ФЕДЕРАЛЬНОЕ АГЕНТСТВО

ПО ТЕХНИЧЕСКОМУ РЕГУЛИРОВАНИЮ И МЕТРОЛОГИИ

НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

ГОСТ Р МЭК 61360-2— 2013

СТАНДАРТНЫЕ ТИПЫ ЭЛЕМЕНТОВ ДАННЫХ С АССОЦИИРОВАННОЙ СХЕМОЙ КЛАССИФИКАЦИИ ЭЛЕКТРИЧЕСКИХ КОМПОНЕНТОВ

Часть 2

Словарная схема EXPRESS

IEC 61360-2:2012
Standard data element types with associated classification scheme for electric components — Part 2: EXPRESS dictionary schema (IDT)

Издание официальное

Москва
Стандартинформ
2016

gost.e-ecolog.ru

Предисловие

1 ПОДГОТОВЛЕН АНО «Международная академия менеджмента и качества бизнеса» на основе собственного аутентичного перевода на русский язык международного 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 17 декабря 2013 г. № 2271-ст

4 Настоящий стандарт идентичен международному стандарту МЭК 61360-2:2012 «Стандартные типы элементов данных с ассоциированной схемой классификации электрических компонентов. Часть 2. Словарная схема EXPRESS (IEC 62264-2:2012 «Standard data element types with associated classification scheme for electric components — Part 2: EXPRESS dictionary schema»).

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 публикуется в ежегодном (по состоянию на 1 января текущего года) информационном указателе «Национальные стандарты», а официальный текст изменений и поправок — в ежемесячном информационном 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 (www.gost.ru)

© Стандартинформ, 2016

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 3
4 Общая словарная схема и ее совместимость со стандартной словарной схемой I SO13584_IEC61360_dictionary_schema 10
4.1 Общие положения 10
4.2 Использование общей словарной схемы для обмена данных, соответствующих требованиям МЭК 61360-1 10
4.3 Совместимость с ИСО 13584-42 11
4.4 Соответствие названий, используемых в МЭК 61360-1 и МЭК 61360-2 11
4.5 Основная структура общей словарной схемы 12
5 Стандартная словарная схема ISO13584_IEC61360_dictionary _schema 13
5.1 Общие положения 13
5.2 Словарная схема 13
5.3 Ссылки на другие схематики 13
5.4 Определения констант 14
5.5 Идентификация словаря 14
5.6 Базовые семантические единицы: определения и использование словаря 15
5.7 Поставщик данных 22
5.8 Данные класса 24
5.9 Тип элемента данных/данные о свойствах 35
5.10 Область данных: система типов 40
5.11 Определения базового типа и сущности 64
5.12 Определения функций 75
6 Стандартная схема языкового ресурса (ISO13584_IEC61360_language_resource_schema) 90
6.1 Краткое описание 90
6.2 Определения типов и сущностей стандартной схемы языкового ресурса ISO13584_IEC61360_language_resource_schema 91
6.3 Определения функций стандартной схемы языкового ресурса ISO13584_IEC61360_language_resource_schema 94
6.4 Определение правила стандартной схемы языкового ресурса ISO13584_IEC61360_language_resource_schema 94
7 Стандартная схема ограничений класса ISO13584_IEC61360_class_constraint_schema 95
7.1 Общие положения 95
7.2 Введение в стандартную схему ограничений класса ISO13584_IEC61360_class_constraint_schema 96
7.3 Определения сущностей стандартной схемы ограничений класса ISO13584_IEC61360_class_constraint_schema 96
7.4 Определения типа стандартной схемы ограничений класса ISO13584_IEC61360_class_constraint_schema 105
7.5 Определения функций стандартной схемы ограничений класса ISO13584_IEC61360_class_constraint_schema 105
7.6 Определение правил стандартной схемы ограничений класса ISO13584_IEC61360_class_constraint_schema 112

8 Стандартная условная схема класса элементов ISO13584_IEC61360_item_class_case_of_schema 113
8.1 Краткое описание 113
8.2 Введение в стандартную условную схему класса элементов ISO13584_IEC61360_item_class_case_of_schema 114
8.3 Определения сущностей стандартной условной схемы класса элементов ISO13584_IEC61360_item_class_case_of_schema 114
8.4 Определения функций стандартной схемы условного класса элементов ISO13584_IEC61360_item_class_case_of_schema 118
8.5 Определения правил стандартной схемы условного класса элементов ISO13584_IEC61360_item_class_case_of_schema 122
Приложение А (справочное) Пример физического файла 124
Приложение В (справочное) Диаграмма EXPRESS-G 128
Приложение С (справочное) Частные словари 138
Приложение D (справочное) Спецификация формата значения 139
Приложение ДА (справочное) Сведения о соответствии ссылочных международных стандартов ссылочным национальным стандартам Российской Федерации 150

Введение

Общая словарная схема ИСО/МЭК, приведенная в настоящем стандарте, основывается на пересечении областей деятельности следующих стандартов:

- МЭК 61360-1;
- ИСО 13584-42.

Выдержки из соответствующих стандартов, содержащие описание их областей деятельности, приведены далее.

МЭК 61360-1:2009

Настоящий стандарт устанавливает базис для четкого и однозначного определения характеристических свойств (типов элементов данных) элементов электротехнических систем, начиная от элементарных компонентов сборочных модулей до комплексных систем. Несмотря на то что изначально предполагалось использовать данный базис для обмена информацией между электронными компонентами, принципы и методы, заложенные в настоящем стандарте, могут быть использованы и в других областях, таких как сборка компоновочных узлов и электротехнических подсистем (систем).

ИСО 13584-42:2010

Настоящий стандарт устанавливает принципы, используемые для определения характеристических классов деталей и свойств деталей, которые предусматриваются для спецификаций деталей независимо от идентификации, определенной поставщиком.

Настоящий стандарт содержит правила и руководства для технических комитетов по стандартизации и поставщиков информации с целью создания онтологий продукции. Онтологии продукции состоят из иерархии характеристических классов деталей, выполненных в соответствии с общей методологией и обеспечивающих согласованность между поставщиками. Правила и руководства, приведенные в настоящем стандарте, содержат методы группировки деталей в характеристические классы деталей с целью образования иерархий; методы сопоставления свойств деталей с характеристическими классами деталей и со словарем элементов, содержащим классы и свойства деталей.

Настоящий стандарт можно рассматривать в качестве нормативной ссылки на модель данных, определяющую обмен данными словаря. EXPRESS-спецификация разработана как общая модель для ИСО 13584 и МЭК 61360 и опубликована в качестве стандарта МЭК 61360-2.

НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

СТАНДАРТНЫЕ ТИПЫ ЭЛЕМЕНТОВ ДАННЫХ С АССОЦИИРОВАННОЙ СХЕМОЙ КЛАССИФИКАЦИИ ЭЛЕКТРИЧЕСКИХ КОМПОНЕНТОВ

Часть 2

Словарная схема EXPRESS

Standard data element types with associated classification scheme for electric components.
Part 2. EXPRESS dictionary schema

Дата введения — 2014—09—01

1 Область применения

В настоящем стандарте установлена формальная модель данных в соответствии с заданной областью применения МЭК 61360-1 и ИСО 13584-42. Настоящий стандарт обеспечивает возможность компьютерного представления данных и обмен ими.

