hour_component <= 23};
  WR2: {0 <= minute_component <= 59};
  WR3: {0 <= second_component <= 59};
  WR4: {1 <= month_component <= 12};
  WR5: {1 <= day_component <= 31};
  WR6: year_component >= 0;
END_ENTITY;

(* A Derived_data_type_definition is type of User_defined_data_type_definition whose value is calculated from Data_in-
stance and or Data_field values. *)
ENTITY derived_data_type_definition
SUBTYPE OF (user_defined_data_type_definition);
  (* The expression specifies the algorithm for deriving the value of an instance of a Derived_data_type_definition.
<note>The syntax for specifying the algorithm for deriving the value of a expression is not defined.</note><example
number="6">To avoid redundancy in a database design and to keep the data up-to-date, the age field is calculated from
the global value today and the field date of birth. age = today - date_of_birth.</example> *)
  expression : text_select;
  (* The resultant_data_type specifies the expected type of the result of the expression based on the types of the values
involved in the expression. *)
  resultant_data_type : data_type_definition_select;
END_ENTITY;

(* A Digital_document is a type of Documentation_reference and a reference to a document in some digital format. *)
ENTITY digital_document
SUBTYPE OF (documentation_reference);
  (* The document_format specifies the format (for example: MSWord, PDF, HTML) of the digital document. *)
  document_format : OPTIONAL label;
  (* The document_size specifies the size of a digital document. *)
  document_size : OPTIONAL label;
  (* The location specifies the location (for example: file structure, URL) from where the digital document can be obtained.
*)
  location : label;
END_ENTITY;

(* A Document_assignment is the assignment of a Documentation_reference to an element of systems engineering. *)
ENTITY document_assignment
SUPERTYPE OF (partial_document_assignment);
  (* The description specifies additional information about the Document_assignment. *)
  description : OPTIONAL text_select;
  (* The documentation specifies the Documentation_reference. *)
  documentation : documentation_reference;
  (* The documented_object specifies an object that is documented by the Document_assignment. *)
  documented_object : instance_definition_select;
END_ENTITY;
```

```
(* A Documentation_reference is a reference to any relevant information. A Documentation_reference is an abstract entity
that shall never be instantiated. *)
ENTITY documentation_reference
ABSTRACT SUPERTYPE OF (ONEOF(digital_document, non_digital_document));
  (* The associated_version specifies the Configuration_element_version for the Documentation_reference. *)
  associated_version : configuration_element_version;
  (* The description specifies additional information about the Documentation_reference. *)
  description : OPTIONAL text_select;
  (* The id specifies the identifier of the Documentation_reference. *)
  id : element_identifier;
  (* The name specifies the word, or words, that are used to refer to the Documentation_reference. *)
  name : label;
UNIQUE
  UR1: id;
END_ENTITY;

(* A Documentation_relationship is the relationship between two Documentation_reference objects. The semantics of the
relationship is defined by the description attribute. *)
ENTITY documentation_relationship;
  (* The description specifies additional information about the Documentation_relationship. *)
  description : OPTIONAL text_select;
  (* The related_documentation specifies the second Documentation_reference in the relationship. *)
  related_documentation : documentation_reference;
  (* The relating_documentation specifies the first Documentation_reference in the relationship. *)
  relating_documentation : documentation_reference;
  (* The relationship_type specifies the word, or words, that are used to refer to the Documentation_relationship. *)
  relationship_type : OPTIONAL label;
WHERE
  correct_relation: relating_documentation :<>: related_documentation;
END_ENTITY;

(* A Effectiveness_measure is an optimization criteria for any number of Requirement_instance objects. *)
ENTITY effectiveness_measure;
  (* The optimization_function specifies the inter-requirement optimization function defined for the Effectiveness_mea-
sure. *)
  optimization_function : textual_specification;
END_ENTITY;

(* An Effectiveness_measure_assignment is the mechanism for including a Requirement_instance in an inter-require-
ment optimization criterion represented by an Effectiveness_measure. All Requirement_instance objects assigned shall
be associated with the same System_view. *)
ENTITY effectiveness_measure_assignment;
  (* The assigned_requirement specifies the Requirement_instance involved in the optimization function by the Effective-
ness_measure_assignment. *)
  assigned_requirement : requirement_instance;
  (* The effectiveness_measure specifies the Effectiveness_measure that the Requirement_instance is assigned to via
the Effectiveness_measure_assignment. *)
  effectiveness_measure : effectiveness_measure;
  (* The weight specifies the relative importance of the assigned_requirement Requirement_instance compared to the
other Requirement_instance objects assigned to the same Effectiveness_measure. *)
  weight : label;
END_ENTITY;

(* An Effectiveness_measure_relationship is the mechanism for indicating a relationship between two Effectiveness_
measure objects. The nature of the relationship is further defined by the description attribute. <note>The Effectiveness_
measure_relationship may be used to indicate the relationship between two optimization criteria.</note> *)
ENTITY effectiveness_measure_relationship;
  (* The description specifies additional information about the Effectiveness_measure_relationship. *)
  description : OPTIONAL text_select;
  (* The relating specifies the second element in the relationship. *)
  related : effectiveness_measure;
```

```
  (* The relating specifies the first element in the relationship. *)
  relating : effectiveness_measure;
END_ENTITY;

(* An Effectivity is the identification of the valid use of an aspect of product data tracked by Date_time or event. <note
number="1">The effectivity may define open ended or closed intervals for the validity of the produced data tracked.</
note> <note number="2">The Effectivity shall be specified either by a primary_definition attribute or by another Effectivity
(related using an Effectivity_relationship) that has an instantiated primary_definition.</note> *)
ENTITY effectivity;
  (* The concerned_organization specifies the Organization in which the Effectivity is valid.<example number="7">The
Effectivity of the same item may be different in the various sites of a system supplier.</example> *)
  concerned_organization : SET [0 : ?] OF organization;
  (* The description specifies additional textual information on the Effectivity. *)
  description : OPTIONAL text_select;
  (* The id specifies the identifier of the Effectivity. *)
  id : OPTIONAL element_identifier;
  (* The primary_definition specifies the Date_time or event when the identified object becomes effective or ceases to be
valid. <note number="3">The meaning of this attribute is further detailed by the attribute role of Effectivity_assignment.</
note> *)
  primary_definition : OPTIONAL event_or_date_select;
  (* The secondary definition is used to define a period of time the start of which is specified as primary definition. This
attribute shall be used only if the attribute role of the Effectivity_assignment indicates that the secondary_definition is
defined as a period of time. *)
  secondary_definition : OPTIONAL period_or_date_select;
  (* The version id specifies the identification of a particular version of the Effectivity. *)
  version_id : OPTIONAL element_identifier;
END_ENTITY;

(* An Effectivity_assignment associates an Effectivity with the object whose Effectivity is controlled by the associated
Effectivity. *)
ENTITY effectivity_assignment;
  (* The assigned_effectivity specifies the assigned Effectivity. *)
  assigned_effectivity : effectivity;
  (* The effective_element specifies the object that has an Effectivity assigned to it. *)
  effective_element : effective_element_select;
  (* The role specifies relationship between the Effectivity and the object that has an effectivity assigned to it. <p>If one of
the values “actual_start”, “actual_stop”, “planned_start”, or “planned_stop” is specified, no secondary definition shall be
given in the assigned Effectivity.</p> <p>Where applicable, the following values shall be used:</p> <ul> <li>actual_peri-
od: The actual period during which the Effectivity lasted;</li> <li>actual_start: The actual Date_time in the past when the
Effectivity became effective; </li> <li>actual_stop: The actual Date_time in the past when the Effectivity ceased to be ef-
fective;</li> <li>planned_period: The period associated with the Effectivity defines a planned period of time during which
the associated object is or was supposed to be effective;</li> <li>planned_start: The Date_time or event associated with
the Effectivity defines the Date_time or event when the Effectivity is or was supposed to start;</li> <li>planned_stop: The
Date_time or event associated with the Effectivity defines the Date_time or event when the Effectivity is or was supposed
to stop;</li><li>required_period: The associated object must be kept effective for this period;</li><li>required_start: The
Date_time or event associated with the Effectivity is a due start Date. Conformance to this bound is expected;</li><li>re-
quired_stop: The Date_time or event associated with the Effectivity is a due stop date. Conformance to this bound is
expected.</li></ul> *)
  role : label;
END_ENTITY;

(* An Effectivity_relationship is a relationship between two Effectivity objects. *)
ENTITY effectivity_relationship;
  (* The description specifies additional information about the Effectivity_relationship. *)
  description : OPTIONAL text_select;
  (* The related specifies the second Effectivity object in the relationship. *)
  related : effectivity;
  (* The relating specifies the first Effectivity object in the relationship. *)
  relating : effectivity;
  (* The relation type specifies the meaning of the relationship. Where applicable, the following values shall be used:<ul>
<li>constraint: The time period between the start and end definition of the related Effectivity shall be within the time period
```

of the relating Effectivity; inheritance: The related Effectivity shall not have a “primary definition” and “secondary definition” specified but inherits the effectivity dates from the relating Effectivity. *)
  relation_type : label;
END_ENTITY;

(* An Element_critical_issue_relationship is the mechanism for indicating that an element (defined by the impact_on_element attribute) is influenced in some respect by impact of a Critical_issue captured by a Critical_issue_impact. The impact is not definite as several Critical_issue_impact objects may be created as a result of a Critical_issue. The nature of the impact is defined by the description attribute. *)
ENTITY element_critical_issue_relationship;
  (* The description specifies the nature of impact for the element identified by the impact_on_element attribute of the element_critical_issue_relationship. *)
  description : OPTIONAL text_select;
  (* The impact_on_element specifies the element influenced by the Critical_issue. *)
  impact_on_element : change_element_select;
  (* The issue_analysis specifies the Critical_issue_impact for which the element_critical_issue_relationship is valid. *)
  issue_analysis : critical_issue_impact;
END_ENTITY;

(* An Element_identifier is an identifier used to represent a unique label for elements of the same type. *)
ENTITY element_identifier;
  (* The identifier_context specifies the organizational scope in which the identifier_value attribute is assumed to be unique for an Element_identifier. *)
  identifier_context : person_organization_select;
  (* The identifier_value specifies an alphanumeric string identifying the element, that the Element_identifier is assigned to, uniquely within its scope. *)
  identifier_value : identifier;
END_ENTITY;

(* An Elementary_maths_space is a set of values that have an existing precise mathematical meaning. *)
ENTITY elementary_maths_space
ABSTRACT SUPERTYPE OF (ONEOF(binary_data_type_definition, boolean_data_type_definition, complex_data_type_definition, event_data_type_definition, integer_data_type_definition, logical_data_type_definition, real_data_type_definition, string_data_type_definition))
SUBTYPE OF (maths_space);
END_ENTITY;

(* An Engineering_process_activity is a part of the process of developing a system. An Engineering_process_activity may be planned, in progress, or already carried out. <note>In the scope of PAS 20542, an Engineering_process_activity is a discrete or composite element of the systems engineering process.</note><example number=”8”>The requirements analysis phase of the systems engineering process would be considered to be an Engineering_process_activity. Performing safety analysis on a certain subsystem would also be considered to be an Engineering_process_activity.</example> *)
ENTITY engineering_process_activity;
  (* The activity_type specifies the purpose of the Engineering_process_activity. Where applicable the following values shall be used:
analysis: The Engineering_process_activity represents an analysis acitivity; design: The Engineering_process_activity is representing a design acitivity; review: The Engineering_process_activity is representing a human review acitivity; design_change: The Engineering_process_activity is representing a change acitivity; trade_off_analysis: The Engineering_process_activity is representing selection acitivity. *)
  activity_type : label;
  (* The actual_end_date specifies the date when the Engineering_process_activity actually finished. *)
  actual_end_date : OPTIONAL date_time;
  (* The actual_start_date specifies the date when the Engineering_process_activity actually started. *)
  actual_start_date : OPTIONAL date_time;
  (* The description specifies additional information about the Engineering_process_activity. *)
  description : OPTIONAL text_select;
  (* The id specifies the identifier of the Engineering_process_activity. *)
  id : element_identifier;
  (* The name specifies the word, or words, that are used to refer to the Engineering_process_activity. *)
  name : label;
  (* The planned_end_date specifies the Date_time when the Engineering_process_activity is or was supposed to be completed. *)

```
  planned_end_date : OPTIONAL period_or_date_select;
  (* The planned_start_date specifies the Date_time when the Engineering_process_activity is or was supposed to start.
*)
  planned_start_date : OPTIONAL event_or_date_select;
  (* The resolved_request specifies the set of Work_request objects that is resolved by the Engineering_process_activity.
*)
  resolved_request : SET [0 : ?] OF work_request;
  (* The status specifies the level of completion of the Engineering_process_activity. *)
  status : OPTIONAL text_select;
INVERSE
  (* The is_controlling specifies the Engineering_process_activity objects that are controlled by this particular Work_order.
*)
  authorization : SET [0 : 1] OF work_order FOR is_controlling;
END_ENTITY;

(* An Engineering_process_activity_element_assignment is the relationship between systems engineering data and the
Engineering_process_activity in which the data are “created in” or “modified in” or “referenced in”. *)
ENTITY engineering_process_activity_element_assignment;
  (* The activity specifies the Engineering_process_activity to which the Engineering_process_activity_element_assign-
ment belongs. *)
  activity : engineering_process_activity;
  (* The description specifies additional information about the Engineering_process_activity_element_assignment. *)
  description : OPTIONAL text_select;
  (* The element specifies the piece of systems engineering data referenced by the Engineering_process_activity. *)
  element : specification_element_select;
  (* The role specifies the function that is performed by the Engineering_process_activity_element_assignment in the
context of the concerned Engineering_process_activity. Where applicable the following values shall be used:<ul><li>con-
trol: The referenced element had influence on completing the Engineering_process_activity;</li><li>created: The refer-
enced element was created in the Engineering_process_activity;</li><li>modified: The referenced element was modified
in the Engineering_process_activity.</li></ul> *)
  role : label;
END_ENTITY;

(* An Engineering_process_activity_relationship is a relationship between two Engineering_process_activity objects. *)
ENTITY engineering_process_activity_relationship;
  (* The description specifies additional information about the Engineering_process_activity_relationship. *)
  description : OPTIONAL text_select;
  (* The related_activity specifies the second of the two Engineering_process_activity objects related by an Engineer-
ing_process_activity_relationship. *)
  related_activity : engineering_process_activity;
  (* The relating_activity specifies the first of the Engineering_process_activity objects related by an Engineering_pro-
cess_activity_relationship. *)
  relating_activity : engineering_process_activity;
  (* The relation_type specifies a textual description of the nature of the relationship between the two Engineer-
ing_process_activity objects involved in the relationship. Where applicable one of the following values shall be
used:<ul><li>alternative: Either the relating_activity or the related_activity shall be performed;</li><li>hierarchy: The
relationship between the relating_activity and the related_activity is such that the relating_activity is divided into the re-
lated_activity;</li><li>sequence: The relating_activity and the related_activity shall be performed in sequence where the
relating_activity precedes the related_activity;</li><li>simultaneous: The relating_activity and the related_activity may be
performed simultaneously.
</li></ul> *)
  relation_type : label;
WHERE
  WR1: relating_activity :<>: related_activity;
END_ENTITY;

(* An Event_data_type_definition is a type of Elementary_maths_space that includes all event values. *)
ENTITY event_data_type_definition
SUBTYPE OF (elementary_maths_space);
END_ENTITY;
```

(* An Execution_time is an estimation of the time it takes for a function to produce the outputs either when it has been activated or as soon as the whole input data set is available. <note number=»1»>The execution time shall not be confused with the time for a physical component to perform the function which would be related to the physical architecture definition in the physical UoF.</note> <example number=»9»> The exact semantics of this entity is defined by the role attribute. The execution time given here records, for instance, the time that is necessary within a process industry to perform a specific action. For instance in a process industry it may be necessary to have a piece of work in a given acid bath for a given period to achieve specific properties.</example> <note number=»2»>An execution_time can be associated with all types of General_function_definition objects.</note> <note number=»3»>The execution_time specifies the time it takes for a function to be executed, not counting interference from other functions in the system.</note> *)
ENTITY execution_time;
(* The role specifies how the time attribute should be interpreted. <example number=»10»>The function calculate_time_to_destination may have a best_case Execution_time of 0 seconds, a worst_case Execution_time of 0.020 seconds, and no value expressed for nominal_case Execution_time.</example> *)
role : timing_type;
(* The time specifies the execution time. *)
time : REAL;
(* The timing specifies the General_function_definition to which the Execution_time applies. *)
timing : functionality_instance_reference;
(* The unit specifies the Execution_time unit for the time attribute. *)
unit : label;
END_ENTITY;

(* A Finite_integer_interval is type of Integer_interval that has lower and upper bounds that constrain the sub-set of integers it contains. *)
ENTITY finite_integer_interval
SUBTYPE OF (integer_interval);
(* The low_index specifies the lowest value that a Data_instance of this type can take. *)
low_index : INTEGER;
(* The size specifies the value domain of the Finite_integer_interval as number of consequtive integer values, counting from the low_index attribute, a Data_instance of this type can take. <note>The upper bound of the value domain of a Finite_integer_interval is given by the expression low_index + size - 1.</note> *)
size : INTEGER;
END_ENTITY;

(* A Finite_real_interval is a type of Real_interval that has lower and upper bounds that constrain the sub-set of real values it contains. *)
ENTITY finite_real_interval
SUBTYPE OF (real_interval);
(* The high_closure specifies whether or not the high_index is included in the maths space. If the value is TRUE, then the value is included in the space, otherwise the value is excluded. *)
high_closure : BOOLEAN;
(* The high_index specifies the highest value that a Data_instance of this type can take. *)
high_index : REAL;
(* The low_closure specifies whether or not the low_index is included in the maths space. If the value is TRUE, then the value is included in the space, otherwise the value is excluded. *)
low_closure : BOOLEAN;
(* The low_index specifies the lowest value that a Data_instance of this type can take. *)
low_index : REAL;
END_ENTITY;

(* A finite_space is a type of Maths_space containing an explicit set of values. <note>Elementary_maths_space objects that are finite (such as Binary_data_type_definition and Logical_data_type_definition objects) should be used in preference to finite_space objects.</note> *)
ENTITY finite_space
SUBTYPE OF (maths_space);
(* The member specifies the set of elements the build up the Finite_space. *)
member : SET [0 : ?] OF data_type_value_select;
END_ENTITY;

(* A Formal_data_interaction_port is an element in the interface between to a Functional_state_context object. *)
ENTITY formal_data_interaction_port;

```
  (* The data specifies the Data_instance object in the interface. *)
  data : data_instance;
  (* The port_of specifies the Functional_state_context object for which the Formal_data_interaction_port provides part of
the interface. *)
  port_of : functional_state_context;
END_ENTITY;

(* A Formal_io_port is a type of Io_port and the definition of an element of the interface to a General_function_definition.
<note number=»1»>If a General_function_definition has an interface one or more formal_io_port objects will specify the
interface.</note> <note number=»2»>Flow ports are classified according to three criteria in the data model:</note> <ol>
<li>Whether the port is formal (attached to a General_function_definition) or actual (attached to a Function_instance,
Fsm_model, Persistent_storage or Io_split_join),</li> <li>Whether the port is an input or output port,</li> <li>Whether
the port is a flow or control port (a flow port carries data whereas control port carries triggering information such as start,
stop, suspend, resume).</li> </ol> From this classification the role the port plays with regard to the Functional_link ob-
jects connected to the port can be derived. A port can «consume» flows (the data on the flow is delivered to the port) or it
can produce flows (the data on the flow is delivered from the port). For instance, a formal_io_port object whose direction
attribute is «output» produces data, while an Actual_io_port whose direction attribute is «input» consumes data. <p>In
the data model we capture this information in the derived role attribute. This attribute is used to ensure that Function-
al_link objects are connected correctly (a Functional_link object must have one producer and one consumer), and that
Io_port_binding objects are applied only to ports where the actual_port in the binding is a producer of information and the
formal_port in the binding is a consumer or vice versa.</p> <note number=»3»>When a Function_instance of the Gener-
al_function_definition is created an Actual_io_port object corresponding to each formal_io_port is created if (and only if)
there is a flow connected to the actual port (and the actual port is related to the corresponding formal_io_port object via a
Io_port_binding object).</note> *)
ENTITY formal_io_port
SUBTYPE OF (io_port);
  (* The port_of specifies the General_function_definition for which this is a port. This attribute is reinforced by the inverse
relationship which indicates that a port_of can only point at a definition but a definition can be referred to by several
ports. *)
  port_of : general_function_definition;
DERIVE
  SELF\io_port.role : port_data_relation := determineformalportrole(SELF);
UNIQUE
  UR1: port_of, SELF\io_port.io_port_number, SELF\io_port.port_type;
END_ENTITY;

(* A Formal_physical_port is a type of Physical_port and represents either an input and an output or an input or an output
of a General_physical_definition. *)
ENTITY formal_physical_port
SUBTYPE OF (physical_port);
  (* The port_of specifies the General_physical_defintion object for which the Formal_physical_port provides part of the
interface. *)
  port_of : general_physical_definition;
END_ENTITY;

(* A Formal_port_position is a type of Visual_element and the representation of the graphical position of a Formal port, a
Formal_io_port, or a Formal_physical_port object. *)
ENTITY formal_port_position
SUBTYPE OF (visual_element);
  (* The position specifies the position of the port. *)
  position : graphics_point;
  (* The positioned_port specifies the formal port that is positioned by the Formal_port_position. *)
  positioned_port : port_position_select;
END_ENTITY;

(* An Fsm_and_state is a type of Fsm_state and the representation of a decomposed state in a finite state machine
where all child Fsm_state objects are active concurrently. An And_state is composed of a minimum of two child Fsm_
states. *)
ENTITY fsm_and_state
SUBTYPE OF (fsm_state);
INVERSE
```

```
  SELF\fsm_state.child_states : SET [2 : ?] OF fsm_state_composition_relationship FOR parent_state;
END_ENTITY;

(* An Fsm_command_interaction_relationship is the mechanism for indicating that a command is issued to a the reference to a function (in the form of a state_function_interaction_port) by a textual_specification. The fsm_command_interaction_relationship is only used within the scope of a finite state machine. The nature of the command is specified in the interaction_type attribute. <note number=»1»>The Fsm_comment_interaction_relationship indicates that a command (function: start, stop, suspend, resume) is issued by a Fsm_transition or Fsm_state. The notation complements the textual definition of actions in states or in transitions.</note> *)
ENTITY fsm_command_interaction_relationship;
  (* The defined_in specifies the textual_specification where the command is issued. *)
  defined_in : fsm_interaction_select;
  (* The interaction_port specifies the state_function_interaction_port through which the command relayed by the Fsm_command_interaction_relationship is sent. *)
  interaction_port : state_function_interaction_port;
  (* The interaction_type specfies the semantics of the Fsm_command_interaction_relationship. Where applicable one of the following values shall be used:<ul><li>start: The referenced function is started by the command;</li> <li>stop: The referenced function is stopped by the command;</li> <li>resume: The referenced function is resumed from a suspended state by the command;</li> <li>suspend: The referenced function is temporarily suspended by the command.</li></ul><note number=»2»>All commands predefined are idempotent, i.e., issuing a «start» command to a started function does not have any effect.</note> *)
  interaction_type : label;
END_ENTITY;

(* An Fsm_data_interaction_binding is a parameter matching relationship between an element in the actual interface of an Actual_io_port and an element of the formal_interface of a Formal_data_interaction_port. *)
ENTITY fsm_data_interaction_binding;
  (* The actual_port specifies the Actual_io_port in the relationship. *)
  actual_port : actual_io_port;
  (* The formal_port specifies the Formal_data_interaction_port in the relationship. *)
  formal_port : formal_data_interaction_port;
END_ENTITY;

(* An Fsm_data_interaction_relationship is the mechanism for indicating the a data_instance is referenced within a textual_specification. The nature of the reference is defined by the interaction_type attribute. <note>The Fsm_data_interation_relationship that the associated data item of an Io_port fsm_interaction_port object is updated or read as a part of a action performed in a state or transition.</note> *)
ENTITY fsm_data_interaction_relationship;
  (* The defined_in specifies the Textual_specification where the command is issued. *)
  defined_in : fsm_interaction_select;
  (* The interaction_port specifies the Formal_data_interaction_port in the relationship. *)
  interaction_port : formal_data_interaction_port;
  (* The interaction_type specfies the semantics of the Fsm_data_interaction_relationship. Where applicable one of the following values shall be used:<ul><li>read: The value of the Data_instance on the Formal_data_interaction_port is referenced  in the Textual_specification identified by the defined_in attribute;</li><li>write: The value of the Data_instance on the Formal_data_interaction_port is written  in the Textual_specification identified by the defined_in attribute.
</li></ul> *)
  interaction_type : label;
END_ENTITY;

(* A Fsm_generic_state is the abstract super class of either a Fsm_state, or a Fsm_transient_state. *)
ENTITY fsm_generic_state
ABSTRACT SUPERTYPE OF (ONEOF(fsm_state, fsm_transient_state));
INVERSE
  (* The destination_state specifies the Fsm_state activated if the Fsm_state_transition is fired. *)
  destination_transition : SET [0 : ?] OF fsm_state_transition FOR destination_state;
END_ENTITY;

(* An Fsm_initial_state_transition is the initial transition to the initial state of an Fsm_or_state in a finite state model. *)
ENTITY fsm_initial_state_transition;
  (* The initial_state specifies the initial state. *)
```

```
  initial_state : fsm_generic_state;
  (* The transition_context specifies the environment for which the initial_state_context is valid. *)
  transition_context : default_context_select;
END_ENTITY;

(* An Fsm_model is a type of General_functionality_instance and represents the entry point to a finite state machine.
<note number=»1»>Fsm_model objects may be referred to within the function breakdown structure and represent the activation and de-activation logic of functions in the functional breakdown structure.</note><note number=»2»>This entity is included in the model to allow a FSM to be included in a General_function_definition and provides the link between the functional view and state view of a system.</note> *)
ENTITY fsm_model
SUBTYPE OF (general_functionality_instance);
  (* The behaviour_model specifies the State_machine_functional_behaviour_model object for which the Fsm_model provides the behaviour specification. *)
  behaviour_model : state_machine_functional_behaviour_model;
  (* The definition specifies the Functional_state_context that contains the finite state machine model. *)
  definition : functional_state_context;
  (* The id specifies the identifier of the Fsm_model. *)
  id : element_identifier;
  (* The name specifies the word, or words, that are used to refer to the Fsm_model. *)
  name : label;
  (* The presentation_id specifies the identity information for a Fsm_model that shall be presented to the user. *)
  presentation_id : OPTIONAL label;
END_ENTITY;

(* An Fsm_or_state is a type of Fsm_state of a finite state machine which, if decomposed, constrains the semantics of the child Fsm_state objects such that only one of the child states may be active when the parent state is active. <note number=»1»>subclass definition probably the best way to document this properly!</note> <note number=»2»>The fsm_or_state corresponds to the or state of statecharts and the normal state of a finite state machine.</note> *)
ENTITY fsm_or_state
SUBTYPE OF (fsm_state);
END_ENTITY;

(* A fsm_state is a type of Fsm_generic_state and a representation of a static state in a finite state machine.
<note>
Within the scope of AP 233 a fsm_state is always an element of a finite state machine. The concept «state» in EACM is in AP 233 terminology a Partial_system_view.
</note> *)
ENTITY fsm_state
ABSTRACT SUPERTYPE OF (ONEOF(fsm_and_state, fsm_or_state))
SUBTYPE OF (fsm_generic_state);
  (* The description specifies additional information on the description.
 *)
  description : OPTIONAL text_select;
  (* The name specifies the word or set of words to which the name is referred to.
 *)
  name : label;
  (* The presentation_id specifies the identity information for a presentation_id that shall be presented to the user.
 *)
  presentation_id : OPTIONAL label;
INVERSE
  (* The parent_state specifies the parent Fsm_state in the relationship. *)
  child_states : SET [0 : ?] OF fsm_state_composition_relationship FOR parent_state;
END_ENTITY;

(* An Fsm_state_composition_relationship is a relationship between two Fsm_state objects defining a parent child relationship between the two objects. <note number=»1»>The parent Fsm_state is identified by the parent_state attribute and the child Fsm_state is identified by the child_state attribute.</note> <note number=»2»>The State Fsm_state_composition_relationship entity is included in the model to support statechart extensions to the traditional finite state machine formalism.</note> *)
ENTITY fsm_state_composition_relationship;
```

```
  (* The child_state specifies the child Fsm_state in the relationship. *)
  child_state : fsm_state;
  (* The parent_state specifies the parent Fsm_state in the relationship. *)
  parent_state : fsm_state;
END_ENTITY;

(* An Fsm_state_transition is a representation of a possible execution path between two Fsm_generic_state objects.
<note>A fsm_state_transition is applicable for use in finite state machines only.</note> *)
ENTITY fsm_state_transition;
  (* The destination_state specifies the Fsm_state activated if the Fsm_state_transition is fired. *)
  destination_state : fsm_generic_state;
  (* The source_state specifies the Fsm_state originally active in the relationship. *)
  source_state : fsm_generic_state;
INVERSE
  (* The assigned_to specifies the Fsm_state_transition to which the textual specification is assigned.
 *)
  transition_label : SET [0 : 1] OF state_transition_specification_assignment FOR assigned_to;
END_ENTITY;

(* An Fsm_transient_state is a type of Fsm_generic_state with the discriminator the state is not, under any conditions, a static state of a finite state machine. <note>In statechart terminology a fsm_transient_state is a pseudo-state. Under no circumstances shall the fsm_transient_state be a static state of a system. If transitions out of a fsm_transient_state exist then these shall either be non guarded or the union of the guarding conditions shall be one, i.e., at least one guard will always evaluate to TRUE.</note> *)
ENTITY fsm_transient_state
SUBTYPE OF (fsm_generic_state);
  (* The state_type specifies the type of a Fsm_transient_state. Where applicable the state_type shall be one of the following values: <ul> <li>History</li> <li>Deep-history</li> <li>Condition</li> <li>Select</li> *)
  state_type : label;
END_ENTITY;

(* An Fsm_transient_state_composition_relationship is a relationship between fsm_state and a fsm_transient_state indicating that the Fsm_transient_state is a sub-state of the Fsm_state. <note>An Fsm_transient_state_composition_relationship may be instantiated only of there is at least one Fsm_state_composition_relationship instantiated for the Fsm_state. An Fsm_state may not only be composed of Fsm_transient_state objects.</note> *)
ENTITY fsm_transient_state_composition_relationship;
  (* The child_state represents the Fsm_transient_state in the Fsm_transient_state_composition_relationship. *)
  child_state : fsm_transient_state;
  (* The parent_state represents the Fsm_state in the Fsm_transient_state_composition_relationship. *)
  parent_state : fsm_state;
END_ENTITY;

(* A Function_instance is a type of General_functionality_instance and a representation of an entity that performs an activity within a system. <note>The functional domain describes what a system does.</note> *)
ENTITY function_instance
SUBTYPE OF (general_functionality_instance);
  (* The definition specifies the General_function_definition that provides the definition for the Function_instance. *)
  definition : general_function_definition;
  (* The id specifies the identifier of the Function_instance. *)
  id : element_identifier;
  (* The name specifies the word, or words, which are used to refer to the Function_instance. *)
  name : label;
  (* The presentation_id specifies the identity information for a Function_instance that shall be presented to the user. *)
  presentation_id : OPTIONAL label;
INVERSE
  (* The port_of specifies the Function_instance object to which the Control_io_port is attached. *)
  control_port : SET [0 : ?] OF control_io_port FOR port_of;
UNIQUE
  UR1: definition, id;
END_ENTITY;
```

(* A Function_reference is the relationship between a Function_instance object and a Fsm_model object. <note number="1">The function_reference is the means for referencing elements in the name space of a Composite_function_definition to the name space of a finite state machine.</note> <note number="2">To complete the referencing mechanism the following entities shall be used: State_function_interaction_port and Name_binding.</note> *)
ENTITY function_reference;
  (* The function_link specifies the Function_instance that is imported into the finite state machine. *)
  function_link : function_instance;
  (* The port_of specifies the Fsm_model into which the Function_instance object is imported. *)
  port_of : fsm_model;
END_ENTITY;

(* A Functional_behaviour_model is the definition of functional behaviour model. A Functional_behaviour_model is an abstract super type that shall never be instantiated. *)
ENTITY functional_behaviour_model
ABSTRACT SUPERTYPE OF (ONEOF(cb_functional_behaviour_model, state_machine_functional_behaviour_model));
  (* The description specifies additional textual information on the Functional_behaviour_model. *)
  description : OPTIONAL text_select;
INVERSE
  (* The assigned_behaviour_model specifies the Functional_behaviour_model assigned. *)
  defines_behaviour_for : SET [0 : 1] OF functional_behaviour_model_assignment FOR assigned_behaviour_model;
END_ENTITY;

(* A Functional_behaviour_model_assignment is an assignment relationship between a Functional_behaviour_model and a Composite_function_definition object whose behaviour is defined by the Functional_behaviour_model. *)
ENTITY functional_behaviour_model_assignment;
  (* The assigned_behaviour_model specifies the Functional_behaviour_model assigned. *)
  assigned_behaviour_model : functional_behaviour_model;
  (* The constrained_function specifies the Composite_function_definition object whose behaviour is constrained by the assigned Functional_behaviour_model. *)
  constrained_function : composite_function_definition;
END_ENTITY;

(* A Functional_decomposition_relationship is the hierarchical relationship between a Composite_function_definition and a General_functionality_instance. <note number="1">The Composite_function_definition is the parent in the relationship and the General_functionality_instance object is the child in the relationship.</note> *)
ENTITY functional_decomposition_relationship;
  (* The child specifies the General_functionality_instance that is part of the composition of the parent. *)
  child : general_functionality_instance;
  (* The description specifies the role that the child plays in the Composite_function_definition specified by the parent attribute. <note number="2">As a child object may be part of many Composite_function_definition objects, the description attribute allows for a description of the role of the child in a particular Composite_function_definition.</note> *)
  description : OPTIONAL text_select;
  (* The parent specifies the Composite_function_definition of which the child is a part of the decomposition. *)
  parent : composite_function_definition;
END_ENTITY;

(* A Functional_link is the relationship between two Io_port objects such that there is a data interaction path between the two ports. <note number="1">The direction of the data path is from the source_port to the destination_port.</note> <note number="2">The data carried by the functional_link is determined by the data attribute of the source_port and destination_port attribute. The same Data_instance object shall be at the source and destination port.</note> *)
ENTITY functional_link;
  (* The control_link specifies whether or not the Functional_link carries control information. *)
  control_link : BOOLEAN;
  (* The description specifies additional information on the Functional_link. *)
  description : OPTIONAL text_select;
  (* The destination_port specifies the Io_port that is the data receiver. *)
  destination_port : io_port;
  (* The name specifies the word, or words, that are used to refer to the Functional_link. *)
  name : label;
  (* The source_port specifies the Io_port that is the data transmitter. *)
  source_port : io_port;

```
DERIVE
  data_on_link : data_instance := source_port.data;
WHERE
  correct_ports: ((destination_port.role = consumer) AND (source_port.role = producer));
END_ENTITY;

(* A Functional_link_allocation_relationship is the mechanism for indicating that a Functional_link_reference shall be routed onto a particular Physical_instance_reference. <note>A prerequisite for the allocation is that the allocated functional_link is allocated to a Physical_instance_reference that is part of the same System_view object.</note> *)
ENTITY functional_link_allocation_relationship;
  (* The allocated_functional_link specifies the Functional_link_reference in the Functional_link_allocation_relationship. *)
  allocated_functional_link : functional_link_reference;
  (* The allocated_to specifies the Physical_instance_reference in the relationship.
 *)
  allocated_to : physical_instance_reference;
  (* The description specifies additional textual information about the Functional_link_allocation_relationship. *)
  description : OPTIONAL text_select;
END_ENTITY;

(* A Functional_link_group is a container for grouping related Functional_link objects. <note>A functional_link_group allows a designer to refer to a group of Functional_link objects as if they constituted a single flow. One reason for using a Functional_link_group in a model is that it allows for the assignment of the same graphical properties to the group objects that are members of the Functional_link_group.</note> *)
ENTITY functional_link_group;
  (* The elements is the set of Functional_link objects that make up the Functional_link_group. *)
  elements : SET [2 : ?] OF functional_link;
  (* The name specifies the word, or words, that are yused to refer to the Functional_link_group. *)
  name : label;
END_ENTITY;

(* A Functional_link_reference is the unambiguous reference to Functional_link objects in the context of a General_functionality_instance that is further decomposed. *)
ENTITY functional_link_reference;
  (* The in_scope_of specifies the Functionality_instance_reference which defines the context of the functional_link identified by the Functional_link_reference. *)
  in_scope_of : functionality_instance_reference;
  (* The reference_functional_link specifies the Functional_link for which the Functional_link_reference is the unambiguous reference. *)
  reference_functional_link : functional_link;
END_ENTITY;

(* A Functional_reference_configuration is a collector for all Functionality_reference_composition_relationship objects that apply for a particular functional view on a system. <note number="1">A Functional_reference_configuration may be assigned to any number of systems through the use of multiple System_functional_configuration objects.</note> <note number="2">The Functional_reference_configuration provides the means for defining multiple non-functional views of single system functional architecture model.</note> *)
ENTITY functional_reference_configuration;
  (* The description specifies additional information about the Functional_reference_configuration. *)
  description : OPTIONAL text_select;
END_ENTITY;

(* A Functional_representation_relationship is the mechanism for indicating that a data_instance, used within the functional view of a specification, is acting as an alias for a Physical_instance from a physical view of the specification. <note>The Functional_representation_relationship is motivated by the need to allow for functional models to reference other elements than data types, while maintaining a rigourous structure in the model. <p>The alternative would be to allow Io_port objects to directly carry objects of other types than of Data_instance. While straightforward, this approach would lead to severe interpretation problems for tools incapable of representing items other than Data_instance objects.</p></note> *)
ENTITY functional_representation_relationship;
  (* The description specifies additional information about the Functional_representation_relationship. *)
  description : OPTIONAL text_select;
```

```
  (* The functional_representation specifies the data_instance defined as an alias for the physical_instance object defined
by the physical_element attribute of the Functional_representation_relationship. *)
  functional_representation : data_instance;
  (* The physical_element specifies the physical_instance involved in the Functional_representation_relationship. *)
  physical_element : physical_instance;
END_ENTITY;

(* A Functional_state_context is a type of Generic_state_context and the enclosure of a finite state machine. <note num-
ber="1">There is a specific Functional_state_context object as the usage of FSMs are different in structured analysis and
OO systems engineering. </note> <note number="2">The Functional_state_context may contain any number of states
and transitions, but no transition may cross the boundaries of the Functional_state_context.</note> *)
ENTITY functional_state_context
SUBTYPE OF (generic_state_context);
END_ENTITY;

(* A Functionality_allocation_relationship is the mechanism for mapping a Functionality_instance_reference to a particu-
lar Physical_instance_reference that shall realize the functionality. *)
ENTITY functionality_allocation_relationship;
  (* The allocated_functionality specifies the Functionality_instance_reference that is being allocated. *)
  allocated_functionality : functionality_instance_reference;
  (* The allocated_to specifies Physical_instance_reference to which the Functionality_allocation_relationship is allocat-
ed. *)
  allocated_to : physical_instance_reference;
  (* The description specifies additional textual information on the Functionality_allocation_relationship. *)
  description : OPTIONAL text_select;
END_ENTITY;

(* A Functionality_instance_reference is the non-ambiguous reference to a General_functionality_instance element in a
Composite_function_definition structure. <note>A Functionality_instance_reference is introduced to allow for allocation
and reference of system specific non-functional characteristics to a system functional description.</note> *)
ENTITY functionality_instance_reference
SUPERTYPE OF (persistent_storage_reference);
  (* The description specifies additional textual information on the Functionality_instance_reference. *)
  description : OPTIONAL text_select;
  (* The id specifies the identifier of the Functionality_instance_reference. *)
  id : element_identifier;
  (* The name specifies the word, or words, that are used to refer to a Functionality_instance_reference. *)
  name : label;
  (* The referenced_functionality_instance specifies the General_functionality_instance that the Functionality_instance_
reference is a reference for. *)
  referenced_functionality_instance : general_functionality_instance;
END_ENTITY;

(* A Functionality_reference_composition_relationship is the mechanism for indicating the hierarchical relationship be-
tween two Functionality_instance_reference objects. *)
ENTITY functionality_reference_composition_relationship;
  (* The child specifies the component functionality in the Functionality_reference_composition_relationship. *)
  child : functionality_instance_reference;
  (* The mirror_of specifies the Functional_decomposition_relationship for which the Functionality_reference_composi-
tion_relationship provides the unambiguous reference. *)
  mirror_of : functional_decomposition_relationship;
  (* The parent specifies the decomposed functionality in the Functionality_reference_composition_relationship. *)
  parent : functionality_instance_reference;
  (* The reference_configuration specifies the Functional_reference_configuration for which the parent and child Func-
tionality_instance_reference objects are valid. *)
  reference_configuration : functional_reference_configuration;
END_ENTITY;

(* A Functionality_reference_relationship is the mechanism for capturing binary relationships between two Functional-
ity_instance_reference objects. The purpose is to allow for capturing information.
```

<example number="11">Requirements on temporal relationships between two Functionality_instance_reference objects.</example> *)

```
ENTITY functionality_reference_relationship;
  (* The first_reference specifies the first Functionality_instance_reference object in the relationship. *)
  first_reference : functionality_instance_reference;
  (* The second_reference specifies the second Functionality_instance_reference object in the relationship. *)
  second_reference : functionality_instance_reference;
END_ENTITY;

(* A General_function_definition provides a structured definition of a function (or a process) at some level of abstraction in a system. Each General_function_definition is either a Composite_function_definition or a Leaf_function_definition. *)
ENTITY general_function_definition
ABSTRACT SUPERTYPE OF (ONEOF(composite_function_definition, leaf_function_definition));
  (* The associated_version specifies the Configuration_element_version for the associated_version.
 *)
  associated_version : configuration_element_version;
  (* The description specifies additional textual specification on the General_function_definition. *)
  description : OPTIONAL text_select;
  (* The id specifies the identifier of the id.
 *)
  id : element_identifier;
  (* The name specifies the word, or words, that are used to refer to the General_function_definition. *)
  name : OPTIONAL label;
INVERSE
  (* The port_of specifies the General_function_definition for which this is a port. This attribute is reinforced by the inverse relationship which indicates that a port_of can only point at a definition but a definition can be referred to by several ports. *)
  formal_port : SET [0 : ?] OF formal_io_port FOR port_of;
UNIQUE
  UR1: id;
END_ENTITY;

(* Each General_functionality_instance is either a Fsm_model, a Function_instance, an Io_split_join or a Persistent_storage. *)
ENTITY general_functionality_instance
ABSTRACT SUPERTYPE OF (ONEOF(fsm_model, function_instance, io_split_join, persistent_storage));
  (* The description specifies additional information about the description.
 *)
  description : OPTIONAL text_select;
INVERSE
  (* The port_of specifies the General_functionality_instance object of which the port_of is part.
 *)
  actual_port : SET [0 : ?] OF actual_io_port FOR port_of;
END_ENTITY;

(* Each General_physical_definition is either a Physical_link_definition or a Physical_node_definition. <note number="1">A General_physical_definition is a specification of a thing which, when realized, becomes a physical item that  a) can be touched b) has weight c) has a non-zero physical size.</note> <note number="2">PAS 20542 provides only an abstract view on the physical components a system is built from. The rationale for this decision is that each system physical component may have different characteristics best represented by the multitude of application protocols available in STEP  represented in detail.</note><p>The limited functionality provided is intended to allow systems engineers to map functions and flows onto components and flows onto links, to relate requirements to the components they apply to and to allow for capturing other properties of the physical component.</p> *)
ENTITY general_physical_definition
ABSTRACT SUPERTYPE OF (ONEOF(physical_link_definition, physical_node_definition));
  (* The associated_version specifies the Configuration_element_version for the General_physical_definition *)
  associated_version : configuration_element_version;
  (* The description specifies additional information about the General_physical_definition. *)
  description : OPTIONAL text_select;
  (* The id specifies the identifier of the General_physical_definition. *)
```

```
  id : element_identifier;
  (* The name specifies the word, or words, that are used to refer to the General_physical_definition. *)
  name : OPTIONAL label;
INVERSE
  (* The port_of specifies the General_physical_defintion object for which the Formal_physical_port provides part of the
interface. *)
  formal_port : SET [0 : ?] OF formal_physical_port FOR port_of;
UNIQUE
  UR1: id;
END_ENTITY;

(* A Generic_state_context is the enclosure of a finite state machine or statechart. Each Generic_state_context is a
Functional_state_context. *)
ENTITY generic_state_context ABSTRACT SUPERTYPE;
  (* The associated_version specifies the the Configuration_element_version for the Generic_state_context. *)
  associated_version : configuration_element_version;
  (* The description specifies additional information on the Generic_state_context. *)
  description : OPTIONAL text_select;
  (* The id specifies the identifier of the Generic_state_context. *)
  id : element_identifier;
  (* The name specifies the word, or words, that are used to refer to the Generic_state_context. *)
  name : OPTIONAL label;
  (* The original_representation specifies how the state machine enclosed by the Generic_state_context was presented.
Where applicable, one of the following shall be used to represent the original_representation:<ul><li>table: The state
machine was  presented in a table structure in the originating tool;</li><li>graph: The state machine was presented using
a graphical notation in the originating tool.</li></ul> *)
  original_representation : label;
  (* The state_machine_model specifies the type of state machine enclosed by the Generic_state_context. Where appli-
cable, one of the following values shall be used:<ul><li>moore: The enclosed state machine is flat. No states are decom-
positioned. The output alphabet from the state machine is generated by Fsm_generic_state objects;</li> <li>mealy: The
enclosed state machine is flat. No states are decompositioned. The output alphabet from the state machine is generated
by Fsm_state_transition and Fsm_initial_state_transition objects;</li> <li>hierarchical: The states in the enclosed state
machine may be composed of other states. The output alphabet may be generated by both Fsm_generic_state, Fsm_
state_transition and Fsm_initial_state_transition objects.</li></ul> *)
  state_machine_model : label;
END_ENTITY;

(* A Graphics_link is a type of Visual_element and a representation of a open graphical figure spanning a list of Graph-
ics_point objects. <note> The graphical information is not directly associated with the objects that are represented by the
Graphics_link object as the objects may have totally different representations if they appear in more than one context.</
note> *)
ENTITY graphics_link
SUBTYPE OF (visual_element);
  (* The associated_with specifies the Configuration_element for which the Graphics_link provides the visual layout. *)
  associated_with : link_select;
  (* The point specifies the list of co-ordinates of the Graphics_link. *)
  point : LIST [2 : ?] OF graphics_point;
END_ENTITY;

(* A graphics_node is a type of Visual_element and a representation of a closed rectangle used for presenting the posi-
tion of an object. *)
ENTITY graphics_node
SUBTYPE OF (visual_element);
  (* The associated_with specifies the object for which the associated_with provides visual placement information.
 *)
  associated_with : node_select;
  (* The bottom_right specifies the Graphics_point object that provides positioning information for the bottom right corner
of the node.
 *)
  bottom_right : graphics_point;
  (* The top_left specifies the Graphics_point object that provides positioning information for the upper left corner of the node.
 *)
```

```
  top_left : graphics_point;
END_ENTITY;

(* A Graphics_point is a point in the coordinate plane. <note number="1">The coordinate plane is valid in the range {0&lt;= x, y &lt;= 1}. No information on the aspect ratio is given.</note> <note number="2">This model is chosen since the dimensions of elements in a graphical view does not have any meaning in the domain. If the aspect ratio is set up in such a way the image is perceived to be distorted it is up to the viewer to adjust the aspect ratio.</note> *)
ENTITY graphics_point;
  (* The x_coordinate specifies the horizontal position in the coordinate plane. A value of 0 is the leftmost position and a value of 1 is the rightmost position. *)
  x_coordinate : REAL;
  (* The y_coordinate specifies the vertical position in the coordinate plane. A value of 0 is the topmost position and a value of 1 indicates to position furthest down. *)
  y_coordinate : REAL;
END_ENTITY;

(* A Graphics_view is the collection of all items carrying graphical information that apply to the contents of a General_function_definition, a General_physical_definition, a Functional_state_context, or a Fsm_state. *)
ENTITY graphics_view;
  (* The definition_for specifies the object for which the Graphics_view is valid. *)
  definition_for : definition_select;
INVERSE
  (* The transformation_for specifies graphics_view for which the  Coordinate_translation_information provides information. *)
  coordinate_definition : SET [0 : ?] OF coordinate_translation_information FOR transformation_for;
  (* The top_view specifies the Graphics_view object for which the Multi_level_view is the top_view. *)
  root_view : SET [0 : 1] OF multi_level_view FOR top_view;
  (* The view specifies the Graphics_view for which the Visual_element is defined. *)
  view_element : SET [1 : ?] OF visual_element FOR view;
END_ENTITY;