В настоящем стандарте установлена общая информационная модель, разработанная комитетами МЭК/ПК 3D и ИСО/ТК 184/ПК 4. Модель обеспечивает практическую реализацию словарных систем, работающих с данными, предоставленными в соответствии с каждым из указанных стандартов.

Настоящий стандарт распространяется на общую словарную схему ИСО/МЭК. Она располагается на пересечении областей применения двух базовых стандартов МЭК 61360-1 и ИСО 13584-42.

Модель EXPRESS представляет собой общую формальную модель двух указанных стандартов и способствует их гармонизации.

МЭК 61360-2 является основным документом. ИСО 13584-42 в своем приложении содержит копию модели EXPRESS МЭК 61360-2.

В ряде подразделов, где использование общей модели EXPRESS дает больше свободы, стандарты МЭК накладывают дополнительные ограничения, рассмотренные в методологической части МЭК 61360-1.

В настоящем стандарте рассмотрены две схемы, определяющие две опции, которые используются для практической реализации. На каждую опцию можно ссылаться как на класс соответствия.

- Словарная схема **ISO13584_IEC61360_dictionary_schema2** обеспечивает моделирование и обмен технических типов элементов данных с ассоциированной схемой классификации, используемой в определении типа элемента данных. Данная словарная схема устанавливает класс соответствия 1 настоящего стандарта.

- Стандартная схема языкового ресурса **ISO13584_IEC61360_language_resource_schema** определяет ресурс, разрешающий использование строк в различных языках. Она извлечена из словарной схемы, может использоваться в других схематиках и основана главным образом на схеме **support_resource_schema** в соответствии с ИСО 10303-41:2000 и может рассматриваться как ее развитие. Она также допускает использование одного особого языка в контексте обмена физического файла без дополнительных затрат, имеющих место при использовании нескольких языков.

В соответствии с ИСО 10303-21 каждая схема определяет один единственный формат обмена. Формат обмена, определенный классом соответствия 1, полностью совместим со стандартами серии ИСО 13584.

Издание официальное

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.

МЭК 61360-1:2009 Стандартные типы элементов данных с ассоциированной схемой классификации для электрических компонентов. Часть 1. Определения. Принципы и методы (IEC 61360-1:2009, Standard data elements types with associated classification scheme for electric items — Part 1: Definitions — Principles and methods)

МЭК 61360-4-DB Стандартные типы элементов данных с ассоциированной схемой классификации для электрических компонентов. Часть 4. Набор ссылок МЭК на стандартные типы элементов данных и классы компонентов (IEC 61360-4-DB:2005, Standard data element types with associated classification scheme for electric components — Part 4: IEC reference collection of standard data element types and component classes Free access to Database)

ИСО/МЭК 8859-1:1998 Информационная технология. 8-битные однобайтные кодированные наборы графических символов. 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)

ИСО/МЭК 10646:2012 Информационная технология. Универсальный многооктетный кодированный набор символов (UCS) [ISO/IEC 10646:2012, Information technology — Universal Coded Character Set (UCS)]

ИСО/МЭК 14977:1996 Информационная технология. Синтаксический метаязык. Расширенная форма Бэкуса-Наура (ISO/IEC 14977:1996, Information technology — Syntactic metalanguage — Extended BNF)

ИСО 639 (все части) Коды для представления названий языков (ISO 639, Codes for the representation of names of languages)

ИСО 843:1997 Информация и документация. Преобразование греческих символов в латинские (ISO 843:1997, Information and documentation — Conversion of Greek characters into Latin characters)

ИСО 3166-1:2006 Коды для представления названий стран и их подразделений. Часть 1. Коды стран (ISO 3166-1:2006, Codes for the representation of names of countries and their subdivisions — Part 1: Country codes)

ИСО 4217:2008 Коды для представления валют и фондов (ISO 4217:2008, Codes for the representation of currencies and funds)

ИСО 8601:2004 Элементы данных и форматы взаимного обмена. Взаимный обмен информацией. Представление даты и времени (ISO 8601:2004, Data elements and interchange formats — Information interchange — Representation of dates and times)

ИСО 10303-11:2004 Промышленные системы автоматизации и интеграция. Представление и обмен данных продукта. Часть 11. Методы описания. Ссылочное руководство 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 Промышленные системы автоматизации и интеграция. Представление и обмен данных продукта. Часть 21. Методы практической реализации. Открытое кодирование 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 10303-41:2005 Промышленные системы автоматизации и интеграция. Представление и обмен данных продукта. Часть 41. Интегрированные характерные ресурсы. Основы описания и поддержки продукта (ISO 10303-41:2005, Industrial automation systems and integration — Product data representation and exchange — Part 41: Integrated generic resource: Fundamentals of product description and support)

ИСО 13584-26:2000 Промышленная система автоматизации и интеграция. Библиотека деталей. Часть 26. Логический ресурс. Идентификация поставщика информации (ISO 13584-26:2000, Industrial automation systems and integration — Parts library — Part 26: Logical resource: Information supplier identification)

ИСО 13584-42:2010 Промышленная система автоматизации и интеграция. Библиотека деталей. Часть 42. Методология описания. Методология структурирования семейств деталей (ISO 13584-42:2010, Industrial automation systems and integration — Parts library — Part 42: Description methodology: Methodology for structuring parts families)

3 Термины и определения

В настоящем стандарте используются следующие термины с соответствующими определениями:

3.1 **абстрактный класс** (abstract class): Класс, все члены которого также являются членами одного из его подклассов.

Примечание 1 — Абстрактные классы используются, если необходимо сгруппировать различные виды объектов в классе иерархии включения классов.

Примечание 2 — В общей словарной модели ИСО 13584/МЭК 61360 могут быть определены как абстрактные классы категоризации, так и абстрактные характеристические классы. Факт абстрактности — это только концептуальная характеристика класса. Данная характеристика не представлена в модели явно.

Примечание 3 — Абстрактные характеристические классы допускают совместное использование по наследованию (например, некоторые видимые свойства различных подклассов, соответствующие различным видам элементов).

3.2 **применимое свойство класса** (applicable property of a class): Применимым свойством обязательно обладает каждая составная часть, являющаяся членом характеристического класса.

Примечание 1 — Каждая составная часть, являющаяся членом характеристического класса, обладает аспектом, соответствующим каждому применимому свойству данного характеристического класса.

Примечание 2 — Вышеуказанное определение является концептуальным. Не существует требования, чтобы все применимые свойства класса использовались для описания всех составных частей данного класса на уровне модели данных.

Примечание 3 — Все применимые свойства суперкласса также являются применимыми свойствами подклассов данного суперкласса.

Примечание 4 — Только свойства, определенные или унаследованные как видимые и импортированные свойства класса, могут быть применимыми свойствами.

Примечание 5 — Для облегчения интеграции библиотек компонентов и электронных каталогов, основанных на ИСО 13584-24:2003 и ИСО 13584-25, необходимо, чтобы только свойства, применимые в классе, использовались для характеристики их реализаций в библиотеках компонентов и электронных каталогах.

3.3 **атрибут** (attribute): Элемент данных для компьютерного описания свойства, соотношения или класса.

Примечание — Атрибут описывает единственную особенность свойства, класса или соотношения.

Пример — Название свойства, код класса, единица измерения, в которой представлены значения свойства.

3.4 **базовая семантическая единица** (basic semantic unit; BSU): Сущность, уникально идентифицирующая некоторый объект области приложения, представленный как словарный элемент.

Пример 1 — Словарь соответствует настоящему стандарту, если он идентифицирует класс, свойства, источники информации и типы данных.

Пример 2 — Словарь соответствует ИСО 13584-24:2003, если он идентифицирует класс, свойства, источники информации, типы данных, содержит таблицы, документы и библиотеки программ.

Пример 3 — В ИСО 13584-511 класс болтов с восьмигранной головкой идентифицирован BSU. Класс точности (свойство) резьбы также идентифицирован BSU.

Примечание — Содержание базовой семантической единицы также может быть представлено международным идентификатором регистрации данных (IRDI).

3.5 **характеристика продукта** (characteristic of a product; product characteristic): Неизменное свойство, характеристика продукта, значение которой фиксировано, как только продукт определен.

Примечание 1 — Изменение значения характеристики продукта означает изменение самого продукта.

Пример — Для шариковых подшипников внутренний диаметр и наружный диаметр являются характеристиками продукта.

Примечание 2 — Адаптировано из ИСО 13584-24:2003, определение 3.12.

3.6 **класс** (class): Абстрактное представление набора подобных продуктов.

Примечание 1 — Продукт, удовлетворяющий требованиям абстрактного представления, определенного как класс, называется членом класса.

Примечание 2 — Класс — это особое понятие, имеющее различный расширенный смысл в различных контекстах.

Пример — Набор продуктов, используемый на конкретном предприятии, и набор всех продуктов, удовлетворяющих требованиям стандартов ИСО, — два примера контекстов. В данных двух контекстах (требования конкретного предприятия и требования ИСО) набор продуктов, рассматриваемый как член одного и того же класса однорядных шариковых подшипников, может быть различным. Так, сотрудники разных предприятий могут игнорировать некоторые существующие однорядные шариковые подшипники.

Примечание 3 — Классы структурируются соотношениями включения в класс.

Примечание 4 — Класс продуктов — это общее понятие, определенное в ИСО 1087-1. Правила, определенные в ИСО 704, рекомендуется использовать для определения обозначений и определений атрибутов классов продуктов.

Примечание 5 — В контексте серии стандартов ИСО 13584 класс является либо характеристическим классом, ассоциированным со свойствами и пригодным для характеристики продуктов, либо классом категоризации, не ассоциированным со свойствами и непригодным для характеристики продуктов.

3.7 **соотношение включения в класс** (class inclusion relationship): Соотношение между классами, определяющее порядок включения в класс: если класс А — это суперкласс класса А1, то в любом контексте любой член А1 также является членом А.

Пример 1 — Набор продуктов, используемый на конкретном предприятии, и набор всех продуктов, удовлетворяющих требованиям стандартов ИСО, — это два различных контекста.

Пример 2 — В любом контексте класс конденсаторов включает класс электролитических конденсаторов.

Примечание 1 — Включение в класс определяет иерархическую структуру классов.

Примечание 2 — Включение в класс — это концептуальное соотношение. Оно не предписывает что-либо на уровне представления данных. Следовательно, оно не предписывает какую-либо конкретную схему базы данных или модель данных.

Примечание 3 — В модели, определенной в настоящем стандарте, использование «представительного» соотношения гарантирует включение в класс. В настоящем стандарте рекомендуется, чтобы использование «условного» соотношения также гарантировало включение в класс.

Примечание 4 — Соотношение включения в класс также называют родовидовым соотношением.

3.8 **член класса** (class member): Продукт, удовлетворяющий требованиям абстрактного представления, определяющего класс.

3.9 **свойство со значением класса** (class valued property): Свойство, имеющее единственное значение для всего характеристического класса продуктов.

Примечание 1 — Значение свойства со значением класса не определяется индивидуально для каждого отдельного продукта характеристического класса. Оно определяется глобально для всего класса.

Примечание 2 — Если все продукты характеристического класса продуктов имеют одно и то же значение конкретного свойства, то определение данного свойства как свойства со значением класса позволяет избежать дублирования значений для любой реализации.

Примечание 3 — Свойства со значением класса также могут использоваться для достижения некоторой общности между различными характеристическими классами, если такая общность не достигается в иерархических структурах.

3.10 **общая словарная модель ИСО 13584/МЭК 61360** (common ISO13584/IEC 61360 dictionary model): Модель данных онтологии продукта, использующая язык моделирования информации EXPRESS и полученная совместными усилиями комитетов ИСО/ТК 184/ПК 4/РГ 2 и МЭК/ПК 3D.

Примечание 1 — Несколько уровней допустимых вариантов практической реализации, рассматриваемых как класс соответствия, определены для общей словарной модели ИСО 13584/МЭК 61360. Класс соответствия 1

состоит из различных схем, задокументированных в настоящем стандарте (дублирует информацию, содержащуюся в настоящем стандарте). Схема расширения словарного агрегата **ISO13584_IEC61360_dictionary_aggregate_extension_schema** задокументирована в ИСО 13584-25 (дублирована в МЭК 61360-5). Прочие классы соответствия задокументированы в ИСО 13584-25 (классы соответствия 2, 3 и 4).

Примечание 2 — В серии стандартов ИСО 13584 каждая отдельная онтология продукта, использующая конкретную область продукта и основанная на общей словарной модели ИСО 13584/МЭК 61360, называется ссылочным словарем для данной области.

3.11 **контекстно-зависимая характеристика продукта** (context dependent characteristic of product): Свойство продукта, значение которого зависит от некоторого контекстного параметра.

Примечание 1 — Для данного продукта контекстно-зависимая характеристика математически определяется как функция, область определения которой задается некоторыми контекстными параметрами, определяющими среду продукта.

Пример — Для шариковых подшипников долговечность — это контекстно-зависимая характеристика, зависящая от радиальной нагрузки, осевой нагрузки и частоты вращения.

Примечание 2 — Адаптировано из ИСО 13584-24:2003, определение 3.22.

3.12 **контекстный параметр** (context parameter): Переменная, значение которой характеризует контекст, в котором рассматривается продукт.

Пример 1 — Динамическая нагрузка на подшипник — это контекстный параметр для данного подшипника.

Пример 2 — Внешняя температура, для которой измеряется сопротивление резистора, — это контекстный параметр для данного резистора.

Примечание 1 — Данное определение заменяет определение по ИСО 13584-24:2003.

Примечание 2 — В серии стандартов ИСО 13584 значение свойства представляется как тип элемента данных.

3.13 **тип элемента данных** (data element type): Единица данных, имеющая идентификацию, описание и значение.

Примечание — В серии стандартов ИСО 13584 значение свойства представляется как тип элемента данных.

3.14 **словарные данные** (dictionary data): Набор данных, представляющий онтологии продукта, возможно ассоциированные с категоризациями продукта.

Примечание 1 — При обмене словарными данными рекомендуется использовать некоторый класс соответствия общей словарной модели ИСО/МЭК.