(* A Hibound_integer_interval is an Integer_interval that has an upper bound that constrains the sub-set of integer values it contains. *)
ENTITY hibound_integer_interval
SUBTYPE OF (integer_interval);
  (* The high_index specifies the highest value that a Data_instance of this type can take. *)
  high_index : INTEGER;
END_ENTITY;

(* A Hibound_real_interval is a type of Real_interval that has an upper bound that constrains the sub-set of real values it contains. *)
ENTITY hibound_real_interval
SUBTYPE OF (real_interval);
  (* The high_closure specifies whether or not the high_index is included in the maths space. TRUE = included, FALSE = excluded. *)
  high_closure : BOOLEAN;
  (* The high_index specifies the highest value that a Data_instance of this type can take. *)
  high_index : REAL;
END_ENTITY;

(* An Implied_external_interaction is the relationship between a Requirement_instance object and a Function_instance or a Physical_instance object such that the existence of the functional or physical element is implied in the requirement. *)
ENTITY implied_external_interaction;
  (* The associated_requirement specifies the Requirement_instance in the relationship. *)
  associated_requirement : requirement_instance;
  (* The implied_external_element specifies the Configuration_element implied by the requirement. *)
  implied_external_element : external_element_select;
INVERSE
  (* The transfer specifies the Implied_external_interaction object for which the Data_transfer is valid. *)
  associated_data : SET [0 : 1] OF data_transfer FOR transfer;
END_ENTITY;
```

(* An Infinite_cardinality is the representation of a positive infinite real number. *)
ENTITY infinite_cardinality;
END_ENTITY;

(* An Initial_state_transition_specification_assignment is the mechanism for relating a Textual_specification defining the actions carried out when the Fsm_initial_state_transition is activated. *)
ENTITY initial_state_transition_specification_assignment;
(* The assigned_to specifies the Fsm_initial_state_transition the specification is associated with. *)
assigned_to : fsm_initial_state_transition;
(* The specification specifies the Textual_specification assigned by the Initial_state_transition_specification_assignment. *)
specification : textual_specification;
END_ENTITY;

(* An Integer_data_type_definition is a type of Elementary_maths_space that includes all integer values. *)
ENTITY integer_data_type_definition
SUBTYPE OF (elementary_maths_space);
END_ENTITY;

(* An Integer_interval is a type of Elementary_maths_space that is a bounded sub-set of all integer values. *)
ENTITY integer_interval
ABSTRACT SUPERTYPE OF (ONEOF(finite_integer_interval, hibound_integer_interval, lobound_integer_interval))
SUBTYPE OF (maths_space);
END_ENTITY;

(* An Io_buffer is a modifier to the temporal behaviour of an Actual_io_port. <note>If an Io_buffer is assigned to an Actual_io_port producing data then data produced is discretized and can only be read once. If an Io_buffer is assigned to an Actual_io_port consuming data, the data received in discretized and stored until read from the port. The behaviour of the buffer is further specified by the continuously_active attribute. For consuming ports the Io_buffer may control the Io_buffer to be active continuously or only in the intervals the function the Io_port is assigned to is active. The continuously_active attribute does not modify the semantics of Actual_io_port objects producing data.</note> *)
ENTITY io_buffer;
(* The assigned_to specifies the Actual_io_port object to which the Io_buffer is assigned. *)
assigned_to : actual_io_port;
(* The continuously_active specifies whether the Io_buffer is active continuously or not. *)
continuously_active : BOOLEAN;
(* The synchronization_semantics specifies the syncronization conditions enforced by the Io_buffer. The synchronization_semantics is one of the following:
synchronous: The function sending information via the Actual_io_port to which the Io_buffer is connected shall remain active until the information have been consumed by the receiver; asynchronous: The function sending infromation via the Actual_io_port to which the Io_buffer is connected may terminate without constraints on whether the information is received or not. *)
synchronisation_semantics : buffer_synchronisation_enumeration;
END_ENTITY;

(* An Io_composition_port is a type of Io_port and a decomposition of a TBD. <note number=»1»>The semantics of the entity Functional_link is to convey a Data_instance from the producing Io_port to the consuming Io_port. The Data_instance shall be the same on both ports. For this reason there is a requirement to include support for decomposing an Io_port such that Functional_link objects may read or write sub sets of the data on an Io_port. The Io_composition_port is the means for taking a Data_instance of type composite_data_type on an Io_port and create an intermediate port whose data is a sub-set of the port it is connected to and is either the source or target of a Functional_link.</note> <note number=»2»>Due to the data abstraction facility included in the ARM, composition can only be made one level at a time. To specify that a Functional_link produces or consumes a Data_instance that is part of a data type structure at level n, then an Io_composition_port is needed to accurately define this.</note> <note number=»3»>The Io_composition_port corresponds to the dot-notation used in most imperative programming languages. For example: a:=a_data_type.a_data_field. </note> *)
ENTITY io_composition_port
SUBTYPE OF (io_port);
(* The is_alias_for specifies the Data_instance object of the port_of attribute that is assigned to the Io_composition_port. The Data_instance shall be at one level down in the data abstraction hierarchy of the data type of the Io_port object that is specified by the port_of attribute. <note number=»4»>This corresponds to the dot notation of programming languages. Assume a composite data type: type a_type = record a_field : another_type; another_

field : another_type; end type; </note> <note number=»5»>The is_alias_for identifies the field of a composite data type that is assigned to the data variable (Data_instance) of the is_alias_for.</note> <note number=»6»>The data attribute is inherited from the entity Io_port.</note> *)
  is_alias_for : data_instance;
  (* The port_of specifies the Io_port of which the Io_composition_port is a part. *)
  port_of : data_composition_select;
DERIVE
  SELF\io_port.role : port_data_relation := port_of.role;
UNIQUE
  UR1: port_of, SELF\io_port.io_port_number;
END_ENTITY;

(* An Io_port is a part of the interface of a functional object. <note number=»1»>An io_port is an abstract entity and shall not be instantiated.</note> *)
ENTITY io_port
ABSTRACT SUPERTYPE OF (ONEOF(actual_io_port, control_io_port, formal_io_port, io_composition_port));
  (* The data specifies the Data_instance that defines the format of the element in the interface. *)
  data : data_instance;
  (* The io_port_number in combination with the port_type attribute and the port_of attribute shall be unique within the set of ports attached to an element. The io_port_number is the identifier of the Io_port. *)
  io_port_number : INTEGER;
  (* The port_type specifies the nature of the Io_port. The port_type is one of the following: input: The port is handling information originating from elements outside the context of the functionality the Io_port is an element of the interface; output: The port is handling information originating from elements inside the context of the functionality the Io_port is an element of the interface; control: The port is a specialisation of an input such that the information provided by this part of the interface is a prerequisite for the activation of the element the Io_port is part of the interface of.
<note number=»2»>The control is provided to allow for representation of IDEF0 control inputs. Semantically there is no difference between a control and an input port_type.</note> mechanism: The port is a specialisation of an input such that the resources provided by this part of the interface is a prerequisite for the activation of the element the Io_port is part of the interface of.
<note number=»3»>The mechanism is provided to allow for representation of IDEF0 control inputs. Semantically there is no difference between a mechanism and an input port_type.</note> *)
  port_type : port_type;
  (* The role specifies the interaction of the Io_port. The role may either produce information or consume information. The role is one of the following: *)
  role : port_data_relation;
END_ENTITY;

(* An Io_port_binding is the mapping from one Actual_io_port object to a Formal_io_port object. <note number=»1»>In programming languages, this corresponds to the mapping between formal and actual parameters in a function call.</note> <note number=»2»>An Io_port_binding is valid only if the Actual_io_port is connected to a Function_instance as its definition attribute has the General_function_definition object that the Formal_io_port is a port_of.</note> *)
ENTITY io_port_binding;
  (* The actual_port specifies the Actual_io_port object in the mapping. *)
  actual_port : actual_io_port;
  (* The formal_port specifies the Formal_io_port object in the mapping. *)
  formal_port : formal_io_port;
WHERE
  WR1: (SELF.formal_port.data :<>: SELF.actual_port.data);
  WR2: formal_port.role <> actual_port.role;
  WR3: correct_binding(SELF);
END_ENTITY;

(* An Io_split_join is a type of General_functionality_instance and a means for splitting up the components of a Functional_link object carrying composite data into several Functional_link objects carrying basic data or merging a Functional_link carrying basic data to a Functional_link carrying composite data. <note>The future of this entity is not clear. It may be removed from the model and the functionality may be replaced by an ordinary function.</note> *)
ENTITY io_split_join
SUBTYPE OF (general_functionality_instance);
END_ENTITY;

```
(* The Issue_source_relationship is the mechanism for relating the elements that played a part in raising an issue to a
critical_issue. *)
ENTITY issue_source_relationship;
  (* The description specifies additional information on the Issue_source_relationship. *)
  description : OPTIONAL text_select;
  (* The issue specifies the Critical_issue that the issue_source is assigned to. *)
  issue : critical_issue;
  (* The issue_source specifies the element that raised the issue. *)
  issue_source : issue_source_select;
END_ENTITY;

(* An Issue_system_assignment is the mechanism for relating a Critical_issue to the System_view or System_instance
that the issue is relevant for. *)
ENTITY issue_system_assignment;
  (* The description specifies additional information on the Issue_system_assignment. *)
  description : OPTIONAL text_select;
  (* The identified_system specifies the System_view or System_instance in the relationship. *)
  identified_system : system_select;
  (* The issue specifies the Critical_issue in the relationship. *)
  issue : critical_issue;
END_ENTITY;

(* A Justification is a textual description of the reasons for the existence of, or design rationale for, part of a system spec-
ification. *)
ENTITY justification;
  (* The assigned_to specifies the object to which the Justification applies. *)
  assigned_to : justification_assignment_select;
  (* The justification_text specifies the textual comment on the object referenced by the assigned_to attribute. *)
  justification_text : text_select;
  (* The role is the word, or words, that are used to describe the nature of the Justification. <example num-
ber=»12»>Roles for a Justification might be «comment», «rationale», or «justification».</example> *)
  role : label;
END_ENTITY;

(* A Justification_relationship represents some relationships between two Justification objects. <note>The exact se-
mantics of the relationship depend on the value of the relationship_type attribute.</note> <example number=»13»>The
relationship between two Justification objects can arise from a situation where the choice of one alternative for a given
part of the design may impact on further choices making potential alternatives invalid.</example> *)
ENTITY justification_relationship;
  (* The description specifies additional textual information about the Justification_relationship. *)
  description : OPTIONAL text_select;
  (* The related specifies the second Justification in the relationship. *)
  related : justification;
  (* The relating specifies the first Justification in the relationship. *)
  relating : justification;
  (* The relationship_type specifies the semantics of the relationship.  *)
  relationship_type : label;
END_ENTITY;

(* A Leaf_function_definition is a type of General_function_definition that is not decomposed any further. <note num-
ber=»1»>A function hierarchy is always terminated by a Leaf_function_definition. In a later release of the system under
design, a Leaf_function might become a composite_function_definition. This case is handled by the use of Configura-
tion_element_version items identifying two different releases of the same object.</note> *)
ENTITY leaf_function_definition
SUBTYPE OF (general_function_definition);
  (* The definition specifies the text that describes what (the service, transformation, state change, etc.) is performed by
the function.The definition specifies the Textual_specification object that defines the specification and specification lan-
guage for the Leaf_function_definition. <note number=»2»>The standard does not feature any means to ensure consis-
tency and correctness of the content of the definition.</note> *)
  definition : textual_specification;
```

```
  (* The function_type specifies the word, or words, used to refer to the type of Leaf_function_definition. The function_
type shall be interpreted in the generic sense as it specifies a class of functions. <note number=»2»>Possible values for
function_type might be «mathematical», «transformational», or «trigonometric».</note> <note number=»3»>The PAS
20542 group shall provide a definition of preferred values for this attribute.</note> *)
  function_type : OPTIONAL label;
  (* The predefined specifies whether the predefined object has a formally defined semantics in the exporting tool. <note
number=»4»>This attribute is a simple method of allowing for tools having large libraries of predefined functions to ex-
change information without having to map too much information onto user defined functions.</note> *)
  predefined : BOOLEAN;
END_ENTITY;

(* A Lobound_integer_interval is a type of Integer_interval that has a lower bound that constrains the sub-set of integer
values it contains. *)
ENTITY lobound_integer_interval
SUBTYPE OF (integer_interval);
  (* The low_index specifies the lowest value that a Data_instance of this type can take. *)
  low_index : INTEGER;
END_ENTITY;

(* A Lobound_real_interval is a type of Real_interval that has a lower bound that constrains the sub-set of real values it
contains. *)
ENTITY lobound_real_interval
SUBTYPE OF (real_interval);
  (* The low_closure specifies whether or not the low_index is included in the maths space. TRUE = included, FALSE =
excluded. *)
  low_closure : BOOLEAN;
  (* The low_index specifies the lowest value that a Data_instance of this type can take. *)
  low_index : REAL;
END_ENTITY;

(* A Logical_data_type_definition is a type of Elementary_maths_space whose value range is: TRUE, FALSE or UN-
KNOWN. <note>Logical_data_type_definition is not modelled as a Finite_space as the concept of a Logical_data_type_
definition has a more precise meaning than a Finite_space would allow.</note> *)
ENTITY logical_data_type_definition
SUBTYPE OF (elementary_maths_space);
END_ENTITY;

(* A Maths_space is a set of mathematical values that a Data_instance of that space can take. <note>For the purposes
of data typing, strings are assumed to be mathematical values.</note> *)
ENTITY maths_space
ABSTRACT SUPERTYPE OF (ONEOF(elementary_maths_space, finite_space, integer_interval, real_interval));
  (* The description specifies additional textual information on the Maths_space. *)
  description : OPTIONAL text_select;
  (* The id specifies the identifier of the Maths_space. *)
  id : element_identifier;
  (* The name specifies the word, or words, that are used to refer to a Maths_space. *)
  name : OPTIONAL label;
UNIQUE
  UR1: id;
END_ENTITY;

(* A Model_defined_requirement_definition is a type of Requirement_definition where the required capability is stated
using some kind of a model expressed in PAS 20542 format. <note>The model may be in any format supported by PAS
20542.</note> *)
ENTITY model_defined_requirement_definition
SUBTYPE OF (requirement_definition);
  (* The assigned_model specifies the model that defines the requirement. *)
  assigned_model : system_view;
  (* The model_relevance specifies in text how the requirement expressed by the model shall be interpreted or which part
of the assigned_model expresses the requirement. *)
  model_relevance : OPTIONAL text_select;
END_ENTITY;
```

```
(* A Multi_level_view is a set of Graphics_view objects that provide presentation information for multiple levels of decomposition within a single view. <note number="1">The Multi_level_view shall not be instantiated if no hierarchy of Graphics_view objects exist for a particular Graphics_view object.</note> <note number="2">The Multi_level_view is provided for representing multiple levels of decomposition of a design diagram.</note> *)
ENTITY multi_level_view;
  (* The description specifies additional textual information on the Multi_level_view. *)
  description : OPTIONAL text_select;
  (* The reference_name specifies the word, or words, that are used ot refer to the Multi_level_view. *)
  reference_name : OPTIONAL label;
  (* The top_view specifies the Graphics_view object for which the Multi_level_view is the top_view. *)
  top_view : graphics_view;
END_ENTITY;

(* A Name_binding is the mapping relationship between a Function_reference object in the name space (part of the hierarchical breakdown) of a Composite_function_definition object and a State_function_interaction_port object. <note>The Name_binding maps the reference to a Function_instance object, used within a Functional_state_context object, to the actual Function_instance object. To accomplish this two port entities are introduced:<ol><li>Function_reference: provides the reference to a Function_instance object .</li><li>State_function_interaction_port: defines the local reference within a Functional_state_context.</li></ol></note> *)
ENTITY name_binding;
  (* The actual_port specifies the Function_reference in the relationship. *)
  actual_port : function_reference;
  (* The formal_port specifies the State_function_interaction_port in the relationship. *)
  formal_port : state_function_interaction_port;
END_ENTITY;

(* A Nominal_value is a type of Value_with_unit and a quantity expressed with a numerical value and a unit. *)
ENTITY nominal_value
SUBTYPE OF (value_with_unit);
  (* The value_component specifies the quantity of the Nominal value. <note number="2">The definition is from AP-214 ARM.</note> *)
  value_component : NUMBER;
INVERSE
  (* The limited_value specifies the Nominal_value further defined by the Plus_minus_bound. *)
  limitation : SET [0 : 1] OF plus_minus_bounds FOR limited_value;
END_ENTITY;

(* A Non_digital_document is a type of Document_reference that provides a reference to a document that is in non-digital form. *)
ENTITY non_digital_document
SUBTYPE OF (documentation_reference);
  (* The location specifies the physical location where the Non_digital_document can be obtained. *)
  location : label;
END_ENTITY;

(* An Oo_action is an executable statement that forms an abstraction of a computational procedure. *)
ENTITY oo_action
SUPERTYPE OF (ONEOF(oo_call_action, oo_create_action, oo_send_action));
  (* The description specifies additional information about the Oo_action. *)
  description : OPTIONAL text_select;
  (* The is_asynchronous specifies whether the Oo_action is executed synchronously or asynchronously. *)
  is_asynchronous : BOOLEAN;
  (* The name specifies the word, or words, that are used to refer to the Oo_action. *)
  name : label;
  (* The script defines the Textual_specification that defines the specification and the specification language for the Oo_action. *)
  script : textual_specification;
END_ENTITY;

(* An Oo_action_state  is a type of Cb_place that represents the execution of an atomic action, typically the invocation of an Oo_operation activation. *)
```

```
ENTITY oo_action_state
SUBTYPE OF (cb_place);
END_ENTITY;

(* An Oo_action_state_transition is a type of Cb_transition that specifies a temporal relationship between two Oo_action_state objects. *)
ENTITY oo_action_state_transition
SUBTYPE OF (cb_transition);
END_ENTITY;

(* An Oo_action_temporal_relationship is a relationship between two Oo_action objects that specifies their temporal order. *)
ENTITY oo_action_temporal_relationship;
  (* The description specifies additional information about the Oo_action_temporal_relationship. *)
  description : OPTIONAL text_select;
  (* The predecessor specifies the preceding Oo_action. *)
  predecessor : oo_action;
  (* The successor specifies the subsequent Oo_action. *)
  successor : oo_action;
END_ENTITY;

(* An Oo_actor is a coherent set of roles that users of Oo_use_case objects play when interacting with them. *)
ENTITY oo_actor;
  (* The description specifies additional information about the Oo_actor. *)
  description : OPTIONAL text_select;
  (* The id specifies the identifier of the Oo_actor. *)
  id : element_identifier;
  (* The name specifies the word, or words, that are used to refer to the Oo_actor. *)
  name : label;
  (* The namespace specifies the element to which the Oo_actor belongs. The names of elements belonging to the same namespace element must be unique within this set. *)
  namespace : OPTIONAL oo_namespace_select;
END_ENTITY;

(* An Oo_argument is a specific initial_value corresponding to an Oo_parameter object. *)
ENTITY oo_argument;
  (* The action specifies the Oo_action object to which the action belongs. *)
  action : oo_action;
  (* The initial_value specifies the actual value of initial_value. *)
  initial_value : text_select;
END_ENTITY;

(* An Oo_association is a type of Oo_generic_association that provides a semantic relationship between elements of the system. *)
ENTITY oo_association
SUBTYPE OF (oo_generic_association);
WHERE
  WR1: SELF\oo_generic_association.reading_direction IN SELF\oo_generic_association.connection;
END_ENTITY;

(* An Oo_association_class is an object that has both Oo_class_association and Oo_class properties. It represents a set of Oo_object objects that are coincidently instantiated with the association instance. *)
ENTITY oo_association_class;
  (* The class specifies the related Oo_class object. *)
  class : oo_class;
  (* The class_association specifies the related Oo_association object. *)
  class_association : oo_generic_association;
  (* The description specifies additional information about the Oo_association_class. *)
  description : OPTIONAL text_select;
END_ENTITY;
```

```
(* An Oo_association_end is a type of Oo_generic_association_end that specifies an end point of an Oo_association. *)
ENTITY oo_association_end
SUBTYPE OF (oo_generic_association_end);
  SELF\oo_generic_association_end.association : oo_association;
END_ENTITY;

(* An Oo_association_end_classifier_relationship is a relationship between an Oo_association_end and a classifier specifying the Oo_association_end_classifier_relationship. *)
ENTITY oo_association_end_classifier_relationship;
  (* The association_end specifies the qualified Oo_association_end object. *)
  association_end : oo_generic_association_end;
  (* The description specifies additional information about the Oo_association_end_classifier_relationship *)
  description : OPTIONAL text_select;
  (* The specification defines the specifying classifier. *)
  specification : oo_extended_classifier_select;
END_ENTITY;

(* An Oo_association_end_qualifier_association is a relationship between an Oo_association_end object and an Oo_attribute object. *)
ENTITY oo_association_end_qualifier_association;
  (* The association_end specifies the qualified Oo_association_end object. *)
  association_end : oo_generic_association_end;
  (* The qualifier specifies the qualifying Oo_attribute object. *)
  qualifier : OPTIONAL oo_attribute;
END_ENTITY;

(* An Oo_association_end_role is a type of Oo_generic_association_end that specifies an end point of an Oo_association_role. *)
ENTITY oo_association_end_role
SUBTYPE OF (oo_generic_association_end);
  (* The base specifies the Oo_association_end that the Oo_association_end_role is based on. *)
  base : OPTIONAL oo_association_end;
  (* The collaboration_multiplicity specifies the number of target instances that may be associated with a single source instance across the given Oo_association_role specified by an Oo_association_end_role. *)
  collaboration_multiplicity : cardinality_association_select;
  (* The role_type specifies the semantics of the Oo_association_end_role. *)
  role_type : oo_classifier_role;
  SELF\oo_generic_association_end.association : oo_association_role;
END_ENTITY;

(* An Oo_association_role is a type of Oo_generic_association that specifies a specific behavior of an Oo_association in a particular context. *)
ENTITY oo_association_role
SUBTYPE OF (oo_generic_association);
  (* The base specifies the Oo_association that the Oo_association_role is based on. *)
  base : OPTIONAL oo_association;
  (* The multiplicity specifies the number of target instances that may be associated with a single source instance across the given Oo_association specified by base. *)
  multiplicity : cardinality_association_select;
WHERE
  WR1: SELF\oo_generic_association.reading_direction IN SELF\oo_generic_association.connection;
END_ENTITY;

(* An Oo_attribute is a named part of a classifier that specifies the range of values that instances of the classifier may hold. *)
ENTITY oo_attribute;
  (* The definition specifies the type definition of the Oo_attribute. *)
  definition : oo_extended_classifier_select;
  (* The description specifies additional information about the Oo_attribute. *)
  description : OPTIONAL text_select;
  (* The id specifies the identifier of the Oo_attribute. *)
```

```
  id : element_identifier;
  (* The name specifies the word, or words, that are used ot refer to the Oo_attribute. *)
  name : label;
  (* The owner specifies the classifier to which the Oo_attribute belongs. *)
  owner : oo_extended_classifier_select;
  (* The visibility specifies how the Oo_attribute may be seen outside its enclosing namespace. Where applicable the
following values should be used:<ul><li>private: Specifies that the Oo_attribute is only visible to elements within the
same namespace;</li> <li>protected: Specifies that the Oo_attribute is only visible to elements of the same namespace
and namespaces that are inherited from this;</li> <li>public: Specifies that the Oo_attribute can be accessed from other
elements including those outside the namespace of the Oo_attribute</li></ul> *)
  visibility : label;
END_ENTITY;

(* An Oo_attribute_instance is an instance of a Oo_attribute. *)
ENTITY oo_attribute_instance;
  (* The attribute_value specifies the actual value of an Oo_attribute_instance. *)
  attribute_value : label;
  (* The definition specifies the Oo_attribute that provides the definition for the Oo_attribute_instance. *)
  definition : oo_attribute;
  (* The owner specifies the classifier to which the Oo_attribute_instance belongs. *)
  owner : oo_extended_classifier_select;
END_ENTITY;

(* An Oo_attribute_link_end_association allocates Oo_attribute_instance objects to Oo_link_end objects. *)
ENTITY oo_attribute_link_end_association;
  (* The attribute_instance specifies the Oo_attribute_instance object that is attached the the Oo_link_end object speci-
fied by link_end. *)
  attribute_instance : oo_attribute_instance;
  (* The description specifies additional information about the Oo_attribute_link_end_association. *)
  description : text_select;
  (* The link_end specifies the Oo_link_end object to which the attribute_instance is attached. *)
  link_end : oo_link_end;
END_ENTITY;