Примечание 2 — Данное определение словарных данных заменяет нижеследующее определение из первого издания МЭК 61360-2 «набор данных, описывающий иерархии характеристических классов продуктов и свойства этих продуктов».

3.15 **словарный элемент** (dictionary element): Набор атрибутов, составляющий словарное описание некоторого объекта области приложения.

Пример 1 — Словарь соответствует настоящему стандарту, если он дает описания классов, свойств, источников информации и типов данных.

Пример 2 — Словарь соответствует ИСО 13584-24:2003, если он дает описания классов, свойств, источников информации, типов данных, а также содержит таблицы, документы и библиотеки программ.

3.16 **семейство продуктов** (family of products): Набор продуктов, представленный одним и тем же характеристическим классом.

Примечание — Данное определение заменяет нижеследующее определение ИСО 13584-24:2003 «простое или характерное семейство деталей».

3.17 **особенность** (feature): Аспект продукта, который может быть описан характеристическим классом и набором пар «значение-свойство».

Примечание 1 — В действительности реализация особенности может быть только встроенной в продукт, аспектом которого она является.

Пример 1 — Головка винта — это особенность, описанная классом головок и рядом свойств головок, зависящих от класса головок. Головка винта существует, только если она принадлежит винту.

Примечание 2 — Особенности представлены классом элементов **item_class,** логический атрибут совместного использования **instance_sharable** которого равен *false*.

Примечание 3 — Атрибут **instance_sharable** допускает задание концептуального статуса элемента: это либо независимый элемент (**instance_sharable** = *true)* или особенность (**instance_sharable** = *false)*. Какие-либо ограничения на уровне представления данных отсутствуют. Общая словарная модель ИСО 13584/МЭК 61360, представляющая некоторые реализации действительности (совместно использующие одно и то же представление EXPRESS с помощью одной или нескольких сущностей языка EXPRESS), рассматривается как зависимая практическая реализация. Не существует механизма, указывающего, допустимо или недопустимо совместное использование значения данных реализации особенности.

Пример 2 — Одна и та же реализация класса головок винта может быть ссылочной для нескольких реализаций класса винтов. Это означает, что существует несколько головок винтов, но все они относятся к одному характеристическому классу и одному набору значений свойств. Атрибут instance_sharable допускает, что изменение одной реализации класса головок винта может привести к изменению нескольких реализаций класса винтов.

3.18 **импортированное свойство** (imported property): Свойство, определенное в одном классе и выбранное другим классом (того же самого или другого ссылочного словаря с помощью условного соотношения), чтобы стать применимым во втором классе.

Примечание 1 — Импортированными из данного класса могут быть только свойства, являющиеся видимыми и/или применимыми в данном классе.

Примечание 2 — Импортирование между классами различных ссылочных словарей допускает повторное использование свойств (определенных, например, в стандартном ссылочном словаре) без их повторного определения.

Примечание 3 — Импортирование между классами одного ссылочного словаря означает признание факта, что некоторые продукты могут выполнять несколько функций. При этом необходимо обеспечение возможности импорта свойств из нескольких классов высшего уровня.

Примечание 4 — Если свойство импортируется в новый класс, то оно сохраняет свой исходный идентификатор. Необходимость дублировать все атрибуты отсутствует.

Примечание 5 — Импортированное свойство является применимым в классе, в который оно импортируется.

3.19 **информация** (information): Факты, понятия и инструкции.

[ИСО 10303-1:1994, статья 3.2.20]

3.20 **информационная модель** (information model): Формальная модель ограниченного набора фактов, понятий или инструкций, удовлетворяющая установленным требованиям.

[ИСО 10303-1:1994, статья 3.2.21]

3.21 **поставщик информации; поставщик** (information supplier; supplier): Организация, предоставляющая онтологию (т. е. словарь данных) или библиотеку поставщика и несущая ответственность за их содержание.

Примечание — Данное определение поставщика заменяет определение поставщика информации по ИСО 13584-1:2001.

3.22 **международный идентификатор регистрации данных** (international registration data identifier; IRDI): Международный уникальный идентификатор заданного объекта области приложения по ИСО/МЭК 11179-5.

Примечание 1 — Только международные идентификаторы регистрации данных, соответствующие ИСО/ТС 29002-5, используются в контексте серии стандартов ИСО 13584.

Примечание 2 — Международные идентификаторы регистрации данных могут использоваться для представления содержания базовой семантической единицы, которая определяет словарный элемент как строку.

Примечание 3 — Международный идентификатор регистрации данных можно также использовать для задания содержания атрибута словарного элемента.

Пример — Единица измерения свойства, значения свойств или ограничения (на свойство) может быть идентифицирована IRDI.

3.23 **представительное соотношение** (is-a relationship): Соотношение включения в класс, ассоциированное с наследованием: если А1 *представляет* А, то каждый продукт, принадлежащий А1, принадлежит А, и все описания в контексте А автоматически дублируются в контексте А1.

Примечание 1 — Данный механизм обычно называют «наследственностью».

Примечание 2 — В общей словарной модели ИСО 13584/МЭК 61360 представительное соотношение может быть определено только между характеристическими классами. Рекомендуется, чтобы оно определяло единственную иерархию и гарантировало, чтобы и видимые, и применимые свойства унаследовались.

3.24 **условное соотношение; условность** (is-case-of relationship; case-of): Механизм импортирования свойств: если А1 *условное соотношение* из А, то определение продуктов из А также покрывает продукты из А1, и А1 может импортировать любое свойство из А.

Примечание 1 — Цель *условного* соотношения — разрешить объединение нескольких иерархий включения в класс, при этом гарантируя, что ссылочные иерархии могут обновляться независимо.

Примечание 2 — Не существует такого ограничения, что условное соотношение определяет единственную иерархию.

Примечание 3 — В общей словарной модели ИСО 13584/МЭК 61360 условное соотношение используется преимущественно в следующих четырех случаях: (1) для связи характеристического класса и класса категоризации, (2) для импорта (в контексте некоторых стандартизованных ссылочных словарей) некоторых свойств, ранее определенных в других стандартизованных ссылочных словарях, (3) для соединения ссылочного словаря пользователя с одним или несколькими стандартизованными ссылочными словарями, (4) для описания продукта с помощью свойств различных классов: если продукты класса А1 выполняют две различные функции и, таким образом, являются логически описанными свойствами, ассоциированными с двумя различными классами А и В, то А1 может быть присоединено представительным соотношением, например, к А и условным соотношением — к В.

Примечание 4 — Конструктивы ресурса EXPRESS для моделирования условных соотношений определены в 4.5 и далее.

3.25 **элемент** (item): Сущность, характеризуемая характеристическим классом (которому она принадлежит) и набором пар «значение-свойство».

Примечание 1 — Данное определение заменяет определение, приведенное в ИСО 13584-24:2003.

Примечание 2 — В серии стандартов ИСО 13584 и продукты, и особенности продуктов (соответствующие составным свойствам) являются элементами.

3.26 **листовой характеристический класс** (leaf characterization class): Конечный характеристический класс, который далее уже не специализируется в более точный характеристический класс.