(* An Oo_behavioral_feature ia a dynamic feature of a classifier. *)
ENTITY oo_behavioural_feature
ABSTRACT SUPERTYPE OF (ONEOF(oo_method, oo_operation, oo_reception));
  (* The name specifies the word, or words, that are used to refer to the Oo_behavioral_feature. *)
  name : label;
  (* The owner specifies the classifier to which the Oo_behavioral_feature belongs. *)
  owner : oo_classifier_select;
  (* The visibility specifies how the Oo_behavioural_feature may be seen outside its enclosing namespace. Where
applicable the following values should be used:<ul><li>private: Specifies that the Oo_behavioural_feature is only visible
to elements within the same namespace;</li> <li>protected: Specifies that the Oo_behavioural_feature is only visible to
elements of the same namespace and namespaces that are inherited from this;</li><li>public: Specifies that the Oo_be-
havioural_feature can be accessed from other elements including those outside the namespace of the Oo_behavioural_
feature.</li></ul> *)
  visibility : label;
END_ENTITY;

(* An Oo_call_action is a type of Oo_action that specifies the call of an operation specified as Oo_operation object. *)
ENTITY oo_call_action
SUBTYPE OF (oo_action);
  (* The operation specifies the Oo_operation that is being called. *)
  operation : oo_operation;
END_ENTITY;

(* An Oo_class is a basic object-oriented notion representing a set of Oo_object elements sharing the same collection
of common features such as Oo_attribute, Oo_operation, Oo_method and relationships in the form of Oo_association,
Oo_generalization, Oo_dependency, Oo_extension or Oo_inclusion. *)
ENTITY oo_class;
```

```
  (* The associated_version specifies the Configuration_element_version for the Oo_class. *)
  associated_version : configuration_element_version;
  (* The description specifies additional information about the Oo_class. *)
  description : OPTIONAL text_select;
  (* The id specifies the identifier of the Oo_class. *)
  id : element_identifier;
  (* The is_active specifies whether an instance of an Oo_class in the form of an Oo_object owns a thread of control. If
the value is TRUE, the Oo_object instance of the is_active has its own thread of control and may run concurrently with
other active instances of Oo_class. If the value is FALSE, the Oo_operation calls are executed under control of the call-
ing active Oo_object. *)
  is_active : BOOLEAN;
  (* The name specifies the word, or words, that are used to refer to the Oo_class. *)
  name : label;
  (* The namespace specifies the element to which the Oo_class belongs. The names of elements belonging to the same
namespace element must be unique within this set. *)
  namespace : OPTIONAL oo_namespace_select;
  (* The visibility specifies how the Oo_class may be seen outside its enclosing namespace. Where applicable the
following values should be used:<ul><li>private: Specifies that the Oo_class is only visible to elements within the same
namespace;</li> <li>public: Specifies that the Oo_class can be accessed from other elements including those outside
the namespace of the Oo_class;</li> <li>protected: Specifies that the Oo_class is only visible to elements of the same
namespace and namespaces that are inherited from this.</li></ul> *)
  visibility : label;
END_ENTITY;

(* An Oo_classifier_role is a specific behaviour of a classifier in a particular context. *)
ENTITY oo_classifier_role;
  (* The multiplicity specifies the number of Oo_classifier_role objects that may be associated with a single sender of a
referencing Oo_message. *)
  multiplicity : cardinality_association_select;
INVERSE
  (* The role_type specifies the semantics of the Oo_association_end_role. *)
  association_end_role : SET [0 : ?] OF oo_association_end_role FOR role_type;
END_ENTITY;

(* An Oo_collaboration specifies the way in which different elements are to be used in accomplishing a particular task. *)
ENTITY oo_collaboration;
  (* The description specifies additional information about the Oo_collaboration. *)
  description : OPTIONAL text_select;
  (* The id specifies the identifier of the Oo_collaboration. *)
  id : element_identifier;
  (* The name specifies the word, or words, that are used to refer to the Oo_collaboration. *)
  name : label;
  (* The representing specifes the element whose behaviour is described by the Oo_collaboration. *)
  representing : oo_classifier_or_operation_select;
END_ENTITY;

(* An Oo_component is a distributable piece of implementation of a system. *)
ENTITY oo_component;
  (* The description specifies additional information about the Oo_component. *)
  description : OPTIONAL text_select;
  (* The id specifies the identifier of the Oo_component. *)
  id : element_identifier;
  (* The name specifies the word, or words, that are used to refer to the Oo_component. *)
  name : label;
  (* The namespace specifies the element to which the Oo_component belongs. The names of elements belonging to the
same namespace element must be unique within this set. *)
  namespace : OPTIONAL oo_namespace_select;
  (* The visibility specifies how the Oo_component may be seen outside its enclosing namespace. Where applicable the
following values should be used:<ul><li>private: Specifies that the Oo_component is only visible to elements within the
same namespace;</li> <li>protected: Specifies that the Oo_component is only visible to elements of the same name-
```

space and namespaces that are inherited from this; public: Specifies that the Oo_component can be accessed from other elements including those outside the namespace of the Oo_component. *)
  visibility : label;
END_ENTITY;

(* An Oo_component_allocation is the mechanism for allocating a Oo_component object to a Physical_instance_reference object. *)
ENTITY oo_component_allocation;
  (* The deployment_location specifies the Physical_instance_reference object on which the Oo_component object specified by resident resides. *)
  deployment_location : physical_instance_reference;
  (* The description specifies additional information about the Oo_component_allocation. *)
  description : OPTIONAL text_select;
  (* The resident specifies the deployed Oo_component object. *)
  resident : oo_component;
END_ENTITY;

(* An Oo_constraint specifies a semantic condition or restriction. *)
ENTITY oo_constraint;
  (* The body specifies the Textual_specification object that defines the specification and the specification language for the Oo_constraint. *)
  body : textual_specification;
END_ENTITY;

(* An Oo_constraint_model_element_relationship is the allocation of an Oo_constraint to an element. *)
ENTITY oo_constraint_model_element_relationship;
  (* The constraint specifies the constraining Oo_constraint. *)
  constraint : oo_constraint;
  (* The model_element specifies the constrained element. *)
  model_element : oo_model_element_select;
END_ENTITY;

(* An Oo_create_action is a type of Oo_action that specifies the creation of a classifier. *)
ENTITY oo_create_action
SUBTYPE OF (oo_action);
  (* The instantiation specifies that the classifier has been created by the Oo_create_action. *)
  instantiation : oo_classifier_select;
END_ENTITY;

(* An Oo_dependency is a relationship between two elements, in which a change to the supplier element will effect the client element. *)
ENTITY oo_dependency;
  (* The client specifies the dependent element of the Oo_dependency. *)
  client : oo_model_element_select;
  (* The description specifies additional information about the Oo_dependency. *)
  description : text_select;
  (* The supplier specifies the independent element of the Oo_dependency. *)
  supplier : oo_model_element_select;
END_ENTITY;

(* An Oo_element_import is the use of a model_element in an Oo_package. *)
ENTITY oo_element_import;
  (* The alias_name specifies an alternative name for the imported model_element that can be used to reference the model_element. *)
  alias_name : OPTIONAL label;
  (* The container specifies the Oo_package that imports the model_element through the container. *)
  container : oo_package;
  (* The model_element specifies the element that is imported. *)
  model_element : oo_model_element_select;
  (* The name specifies the word, or words, that are used to refer to the Oo_element_import. *)
  name : label;

```
  (* The visibility specifies how the Oo_element_import may be seen outside its enclosing namespace. Where applicable
the following values should be used:<ul><li>private: Specifies that the Oo_element_import is only visible to elements
within the same namespace;</li> <li>protected: Specifies that the Oo_element_import is only visible to elements of the
same namespace and namespaces that are inherited from this;</li> <li>public: Specifies that the Oo_element_import can
be accessed from other elements including those outside the namespace of the Oo_element_import.</li></ul> *)
  visibility : label;
END_ENTITY;

(* An Oo_element_residence identifies the Oo_component object in which an implementation element resides. *)
ENTITY oo_element_residence;
  (* The description specifies additional information about the Oo_element_residence. *)
  description : OPTIONAL text_select;
  (* The implementation_location specifies the Oo_component on which the element resides. *)
  implementation_location : oo_component;
  (* The resident specifies the element residing on the Oo_component specified by implementation_location. *)
  resident : oo_model_element_select;
  (* The visibility specifies how the Oo_element_residence may be seen outside its enclosing namespace. Where
applicable the following values should be used:<ul><li>private: Specifies that the Oo_element_residence is only visible
to elements within the same namespace;</li> <li>protected: Specifies that the Oo_element_residence is only visible to
elements of the same namespace and namespaces that are inherited from this;</li> <li>public: Specifies that the Oo_el-
ement_residence can be accessed from other elements including those outside the namespace of the Oo_element_resi-
dence.</li></ul> *)
  visibility : label;
END_ENTITY;

(* An Oo_extension relates an extension Oo_use_case to a base Oo_use_case. *)
ENTITY oo_extension;
  (* The base specifies the original Oo_use_case object in the Oo_extension. *)
  base : oo_use_case;
  (* The condition defines the Textual_specification object that defines a condition that has to be fulfilled in order for the
extension to take place. *)
  condition : text_select;
  (* The extension specifies the extending Oo_use_case object in the Oo_extension. *)
  extension : oo_use_case;
  (* The extension_point specifies the Oo_extension_point that defines where information shall be added. *)
  extension_point : oo_extension_point;
END_ENTITY;

(* An Oo_extension_point is the position at which an Oo_use_case object may be extended through an Oo_extension
object. *)
ENTITY oo_extension_point;
  (* The location specifies the Textual_specification object that defines the location of the Oo_extension_point in textual
form. *)
  location : text_select;
  (* The use_case specifies the Oo_use_case object for which the Oo_extension_point is an extension point. *)
  use_case : oo_use_case;
END_ENTITY;

(* An Oo_generalization is a taxonomic relationship between a more general element and a more specific element. *)
ENTITY oo_generalization;
  (* The child specifies the more specific element. The child is fully consistent with the parent and may contain additional
information. *)
  child : oo_generalizable_element_select;
  (* The discriminator is the name of a partition of all child elements that have the same parent element. *)
  discriminator : label;
  (* The parent specifies the more general element. *)
  parent : oo_generalizable_element_select;
END_ENTITY;

(* An Oo_generic_association is either an Oo_association or an Oo_association_role and specifies a semantic relation-
ship between elements of an object oriented system specification. *)
```

```
ENTITY oo_generic_association
ABSTRACT SUPERTYPE OF (ONEOF(oo_association, oo_association_role));
  (* The description specifies additional information about the Oo_generic_association. *)
  description : OPTIONAL text_select;
  (* The id specifies the identifier of the Oo_generic_association. *)
  id : element_identifier;
  (* The name specifies the word, or words, that are used to refer to the Oo_generic_association. *)
  name : label;
  (* The reading_direction specifies the Oo_generic_association_end object to which the name of the Oo_generic_association is directed. <example number="14">A reading_direction between an Oo_class named Adult and another Oo_class named Child is named Parent. The reading direction specifies the Child as the directed object; for example, the association reads "Adult is Parent of Child".</example> *)
  reading_direction : oo_generic_association_end;
  (* The visibility specifies how the Oo_generic_association may be seen outside its enclosing namespace. Where applicable the following values should be used:<ul><li>private: Specifies that the Oo_generic_association is only visible to elements within the same namespace;</li> <li>protected: Specifies that the Oo_generic_association is only visible to elements of the same namespace and namespaces that are inherited from this;</li> <li>protected: Specifies that the Oo_generic_association is only visible to elements of the same namespace and namespaces that are inherited from this.</li></ul> *)
  visibility : label;
INVERSE
  (* The association specifies the Oo_association object of which the Oo_generic_association_end is an end. *)
  connection : SET [2 : ?] OF oo_generic_association_end FOR association;
END_ENTITY;

(* An Oo_generic_association_end is an end point of an Oo_association. *)
ENTITY oo_generic_association_end
ABSTRACT SUPERTYPE OF (ONEOF(oo_association_end, oo_association_end_role));
  (* The aggregation specifies the whole-part relationship of the associated elements. *)
  aggregation : label;
  (* The association specifies the Oo_association object of which the Oo_generic_association_end is an end. *)
  association : oo_generic_association;
  (* The description specifies additional information about the Oo_generic_association_end. *)
  description : OPTIONAL text_select;
  (* The id specifies an identifier for the Oo_generic_association_end. *)
  id : element_identifier;
  (* The is_navigable specifies whether traversal from a source instance to its associated target is possible. *)
  is_navigable : BOOLEAN;
  (* The multiplicity specifies the number of target instances that may be associated with a single source instance across the Oo_generic_association specified by the association attribute. *)
  multiplicity : cardinality_association_select;
  (* The name specifies the word, or words, that are used to refer to the Oo_generic_association_end. *)
  name : label;
  (* The visibility specifies how the Oo_generic_association_end may be seen outside its enclosing namespace. Where applicable the following values should be used:<ul><li>private: Specifies that the Oo_generic_association_end is only visible to elements within the same namespace;</li> <li>protected: Specifies that the Oo_generic_association_end is only visible to elements of the same namespace and namespaces that are inherited from this;</li> <li>public: Specifies that the Oo_generic_association_end can be accessed from other elements including those outside the namespace of the Oo_generic_association_end.</li></ul> *)
  visibility : label;
END_ENTITY;

(* An Oo_inclusion indicates that an instance of an Oo_use_case object will also contain the behaviour as specified by another Oo_use_case. *)
ENTITY oo_inclusion;
  (* The addition specifies the included Oo_use_case object. *)
  addition : oo_use_case;
  (* The base specifies the including Oo_use_case object. *)
  base : oo_use_case;
END_ENTITY;
```

```
(* An Oo_instance_classifier_relationship is a relationship between an instance and the classifier of which the Oo_in-
stance_classifier_relationship is an instance. *)
ENTITY oo_instance_classifier_relationship;
  (* The classifier specifies the object that provides the definition for the Oo_instance_classifier_relationship. *)
  classifier : oo_extended_classifier_select;
  (* The description specifies additional information about the Oo_instance_classifier_relationship. *)
  description : text_select;
  (* The instance specifies the Oo_instance_classifier_relationship that is to be related to its specific action. *)
  instance : oo_instance_select;
END_ENTITY;

(* An Oo_interaction specifies how messages are sent between instances to perform a specific task. *)
ENTITY oo_interaction;
  (* The description specifies additional information about the Oo_interaction. *)
  description : OPTIONAL text_select;
  (* The id specifies the identifier of the Oo_interaction. *)
  id : element_identifier;
  (* The interaction_context specifies the Oo_collaboration object that defines the context for the Oo_interaction. *)
  interaction_context : oo_collaboration;
  (* The name specifies the word, or words, that are used to refer to the Oo_interaction. *)
  name : label;
END_ENTITY;

(* An Oo_interface is a collection of Oo_operation objects that may be used for defining a service offered by an instance.
*)
ENTITY oo_interface;
  (* The associated_version specifies the Configuration_element_version for the Oo_interface. *)
  associated_version : configuration_element_version;
  (* The description specifies additional information about the Oo_interface. *)
  description : OPTIONAL text_select;
  (* The id specifies the identifier of the Oo_interface. *)
  id : element_identifier;
  (* The name specifies the word, or set of words, that is used to refer to the Oo_interface. *)
  name : label;
  (* The namespace specifies the element to which the namespace belongs. The names of elements belonging to the
same namespace element must be unique within this set. *)
  namespace : OPTIONAL oo_namespace_select;
END_ENTITY;

(* An Oo_link is a connection between instances. It is an instance of an Oo_association object. *)
ENTITY oo_link;
  (* The definition specifies the Oo_association object that provides the definition for the Oo_generic_association. *)
  definition : oo_generic_association;
INVERSE
  (* A link allocates Oo_attribute_instance objects to Oo_link_end objects. *)
  connection : SET [2 : ?] OF oo_link_end FOR link;
END_ENTITY;

(* An Oo_link_end is an end point of an Oo_link object. It is an instance of an Oo_association_end object. *)
ENTITY oo_link_end;
  (* The definition specifies the Oo_association_end object that provides the definition for the Oo_link_end. *)
  definition : oo_generic_association_end;
  (* The instance specifies the instance to which the Oo_link_end is attached. *)
  instance : oo_instance_select;
  (* A link allocates Oo_attribute_instance objects to Oo_link_end objects. *)
  link : oo_link;
WHERE
  WR1: definition.association :=: link.definition;
END_ENTITY;

(* An Oo_message is a communication between instances that conveys information with the expectation that activity will ensue. *)
```

```
ENTITY oo_message;
  (* The action specifies the Oo_action object that will, when executed, dispatch an instance of the Oo_message. *)
  action : oo_action;
  (* The communication specifies the Oo_association_role object that defines the roles that the sender and receiver
objects must conform to. *)
  communication : OPTIONAL oo_association_role;
  (* The interaction specifies the Oo_interaction object which the interaction is a part of. *)
  interaction : oo_interaction;
  (* The name specifies the word, or set of words, that is used to refer to the Oo_message. *)
  name : label;
  (* The receiver specifies the receving Oo_classifier_role object. *)
  receiver : oo_classifier_role;
  (* The sender specifies the sending Oo_classifier_role object. *)
  sender : oo_classifier_role;
  (* The sequence_number specifies the order of the Oo_message in a sequence of Oo_message objects in the overall
interaction. *)
  sequence_number : LIST [1 : ?] OF natural_number;
END_ENTITY;

(* An Oo_message_temporal_relationship is a relationship between two Oo_message objects that specifies their tempo-
ral order. *)
ENTITY oo_message_temporal_relationship;
  (* The predecessor specifies the preceding Oo_message object. *)
  predecessor : oo_message;
  (* The successor specifies the subsequent Oo_message object. *)
  successor : oo_message;
END_ENTITY;

(* An Oo_method is an implementation of an Oo_operation. *)
ENTITY oo_method
SUBTYPE OF (oo_behavioural_feature);
  (* The body specifies the Textual_specification object that defines the specification and the specification language for
the Oo_method. *)
  body : textual_specification;
  (* The description specifies additional information about the Oo_method. *)
  description : OPTIONAL text_select;
  (* The id specifies the identifier of the Oo_method. *)
  id : element_identifier;
  (* The specification specifies the Oo_operation of which the Oo_method is an implementation *)
  specification : oo_operation;
END_ENTITY;

(* An Oo_model_element_stereotype_relationship is a relationship between an Oo_stereotype object and another object.
*)
ENTITY oo_model_element_stereotype_relationship;
  (* The model_element specifies the related element. *)
  model_element : oo_model_element_select;
  (* The stereotype specifies the Oo_stereotype that is related. *)
  stereotype : oo_stereotype;
END_ENTITY;

(* An Oo_model_element_tagged_value_relationship is a relationship between an Oo_tagged_value object and another
object employing the Oo_tagged_value object. *)
ENTITY oo_model_element_tagged_value_relationship;
  (* The model_element specifies the relating element. *)
  model_element : oo_model_element_select;
  (* The tagged_value specifies the related Oo_tagged_value object. *)
  tagged_value : oo_tagged_value;
END_ENTITY;

(* An Oo_object is an instance of an Oo_class. *)
ENTITY oo_object;
```

```
  (* The definition specifies the Oo_class of which the Oo_object is an instance. *)
  definition : oo_class;
  (* The description specifies additional information about the Oo_object. *)
  description : OPTIONAL text_select;
  (* The id specifies the identifier of the Oo_object. *)
  id : element_identifier;
END_ENTITY;

(* An Oo_operation is a service that can be requested to effect behaviour. *)
ENTITY oo_operation
SUBTYPE OF (oo_behavioural_feature);
  (* The concurreny specifies how a call of the Oo_operation is to be executed. *)
  concurrency : label;
  (* The description specifies additional information about the Oo_operation. *)
  description : OPTIONAL text_select;
  (* The id specifies the identifier of the Oo_operation. *)
  id : element_identifier;
  (* The is_abstract specifies whether an Oo_operation may be instantiated or not. If the value is TRUE, the Oo_operation is abstract and may not be directly instantiated, only derivatives of the is_abstract may be instantiated. If the value is FALSE the is_abstract is not abstract and may be instantiated. *)
  is_abstract : BOOLEAN;
  (* The specification specifies the Textual_specification object that defines the specification and the specification language for the Oo_operation. *)
  specification : textual_specification;
END_ENTITY;

(* An Oo_operation_interface_association is a relationship between an Oo_operation object and an Oo_interface object.
*)
ENTITY oo_operation_interface_association;
  (* The description specifies additional information about the Oo_operation_interface_association. *)
  description : OPTIONAL text_select;
  (* The interface specifies the Oo_interface object of which the operation is part. *)
  interface : oo_interface;
  (* The operation specifies the Oo_operation that is part of the Oo_interface specified by the interface. *)
  operation : oo_operation;
END_ENTITY;

(* An Oo_package is a general purpose mechanism for grouping objects. *)
ENTITY oo_package
SUPERTYPE OF (oo_view);
  (* The associated_version specifies the Configuration_element_version for the Oo_package. *)
  associated_version : configuration_element_version;
  (* The description specifies additional information about the Oo_package. *)
  description : OPTIONAL text_select;
  (* The id specifies the identifier of the Oo_package. *)
  id : element_identifier;
  (* The name specifies the word, or set of words, that is used to refer to the Oo_package. *)
  name : label;
  (* The visibility specifies how the Oo_package may be seen outside its enclosing namespace. Where applicable the following values should be used:private: Specifies that the Oo_package is only visible to elements within the same namespace;</li> <li>protected: Specifies that the Oo_package is only visible to elements of the same namespace and namespaces that are inherited from this;</li> <li>public: Specifies that the Oo_package can be accessed from other elements including those outside the namespace of the Oo_package.</li></ul> *)
  visibility : label;
INVERSE
  (* The container specifies the Oo_package that imports the model_element through the container. *)
  element_import : SET [0 : ?] OF oo_element_import FOR container;
END_ENTITY;

(* An Oo_parameter is a variable that can be changed, passed or returned.
 *)
```

```
ENTITY oo_parameter;
  (* The behavioural_feature specifies the Oo_behavioural_feature to which the behavioural_feature belongs. *)
  behavioural_feature : oo_behavioural_feature;
  (* The default_value specifies the value that is given to the default_value when it is created or if its value is out of the
allowed value range.
 *)
  default_value : label;
  (* The kind specifies the direction of the kind. Where applicable the following values should be used:
 *)
  kind : label;
  (* The name specifies the word, or set of words, by which the name is referred.
 *)
  name : label;
  (* The parameter_type specifies the type definition of the parameter_type.
 *)
  parameter_type : oo_classifier_select;
  (* The visibility specifies how the visibility may be seen outside its enclosing namespace. Where applicable the following
values should be used:
 *)
  visibility : label;
END_ENTITY;

(* An oo_reception declares that a classifier specified by owner is prepared to react to the receipt of an Oo_signal. A
classifier is one of the following: Physical_instance, Oo_class, Oo_actor, Oo_use_case, Oo_signal, Oo_component, or
Oo_interface.
 *)
ENTITY oo_reception
SUBTYPE OF (oo_behavioural_feature);
  (* The signal specifies the Oo_signal that the owner is prepared to react to.
 *)
  signal : oo_signal;
  (* The specification defines the specifies for the specification.
 *)
  specification : text_select;
END_ENTITY;

(* An oo_send_action specifies an asynchronous signalling action.
 *)
ENTITY oo_send_action
SUBTYPE OF (oo_action);
  (* The signal specifies the Oo_signal that is being raised.
 *)
  signal : oo_signal;
END_ENTITY;

(* An oo_signal specifies an asynchronous communication between instances.
 *)
ENTITY oo_signal;
  (* The name specifies the word, or set of words, by which the name is referred.
 *)
  name : label;
  (* The namespace specifies the element to which the namespace belongs. The names of elements belonging to the
same namespace element must be unique within this set.
 *)
  namespace : OPTIONAL oo_namespace_select;
END_ENTITY;

(* An oo_signal_behavioral_feature_relationship specifies the relationship between an Oo_signal and Oo_behavioral_
feature object as its behavioural_feature_context.
 *)
ENTITY oo_signal_behavioural_feature_relationship;
```

```
  (* The behavioural_feature_context specifies the Oo_behavioral_feature object that defines context for the Oo_signal
specified by raised_signal.
 *)
  behavioural_feature_context : oo_behavioural_feature;
  (* The raised_signal specifies the raised Oo_signal.
 *)
  raised_signal : oo_signal;
END_ENTITY;