Пример — Винт с утопленной плоской головкой и крестообразным шлицем (тип Y) и шестигранный винт с головкой под торцевой ключ с метрической мелкой резьбой — это листовые характеристические классы, определенные в ИСО 13584-511.

3.27 **нелистовой характеристический класс** (non-leaf characterization class): Промежуточный характеристический класс, который далее специализируется в более точный характеристический класс.

Пример — Компонент с наружной резьбой и болт/винт с метрической резьбой — это нелистовые характеристические классы, определенные в ИСО 13584-511.

3.28 **неколичественный тип элемента данных** (non-quantitative data element type): Тип элемента данных, который задает или описывает объект с помощью кодов, аббревиатур, названий, ссылок или описаний.

3.29 **составная часть** (part): Материал или функциональный элемент, составляющий продукт.

[ИСО 13584-1:2001, статья 3.1.16]

3.30 **библиотека деталей** (parts library): Компьютерная онтология продукта и компьютерное описание набора продуктов, ссылающихся на данную онтологию.

Примечание — Настоящее определение заменяет определение, приведенное в первом издании настоящего стандарта.

3.31 **продукт** (product): Объект или вещество, изготовленные естественным или искусственным путем.

Примечание — В настоящем стандарте термин «продукт» взят в своем самом широком смысле, включая устройства, системы и установки, а также материалы, процессы, программное обеспечение и услуги.

3.32 **категоризация продукта; категоризация детали; категоризация** (product categorization; part categorization; categorization): Рекурсивное деление набора продуктов на подмножества для особых целей.

Примечание 1 — Подмножества, используемые при категоризации продукта, образуют классы категоризации продукта или категории продукта.

Примечание 2 — Категоризация продукта — это не онтология продукта. Она не может использоваться для характеристики продукта.

Примечание 3 — Свойства не могут ассоциироваться с категоризациями.

Примечание 4 — В зависимости от целевого использования можно взять несколько категоризаций для одного набора продуктов.

Пример — Классификация системы стандартных продуктов и услуг ООН (UNSPSC) — пример категоризации продукта, разработанной для анализа расходов.

Примечание 5 — Использование *условного* соотношения и нескольких иерархий характеристических классов продуктов может быть соединено с иерархией категоризации для создания единой структуры.

3.33 **класс категоризации продукта; класс категоризации детали; класс категоризации** (product categorization class; part categorization class; categorization class): Класс продуктов, составляющих элемент категоризации.

Пример — Производство компонентов и поставки, промышленная оптика — это примеры класса категоризации продукта, определенного UNSPSC.

Примечание 1 — Настоящий стандарт не содержит правил выбора классов категоризации. Данное понятие: (1) отлично от понятия характеристического класса, (2) объясняет, что один и тот же характеристический класс может быть соединен с любым количеством классов категоризации.

Примечание 2 — Не существует свойств, ассоциированных с классом категоризации.

3.34 **характеризация продукта; характеризация детали** (product characterization; part characterization): Описание продукта с помощью характеристического класса продуктов, которому он принадлежит, и набора пар «значение-свойство».

Пример — Болт с шестигранной головкой hexagon_head_bolt_ISO_4014 (марка продукта = A, тип резьбы = M, длина = 50, диаметр = 8) — это пример характеризации продукта.

3.35 **характеристический класс продуктов; характеристический класс деталей; характеристический класс** (product characterization class; part characterization class; characterization class): Класс продуктов, выполняющих одну функцию и имеющих общие свойства.

Примечание — Характеристический класс продуктов может быть определен на различных уровнях детализации. При этом определяется иерархия включения классов.

Пример — Болт/винт с метрической резьбой и болт с шестигранной головкой — это примеры характеристических классов продуктов, определенные в ИСО 13584-511. Первый характеристический класс включен во второй. Транзистор и биполярный силовой транзистор — это примеры характеристических классов продуктов, определенные МЭК 61360-DB. Здесь второй класс включен в первый.

3.36 **онтология продукта; онтология детали; онтология** (product ontology; part ontology; ontology): Модель знания о продукте, полученная путем формального и согласованного представления понятий области существования продукта в терминах идентифицированных характеристических классов, соотношений между классами и идентифицированных свойств.

Примечание 1 — Онтологии продуктов основаны на рассмотрении модели реализации класса, позволяющей распознавать и давать обозначения наборам продуктов (называемым характеристическими классами) с подобными функциями (например, *шариковый подшипник, конденсатор)*. Онтологии продуктов также позволяют селектировать внутри класса различные подмножества продуктов (называемые реализациями), рассматриваемые как идентичные. Правила, определенные в ИСО 1087-1, рекомендуется использовать для формулировки обозначений и определений характеристических классов. Реализации не имеют определений. Они идентифицируются классом, которому они принадлежат, и набором пар «свойство-значение».

Примечание 2 — Онтологии работают не со словами, а с понятиями независимо от выбора языка.

Примечание 3 — Слово «согласованный» означает, что концептуализация согласована в некотором сообществе.

Примечание 4 — Слово «формальный» означает, что онтология должна быть представлена на компьютере. Некоторые уровни компьютерного обоснования применимы к онтологиям: проверка непротиворечивости, выполнение умозаключений.

Примечание 5 — Слово «идентифицированный» означает, что каждый характеристический класс и свойства онтологии ассоциированы с глобально уникальным идентификатором, обеспечивающим ссылку на данное понятие из любого контекста.

Примечание 6 — Модель данных онтологии, рекомендованная в настоящем стандарте, — это общая словарная модель ИСО 13584/МЭК 61360, простейшая версия которой задокументирована в настоящем стандарте. Более полные версии задокументированы в ИСО 13584-25 и МЭК 61360-5 (классы соответствия 1, 2, 3 и 4 для обоих документов).

Примечание 7 — В настоящем стандарте онтология продукта, использующая конкретную область продукта, соответствующую общей словарной модели ИСО 13584/МЭК 61360, называется ссылочным словарем для данной области.

Пример — Ссылочный словарь электрических компонентов, определенный в МЭК 61360-4-DB, является онтологией продукта для электрических компонентов, соответствующих общей словарной модели ИСО 13584/МЭК 61360. Данная модель согласована всеми членами подразделений комитета МЭК/ПК 3D. Корпоративный ссылочный словарь согласовывается уполномоченными экспертами от имени компании.

3.37 **свойство** (property): Определенный параметр, удобный для описания и дифференциации продуктов.

Примечание 1 — Свойство описывает один аспект рассматриваемого объекта.

Примечание 2 — Свойство определяется полной совокупностью своих ассоциированных атрибутов. Типы и количество атрибутов, описывающих свойство с высокой точностью, задокументированы в настоящем стандарте.

Примечание 3 — Настоящий стандарт имеет три различных идентифицированных вида свойств: характеристики продукта, контекстные параметры и контекстно-зависимые характеристики продукта.

Примечание 4 — Данное определение свойства заменяет его определение, приведенное в предшествующем издании настоящего стандарта.

Примечание 5 — В серии стандартов ИСО 13584 значение свойства представлено как тип элемента данных.

3.38 **тип данных свойства** (property data type): Допустимый набор значений свойства.

3.39 **класс определения свойства** (property definition class): Характеристический класс продуктов, в контексте которого определены свойства продукта.

Примечание — В общей словарной модели ИСО 13584/МЭК 61360 каждое свойство продукта имеет один класс определения свойства, задающий область приложений. Свойство имеет смысл только для данного класса и для всех его подклассов. Говорят, что свойство видимо в данной области.

3.40 **количественный тип элемента данных** (quantitative data element type): Тип элемента данных с числовым значением, представляющим физическое количество, количество информации или перечисляемое количество объектов.

3.41 **ссылочный словарь** (reference dictionary): Онтология продукта, соответствующая общей словарной модели ИСО 13584/МЭК 61360.

Примечание — В серии стандартов ИСО 13584 онтология продукта, обращающаяся к некоторой области продукта, основанной на общей словарной модели ИСО 13584/МЭК 61360, называется ссылочным словарем для данной области.

3.42 **конструктив ресурса** (resource construct): Набор сущностей языка EXPRESS, его типов, функций, правил и ссылок, которые вместе определяют корректное описание данных.

Примечание — Данное определение адаптировано из ИСО 10303-1:1994.

3.43 **подкласс** (subclass): Класс, расположенный на одну ступень ниже другого класса в иерархии включения классов.

Примечание — В общей словарной модели ИСО 13584/МЭК 61360 иерархии включения классов определены представительными соотношениями. Они могут также быть установлены условными соотношениями.

3.44 **суперкласс** (superclass): Класс, расположенный на одну ступень выше другого класса в иерархии включения классов.

Примечание 1 — В общей словарной модели ИСО 13584/МЭК 61360 иерархии включения классов определены представительными соотношениями. Они также могут быть установлены условными соотношениями.

Примечание 2 — В общей словарной модели ИСО 13584/МЭК 61360 класс имеет один суперкласс, установленный представительным соотношением.

3.45 **библиотека поставщика** (supplier library): Библиотека деталей, для которой поставщик информации отличается от пользователя библиотеки.

Примечание — Данное определение заменяет определение по ИСО 13584-1:2001.

3.46 **видимое свойство** (visible property): Свойство с определением, имеющим смысл в области применения данного характеристического класса, оно не обязательно применимое к продуктам из данного класса.

Примечание 1 — Фраза «имеет смысл в области применения данного характеристического класса» означает, что наблюдатель для любого характеристического класса продуктов может определить, действительно ли данное свойство применимо и, если да, какому аспекту продукта оно соответствует.

Примечание 2 — Понятие видимого свойства допускает совместное использование определения свойства в характеристическом классе продуктов, где данное свойство не обязательно применимо.

Пример — Свойство «нерезьбовая длина» имеет смысл для любого класса винтов, но применимо только для винтов, имеющих нерезьбовую часть. Оно может быть определено как видимое на уровне «винт», но применимо только в некоторых подклассах.

Примечание 3 — Все видимые свойства суперкласса, являющегося характеристическим классом продуктов, также являются видимыми свойствами для его подклассов.

Примечание 4 — Для облегчения интеграции библиотек компонентов и электронных каталогов, основанных на ИСО 13584-24 и ИСО 13584-25, указанные части ИСО 13584 требуют, чтобы только свойства, применимые в классе, использовались для характеристики его реализаций в библиотеках компонентов и электронных каталогах.

Примечание 5 — Данное определение видимого свойства заменяет предшествующее определение по ИСО 13584-24.

4 Общая словарная схема и ее совместимость со стандартной словарной схемой ISO13584_IEC61360_dictionary_schema

4.1 Общие положения

В следующих подразделах представлена архитектура общей словарной схемы. Приведены пояснения, почему одна и та же информационная модель используется в стандартах и как обеспечивается их совместимость.

Общая словарная схема объединяет требования серии стандартов МЭК 61360 и ИСО 13584. Следовательно, она содержит ресурсы, удовлетворяющие особым требованиям обеих серий стандартов. Указанные ресурсы обеспечиваются либо как вспомогательные функции, либо как подтипы для типов, определенных для выполнения общих требований.

4.2 Использование общей словарной схемы для обмена данных, соответствующих требованиям МЭК 61360-1

Для обмена данных, соответствующих требованиям МЭК 61360-1, общая словарная схема используется следующим образом:

a) для обмена словарных элементов, соответствующих МЭК 61360-1, не требуются какие-либо особые расширения серии стандартов ИСО 13584 для поддержки многоязыковых функций. Однако указанные расширения (например, **present_translations, translated_label** и **translated_text**) должны использоваться в структуре обмена для обеспечения совместимости;

b) если класс компонентов имеет суперкласс, то кодированное название **coded_name** должно определяться как код значения **value_code** в области классифицирующего типа элементов данных суперкласса;

c) если классифицирующий тип элемента данных существует внутри особого класса компонентов, то для каждого значения в своей области должны быть определены подкласс и его термин;

d) классифицирующий тип элемента данных (вспомогательный тип в классе соответствия 2 в общей словарной схеме) всегда должен быть доступен для классов компонентов, определенных в соответствии с МЭК 61360-1;

e) допустимы только единицы системы СИ. При этом общая словарная схема дает возможность использования большого количества других систем единиц измерения. Однако при использовании данной схемы для обмена количественных типов элементов данных, соответствующих требованиям серии стандартов МЭК 61360, допустима только система СИ.

4.3 Совместимость с ИСО 13584-42

Практическая реализация, соответствующая настоящему стандарту, должна поддерживать все сущности, типы и ассоциированные ограничения, принадлежащие поддерживаемому классу соответствия. Следовательно, соответствие классу соответствия 1 настоящего стандарта требует, чтобы поддерживались все сущности, типы и ассоциированные ограничения, определенные в общей словарной схеме. Данные ИСО 13584, удовлетворяющие требованиям общей словарной схемы, могут, таким образом, обрабатываться в рамках практической реализации МЭК 61360, удовлетворяющей требованиям класса соответствия 1, включающего все особенности класса соответствия 1. В ИСО 13584 имеется особый класс соответствия 3, содержащий все сущности, типы и ассоциированные ограничения, определенные в общей словарной схеме. Практическая реализация, удовлетворяющая требованиям ИСО 13584 и требованиям данного класса соответствия, должна, таким образом, поддерживать данные по МЭК, принадлежащие классу соответствия 1 настоящего стандарта.

4.4 Соответствие названий, используемых в МЭК 61360-1 и МЭК 61360-2

Из-за некоторых ограничений (например, язык EXPRESS не допускает пробелов в названии сущности) в приложениях появляется ряд подобных названий на языке EXPRESS, полученных путем замены пробелов в названиях на нижнюю черту (например, словосочетание «предпочтительное имя» **preferred name** представляется как **preferred_name**).

В других местах модели EXPRESS используются названия, отличные от рекомендованных МЭК 61360-1. Это есть следствие попыток получить информационную модель EXPRESS, общую с библиотеками деталей.

В таблице 1 сравниваются названия, используемые в двух частях МЭК 61360.