(* An oo_stereotype specifies a new type of element that extends the semantics of the meta-model.
 *)
ENTITY oo_stereotype;
  (* The associated_version specifies the Configuration_element_version for the associated_version.
 *)
  associated_version : configuration_element_version;
  (* The description specifies additional information about the description.
 *)
  description : OPTIONAL text_select;
  (* The id specifies the identifier of the id.
 *)
  id : element_identifier;
  (* The name specifies the word, or set of words, by which the name is referred.
 *)
  name : label;
  (* The visibility specifies how the visibility may be seen outside its enclosing namespace. Where applicable the following
values should be used:
 *)
  visibility : label;
END_ENTITY;

(* An oo_stimulus reifies a communication between two instances.
 *)
ENTITY oo_stimulus;
  (* The dispatch_action specifies the action performed by the dispatch_action.
 *)
  dispatch_action : oo_action;
  (* The receiver specifies the receiving instance of the receiver.
 *)
  receiver : oo_instance_select;
  (* The sender specifies the sending instance of the sender.
 *)
  sender : oo_instance_select;
END_ENTITY;

(* The oo_stimulus_argument is a relationship between an Oo_stimulus object and an instance serving as argument for
the stimulus.
 *)
ENTITY oo_stimulus_argument;
  (* The argument specifies the instance that serves as argument for the Oo_stimulus object specifies by stimulus.
 *)
  argument : oo_instance_select;
  (* The stimulus specifies the Oo_stimulus object of which the stimulus is an argument.
 *)
  stimulus : oo_stimulus;
END_ENTITY;

(* An oo_tagged_value defines a property as a name-initial_value pair.
 *)
ENTITY oo_tagged_value;
  (* The id specifies the identifier of the id.
 *)
```

```
  id : element_identifier;
  (* The initial_value specifies the actual value of the initial_value.
 *)
  initial_value : text_select;
  (* The name specifies the word, or set of words, by which the name is referred. The name is reffered to as the tag.
 *)
  name : label;
END_ENTITY;

(* An oo_use_case specifies a sequence of actions that a system or other entity can perform, interacting with Oo_actor
objects of the system.
 *)
ENTITY oo_use_case;
  (* The description specifies additional information about the Oo_use_case. *)
  description : OPTIONAL text_select;
  (* The id specifies the identifier of the Oo_use_case. *)
  id : element_identifier;
  (* The name specifies the word, or set of words,that are used to refer to the Oo_use_case. *)
  name : label;
  (* The namespace specifies the element to which the Oo_use_case belongs. The names of elements belonging to the
same namespace element must be unique within this set. *)
  namespace : OPTIONAL oo_namespace_select;
END_ENTITY;

(* An Oo_view is a graphical representation of a collection of model_elements. *)
ENTITY oo_view
SUBTYPE OF (oo_package);
  (* The view_type specifies the type of the Oo_view. Where applicable the following values should be used:<ul><li>activ-
ity: The Oo_view represents a state machine where all or most of the states are Oo_action_state objects and all or most
of the transitions are Oo_action_state_transition objects;</li> <li>collaboration: The Oo_view represents interactions and
links between a collection of objects;</li> <li>component: The Oo_view represents the organizations and dependencies
among Oo_component objects;</li> <li>deployment: The Oo_view represents the run-time configuration of Physical_in-
stance objects and the Oo_component objects executing on them;</li> <li>sequence: The Oo_view represents an inter-
action of a collection of objects in a time sequence;</li> <li>statechart: The Oo_view represents a state machine;</li>
<li>static structure: The Oo_view represents a collection of declarative (static) objects;</li> <li>use case: The Oo_view
shows the relationships among Oo_actor objects and Oo_use_case objects within a system.</li></ul> *)
  view_type : label;
END_ENTITY;

(* An Oo_view_context_element_relationship specifies the allocation of an object to a Oo_view. *)
ENTITY oo_view_context_element_relationship;
  (* The represented_model_element specifies the object to be mapped. *)
  represented_model_element : oo_extended_model_element_select;
  (* The view specifies the Oo_view that the model_element is allocated to.
 *)
  view : oo_view;
END_ENTITY;

(* An Oo_view_relationship is a relationship between two Oo_view objects. *)
ENTITY oo_view_relationship;
  (* The base specifies the relating Oo_view object. *)
  base : oo_view;
  (* The description specifies additional information about the Oo_view_relationship.
 *)
  description : text_select;
  (* The reference_view specifies the related Oo_view object. *)
  reference_view : oo_view;
  (* The relationship_type specifies the type of the Oo_view_relationship. <example number="15">An Oo_view of the
view_type sequence can be a specialization of another Oo_view of the view_type use case. For this example the type
would be given the value specialization.</example> *)
  relationship_type : label;
END_ENTITY;
```

```
(* An oo_view_system_view_relationship is a relationship between an Oo_view and a System_view.
 *)
ENTITY oo_view_system_view_relationship;
  (* The oo_view specifies the Oo_view object for that the system_context is being defined.
 *)
  oo_view : oo_view;
  (* The system_context specifies the System_view object that defines a context for the oo_view.
 *)
  system_context : system_view;
END_ENTITY;

(* An organization is a group of people involved in a particular business process. <note number="1">This is the AP 214
definition.</note> *)
ENTITY organization;
  (* The delivery_address specifies the address where goods are delivered. *)
  delivery_address : OPTIONAL address;
  (* The description specifies additional information about the Organization. *)
  description : OPTIONAL text_select;
  (* The id specifies the identifier of the Organization. <note number="2">The assignment of this attribute is usually
controlled by a registration authority. The registration authority may be a public organization that assigns identifiers to
corporations, or it may be the parent corporation that assigns component identifiers to its components.</note><example
number="16">The id may be the code assigned to the id for a stock market listing, or it may be a department number.</
example> *)
  id : basic_identifier;
  (* The name specifies the word, or set of words, that is used to refer to the Organization. *)
  name : label;
  (* The postal_address specifies the address where letter mail is delivered. *)
  postal_address : OPTIONAL address;
  (* The visitor_address specifies the address where the Organization receives visitors. *)
  visitor_address : OPTIONAL address;
END_ENTITY;

(* An Organization_relationship is used to represent an arbitrary relationship between two Organization objects. <note
number="1">It also allows for the description of the respective role of an Organization in a Project (Prime contractor,
partners, sub-contractors).</note> <example number="17">The relationship between two Organization objects may be
"system contractor" or "partner".</example> *)
ENTITY organization_relationship;
  (* The description specifies a textual string that captures additional information on the nature of the relationship between
the two Organization objects. *)
  description : OPTIONAL text_select;
  (* The name specifies a name that is associated with the relationship between the two Organization objects. *)
  name : label;
  (* The related_organization specifies the second of the two Organization objects involved in the relationship. <note
number="2">The exact semantics of this attribute depend of the value of the description attribute.</note> *)
  related_organization : organization;
  (* The relating_organization specifies the first of the two Organization objects involved in the relationship. <note num-
ber="3">The exact semantics of this attribute depend of the value of the description attribute.</note> *)
  relating_organization : organization;
WHERE
  WR1: related_organization :<>: relating_organization;
END_ENTITY;

(* A Package is a set of data used to group objects that are related in some way. <note number="1">The classification
criteria is user defined.</note> <note number="2">Package can be used to group information on any level of design
abstraction, from a small set of Requirement_instance objects that somehow are related to the set of design information
that is the outcome of a design Engineering_process_activity.</note> *)
ENTITY package
SUPERTYPE OF (selection_package);
  (* The description specifies additional information about the Package. *)
  description : OPTIONAL text_select;
```

```
  (* The discriminator specifies the common classification criteria that apply to all elements of the Package. <note number="3">The criteria defined by the discriminator is not formally rigourous. The standard does not prevent classifications that are in conflict with the discriminator.</note> *)
  discriminator : text;
  (* The id specifies the identifier of the Package. *)
  id : element_identifier;
  (* The name specifies the word, or set of words, that is used to refer to the Package. *)
  name : label;
INVERSE
  (* The package specifies the Package to which the element is assigned
 *)
  element : SET [0 : ?] OF package_element_assignment FOR package;
UNIQUE
  UR1: id;
END_ENTITY;

(* A Package_classification_assignment is the mechanism for assigning a Package to a Package_classification_system such that the Package is a member at the top-level of the classification system. *)
ENTITY package_classification_assignment;
  (* The assigned_package specifies the Package in the relationship. *)
  assigned_package : package;
  (* The classification_system specifies the Package_classification_system in the relationship. *)
  classification_system : package_classification_system;
END_ENTITY;

(* A Package_classification_system is a representation of a classification system consisting of multiple Package objects. <note>The purpose of the Package_classification_system is to support reuse in large design databases. The classification mechanism allows for creation of multiple classification criteria. Classification may be taken down to any depth by nesting Package objects.</note> *)
ENTITY package_classification_system;
  (* The description specifies additional textual information about the Package_classification_system. *)
  description : OPTIONAL text_select;
  (* The id specifies the identifier of the Package_classification_system. *)
  id : element_identifier;
  (* The name specifies the word, or set of words, that is used to refer to the Package_classification_system. *)
  name : label;
END_ENTITY;

(* A package_element_assignment is the assignment of an element to a particular Package.
<note>
The exact list of what can be assigned to a Package shall be evaluated.
</note> *)
ENTITY package_element_assignment;
  (* The description specifies additional information about the description.
 *)
  description : OPTIONAL text_select;
  (* The element specifies the instance_definition_select, a Configuration_element_version, a Configuration_element, an Engineering_process_activity or a Project that is assigned to a Package.
 *)
  element : package_element_select;
  (* The package specifies the Package to which the element is assigned
 *)
  package : package;
  (* The reference_name specifies the word or set of words by which the element defined by the element attribute is referred to in the context of the package defined by the package attribute.
 *)
  reference_name : OPTIONAL label;
END_ENTITY;

(* A Package_hierarchy_relationship is the definition of a parent child relationship between two Package objects.<note number="1">The purpose of this entity is to allow for classification structures ranging from generic to specific - such as
```

the classification of vehicles into more specific classes.</note> <note number="2">Classification in this context does not necessarily imply inheritance of characteristics as inheritance is defined by object oriented software engineering methods and languages.</note> *)
ENTITY package_hierarchy_relationship;
(* The sub_package specifies the child Package in the relationship. *)
sub_package : package;
(* The super_package specifies the parent Package in the relationship. *)
super_package : package;
END_ENTITY;

(* A Partial_document_assignment is a type of Document_assignment with the discriminator that only a part of the assigned is of interest to the element the document is assigned to. *)
ENTITY partial_document_assignment
SUBTYPE OF (document_assignment);
(* The document_portion specifies the subset of the assigned document that is of interest for the Partial_document_assignment. *)
document_portion : label;
END_ENTITY;

(* A partial_system_view is a type of System_view and a domain, mode or scenario oriented view of a system.
<note
number="1">
The model makes no assumption on the complexity of a view. It may be anything from utterly simple to very complex. The partial_system_view is intended to collect requirements in the requirements elicitation phase, to capture domain, state or scenario specific information of a system. A partial_system_definition on its own does not represent a specification for a particular system.
</note><note
number="2">
To include the elements in a partial_system_view into a system specification it shall be associated with a System_definition object by the use of a System_view_assignment object. For partial_system_view objects it is possible to capture information on information such as life-cycle viewpoint assumed for the particular system, fidelity via the attribute is_relevant_for.
</note> *)
ENTITY partial_system_view
SUBTYPE OF (system_view);
(* The is_relevant_for specifies the System_view_context object that specifies lifecycle, fidelity and view point for a is_relevant_for.
*)
is_relevant_for : system_view_context;
INVERSE
(* The assigned_view specifies the assigned Partial_system_view. *)
assigned_to_systems : SET [1 : ?] OF system_view_assignment FOR assigned_view;
END_ENTITY;

(* A Partial_system_view_relationship is a relationship between two Partial_system_view objects. The semantics of the relationship is specified by the relationship_type attribute. *)
ENTITY partial_system_view_relationship
SUPERTYPE OF (triggered_system_view_relationship);
(* The description specifies additional textual information about the Partial_system_view. *)
description : OPTIONAL text_select;
(* The related specifies the second Partial_system_view object in the relationship. *)
related : partial_system_view;
(* The relating specifies the first Partial_system_view object in the relationship. *)
relating : partial_system_view;
(* The relationship_type specifies the semantics of the relationship_type. <example number="18">For a launcher a viable partition is to describe the system in two views: "on ground" and "lauched". A Partial_system_view_relationship may be established between the Partial_system_view object with the relationship_type set to "precedence", indicating that the ground_view always preceeds the lauched view when the system is operational.
</example>Where applicable one of the following values shall be used:
overlap: The related and relating Partial_system_view objects contain specifications that are redundant. (The specifications are assumed to at the same level of granularity). detail: The related object is providing a more detailed view on the system compared with

the relating objects; precedence: The related Partial_system_view object describe a view of the system which preceedes the relating view of the system. *)
  relationship_type : label;
  (* The system_definition_context specifies the System_definition object for which the Partial_system_view is valid. *)
  system_definition_context : system_definition;
WHERE
  correct_relationship: related :<>: relating;
END_ENTITY;

(* A persistent_storage is a type of General_functionality_instance which allows for persistent storage of discrete Data_instance objects. <note number="1">Persistent storage should not be confused with continuous storage - such as a water tank. The latter is a function in its own right and shall be represented by a General_function_definition object defining the functional characteristics of the tank.</note> <note number="2">Data_instance objects are used to represent what is stored no matter what the persistent_storage is designed to store. The interface to persistent_storage is defined by Actual_io_port objects. What is actually stored in the persistent_storage is defined by the data associated with the port.</note> *)
ENTITY persistent_storage
SUBTYPE OF (general_functionality_instance);
  (* The id specifies the identifier of the Persistent_storage. *)
  id : element_identifier;
  (* The name specifies the word, or set of words, that are used to refer to the Persistent_storage. *)
  name : label;
  (* The permanent specifies whether the storage is permanent or volatile. <note number="3">Volatility in this context is application dependent. The purpose of this attribute is to define whether the information stored is available after some kind of "restart" or "reset" of the system.</note> *)
  permanent : LOGICAL;
  (* The presentation_id specifies the identity information for a Persistent_storage that shall be presented to the user. *)
  presentation_id : OPTIONAL label;
  (* The read_only specifies whether it is possible to write to the Store or not. <note number="4">How a read only store gets values is not specified in the standard but left to the design tools using this feature. It is possible that a store could have different values for read_only in different operating modes (i.e., operational data may be loaded during ground warm up but is read only during flight).</note> *)
  read_only : LOGICAL;
  (* The storage_access specifies how the elements in the Persistent_storage are accessed. Where applicable one of the following values shall be used:
queue: First in - first out semantics applies; stack: Last in - first out semantics applies; random_access: The access semantics is not constrained; other: The access semantics is not specified. *)
  storage_access : label;
  (* The store_size specifies the upper bound of elements that can be held in the Persistent_storage. *)
  store_size : OPTIONAL INTEGER;
UNIQUE
  UR1: id;
END_ENTITY;

(* A Persistent_storage_equivalence_relationship is the relationship between two Persistent_storage objects to indicate that the two objects are representations of the same storage object. <note>The purpose of the persistent_storage_equivalence_relationship is to indicate that two storage objects are referencing the same storage facility. This is because we can see a hierarchy of functions, but the storage mechanism is generally flat.</note> *)
ENTITY persistent_storage_equivalence_relationship;
  (* The equivalent_storage specifies the set of persistent_storage_reference objects which shall be treated as references to the same storage in the scope of a functional_reference_configuration. *)
  equivalent_storage : SET [2 : ?] OF persistent_storage_reference;
  (* The valid_context specifies the System_view for which the equivalence relationship is valid. *)
  valid_context : functional_reference_configuration;
END_ENTITY;

(* A Persistent_storage_reference is a type of Functionality_instance_reference that provides the mechanism for unambiguous reference to a persistent_storage object. *)
ENTITY persistent_storage_reference
SUBTYPE OF (functionality_instance_reference);
  SELF\functionality_instance_reference.referenced_functionality_instance : persistent_storage;
END_ENTITY;

```
(* A Person is an individual human being who has some relationship to product data. *)
ENTITY person;
  (* The Address specifies the location where the Person can be reached. *)
  address : OPTIONAL address;
  (* The first_name specifies the first name of the first_name.
 *)
  first_name : OPTIONAL label;
  (* An id specifies the identifier of a Person. *)
  id : basic_identifier;
  (* The last_name specifies the surname of the last_name.
 *)
  last_name : OPTIONAL label;
  (* The middle_names specifies any middle names of the middle_names.
 *)
  middle_names : OPTIONAL LIST [1 : ?] OF label;
  (* The prefix_titles specifies any prefix titles of the prefix_titles.
 *)
  prefix_titles : OPTIONAL LIST [1 : ?] OF label;
  (* The suffix_titles specifies the suffix titles of the suffix_titles.
 *)
  suffix_titles : OPTIONAL LIST [1 : ?] OF label;
UNIQUE
  UR1: id;
END_ENTITY;

(* A Person_in_organization is the specification of a Person in the context of an Organization. *)
ENTITY person_in_organization;
  (* The associated_organization specifies the Organization. *)
  associated_organization : organization;
  (* The associated_person specifies the Person. *)
  associated_person : person;
  (* The description specifies additional information about the Person_in_organization. *)
  description : OPTIONAL text_select;
  (* The role specifies the relationship between the Person and the Organization. *)
  role : label;
END_ENTITY;

(* A Person_organization_assignment is an object that associates an Organization or a Person_in_organization with
product data. <note>This assignment provides additional information for the associated object. The provision of such
data through this assignment has an organizational character whereas some objects require the same kind of mandatory
data in order to be semantically complete. This assignment shall not be used to associate the corresponding organiza-
tional data with an object whose attributes are referencing the organizational data directly.</note> *)
ENTITY person_organization_assignment;
  (* The assigned_person_organization specifies the concerned Person_in_organization or Organization. *)
  assigned_person_organization : person_organization_select;
  (* The assigned_to specifies the element to which the Person_organization_assignment applies. *)
  assigned_to : person_organization_assignment_select;
  (* The description specifies additional information about the Person_organization_assignment. *)
  description : OPTIONAL text_select;
  (* The role specifies the responsibility of the assigned Person_in_organization or Organization with respect to the
object that it is applied to. Where applicable, the following values shall be used:<ul> <li>creator: The referenced object
has been created by the assigned Person or Organization;</li> <li>custodian: The assigned Person or Organization is
responsible for the existence and integrity of the referenced object;</li> <li>customer: The assigned Person or Organi-
zation acts as a purchaser or consumer of the referenced object;</li> <li>designer: The assigned Person or Organiza-
tion is the one that delivers the data describing the referenced object (such as a system specification);</li> <li>editor:
The assigned Person or Organization is responsible for making any changes to any attribute of the referenced object;</
li> <li>id_owner: The assigned Person or Organization is the one responsible for the designation of an identifier;</li>
<li>location: The assigned Organization is the place where the referenced object can be found or where it takes place;</
li> <li>manufacturer: The assigned Person or Organization is the one that produces the actual (physical) object;</li>
<li>owner: The assigned Person or Organization owns the referenced object, and has final say over its disposition and
any changes to it;</li> <li>supplier: The assigned Person or Organization is the one that delivers the actual (physical)
```

```
object;</li> <li>wholesaler: The assigned Person or Organization is in the sales chain between the manufacturer and the supplier.</li> </ul> *)
  role : label;
END_ENTITY;

(* A Physical_binding is the mapping between a Formal_physical_port and an Actual_physical_port such that the ports are connected to each other. *)
ENTITY physical_binding;
  (* The actual_port specifies the Actual_physical_port in the mapping. *)
  actual_port : actual_physical_port;
  (* The formal_port specifies the Formal_physical_port in the mapping. *)
  formal_port : formal_physical_port;
WHERE
  WR1: formal_port.port_of :=: actual_port.port_of.definition;
END_ENTITY;

(* A Physical_composition_relationship is a mechanism that relates a Physical_instance in the role of a component to an assembly represented by a General_physical_definition object. <note>A Physical_composition_relationship is used to describe the breakdown structure of a component. The General_physical_definition object can thus be composed of a set of sub-components. Therefore, the relationship points on one hand at the definition and on the other hand at the instances that belong to the component breakdown.</note> *)
ENTITY physical_composition_relationship;
  (* The assembly specifies the General_physical_definition in the relationship. *)
  assembly : general_physical_definition;
  (* The component specifies the Physical_instance in the relationship. *)
  component : physical_instance;
  (* The description specifies additional information about the relationship. *)
  description : OPTIONAL text_select;
END_ENTITY;

(* A Physical_connection is the mechanism for associating two Physical_port objects to each other. <note number="1">The Physical_connection specifies an ideal connection between two ports. It is assumed there is no energy loss in the connection. In case a non-ideal connection is desired then connection in itself shall be represented by a General_physical_definition or Physical_instance object.</note> <note number="2">In a functional specification the closest corresponding entity to Physical_connection is a Functional_link.</note> <note number="3">The Physical_connection is bi-directional.</note> *)
ENTITY physical_connection;
  (* The connected specifies the second Physical_port in the relationship. *)
  connected : physical_port;
  (* The connecting specifies the first Physical_port in the relationship. *)
  connecting : physical_port;
END_ENTITY;

(* A Physical_instance is the occurrence of a General_physical_definition within a system specification. *)
ENTITY physical_instance;
  (* The definition specifies the General_physical_definition that provides the specification of the Physical_instance. *)
  definition : general_physical_definition;
  (* The description specifies additional information about the Physical_instance. *)
  description : OPTIONAL text_select;
  (* The id specifies the identifier of the Physical_instance. *)
  id : element_identifier;
  (* The name specifies the word, or set of words, that is used to refer to the Physical_instance. *)
  name : label;
  (* The presentation_id specifies the identity information for a Physical_instance that shall be presented to the user. *)
  presentation_id : OPTIONAL label;
INVERSE
  (* The port_of specifies the Physical_instance for which the port_of provides part of the interface. *)
  actual_port : SET [0 : ?] OF actual_physical_port FOR port_of;
UNIQUE
  UR1: definition, id;
END_ENTITY;
```

(* A Physical_instance_reference is the non-ambiguous reference to a Physical_instance element in a Physical_node_definition structure. <note>A Physical_instance_reference is introduced to allow for allocation and reference of system specific non-functional characteristics to a system physical description.</note> *)
ENTITY physical_instance_reference;
(* The description specifies additional textual information about the Physical_instance_reference. *)
description : OPTIONAL text_select;
(* The id specifies the identifier of the Physical_instance_reference. *)
id : element_identifier;
(* The name specifies the word, or set of words, that is used to refer to the Physical_instance_reference. *)
name : label;
(* The reference_for_instance specifies the Physical_instance that the Physical_instance_reference is a reference for. *)
reference_for_instance : physical_instance;
END_ENTITY;

(* A Physical_link_definition is a type of General_physical_definition with the discriminator that the main purpose is transfer of information or material. *)
ENTITY physical_link_definition
SUBTYPE OF (general_physical_definition);
END_ENTITY;

(* A Physical_node_definition is a type of General_physical_definition with the discriminator that it is used for processing (transforming) material and or information from one format to another. <note>The component defined by a Physical_node_definition is essentially technology free. Within the scope of this PAS, no assumption is made in regard to the nature of the physical thing.</note> <example number="19">For instance, in order to perform a certain amount of operation in a given time slot, a human being can be used (reliable enough, accurate enough) whereas for some other kind of processing a powerful computer is required. The part designer will use either of the technologies to satisfy the requirement.</example> <example number="20">In order to perform heat protection several technology alternatives can be used based on several kinds of material. The Physical_node_definition object will describe the performances that have to be met (temperature range, time period it shall last, cost, etc.), whereas the real manufacturing of the component will use a specific material to meet those performances.</example> *)
ENTITY physical_node_definition
SUBTYPE OF (general_physical_definition);
END_ENTITY;

(* A Physical_port is a representation of an element in the interface to a Physical_instance or a General_physical_definition. *)
ENTITY physical_port
ABSTRACT SUPERTYPE OF (ONEOF(actual_physical_port, formal_physical_port));
(* The description specifies additional information about the Physical_port. *)
description : OPTIONAL text_select;
(* The direction specifies the direction of data (material, energy, etc.) transfer supported by the Physical_port.The direction is one of the following: input: The Physical_port is the input of a transfer; output: The Physical_port is the output of a transfer; bi-directional: The Physical_port may be both input or output of a transfer. *)
direction : label;
(* The name specifies the word, or set of words, that are used to refer to the Physical_port. *)
name : label;
END_ENTITY;

(* A Physical_reference_configuration is a collector for all Physical_reference_relationship objects that apply for a particular physical view on a system. <note number="1">A Physical_reference_configuration may be assigned to any number of systems through the use of multiple System_physical_configuration objects.</note> <note number="2">The Physical_reference_configuration provides the means for defining multiple non-functional views on top ofa single system physical architecture model.</note> *)
ENTITY physical_reference_configuration;
(* The description specifies additional information about the Physical_reference_configuration. *)
description : OPTIONAL text_select;
END_ENTITY;

(* A Physical_reference_relationship is the mechanism for indicating a hierarchical relationship between two Physical_instance_reference objects. *)
ENTITY physical_reference_relationship;

```
  (* The child specifies the component Physical element in the Physical_reference_relationship. *)
  child : physical_instance_reference;
  (* The mirror_of specifies the Physical_composition_relationship for which the Physical_reference_relationship provides
the unambiguous reference. *)
  mirror_of : physical_composition_relationship;
  (* The parent specifies the decomposed Physical_instance_reference in the Physical_reference_relationship. *)
  parent : physical_instance_reference;
  (* The valid_configuration specifies the Physical_reference_configuration for which the parent and child Physical_in-
stance_reference are valid. *)
  valid_configuration : physical_reference_configuration;
END_ENTITY;

(* A Plus_minus_bounds is the specification of the allowable deviation from a numerical value applied to a Nominal_val-
ue. *)
ENTITY plus_minus_bounds;
  (* The distribution_function specifies a mathematical function describing the projected distribution of measurement
values. *)
  distribution_function : OPTIONAL textual_specification;
  (* The limited_value specifies the Nominal_value further defined by the Plus_minus_bound. *)
  limited_value : nominal_value;
  (* The lower_bound specifies the value of the tolerance that shall be subtracted from the exact value to establish the
minimum allowed value. *)
  lower_bound : NUMBER;
  (* The significant_digits specifies the number of decimal digits indicating the accuracy of the lower bound and upper
bound values. *)
  significant_digits : OPTIONAL INTEGER;
  (* The upper_bound specifies the value of the tolerance that shall be added to the exact value to establish the maxi-
mum allowed value. *)
  upper_bound : NUMBER;
END_ENTITY;

(* A Project is an identified programme of work. <note>A Project may be further characterized by planned or actual dates,
allowed budget, or resources.</note> <example number="21">For the development of a new system, a project is set up
that is responsible for the development decisions as well as for the accounting of the costs.</example> *)
ENTITY project;
  (* The actual_end_date specifies the date when the Project was actually finished. *)
  actual_end_date : OPTIONAL date_time;
  (* The actual_start_date specifies the date when the Project was actually started. *)
  actual_start_date : OPTIONAL date_time;
  (* The description specifies additional information about the Project. *)
  description : OPTIONAL text_select;
  (* The id specifies the identifier of the Project. *)
  id : element_identifier;
  (* The name specifies the word, or group of words, that is used to refer to the Project. *)
  name : label;
  (* The planned_end_date specifies either the date when the Project is or was supposed to be finished or the duration of
the Project. *)
  planned_end_date : OPTIONAL period_or_date_select;
  (* The planned_start_date specifies the date when the Project is or was supposed to be started. *)
  planned_start_date : OPTIONAL event_or_date_select;
  (* The work_program specifies the Engineering_process_activity objects that are carried out within the Project. *)
  work_program : SET [0 : ?] OF engineering_process_activity;
END_ENTITY;

(* A Project_event_reference is the definition of a point in time established relative to an event. *)
ENTITY project_event_reference;
  (* The description specifies additional information about the Project_event_reference. *)
  description : OPTIONAL text_select;
  (* The event type specifies the kind of event that serves as reference. <example number="22">The two events “start of
system analysis” and “end of detailed design” are examples of an event_type.</example> *)
  event_type : label;
```

```
  (* The offset specifies the amount of time before or after the defined event that shall be used to calculate the actual
point in time. *)
  offset : value_with_unit;
END_ENTITY;

(* A project_relationship is a relationship between two Project objects.
 *)
ENTITY project_relationship;
  (* The description specifies additional information about the description.
 *)
  description : OPTIONAL text_select;
  (* The related specifies the second of the two Project objects related by a related.
<note
number="1">
The semantics of this attribute are defined by the attribute relation_type.
</note> *)
  related : project;
  (* The relating specifies the first of the two Project objects related by a relating.
<note
number="2">
The semantics of this attribute are defined by the attribute relation_type.
</note> *)
  relating : project;
  (* The relation_type specifies the meaning of the relationship. Where applicable, the following values shall be used:
 *)
  relation_type : label;
END_ENTITY;

(* A Property_assignment is a mechanism of associating a Property_value with an element. *)
ENTITY property_assignment;
  (* The assigned_to specifies the element with which the property is associated. <note>The list of entities shall be scrutinized by the PAS 20542 group.</note> *)
  assigned_to : property_assignment_select;
  (* The assignment_rationale specifies the motivation for assigning a Property_value. *)
  assignment_rationale : OPTIONAL text_select;
  (* The measurement_method specifies the method used to obtain the Property_assignment. Where applicable one of
the following values shall be used:<ul> <li>measurement;</li> <li>estimation.</li> </ul> *)
  measurement_method : label;
  (* The property specifies the property assigned. *)
  property : property_value;
  (* The property_name specifies the word, or set of words, that is used to refer to the Property_value in the context of
the element to which the Property_value is assigned. *)
  property_name : label;
END_ENTITY;

(* A Property_definition is the definition of a special quality. <note>A Property may reflect physics or arbitrary user defined
measurements.</note> *)
ENTITY property_definition;
  (* The allowed_unit specifies the unit or set of units that are accepted.<example number="23">A company may accept a
mass specified in kilograms or tonnes, but not in grams or pounds.</example> *)
  allowed_unit : SET [0 : ?] OF unit;
  (* The description specifies additional information about the Property_definition. *)
  description : OPTIONAL text_select;
  (* The property_type specifies the nature of the property. <example number="24">Possible values for property_type
include "electrical" and "time".</example> *)
  property_type : label;
END_ENTITY;

(* A Property_relationship is a relationship between two Property objects. <example number="25">Property_relationship
may be used to indicate that the value of one Property_definition can be derived from the value of another Property_definition.</example> *)
```

```
ENTITY property_relationship;
  (* The description specifies additional information about the Property_relationship. *)
  description : OPTIONAL text_select;
  (* The related specifies the second of the two Property_definition objects related by the Property_relationship. *)
  related : property_definition;
  (* The relating specifies the first of the two Property_definition objects related by the Property_relationship. *)
  relating : property_definition;
  (* The relation_type specifies the meaning of the relationship. Where applicable, the following values shall be used:<ul>
<li>dependency: The related Property_definition is dependent upon the relating Property_definition;</li> <li>hierarchy:
The application object defines a hierarchical relationship where the related Property_definition is on a lower level than
the relating Property_definition.</li> </ul> *)
  relation_type : label;
END_ENTITY;

(* A Property_value is the representation of a characteristic of a Property_definition. *)
ENTITY property_value;
  (* The definition specifies the Property_definition that the Property_value characterizes. *)
  definition : property_definition;
  (* The property_value_name specifies the word, or group of words, that is used to refer to the Property_value. <exam-
ple number="26">"l1" or "vol2" are examples for the value name of a property_value_name.</example> *)
  property_value_name : label;
  (* The specified_value specifies either a text description or a numerical measure representing the Property_value.
<note>If the specified value is represented as a numerical value, it is either the actual value, a range of possible values,
a limit for the value of a Property_value, or a list of these.</note> *)
  specified_value : property_value_select;
END_ENTITY;

(* A Property_value_function is a mechanism for representing a function that is used to calculate the relative value (with
regard to optimality) of a Property_value. *)
ENTITY property_value_function;
  (* The defines_property_value_merit specifies the Property_value whose relative value is calculated by the function_
specification. *)
  defines_property_value_merit : property_value;
  (* The function_specification specifies the function used for determining the relative value of the Property_value defined
by the attribute defines_property_value_merit. *)
  function_specification : text_select;
END_ENTITY;

(* A Property_value_relationship is a relationship between two Property_value objects. *)
ENTITY property_value_relationship;
  (* The related specifies the second Property_value in the relationship. *)
  related : property_value;
  (* The relating specifies the first Property_value in the relationship. *)
  relating : property_value;
  (* The relation_description specifies additional information about the Property_value_relationship. *)
  relation_description : OPTIONAL text_select;
  (* The relation_type specifies the meaning of the relationship. <note>The PAS 20542 group shall consider defining a list
of preferred values for this attribute.</note> *)
  relation_type : label;
END_ENTITY;

(* A Rank_assignment is the assignment of an element to a particular Ranking_element. <note>The assignment to a
particular priority is assumed to be relative to a set of alternatives.</note> *)
ENTITY rank_assignment;
  (* The assigned_rank specifies the Ranking_element to which the element is assigned. *)
  assigned_rank : ranking_element;
  (* The assigned_to_element specifies the element that is assigned. *)
  assigned_to_element : ranked_element_select;
  (* The assignment_criteria specifies the criteria that were applied for assigning the assigned element to a particular
Ranking_element. *)
  assignment_criteria : OPTIONAL label;
```

```
  (* The assignment_description specifies additional information about the Rank_assignment. *)
  assignment_description : OPTIONAL text_select;
  (* The relevant_elements specifies a Rank_group that holds a number of elements ranked according to the same crite-
ria. *)
  relevant_elements : rank_group;
END_ENTITY;

(* A Rank_group is a placeholder for elements that are compared and ranked in accordance with a specified criterion. *)
ENTITY rank_group;
  (* The classification_criteria specifies the purpose of the classification_criteria. It defines the criteria that are common to
all elements assigned to a Rank_group. *)
  classification_criteria : ranking_type;
  (* The name specifies the word, or set of words, that is used to refer to the Rank_group. *)
  name : label;
INVERSE
  (* The relevant_elements specifies a Rank_group that holds a number of elements ranked according to the same crite-
ria. *)
  compared_element : SET [0 : ?] OF rank_assignment FOR relevant_elements;
END_ENTITY;

(* A Rank_relation is the the mechanism for representing partially orders between two Ranking_element objects. *)
ENTITY rank_relation;
  (* The higher_rank specifies the higher priority Ranking_element. *)
  higher_rank : ranking_element;
  (* The lower_rank specifies the lower priority Ranking_element. *)
  lower_rank : ranking_element;
END_ENTITY;

(* A Ranking_element is the representation of a distinct priority level in a Ranking_system. *)
ENTITY ranking_element;
  (* The description specifies additional information about the Ranking_element. *)
  description : OPTIONAL text_select;
  (* The name specifies the word, or set of words, that is used to refer to a Ranking_element. *)
  name : label;
INVERSE
  (* The lower_rank specifies the lower priority Ranking_element. *)
  higher_ranked_element : SET [0 : ?] OF rank_relation FOR lower_rank;
  (* The higher_rank specifies the higher priority Ranking_element. *)
  lower_ranked_element : SET [0 : ?] OF rank_relation FOR higher_rank;
END_ENTITY;

(* A Ranking_system is a mechanism for representing discrete ranges of priority levels. *)
ENTITY ranking_system;
  (* The description specifies additional information about the Ranking_system. *)
  description : OPTIONAL text_select;
  (* The highest_rank specifies the highest priority level in the Ranking_system. *)
  highest_rank : ranking_element;
  (* The name specifies the word, or set of words, that are used to refer to the Ranking_system. *)
  name : label;
END_ENTITY;

(* A Real_data_type_definition is a type of Elementary_maths_space that includes all real values. *)
ENTITY real_data_type_definition
SUBTYPE OF (elementary_maths_space);
END_ENTITY;

(* A Real_interval is a type of Maths_space that is a bounded sub-set of all real values. *)
ENTITY real_interval
ABSTRACT SUPERTYPE OF (ONEOF(finite_real_interval, hibound_real_interval, lobound_real_interval))
SUBTYPE OF (maths_space);
END_ENTITY;
```

(* A Realized_system is the reference to a physically realized system. A Realized_system is realized in accordance with a specification identified by the realisztion_of attribute. <note number="1">The data model has been extended to cover real systems in order to correctly represent verification plans and verification results.</note> *)
ENTITY realized_system;
 (* The description specifies additional information on the Realized_system. *)
 description : text_select;
 (* The id specifies the identifier of the Realized_system. <note number="2">The id typically specifies the serial number of the Realised_system.</note> *)
 id : element_identifier;
 (* The name specifies the word, or set of words, that is used to refer to the Realized_system. *)
 name : label;
 (* The realization_of specifies the system_instance that provided the specification for the realization of the Realized_system. *)
 realization_of : system_instance;
UNIQUE
 UR1: id, realization_of;
END_ENTITY;

(* A Realized_system_composition_relationship is the hierarchy relationship between two Realized_system objects. The super-system in the composition is identified via the system attribute and the sub-system is identified via the component attribute. <note>The Realized_system_composition_relationship specifies that a system of system structure has been formed for some purpose. In the scope of PAS 20542, the most likely reason is for verification that realized systems are conformant with the specification that the realization was based on.</note> *)
ENTITY realized_system_composition_relationship;
 (* The component specifies the sub-system in the Realized_system_composition_relationship. *)
 component : realized_system;
 (* The description specifies additional information on the Realized_system_composition_relationship. *)
 description : text_select;
 (* The mirror_of specifies the System_composition_relationship that is realized in a physical system and indicated by the Realized_system_composition_relationship. *)
 mirror_of : system_composition_relationship;
 (* The system specifies the super-system in the Realized_system_composition_relationship. *)
 system : realized_system;
END_ENTITY;

(* A Record_data_type_definition is a type of User_defined_data_type_definition that combines a number of values into a single item (a record). There is a dependency between the combined values that limits their use independently. <example number="27">Assume the following declaration:
record EmployeeDetails
name : String; date_of_birth : Date; age : Integer Derived = TODAY - date_of_birth; end record; The date_of_birth is the date_of_birth of the employee named in the record. If the date of birth is separated from the rest of the details of the employee then it becomes just a date. </example> *)
ENTITY record_data_type_definition
SUBTYPE OF (user_defined_data_type_definition);
END_ENTITY;

(* A Record_data_type_member is a relationship between a Record_data_type_definition and a Data_field it contains. There is no order amongst the occurrences. All of the members of a Record_data_type_definition must be present. *)
ENTITY record_data_type_member;
 (* The child specifies the Data_field object in the relationship. *)
 child : data_field;
 (* The parent specifies the Record_data_type_definition in the relationship. *)
 parent : record_data_type_definition;
END_ENTITY;

(* A Recursive_data_type_definition is a type of User_defined_data_type_definition and a redefinition of another data type definition. At its simplest a Recursive_data_type_definition renames or re-describes another data type. At its most complex, a Recursive_data_type_definition attaches new properties, possibly overriding ones of the redefined User_defined_data_type_definition or Maths_space. A recursive data type inherits the properties of the data type it redefines. A recursive data type may have properties that the data type it redefines does not have. A recursive data type can have alternate values for properties that the type it redefines has. Redefined properties may conflict

```
with existing properties. In cases of conflict it shall be assumed that the properties of the recursive_data_type_definition takes precedence over the properties of the redefined data type.</p> *)
ENTITY recursive_data_type_definition
SUBTYPE OF (user_defined_data_type_definition);
  (* The redefines specifies the Maths_space or User_defined_data_type_definition that the Recursive_data_type_definition is based on. *)
  redefines : data_type_definition_select;
END_ENTITY;

(* A Requirement_allocation_property_relationship is a mechanism for relating a Requirement_instance allocated to an element to a Property_value assigned to that element. <note>This construct allows for tracing the consequences or implications of allocating a requirement to an element.</note> *)
ENTITY requirement_allocation_property_relationship;
  (* The allocated_requirement specifies the requirement in the relationship. *)
  allocated_requirement : requirement_allocation_relationship;
  (* The define_property_value specifies the Property_value that is influenced or defined by the requirement. *)
  define_property_value : property_value;
  (* The description specifies additional information about the Requirement_allocation_property_relationship. *)
  description : OPTIONAL text_select;
END_ENTITY;

(* A Requirement_allocation_relationship is the mechanism for allocating Requirement_instance to another element such that the element shall fulfil or has been found compliant with the capabilities stated by the requirement. <note>The semantics of the Requirement_allocation_relationship are further defined by the role attribute.</note> *)
ENTITY requirement_allocation_relationship
SUPERTYPE OF (specific_requirement_allocation_relationship);
  (* The description specifies additional information about the Requirement_allocation_relationship. *)
  description : OPTIONAL text_select;
  (* The relation_to specifies the element that the Requirement_instance is allocated to. *)
  relation_to : requirement_allocation_select;
  (* The requirement specifies the Requirement_instance in the relationship. *)
  requirement : requirement_instance;
  (* The role specifies the semantics of the relation. Where applicable the following values shall be used:<ul> <li>allocation: The application_object specifies an allocation relationship where the requirement object is allocated to the relation_to object. The allocation relation means that the relation_to object shall be designed in such a way that the requirement is fulfilled;</li> </li>fulfils: The application_object specifies a fulfils relationship where the requirement stated in the requirement object is fulfilled by the relation_to object. The fulfils relation means that the relation_to object has been evaluated and found to be compliant with the requirement object.</li> </ul> *)
  role : label;
END_ENTITY;

(* A Requirement_class is a collector for requirements of similar characteristics. <note number="1">The standard does not specify a fixed structure for requirement classification. Instead a dynamical structure is provided by the Requirement_class and the Requirement_class_relationship.</note> *)
ENTITY requirement_class;
  (* The description specifies additional information about the Requirement_class. <note number="2">The description may carry information identifying the characteristics of requirements that are in scope of the description.</note> *)
  description : OPTIONAL text_select;
  (* The id specifies the identifier of the Requirement_class. *)
  id : element_identifier;
  (* The name specifies the word, or set of words, that is used to refer to the Requirement_class. *)
  name : label;
UNIQUE
  UR1: id;
END_ENTITY;

(* A Requirement_class_relationship is a relationship between two Requirement_class objects. <note>The semantics of the relationship are specified by the relationship_type attribute.</note> *)
ENTITY requirement_class_relationship;
  (* The description specifies additional textual information about the specialisation: The related_class Requirement_class is a specialisation of the relating_class Requirement_class. *)
```

```
  description : OPTIONAL text_select;
  (* The related_class specifies the second Related_class in the relationship. *)
  related_class : requirement_class;
  (* The relating_class specifies the first Relating_class in the relationship. *)
  relating_class : requirement_class;
  (* The relationship_type specifies the nature of the relationship. Where applicable one of the following values shall be
used:<ul> <li>specialization: The related_class Requirement_class is a specialization of the relating_class Require-
ment_class;</li. <li>equivalence: The related_class Requirement_class is a synonym of the relating_class Requirement_
class.</li> </ul> *)
  relationship_type : label;
END_ENTITY;

(* A Requirement_composition_relationship represents the decomposition relationship between a Requirement_defini-
tion and a Requirement_occurence. <note number="1">A Requirement_definition object may be decomposed into any
number of Requirement_occurence objects.</note> <note number="2">One important assumption is that the model
will be employed for representation of multiple versions or variants of a system specification. Consequently there is a
substantial benefit if common design elements among the versions could be shared or reused. These assumptions have
led to a model where four entities are involved in representing a single requirement as seen from a user's point of view.
The structure is outlined below. <p>The entity Requirement_definition holds the description of what is required. In the
model there are three sup-types of this entity that support different representations of what is required. No assumptions
are made as to the context in which the requirement is used which means that no inter-requirement relationships are
defined on the Requirement_definition entity. A Requirement_definition object may provide definition for any number of
requirement_occurence objects.</p><p>The Requirement_instance entity is a context dependent representation of a
requirement. A Requirement_instance object may be assigned to a System_view object (in the system architecture UoF).
All relationships between requirements are defined on the Requirement_instance entity.</p><p>The requirement_com-
position_relationship entity is the means for decomposing requirements. For each added requirement in a composition
a new Requirement_composition_relationship object has to be instantiated.</p><p>The Requirement_occurence entity
is an intermediate representation of a requirement. It is included in the model to allow for maximum traceability and
for improving the support for reusing a requirement in more than one context (different system versions or systems). A
requirement_occurence object using exactly one Requirement_definition object as its definition may serve as a definition
for any number of Requirement_instance objects.</p><p>Consequently, the requirement_occurence entity separates the
static requirement breakdown structure from the dynamic system specific requirement representation (represented by the
Requirement_instance entity) and it implements loose coupling between parent and child requirements in a requirement
breakdown structure.</p><p>The first aspect is important as it allows multiple context dependent objects ( Requirement_
instance ) to reference the same context independent object.</p> <p>The second aspect is that a Requirement_defini-
tion object that is part of a composition (via a requirement_occurence and requirement_composition_relationship objects)
is not tightly linked to the parent Requirement_definition object. The construct allows the information model to represent
a specific requirement as part of a requirement composition hierarchy in one situation while in another situation it may be
viewed as a top-level requirement.</p></note> *)
ENTITY requirement_composition_relationship;
  (* The child_requirement specifies the decomposing Requirement_occurence object in the relationship. *)
  child_requirement : requirement_occurence;
  (* The description specifies additional textual information about the Requirement_composition_relationship. *)
  description : OPTIONAL text_select;
  (* The index specifies an identifier for the Requirement_occurence identified by the child_requirement in the context of
the Requirement_definition defined by the parent_definition attribute. <note number="3">The index is the means of rep-
resenting the identifier often used to identify a requirement in requirement management tools. In PAS 20542, this idenfier
is built up stepwise one position at a time. If numerical indexing is used and a particular child_requirement is the third in
the composition defined by the parent_definition, then the index attribute shall be 3.</note> *)
  index : label;
  (* The parent_definition specifies the decomposed Requirement_definition n the relationship. *)
  parent_definition : requirement_definition;
UNIQUE
  UR1: index, parent_definition;
END_ENTITY;

(* A Requirement_definition is the context independent definition of a requirement. <note>The combination of Requirement_
definition, Requirement_occurrence and Requirement_composition_relationship objects define the mechanism for breaking
down complex requirements to their basic parts. The use of three separate entities allows for representing that a particular
requirement_definition object is part of several breakdown hierarchies and the use of a Requirement_occurrence object for
each individual hierarchy allows for the unambiguous and insulated representation of each breakdown hierarchy.</note> *)
```

```
ENTITY requirement_definition
ABSTRACT SUPERTYPE OF (ONEOF(model_defined_requirement_definition, structured_requirement_definition, textu-
al_requirement_definition));
  (* The associated_version specifies the Configuration_element_version for the Requirement_definition. *)
  associated_version : configuration_element_version;
  (* The id specifies the identifier of the Requirement_definition. *)
  id : element_identifier;
  (* The name specifies the word, or set of words, that is used to refer to the Requirement_definition. *)
  name : OPTIONAL label;
INVERSE
  (* The parent_definition specifies the decomposed Requirement_definition n the relationship. *)
  composed_of : SET [0 : ?] OF requirement_composition_relationship FOR parent_definition;
  (* The requirement specifies the Requirement_definition in the assignment. *)
  in_requirement_class : SET [0 : 1] OF requirement_requirement_class_assignment FOR requirement;
UNIQUE
  UR1: id;
END_ENTITY;

(* A Requirement_instance is a context dependent representation of a requirement for a particular system. <note>The
Requirement_occurence and Requirement_definition objects provide the definition of what is required.</note> *)
ENTITY requirement_instance;
  (* The definition specifies the Requirement_occurrence that provides additional information on what is required. *)
  definition : requirement_occurence;
  (* The id specifies the identifier of the Requirement_instance. *)
  id : element_identifier;
  (* The name specifies the word, or set of words, that is used to refer to the Requirement_instance. *)
  name : label;
INVERSE
  (* The associated_requirement specifies the Requirement_instance in the relationship. *)
  implied_external : SET [0 : ?] OF implied_external_interaction FOR associated_requirement;
UNIQUE
  UR1: definition, id;
END_ENTITY;

(* A Requirement_occurence is the placeholder for a static instance of a requirement in a requirement breakdown hierar-
chy. *)
ENTITY requirement_occurence;
  (* The definition specifies the Requirement_definition object that provides the definition of what is required. *)
  definition : requirement_definition;
  (* The id specifies the identifier of the Requirement_occurrence. *)
  id : element_identifier;
  (* The name specifies the word, or set of words, that is used to refer to the Requirement_occurence. *)
  name : label;
INVERSE
  (* The child_requirement specifies the decomposing Requirement_occurence object in the relationship. *)
  child_of : SET [0 : ?] OF requirement_composition_relationship FOR child_requirement;
UNIQUE
  UR1: definition, id;
END_ENTITY;