Таблица 1 — Таблица перекрестных ссылок

Название в МЭК 61360-2	Название в МЭК 61360-1
component_class	Класс компонентов
condition_DET	Условный тип элемента данных
dependent_P_DET	Тип элемента данных
det_classification	Класс типов элементов данных
(DER)dic_identifier	Идентификатор
dic_value	Значение
material_class	Класс материалов
Meaning	Смысл значения
non_dependent_P_DET	Тип элемента данных
preferred_symbol	Предпочтительный буквенный символ
revision	№ пересмотра
source_doc_of_definition	Исходный документ для определения типа элемента данных
source_doc_of_definition	Исходный документ для определения класса компонентов

Окончание таблицы 1

Название в МЭК 61360-2	Название в МЭК 61360-1
synonymous_symbols	Синонимические буквенные символы
unit	Единица измерения
value_code	Код значения
version	№ версии

4.5 Основная структура общей словарной схемы

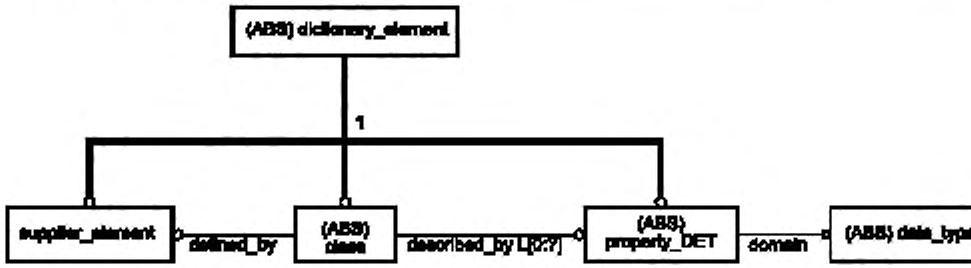
Данный подраздел содержит объяснение основных конструктивов ресурса, используемых общей словарной схемой:

- **dictionary_element** — любой элемент, определенный в словаре;
- **supplier_element** — содержит данные о поставщиках словарных элементов (классы, свойства, типы данных); класс, моделирующий словарные элементы классов (семейств), описанные свойствами;
- **property_DET** — словарный элемент свойства;
- **data_type** — определяет тип свойства.

Указанные части словарной схемы детально представлены в разделе 5: **ISO13584_IEC61360_dictionary_schema**.

В рассматриваемом представлении общей словарной схемы некоторые обзорные диаграммы даны как модели планирования (см. рисунки 1—11). Указанные модели планирования используют графические обозначения EXPRESS-G для языка EXPRESS. Для упрощения рассматриваемых диаграмм некоторые соотношения, определенные в модели EXPRESS, опущены. На рисунке 1 приведена модель планирования основной структуры общей словарной схемы. Большинство указанных рисунков содержат обзорные модели (модели планирования). Их уровень детализации соответствует уровню рассмотрения.

Для упрощения рассматриваемых диаграмм некоторые соотношения, определенные в модели EXPRESS, опущены. На рисунке 1 приведена модель планирования основной структуры общей словарной модели ИСО 13584/МЭК 61360.



(ABS) dictionary_element	Словарный элемент
supplier_element	Элемент поставщика
defined_by	Определен с помощью …
(ABS) class	Класс
described_by L[0:?]	Описан массивом …
(ABS property_DET)	Тип элемента данных свойства
domain	Область
(ABS) data_type	Тип данных

Рисунок 1 — Словарная схема

5 Стандартная словарная схема ISO13584_IEC61360_dictionary _schema

5.1 Общие положения

Данный раздел, представляющий основную часть общей информационной модели ИСО 13584-42 и МЭК 61360, содержит полный листинг словарной схемы на языке EXPRESS, аннотированный с комментариями и пояснительным текстом. Порядок расположения текста в данном разделе определен главным образом порядком, установленным языком EXPRESS.

5.2 Словарная схема

В первую очередь схему необходимо объявить.

Пример представления на языке EXPRESS:

```
*)
SCHEMA ISO13584_IEC61360_dictionary_schema;
(*
```

5.3 Ссылки на другие схематики

Данный подраздел содержит ссылки на другие схематики EXPRESS, используемые в данной словарной схеме. Их источник указан в соответствующем комментарии.

Пример представления на языке EXPRESS:

```
*)
REFERENCE FROM support_resource_schema(identifier, label, text);

REFERENCE FROM person_organization_schema(organization, address);

REFERENCE FROM measure_schema;

REFERENCE FROM ISO13584_IEC61360_language_resource_schema;

REFERENCE FROM ISO13584_IEC61360_class_constraint_schema;

REFERENCE FROM ISO13584_IEC61360_item_class_case_of_schema;

REFERENCE FROM ISO13584_external_file_schema
     (external_item,
     external_file_protocol,
     external_content,
     not_translatable_external_content,
     not_translated_external_content,
     translated_external_content,
     language_specific_content,
     http_file,
     http_class_directory,
     http_protocol);
(*
```

Примечание — Схематика, указанная выше, описана в следующих документах:

support_resource_schema	ИСО 10303
person_organization_schema	ИСО 10303
measure_schema	ИСО 10303
ISO13584_IEC61360_language_resource_schema (схема дублирована для удобства в настоящем стандарте)	МЭК 61360-2
ISO13584_IEC61360_class_constraint_schema (схема дублирована для удобства в настоящем стандарте)	МЭК 61360-2

ISO13584_IEC61360_item_class_case_of_schema МЭК 61360-2
(схема дублирована для удобства в настоящем стандарте)
ISO13584_external_file_schema ИСО 13584-24

5.4 Определения констант

Данный подраздел содержит определения целочисленных констант, используемые позже в определениях типа (см. 5.11).

Пример представления на языке EXPRESS:

```
*)
CONSTANT
      dictionary_code_len: INTEGER := 131;
      property_code_len: INTEGER := 35;
      class_code_len: INTEGER := 35;
      data_type_code_len:INTEGER := 35;
      supplier_code_len: INTEGER := 149;
      version_len: INTEGER := 10;
      revision_len: INTEGER := 3;
      value_code_len: INTEGER := 35;
      pref_name_len: INTEGER := 255;
      short_name_len: INTEGER := 30;
      syn_name_len: INTEGER := pref_name_len;
      DET_classification_len: INTEGER := 3;
      source_doc_len: INTEGER := 255;
      value_format_len: INTEGER := 80;
      sep_cv: STRING := '#';
      sep_id: STRING := '#';
END_CONSTANT;
(*
```

5.5 Идентификация словаря

Сущность «идентификация словаря **dictionary_identification**» позволяет идентифицировать конкретную версию конкретного словаря конкретного поставщика информации (стандартную или произвольную). Данная сущность содержит код **code**, определенный поставщиком словаря и идентифицирующий словарь, № версии и № пересмотра, характеризующие конкретное состояние данного словаря.

Примечание 1 — Случаи, когда № версии и № пересмотра словаря получают приращения, определены в МЭК 61360-1.

Пример представления на языке EXPRESS:

```
*)
ENTITY dictionary_identification;
      code: dictionary_code_type;
      version: version_type;
      revision: revision_type;
      defined_by: supplier_bsu;
DERIVE
      absolute_id: identifier :=
            defined_by.absolute_id + sep_id + code + sep_cv + version;

UNIQUE
      UR1: absolute_id;
END_ENTITY; -- dictionary_identification
(*
```

Определения атрибутов:
code: код, характеризующий словарь.
version: № версии, характеризующий рассматриваемую версию словаря.
revision: № пересмотра, характеризующий пересмотр словаря.
defined_by: поставщик, определяющий словарь.
absolute_id: уникальная идентификация словаря.
Пояснения к тексту программы:
UR1: идентификатор словаря, определенный атрибутом **absolute_id,** является уникальным.