(* A Requirement_relationship is a means of representing relationships among Requirement_instance objects. <note
number="1">The Requirement_relationship may record the existence of relationships among a set of requirements or
may be used to capture relationships between sets of requirements and individual requirements.</note> <note num-
ber="2">The Requirement_relationship may be used, for example, to identify alternate or derived between requirements
in different System_view objects.</note> <note number="3">There are two ways of using the Requirement_relationship:
<ol><li>To relate a number of requirements. The purpose of the relation is specified by the relationship type attribute.
Any number of Requirement_relationship_input_assignment objects relating Requirement_relationship objects and a
number of Requirement_instance objects maintains such a relation. In this use no new Requirement_relationship_result-
ing_relationship are instantiated as a result of the establishment of the Requirement_relationship.</li><li>The creation
or identification of requirements that are dependent on an established requirement_relationship. As in the first case, any
number of Requirement_relationship_input_assignment objects relating a requirement_relationship and a number of
```

```
Requirement_instance objects build up the input to the relationship. To indicate the output of the analysis any number of Requirement_relationship_resulting_relationship objects may be used to indicate requirements created or identified as results based on the requirement relationship.</li></note> *)
ENTITY requirement_relationship;
  (* The description specifies additional information on the Requirement_relationship. *)
  description : OPTIONAL text_select;
  (* The relationship_type specifies the type of relationship. Where applicable one of the following values shall be used:<ul> <li>alternate: The requirements associated via Requirement_relationship_input_assignment objects to the Requirement_relationship are mutually exclusive or equivalent;</li> <li>derived: The requirements associated via Requirement_relationship_input_assignment objects are used to derive additional Requirement_instance objects. The new requirements are indicated using Resulting_relationship objects.</li> <ul> *)
  relationship_type : label;
END_ENTITY;

(* A Requirement_relationship_context_assignment is the mechanism for associating a Requirement_relationship or Requirement_allocation_relationship to a System_view for which the relationship is valid. <note>The Requirement_relationship_context_assignment allows for specifying that a relationship is valid for more than one System_view, which allows for reuse of the specification elements over multiple versions of a system specification.</note> *)
ENTITY requirement_relationship_context_assignment;
  (* The assigned_requirement_relationship specifies the relationship assigned to a system_view by a assigned_requirement_relationship.
  *)
  assigned_requirement_relationship : assigned_requirement_relationship_select;
  (* The description specifies additional information about the Requirement_relationship_context_assignment. *)
  description : OPTIONAL text_select;
  (* The system_context specifies the system_view that the requirement relationship is assigned to. *)
  system_context : system_view;
END_ENTITY;

(* A Requirement_relationship_input_assignment is the mechanism for assigning a Requirement_instance to a Requirement_relationship. *)
ENTITY requirement_relationship_input_assignment;
  (* The assigned_instance specifies the Requirement_instance in the relation. *)
  assigned_instance : requirement_instance;
  (* The input_requirement specifies the Requirement_relationship in the relation. *)
  input_requirement : requirement_relationship;
END_ENTITY;

(* A Requirement_relationship_resulting_relationship is the mechanism for relating a Requirement_instance to a Requirement_relationship indicating that the Requirement_instance was implied by the Requirement_relationship. *)
ENTITY requirement_relationship_resulting_relationship;
  (* The motivation specifies additional information about why the Requirement_relationship_resulting_relationship was created. *)
  motivation : OPTIONAL text_select;
  (* The Requirement_relationship specifies the Requirement_relationship. *)
  requirement_relationship : requirement_relationship;
  (* The resulting_requirement specifies the newly created requirement. *)
  resulting_requirement : requirement_instance;
  (* The role ought to be deleted! *)
  role : label;
END_ENTITY;

(* A Requirement_requirement_class_assignment is the mechanism for assigning a Requirement_definition to a Requirement_class such that the characteristics of the Requirement_definition are compliant with those defined for the Requirement_class. *)
ENTITY requirement_requirement_class_assignment;
  (* The class specifies the Requirement_class in the assignment. *)
  class : requirement_class;
  (* The motivation specifies the reason for the assignment. *)
  motivation : OPTIONAL text_select;
  (* The requirement specifies the Requirement_definition in the assignment. *)
```

```
  requirement : requirement_definition;
END_ENTITY;

(* A Requirement_system_view_assignment is the mechanism for assigning a Requirement_instance to a System_view
indicating that the requirement applies to the System_view. *)
ENTITY requirement_system_view_assignment
SUPERTYPE OF (root_requirement_system_view_assignment);
  (* The description specifies additional information about the Requirement_system_view_assignment. *)
  description : OPTIONAL text_select;
  (* The requirement specifies the Requirement_instance in the relationship. *)
  requirement : requirement_instance;
  (* The system_view specifies the System_view in the relationship. *)
  system_view : system_view;
END_ENTITY;

(* A Requirement_traces_to_requirement_relationship is a relationship between two Requirement_instance objects
assigned to different Partial_system_view objects establishing a traceability link between the two requirements. <note
number="1">The relationship is valid for a particular System_view only.</note> <note number="2">This relationship
is motivated by the need to maintain traceability between different System_view objects of a system. Furthermore it is
implied that the source and traced requirements are not assigned to the same System_view.</note> *)
ENTITY requirement_traces_to_requirement_relationship;
  (* The motivation specifies additional textual information about the nature of the Requirement_traces_to_requirement_
relationship. *)
  motivation : OPTIONAL text_select;
  (* The source_requirement specifies the first Requirement_instance in the relationship. *)
  source_requirement : requirement_instance;
  (* The traced_requirement specifies the second Requirement_instance in the relationship. *)
  traced_requirement : requirement_instance;
  (* The valid_context specifies the System_view for which the Requirement_traces_to_requirement_relationship is valid.
*)
  valid_context : system_definition;
END_ENTITY;

(* A Root_requirement_system_view_assignment is a type of Requirement_system_view_assignment and the mecha-
nism for assigning a Requirement_instance which is the root of a requirement composition structure to a System_view.
For example, the assigned requirement shall not be the child in a requirement composition structure.<p>The index attri-
bute defines the first element in the presentation index for the requirement in the particular system.</p> *)
ENTITY root_requirement_system_view_assignment
SUBTYPE OF (requirement_system_view_assignment);
  (* The index specifies an identifier for presentation of the Requirement_instance identified by the attribute require-
ment inherited from Requirement_system_view_assignment in the context of the System_view defined by the sys-
tem_view attribute. <example number="28">The index for a particular For example, the assigned requirement shall not
be the child in a requirement composition structure may be 5, indicating that the Requirement_instance identified by
the requirement attribute (inherited from Requirement_system_view_assignment) is the fifth requirement for the Sys-
tem_view identified by the system_view attribute (likewise inherited from Requirement_system_view_assignment).</
example> <note>The index is the means for representing the identifier often used to identify a requirement in require-
ment management tools. In PAS 20542, this idenfier is built up stepwise one position at a time for each level in the
composition. For example, if numerical indexing is used and a particular child_requirement is the third in the composi-
tion defined by the system_view attribute then the index attribute shall be "3".</note> <p>Further decomposition of the
requirement shall add information to the index one position at a time. For example, if the requirement is decomposed
into 4 sub-requirements then these may be indexed 1..4 using the index attribute of the Requirement_composition_re-
lationship.</p><p>The  index of a requirement in a composition structure is reconstructed  by concatinating the index
attributes of each Requirement_composition_relationship encountered while traversing a decomposed requirement
from the topmost item to a leaf requirement.</p> *)
  index : label;
UNIQUE
  UR1: index, SELF\requirement_system_view_assignment.system_view;
END_ENTITY;

(* A Selection_package is a type of Package with the discriminator that the number of elements that may be selected
from the Package for a particular system is constrained. <note number="1">The attribute selection_type specifies
```

the semantics of the selection_package.</note> <note number="2">The selection_package allows a specification that provides a constraint that only one of the elements may be selected from the elements in the Package.</note> *)
ENTITY selection_package
SUBTYPE OF (package);
(* The selection_type specifies the selection constraint. Where applicable one of the following values shall be used:<ul and: One element of the package may be selected for a particular system; or: Any number of elements of the package may be selected for a particular system. *)
selection_type : label;
END_ENTITY;

(* A Single_cardinality is a single integer representing a numerical constraint. *)
ENTITY single_cardinality;
(* The defined_value specifies the numerical constraint. *)
defined_value : single_cardinality_select;
END_ENTITY;

(* A Specific_requirement_allocation_relationship is a type of Requirement_allocation_relationship that serves as a mechanism for the allocation or tracing of a requirement_instance to an element that cannot be unambiguously identified in a functional breakdown structure, for example, an Fsm_generic_state or Cb_place. <note number="1">The method for identifying elements in the breakdown structure is to identify the Functionality_instance_reference closest in the breakdown structure to the Fsm_generic_state or Cb_place via the related_to attribute and to specifically identify the traced object via the specific_element attribute.</note> <notev number="2">In cases where the Specific_requirement_allocation_relationship indicates a requirement allocation to an Fsm_generic_state, the related_to attribute shall identify the Fsm_model object that defines the context of the finite state machine.</note> <note number="3">In cases where the Specific_requirement_allocation_relationship indicates a requirement allocation to a Cb_place, then the related_to attribute shall identify the Functionality_instance_reference that, in turn, identifies the Composite_function_definition (via a Function_instance) which is constrained by the Functional_behaviour_model of which the Cb_place identified by the Specific_element attribute is an element.</note> *)
ENTITY specific_requirement_allocation_relationship
SUBTYPE OF (requirement_allocation_relationship);
(* The specific_element specifies the object that the requirement is allocated to via the Specific_requirement_allocation_relationship. *)
specific_element : specific_element_select;
WHERE
WR1: 'SYSTEMS_ENGINEERING_DATA_REPRESENTATION.FUNCTIONALITY_INSTANCE_REFERENCE' IN TYPEOF(SELF\requirement_allocation_relationship.relation_to);
END_ENTITY;

(* A Specification_state_assignment is the mechanism for assigning a Textual_specification to a Fsm_state such that the specification is implemented by the state. *)
ENTITY specification_state_assignment;
(* The assigned_to specifies the Fsm_state in the assignment. *)
assigned_to : fsm_state;
(* The specification specifies the Textual_specification in the assignment. *)
specification : textual_specification;
END_ENTITY;

(* A Start_order is a type of Work_order that serves as the authorization for one or more Engineering_process_activity objects to be performed. <note>A Start_order captures all kinds of work orders except for order for changes - which is handled by Change_order.</note> *)
ENTITY start_order
SUBTYPE OF (work_order);
(* The start_order_type specifies the type of activity authorized by the Start_order. Where applicable one of the following values shall be used:
design_activity: authorisation to start design activity for a system; impact_analysis: authorisation to start analysis activity for identifying the impact of a critical issue. *)
start_order_type : label;
END_ENTITY;

(* A Start_request is a type of Work_request that solicits the commencement of some work. <note>A Start_request captures all kinds of requests to start activities except for change related information.</note> *)
ENTITY start_request

```
SUBTYPE OF (work_request);
  (* The request_type specifies the nature of the Start_request. Where applicable, one of the following values
shall be used:<ul><li>cost_reduction: A request aimed at reducing the engineering and manufacturing costs of an
item;</li> <li>customer_rejection: A request resulting from a rejection by a customer;</li> <li>customer_request:
A request for an activity that is necessary to solve the request of a customer;</li> <li>durability_improvement: A
request aimed at extending the life time of an item;</li> <li>government_regulation: A request resulting from legal
requirements;</li> <li>procurement_alignment: A request to adjust the purchasing process of different items;</li>
<li>security_reason: A request for an activity that is necessary from a security point of view;</li> <li>standardiza-
tion: A request to unify variants of an item;</li> <li>supplier_request: A request for an activity necessary to solve
the request of a supplier;</li> <li>technical_improvement: A request aimed at improving the technical aspects of an
item.</li> </ul> *)
  request_type : label;
END_ENTITY;

(* A State_context_relationship is a mechanism for relating a Fsm_generic_state to a Functional_state_context indicating
that the Fsm_state is a top-level state in the Functional_state_context. <note>This entity provides the link between a
Fsm_state (top-level state if a statechart) and the environment in which a finite state machine exists.</note> *)
ENTITY state_context_relationship;
  (* The in_context specifies the Functional_state_context in the relationship. *)
  in_context : generic_state_context;
  (* The state specifies the Fsm_generic_state in the relationship. *)
  state : fsm_generic_state;
END_ENTITY;

(* A State_function_interaction_port is an element of the interface to a Functional_state_context. <note>The State_func-
tion_interaction_port is a means for importing knowledge of functions outside a Functional_state_context into a Function-
al_state_context. Explicit import of such knowledge is required as the Functional_state_context maintains it own name
space.</note> *)
ENTITY state_function_interaction_port;
  (* The port_of specifies the Functional_state_context interface of which the State_function_interaction_port is part. *)
  port_of : functional_state_context;
END_ENTITY;

(* A State_machine_functional_behaviour_model is a type of Functional_behaviour_model with the discriminator dynamic
behaviour is captured using finite state machine and related formalisms. *)
ENTITY state_machine_functional_behaviour_model
SUBTYPE OF (functional_behaviour_model);
INVERSE
  (* The behaviour_model specifies the State_machine_functional_behaviour_model object for which the Fsm_model
provides the behaviour specification. *)
  behaviour_constraint : SET [1 : ?] OF fsm_model FOR behaviour_model;
END_ENTITY;

(* A state_transition_specification_assignment is an assignment of a textual specification to a Fsm_state_transition.
<note
number="1">
The state_transition_specification_assignment is the mechanism for assigning a guarding expression to a Fsm_state_
transition in a Moore style finite state machine or the assignment of a guarding expression and a set of action statements
to a Fsm_state_transition in a Mealy style finite state machine.
</note><note
number="2">
The language of the assigned specification is not prescribed by AP 233.
</note> *)
ENTITY state_transition_specification_assignment;
  (* The assigned_to specifies the Fsm_state_transition to which the textual specification is assigned.
 *)
  assigned_to : fsm_state_transition;
  (* The specification specifies the textual specification assigned to the Fsm_state_transition.
 *)
  specification : textual_specification;
END_ENTITY;
```

(* A String_data_type_definition is a type of Elementary_maths_space containing all of the strings up to length n that can be formed from combinations of the allowable characters. Where n is the length of the string. If n is not given, the string can have infinite length. *)
ENTITY string_data_type_definition
SUBTYPE OF (elementary_maths_space);
(* The size specifies the space occupied by the String_data_type_definition. <note number="1">The low bound of the Finite_integer_interval is generally not used but is available when a future use (such as the identification of sub-strings within strings) becomes apparent . The low bound should be set to zero.</note> <note number="2">The size of the Finite_integer_interval is the maximum number of characters and instance of a String_data_type_definition can have. </note> A size need not be given for a String_data_type_definition in which case it has an unknown length. The size need not be specified for a particular String_data_type_definition object. *)
size : OPTIONAL finite_integer_interval;
END_ENTITY;

(* A Structured_requirement_definition is a type of Requirement_definition with the discriminator that the required capability is expressed as a value. <note number="1">This entity provides the capability to represent requirements that are structured and typed - as opposed to requirements expressed in text.</note> <note number="2">The name of this entity shall be reconsidered. The current name reflects the fact that the property model is used to store the requirement. A potentially better name is Structured_requirement_definition.</note> *)
ENTITY structured_requirement_definition
SUBTYPE OF (requirement_definition);
(* The required_characteristic specifies the Property_value that provides the structured definition of the requirement. *)
required_characteristic : property_value;
END_ENTITY;

(* A System_composition_relationship is the decomposition relationship between a System_definition object and a System_instance object that is a sub-system of the System_definition object. *)
ENTITY system_composition_relationship;
(* The component_system specifies the sub-system in the relationship. *)
component_system : system_instance;
(* The decomposed_system specifies the system in the relationship. *)
decomposed_system : system_definition;
(* The description specifies additional textual information about the relationship. *)
description : OPTIONAL text_select;
(* The relationship describes the type of the System_composition_relationship. Where applicable one of the following values shall be used: mandatory: The sub-system must be in the system breakdown structure of a real system for it to be valid; optional: The sub-system may, but is not required to, be in the system breakdown structure for any real system. *)
relationship_type : label;
END_ENTITY;

(* A System_definition is a type of System_view and a representation of a system specification for a life cycle of a system. <note>The specification may be in any degree of completeness. The system_definition, via relationships to the collection of detailed entities (function_definition etc), forms the system specification at that point in its evolution. At completion, the System_definition will form the complete system specification for the system life cycle that the System_definition defines.</note> *)
ENTITY system_definition
SUBTYPE OF (system_view);
(* The life_cycle_stage specifies the phase in the system life-cycle the System_definition is valid for. <example number="29">The value for life_cycle_stage may be "operational" indicating that the System_definition provides the specification for the operational stage of the system under specification.</example> *)
life_cycle_stage : label;
INVERSE
(* The system_definition_context specifies the System_definition object for which the Partial_system_view is valid. *)
assigned_to_system : SET [0 : ?] OF partial_system_view_relationship FOR system_definition_context;
(* The system_specification specifies the System_definition to which the Partial_system_view is assigned. *)
scenarios_for_system : SET [0 : ?] OF system_view_assignment FOR system_specification;
END_ENTITY;

(* A system_functional_configuration is the mechanism for assigning a functional_reference_configuration to a context_function_relationship.

<note>
The system_functional_configuration defines the non-functional characteristics such as property values and allocation information that is valid in one particular view on a system.
</note> *)
ENTITY system_functional_configuration;
(* The functional_configuration specifies the functional_reference_configuration that is assigned by the functional_configuration.
*)
functional_configuration : functional_reference_configuration;
(* The system specifies the context_function_relationship to which the functional_reference_configuration is assigned.
*)
system : context_function_relationship;
END_ENTITY;

(* A System_instance is the instantiation of a system specification in a particular context. <note number="1">A System_instance does not imply a realized system, only that a particular specification is planned to be used for the realization of a system in a system composition structure.</note> <note number="2">Each System_instance gets its definition from exactly one System_definition object, defined by the attribute definition.</note> *)
ENTITY system_instance;
(* The definition specifies the System_definition object that provides the specification for a particular definition. There is exactly one System_definition object acting as the definition for a particular definition.
*)
definition : system_definition;
(* The description specifies additional information about the System_instance. *)
description : OPTIONAL text_select;
(* The id specifies the element_identifier of the System_instance. *)
id : element_identifier;
(* The name specifies the word, or set of words, that is used to refer to the System_instance. *)
name : label;
UNIQUE
UR1: definition, id;
END_ENTITY;

(* A system_instance_relationship is a representation of an interaction relationship between two or more System_instance objects.
<note>
system_instance_relationship is used to capture the context of a system in an entity-relationship modelling style. The idea is to model a set of interacting systems and the relationships between them. This means defining the cardinality constraints between the systems involved in a relation. All objects involved in a system_instance_relationship must be part of the same system breakdown structure.
</note>
This is a new modelling philosophy from point of view of the model and corresponds to system context view argued for by David Oliver. An example of a system context diagram is available on page 210 in Engineering Complex Systems with Models and Objects, by David Oliver. The actual relationship is formed by System_instance, System_instance_relationship_port and system_instance_relationship objects. *)
ENTITY system_instance_relationship;
(* The description specifies additional information about the description.
*)
description : text_select;
(* The name specifies the word, or set of words, by which the name is referred.
*)
name : label;
INVERSE
(* The defined_relationship specifies the System_instance_relationship referenced. *)
connected_port : SET [2 : ?] OF system_instance_relationship_port FOR defined_relationship;
END_ENTITY;

(* A System_instance_relationship_port is the mechanism for associating a System_instance object to a System_instance_relationship object. <note>The system_instance_relationship_port also specifies the cardinality of the System_instance object involved in the relation.</note> <example number="30">In a cellular telephone system it may be appropriate to express that for each cell there may be in the range of 0-5000 terminals concurrently. In PAS 20542 this is expressed by instantiating a System_instance_relationship relationship between a cell systems System_instance and

cellular telephone system System_instance with System_instance_relationship_port objects capturing the cardinality information for each system in the relationship. For the example the system_instance_relationship_port for the cellular telephone system system_instance the cardinality figure would be the interval 0-5000. </example> *)
ENTITY system_instance_relationship_port;
(* The cardinality specifies the range of System_instance objects that are involved in the relation. Regardless of the cardinality specified, exactly one System_instance object is always specified by the port_of attribute. Cardinality defines the number of System_instance objects that are involved in the System_instance_relationship defined by the defined_relationship attribute. *)
cardinality : cardinality_association_select;
(* The defined_relationship specifies the System_instance_relationship referenced. *)
defined_relationship : system_instance_relationship;
(* The port_of specifies the System_instance referenced. *)
port_of : system_instance;
END_ENTITY;

(* A System_instance_replication_relationship is the relation between two System_instance objects where a new System_instance object replaces the original one in a system. <note>The entity does not adequately capture the requirement. Additional work is required.</note> *)
ENTITY system_instance_replication_relationship;
(* The rationale specifies the motivation for replacing the System_instance. *)
rationale : OPTIONAL text_select;
(* The replaced_application specifies the original System_instance. *)
replaced_application : system_instance;
(* The relacing_application specifies the replaced System_instance. *)
replacing_application : system_instance;
END_ENTITY;

(* A System_physical_configuration is the mechanism for assigning a Physical_reference_configuration to a Context_physical_relationship. *)
ENTITY system_physical_configuration;
(* The physical_configuration specifies the Physical_reference_configuration that is assigned by the System_physical_configuration. *)
physical_configuration : physical_reference_configuration;
(* The system specifies the Context_physical_relationship to which the Physical_reference_configuration is assigned. *)
system : context_physical_relationship;
END_ENTITY;

(* A System_substitution_relationship is a relationship that indicates that one System_composition_relationship may be substituted for another System_composition_relationship. The subjects of the substitution are the related component_system System_instance objects of both System_composition_relationship objects. The decomposed_system System_definition shall be the same in both System_composition_relationship objects. *)
ENTITY system_substitution_relationship;
(* The base specifies the System_composition_relationship that is replaceable. *)
base : system_composition_relationship;
(* The description specifies textual additional information on the System_substitution_relationship. *)
description : OPTIONAL text_select;
(* The substitute specifies the system_composition_relationship that may be used instead of the base System_composition_relationship. *)
substitute : system_composition_relationship;
END_ENTITY;

(* A System_view is a partial or complete view on a system under specification. <note number="1">A System_view is a collector for all information that is of interest for a system specification.</note> <note number="2">System_view captures the concept of a particular perspective to the system model, as if viewed from a specific stakeholder. It also provides all system related entities with basic capabilities such as version management, persistent Ids, names etc.</note> *)
ENTITY system_view
ABSTRACT SUPERTYPE OF (ONEOF(partial_system_view, system_definition));
(* The associated_version specifies the Configuration_element object for which the Configuration_element_version object represents a version. *)
associated_version : configuration_element_version;
(* The description specifies additional textual information about the System_view. *)

```
  description : OPTIONAL text_select;
  (* The id specifies the identifier of the System_view. *)
  id : element_identifier;
  (* The name specifies the word, or set of words, that is used to refer to the System_view. *)
  name : label;
INVERSE
  (* The associated_context specifies the System_view that is valid for the relationship. *)
  system : SET [0 : ?] OF context_function_relationship FOR associated_context;
UNIQUE
  UR1: id;
WHERE
  WR1: at_most_one_system_function_assigned(SELF);
END_ENTITY;

(* A System_view_assignment is the assignment of a Partial_system_view to a System_definition. <note>The Partial_
system_view assigned to the System_definition represents a part of the complete system specification.</note> *)
ENTITY system_view_assignment;
  (* The assigned_view specifies the assigned Partial_system_view. *)
  assigned_view : partial_system_view;
  (* The assignment_comment specifies additional textual information about the assignment. *)
  assignment_comment : OPTIONAL text_select;
  (* The system_specification specifies the System_definition to which the Partial_system_view is assigned. *)
  system_specification : system_definition;
END_ENTITY;

(* A System_view_context is a description of the fidelity and system viewpoint for a Partial_system_view. <note>The
System_view_context carries a number of attributes that would be more efficient if they could be assigned individually.</
note> *)
ENTITY system_view_context;
  (* The description specifies additional textual information about the System_view_context. *)
  description : OPTIONAL text_select;
  (* The fidelity specifies a classification of the level of specification detail the partial_system_view that references the
fidelity. <example number="31">The value for fidelity may be "user" to indicate that the associated partial_system_view
defines a systems specification at user level of detail.</example> *)
  fidelity : label;
  (* The system_viewpoint specifies the particular viewpoint that is valid for the System_view_context. <example num-
ber="32">The system_viewpoint might be "maintainer" indicating that a Partial_system_view referencing the sys-
tem_viewpoint provides specifications that are valid from a maintainer's point of view.</example> The system_viewpoint
may also be used to indicate that specifications of the referencing Partial_system_view are relevant for a very specific
domain. <example number="33">The system_viewpoint may be "temporal_safety_analysis" indicating that a partial_sys-
tem_view object referencing the system_viewpoint provides a sub-set of system specifications that defines the temporal
requirements of the system under specification .</example> *)
  system_viewpoint : label;
END_ENTITY;

(* A Textual_paragraph is a structured representation of text. <note number="1">The name shall probably be change to
reflect the fact that the entity does not provide a template only but also textual information.</note> *)
ENTITY textual_paragraph;
  (* The element_value specifies the textual component of the Textual_paragraph. *)
  element_value : LIST [1 : ?] OF text_elements;
  (* The name specifies the word, or set of words, that is used to refer to the Textual_paragraph. <note number="2">The
name attribute may be a reference to formatting information for the text.</note> *)
  name : label;
END_ENTITY;

(* A Textual_requirement_definition is a type of Requirement_definition and a definition of a requirement expressed in
text. <note>This is the most "primitive" representation of requirements in PAS 20542.</note> *)
ENTITY textual_requirement_definition
SUBTYPE OF (requirement_definition);
  (* The description specifies the required capabilities. *)
  description : text_select;
END_ENTITY;
```

(* A Textual_section is text in which constituent elements (paragraphs) are named. <note>The Textual_section is provided for situations where the structure of the textual information needs further formatting.</note> *)
ENTITY textual_section;
(* The element specifies the individual element of the template. *)
element : LIST [1 : ?] OF textual_paragraph;
(* The name specifies the word, or set of words, that is used to refer to the Textual_section. *)
name : label;
END_ENTITY;

(* A Textual_specification is text expressed using a defined langauge. <note number="1">The language may or may not be computer interpretable.</note> *)
ENTITY textual_specification;
definition : text_select;
(* The definition_language specifies with a word or set of words the language that the specification is written in. <note number="3"> Possible values for definition_language could be "ANSI C", "structured text", etc.</note> <note number="4">The list of preferred languages shall be revisited.</note> *)
definition_language : label;
END_ENTITY;

(* A Textual_table is a representation of a row structure where a Textual_section represents each position in the structure. <note number="1">A Textual_section may hold a list of Textual_paragraph objects. Each Textual_paragraph represents a column in the Textual_table.</note> *)
ENTITY textual_table;
(* The name specifies the word, or set of words, that is used to refer to the Textual_table. *)
name : label;
(* The table_row specifies the number of rows in theTextual_table. <note number="2">The number of rows shall be greater than zero.</note> *)
table_row : LIST [1 : ?] OF textual_section;
END_ENTITY;

(* A Triggered_system_view_relationship is a type of Partial_system_view_relationship and an indicator that TBD <note number="1">The model support specification of system modes (each Partial_system_view can be viewed as a high level mode of a system). A Triggered_system_view_relationship is a Partial_system_view_relationship where the relation between relating and related Partial_system_view is such that the system can make a transition from one mode to another when the transition_condition attribute is fulfilled.</note> <note number="2">The relationship is unidirectional, the Triggered_system_view_relationship only specifies a transition from the related to the relating Partial_system_view.</note> *)
ENTITY triggered_system_view_relationship
SUBTYPE OF (partial_system_view_relationship);
(* The transition_condition specifies, in text, the condition that shall be fulfilled. <note number="3">If the transition is fulfilled the TBD.</note> *)
transition_condition : text;
END_ENTITY;

(* An Undefined_data_type_definition is a type of User_defined_data_type_definition whose value domain and range are not defined. <note>The Undefined_data_type_definition shall be used in all cases where the data type of a Data_instance is unknown.</note> *)
ENTITY undefined_data_type_definition
SUBTYPE OF (user_defined_data_type_definition);
END_ENTITY;

(* A Union_data_type_definition is a type of User_defined_data_type_definition structure which can take the value of one of its components at any time. *)
ENTITY union_data_type_definition
SUBTYPE OF (user_defined_data_type_definition);
END_ENTITY;

(* An Union_data_type_member is a relationship between a Union_data_type_definition and a Data_field it contains. The members of a Union_data_type_definition are mutually exclusive. *)
ENTITY union_data_type_member;
(* The child specifies the Data_field object in the relationship
*)

```
  child : data_field;
  (* The parent specifies the Union_data_type_definition object in the relationship. *)
  parent : union_data_type_definition;
END_ENTITY;

(* A Unit is the occurrence of a unit of measurement. *)
ENTITY unit;
  (* The unit_name specifies the name of the unit of measurement, using (ideally) an appropriate ISO standard of
Unit terms. <note number="1">Examples would be: s (seconds), m (meters); kg (kilograms), etc.</note> <note num-
ber="2">The unit_name shall correspond with standard measurement types.</note> *)
  unit_name : label;
END_ENTITY;

(* A User_defined_data_type_definition is a data type with a value domain not directly defined within PAS 20542. *)
ENTITY user_defined_data_type_definition
ABSTRACT SUPERTYPE OF (ONEOF(abstract_data_type_definition, aggregate_data_type_definition, derived_data_
type_definition, record_data_type_definition, recursive_data_type_definition, undefined_data_type_definition, union_
data_type_definition));
  (* The associated_version specifies the Configuration_element_version for the User_defined_data_type_definition. *)
  associated_version : configuration_element_version;
  (* The description specifies additional information about the User_defined_data_type_definition. *)
  description : OPTIONAL text_select;
  (* The id specifies the identifier of the User_defined_data_type_definition. *)
  id : element_identifier;
  (* The name specifies the word, or set of words, that are used to refer to the User_defined_data_type_definition. *)
  name : label;
UNIQUE
  UR1: id;
END_ENTITY;

(* A Value_limit is a type of Value_with_unit and a qualified numerical value representing either the lower limit or the up-
per limit of a particular characteristic. <example number="34">"30.5 max" and "5 min" are examples for a Value_limit.</
example> *)
ENTITY value_limit
SUBTYPE OF (value_with_unit);
  (* The limit specifies the limit of the value.
 *)
  limit : NUMBER;
  (* The limit qualifier specifies the type of limit. Where applicable the following values shall be used:<ul> <li>max: The
specified limit is an upper limit;</li> <li>min: The specified limit is a lower limit.</li> </ul> *)
  limit_qualifier : label;
END_ENTITY;

(* A Value_list is an ordered collection of Value_with_unit objects for a Property_value. <example number="35">Prop-
erty_definition may be composed of different property values such as "mass", "speed" and "age" which are altogether
necessary in a given context. The value_list collects all of them in a given order, such that each is identifiable by its index
in the list.</example> *)
ENTITY value_list;
  (* The values specifies the ordered collection of Value_with_unit objects that together are provided for a Property value.
*)
  values : LIST [1 : ?] OF value_with_unit;
END_ENTITY;

(* A Value_range is a type of Value_with_unit and a pair of numerical values representing the range in which the value
of a Property_value shall lie. <note>This kind of tolerance may replace the nominal value of a Property_value. All values
within the range are considered to be equally good. Usually, this way of dimensioning is used for clearances of large
number of objects with a particular specification.</note> *)
ENTITY value_range
SUBTYPE OF (value_with_unit);
  (* The distribution_function specifies the forecasted distribution function of a large set of measurements. *)
  distribution_function : OPTIONAL textual_specification;
```

```
  (* The lower_limit specifies the minimum acceptable value of the Property_value that is constrained by the Value range.
*)
  lower_limit : NUMBER;
  (* The upper_limit specifies the maximum acceptable value of the Property_value that is constrained by the Value
range. *)
  upper_limit : NUMBER;
END_ENTITY;

(* A Value_with_unit is either a single numerical measure, or a range of numerical measures with upper, lower, or upper
and lower bounds. <note>Each value_with_unit is either a Value_limit, a Value_range, or a Nominal_value.</note> *)
ENTITY value_with_unit
ABSTRACT SUPERTYPE OF (ONEOF(nominal_value, value_limit, value_range));
  (* The significant_digits specifies the number of significant digits for a particular Value_with_unit. *)
  significant_digits : OPTIONAL INTEGER;
  (* The unit_component specifies the measurement unit valid for the Value_with_unit. *)
  unit_component : OPTIONAL unit;
END_ENTITY;

(* A Verification_report_for_verification_specification is the mechanism for relating the result of a Verification_specifica-
tion_allocation to a Verification_result. The result of the verification activity is captured by the verification_report_entry
attribute. *)
ENTITY verification_report_for_verification_specification;
  (* The description specifies additional information on the Verification_report_for_verification_specification. *)
  description : text_select;
  (* The specific_verification_element specifies the Verification_specification_allocation which has been used for verifica-
tion. *)
  specific_verification_element : verification_specification_allocation;
  (* The verification_report_entry sspecifies a description of the result of the verification. *)
  verification_report_entry : verification_result;
END_ENTITY;

(* A Verification_result is the verdict of a verification activity. *)
ENTITY verification_result;
  (* The description the overall verdict of a verification activity. Results from the application of individual verification_speci-
fication object to elements under verification is captured by the entity verification_report_for_verification_specification.
The description specifies additional information on the Verification_result. *)
  description : text_select;
  (* The id specifies the identifier of the Verification_result. *)
  id : element_identifier;
  (* The name specifies the word, or set of words, that are used to refer to the Verification_result. *)
  name : label;
  (* The system_under_verification specifies the Verification_specification_system_view_relationship that the Verifica-
tion_result is valid for. *)
  system_under_verification : verification_specification_system_view_relationship;
UNIQUE
  UR1: id, system_under_verification;
END_ENTITY;

(* A Verification_specification is a specification of how a system, functionality or physical element of a system shall be
verified. <note number="1">A Verification_specification uses the requirement structure to define how a system shall be
tested. The rationale for this is that the requirements structure does allow representation not limited to text. Via the Mod-
el_defined_requirement_definition entity it is possible to specify that models or external files shall be used in the verifica-
tion activity.</note> <note number="2">The Verification_specification does not imply traditional testing. It may be formal
verification, analysis or inspection of a system or any other method.</note> *)
ENTITY verification_specification;
  (* The definition specifies the Requirement_definition that defines what shall be tested. *)
  definition : requirement_occurence;
  (* The description specifies additional information about the Verification_specification. *)
  description : OPTIONAL text_select;
  (* The id specifies the identifier of the Verification_specification. *)
  id : element_identifier;
```

```
  (* The name specifies the word, or set of words, that is used to refer to the Verification_specification. *)
  name : label;
  (* The verification_method specifies how the test shall be carried out for a particular system. *)
  verification_method : label;
END_ENTITY;

(* A Verification_specification_allocation is a mechanism for allocating a Verification_specification to a System_instance,
a Physical_instance, a Function_instance, or a Requirement_instance such that the Verification_specification shall be
applied to the allocated object. *)
ENTITY verification_specification_allocation;
  (* The description specifies additional information about the Verification_specification_allocation. *)
  description : OPTIONAL text_select;
  (* The relevant_for specifies the object that the Verification_specification is allocated to. *)
  relevant_for : verification_allocation_select;
  (* The specification specifies the Verification_specification. *)
  specification : verification_specification;
END_ENTITY;

(* A Verification_specification_system_view_relationship is the mechanism for assigning a Verification_specification to a
System_view indicating the test specification shall be applied for verifying and or validating the system. *)
ENTITY verification_specification_system_view_relationship;
  (* The assigned_verification specifies the assigned Verification_specification. *)
  assigned_verification : verification_specification;
  (* The description specifies additional information about the Verification_specification_system_view_relationship. *)
  description : OPTIONAL text_select;
  (* The index specifies an identifier for the Verification_specification identified by the attribute assigned_verification in the
context of the system_view defined by the system_view attribute. *)
  index : label;
  (* The system_view specifies the System_view to which the Verification_specification is assigned. *)
  system_view : system_view;
END_ENTITY;

(* A View_relationship is a hierarchical relation between two Visual_element objects such that the child object is a part
of the Graphics_view of the parent object. <note number="1">If a View_relationship exists between two Graphics_view
objects the same hierarchical relationship shall exist between the General_function_definition, General_physical_defi-
nition or Fsm_generic_state objects involved in the hierarchy.</note> <note number="2">Each Graphics_view object is
potentially part of many Multi_level_view objects. To allow for overlapping views each View_relationship shall, via the
valid_in attribute define the Multi_level_view for which the relationship is valid.</note> *)
ENTITY view_relationship;
  (* The child specifies the Graphics_view that is superimposed in the Graphics_view of the parent. *)
  child : graphics_view;
  (* The parent specifies the Graphics_view that acts as a context of the child Graphics_view. *)
  parent : graphics_view;
  (* The valid_in specifies the Multi_level_view for which the relation is valid. *)
  valid_in : multi_level_view;
END_ENTITY;

(* A Visual_element is the super-type for all entities carrying graphical information for an object.  A Visual_element is a
Graphics_node, an Actual_port_position, a Formal_port_position, or a Graphics_link. *)
ENTITY visual_element
ABSTRACT SUPERTYPE OF (ONEOF(actual_port_position, formal_port_position, graphics_link, graphics_node));
  (* The view specifies the Graphics_view for which the Visual_element is defined. *)
  view : graphics_view;
END_ENTITY;

(* A Work_order is the authorization for one or more Engineering_process_activity objects to be performed. *)
ENTITY work_order
ABSTRACT SUPERTYPE OF (ONEOF(change_order, start_order));
  (* The description specifies additional information about the Work_order. *)
  description : OPTIONAL text_select;
  (* The id specifies the identifier of the Work_order. *)
```

```
  id : element_identifier;
  (* The is_controlling specifies the Engineering_process_activity objects that are controlled by this particular Work_order.
*)
  is_controlling : SET [1 : ?] OF engineering_process_activity;
  (* The status specifies the condition of the Work_order. Where applicable, the following values shall be used:<ul>
<li>in_work: The request is being developed;</li> <li>issued: The request has been completed and reviewed, and im-
mediate action takes place;</li> <li>resolved: The request is resolved; the actions as defined by the request have been
completed and no further work is required.</li> </ul> *)
  status : label;
  (* The version_id specifies the identification of a particular version of a Work_order. *)
  version_id : OPTIONAL element_identifier;
END_ENTITY;

(* A Work_request is the solicitation for some work to be done. *)
ENTITY work_request
ABSTRACT SUPERTYPE OF (ONEOF(change_request, start_request));
  (* The description specifies additional information about the Work_request. *)
  description : text_select;
  (* The id specifies the identifier of the Work_request. *)
  id : element_identifier;
  (* The notified_person specifies the Person or the Organization that shall be informed about the Work_request and the
date when the Person or Organization shall be informed. *)
  notified_person : SET [0 : ?] OF date_and_person_organization;
  (* The requestor specifies the Person or Organization who issued the request and the date when this Person or Organi-
zation issued the request. *)
  requestor : date_and_person_organization;
  (* The scope specifies the objects that are subject to the Work_request. *)
  scope : SET [0 : ?] OF specification_element_select;
  (* The status specifies the status of the Work_request. Where applicable, the following values shall be used:<ul> <li>in_
work: The request is being developed;</li> <li>issued: The request has been completed and reviewed, and immediate
action takes place;</li> <li>resolved: The request is resolved; the actions as defined by the request have been completed
and no further work is required.</li> </ul> *)
  status : label;
  (* The version_id specifies the identification of a particular version of a Work_request. *)
  version_id : OPTIONAL element_identifier;
END_ENTITY;

  FUNCTION at_most_one_system_function_assigned (a_system_view : system_view): LOGICAL;
    LOCAL
      no_of_system_functions : INTEGER := 0;
    END_LOCAL;

    IF SIZEOF(a_system_view.system) > 0 THEN
      REPEAT i := 1 TO SIZEOF(a_system_view.system);
        IF a_system_view.system[i].role = system_function THEN
          no_of_system_functions := no_of_system_functions + 1;
        END_IF;
      END_REPEAT;
    END_IF;
    RETURN (no_of_system_functions <= 1);
  END_FUNCTION;

  FUNCTION correct_binding (binding : io_port_binding): BOOLEAN;
    LOCAL
      function_interface : function_instance;
    END_LOCAL;

    IF ('SYSTEM_ENGINERING_AND_DESIGN.FUNCTION_INSTANCE' IN TYPEOF(binding.actual_port.port_of)) THEN
      RETURN (FALSE);
    END_IF;
    function_interface := binding.actual_port.port_of;
```

```
    IF (binding.formal_port.port_of :=: function_interface.definition) THEN
      RETURN (TRUE);
    ELSE
      RETURN (FALSE);
    END_IF;
  END_FUNCTION;

  FUNCTION determineactualportrole (p : actual_io_port): port_data_relation;
    IF (p.port_type = input) OR (p.port_type = control) OR (p.port_type = mechanism) THEN
      RETURN (consumer);
    ELSE
      RETURN (producer);
    END_IF;
  END_FUNCTION;

  FUNCTION determineformalportrole (p : formal_io_port): port_data_relation;
    IF (p.port_type = input) OR (p.port_type = control) OR (p.port_type = mechanism) THEN
      RETURN (producer);
    ELSE
      RETURN (consumer);
    END_IF;
  END_FUNCTION;



END_SCHEMA;
```

Приложение ДА
(справочное)

Сведения о соответствии ссылочных международных стандартов ссылочным национальным стандартам Российской Федерации

Таблица ДА.1

Обозначение ссылочного международного стандарта	Степень соответствия	Обозначение и наименование соответствующего национального стандарта
ИСО/МЭК 8824-1	IDT	ГОСТ Р ИСО/МЭК 8824-1:2001 Информационная технология. Абстрактная синтаксическая нотация версии один (АСН.1). Часть 1. Спецификация основной нотации
ИСО 10303-1:1994	IDT	ГОСТ Р ИСО 10303-1:1999 Системы автоматизации производства и их интеграция. Представление данных об изделии и обмен этими данными. Часть 1. Общие представления и основополагающие принципы
ИСО 10303-11:2004	IDT	ГОСТ Р ИСО 10303-11:2009 Системы автоматизации производства и их интеграция. Представление данных об изделии и обмен этими данными. Часть 11. Методы описания. Справочное руководство по языку EXPRESS
ИСО 10303-21:2002	IDT	ГОСТ Р ИСО 10303-21:2002 Системы автоматизации производства и их интеграция. Представление данных об изделии и обмен этими данными. Часть 21. Методы реализации. Кодирование открытым текстом структуры обмена
ИСО 10303-22:1998	IDT	ГОСТ Р ИСО 10303-22:2002 Системы автоматизации производства и их интеграция. Представление данных об изделии и обмен этими данными. Часть 22. Методы реализации. Стандартный интерфейс доступа к данным
Примечание — В настоящей таблице использовано следующее условное обозначение степени соответствия стандартов: -IDT — идентичные стандарты.		

Библиография

[1] Electronics Industries Alliance (EIA), November 1997, EIA 632, *Processes for Engineering a System*

[2] National Institute of Standards and Technology, December 1993, FIPS PUB 183, *Integration Definition for Functional Modeling (IDEF0)*, Federal Information Processing Standards Publication 183

[3] The Institute of Electrical and Electronics Engineers, Inc, 1994, IEEE Std 1220, *IEEE trial-use Standard for Application and Management of the Systems Engineering Process*

[4] The Institute of Electrical and Electronics Engineers, Inc, December 1998, IEEE Std 1220, *IEEE Standard for Application and Management of the Systems Engineering Process*

[5] The Institute of Electrical and Electronics Engineers, Inc, 1990, IEEE Std 610.12, *IEEE Standard Glossary of Software Engineering Terminology*

[6] International Organization for Standardization (ISO), November 1997, ISO 15288, *Life-Cycle Management — System Life Cycle Processes*

[7] Military Standards, May 1992, MIL-STD-499B, *Systems Engineering* (draft Military Standard)

[8] The Software Engineering Institute (SEI), 1995, SE-CMM, version 1.1, *The Systems Engineering Capability Maturity Model*

[9] BAE SYSTEMS, 2006-02-12, ISO 10303 WG3 N1234, version 4, Julian Johnson and Mark Taylor, *SEDRES-AP233 WD#5 Overview pack*

[10] Institut für Instrumentelle Mathematik IIM-2:1962, *Kommunikation mit Automaten*, Carl Adam Petri

[11] ИСО/МЭК 8859-1:1998 Информационные технологии. 8-битовые однобайтовые наборы кодированных графических знаков. Часть 1. Латинский алфавит № 1

ISO/IEC 8859-1:1998 Information technology — 8-bit single-byte coded graphic character sets — Part 1: Latin alphabet No. 1

[12] ИСО 10303-203:2011 Системы промышленной автоматизации и интеграция. Представление данных о продукции и обмен данными. Часть 203. Протокол применения: проектирование механических деталей и узлов с контролируемой конфигурацией 3D (модульная версия)

ISO 10303-203:2011 Industrial automation systems and integration — Product data representation and exchange — Part 203: Application protocol: Configuration controlled 3D design of mechanical parts and assemblies

[13] ИСО 10303-209:2001 Системы промышленной автоматизации и интеграция. Представление данных о продукции и обмен данными. Часть 209. Протокол прикладной программы: анализ композитных и металлических конструкционных деталей и связанное с ним проектирование

ISO 10303-209:2001 Industrial automation systems and integration — Product data representation and exchange — Part 209: Application protocol: Composite and metallic structural analysis and related design

[14] ИСО 10303-210:2011 Системы промышленной автоматизации и интеграция. Представление данных о продукции и обмен данными. Часть 210. Протокол прикладной программы. Электронная сборка, межкомпонентное соединение и проектирование компоновки

ISO 10303-210:2011 Industrial automation systems and integration — Product data representation and exchange — Part 210: Application protocol: Electronic assembly, interconnection, and packaging design

[15] ИСО 10303-212:2001 Системы промышленной автоматизации и интеграция. Представление данных о продукции и обмен данными. Часть 212. Протокол прикладной программы. Электротехнический расчет и установка

ISO 10303-212:2001 Industrial automation systems and integration — Product data representation and exchange — Part 212: Application protocol: Electrotechnical design and installation

[16] ИСО 10303-214:2010 Системы промышленной автоматизации и интеграция. Представление данных о продукции и обмен данными. Часть 214. Протокол прикладной программы: базовые данные для процессов проектирования транспортных средств

ISO 10303-214:2010 Industrial automation systems and integration — Product data representation and exchange — Part 214: Application protocol: Core data for automotive mechanical design processes

[17] ИСО 10303-221:2007 Системы промышленной автоматизации и интеграция. Представление данных о продукции и обмен данными. Часть 221. Протокол прикладной программы. Функциональные данные для технологических установок и их схематическое представление

ISO 10303-221:2007 Industrial automation systems and integration — Product data representation and exchange — Part 221: Application protocol: Functional data and their schematic representation for process plants

[18] ИСО 10303-227:2005 Системы промышленной автоматизации и интеграция. Представление данных о продукции и обмен данными. Часть 227. Протокол прикладной программы. Пространственная конфигурация установки

ISO 10303-227:2005 Industrial automation systems and integration — Product data representation and exchange — Part 227: Application protocol: Plant spatial configuration

[19] ИСО 10303—232:2002 Системы промышленной автоматизации и интеграция. Представление данных о продукции и обмен данными. Часть 232. Протокол прикладной программы. Пакетирование технических данных для оперативной информации и обмена

ISO 10303—232:2002 Industrial automation systems and integration — Product data representation and exchange — Part 232: Application protocol: Technical data packaging core information and exchange

[20] ИСО 10303—239:2005 Системы промышленной автоматизации и интеграция. Представление данных о продукции и обмен данными. Часть 239. Протокол прикладной программы. Поддержка жизненного цикла продукции

ISO 10303—239:2005 Industrial automation systems and integration — Product data representation and exchange — Part 239: Application protocol: Product life cycle support

УДК 658.52.011.56 ОКС 25.040.40 Т 58

Ключевые слова: системы промышленной автоматизации, интеграция, жизненный цикл систем, управление производством

Редактор *Т.С. Никифорова*
Технический редактор *В.Н. Прусакова*
Корректор *Г.В. Яковлева*
Компьютерная верстка *Ю.В. Поповой*

Сдано в набор 09.11.2015. Подписано в печать 15.12.2015. Формат 60 ×84 1/8. Гарнитура Ариал.
Усл. печ. л. 46,50. Уч.-изд. л. 41,44.

Набрано в ИД «Юриспруденция», 115419, Москва, ул. Орджоникидзе, 11.
www. jurisizdat. ru gost.e-ecolog.ru

Издано во ФГУП «СТАНДАРТИНФОРМ», 123995 Москва, Гранатный пер., 4
gost.e-ecolog.ru gost.e-ecolog.ru

gost.e-ecolog.ru