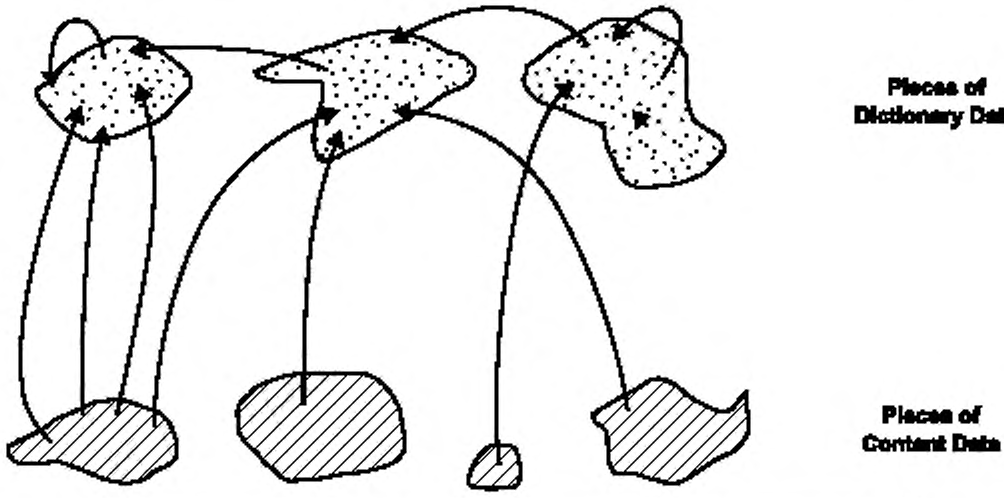
Примечание 2 — Порядок представления стандартных номеров стандартных документов установлен в разделах 5.1 и 5.2 ИСО 13584-26:2000.

5.6 Базовые семантические единицы: определения и использование словаря

5.6.1 Требования обмена

При обмене данными словарей и библиотек деталей данные обычно делят на части. Например, словарь можно обновить несколькими классами, которые устанавливают их суперкласс путем ссылки на уже существующие классы, или (если обмениваются содержанием библиотек) путем ссылки на словарные элементы (только ссылки без включения).

Таким образом, первое требование — это иметь возможность обменивать блоки данных и второе — иметь соотношения между указанными блоками (см. рисунок 2).



Pieces of Dictionary Data	Блоки данных словаря
Pieces of Content Data	Блоки данных содержания

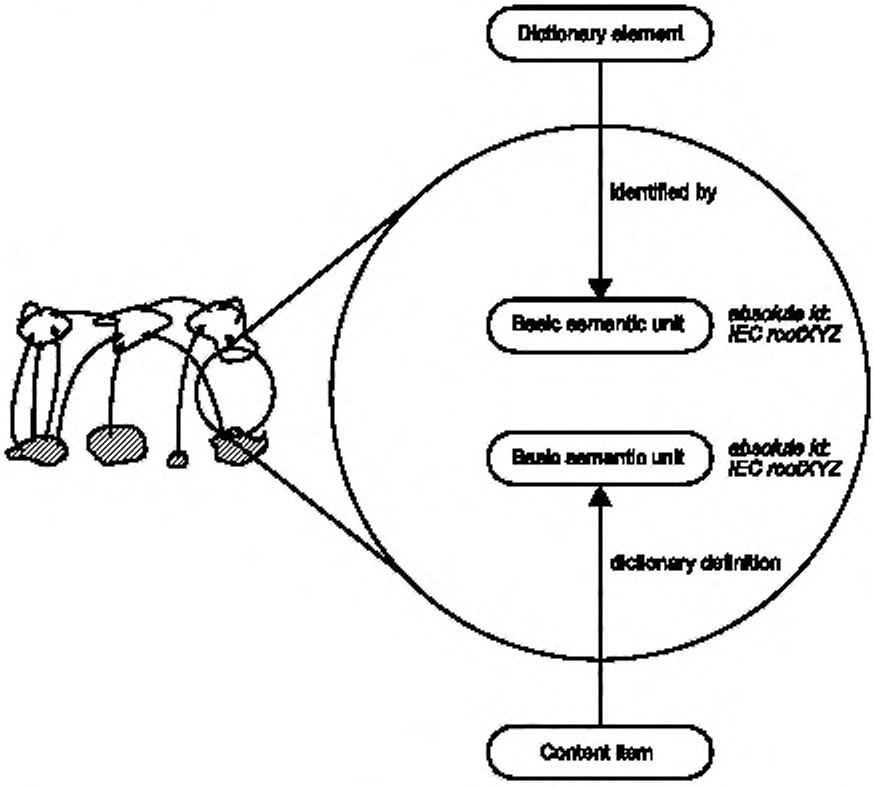
Рисунок 2 — Блоки данных и соотношения между блоками

В ИСО 10303-21 установлено, что каждый из указанных блоков соответствует физическому файлу. Атрибуты языка EXPRESS (см. ИСО 10303-11) могут содержать только ссылки на данные внутри того же физического файла. Таким образом, невозможно использовать атрибуты языка EXPRESS прямо для внутренних ссылок.

5.6.2 Трехуровневая архитектура словарных данных

5.6.2.1 Общие положения

В данном подпункте понятие базовой семантической единицы **basic_semantic_unit** (BSU) вводится как средство применения указанных внутренних ссылок. BSU обеспечивает универсально уникальную идентификацию словарных описаний (см. рисунок 3).



Dictionary element	Словарный элемент
identified by	Идентифицирован ...
Basic semantic unit	Базовая семантическая единица
absolute id	Абсолютный идентификатор
dictionary definition	Словарное определение
Content item	Элемент контента

Рисунок 3 — Практическая реализация «внутреннего» соотношения с помощью базовой семантической единицы

Предположим, что для некоторого контента необходима ссылка на заданное словарное описание.

Пример 1 — Для передачи значения свойства компонента.

Это достигается путем ссылки на базовую семантическую единицу с помощью атрибута словарного определения **dictionary_definition**.

Словарное описание (**dictionary_element**) ссылается на базовую семантическую единицу с помощью атрибута идентификации **identified_by**. Рассматриваемое косвенное соотношение следует из соответствия абсолютных идентификаторов базовой семантической единицы.

Необходимо отметить, что:

- словарный элемент и элемент содержания могут находиться в одном физическом файле, но не обязательно;

- словарный элемент не всегда нужен для обмена элементами содержания, который на него ссылается. В данном случае предполагается, что словарный элемент уже присутствует в словаре целевой системы. И наоборот, словарными данными можно обмениваться без данных контента;

- базовая семантическая единица может быть одной-единственной реализацией в случае, если реализации и словарного элемента, и элемента содержания содержатся в одном физическом файле;
- тот же механизм применяется для ссылок между различными словарными элементами.

Пример 2 — Между классом компонентов и ассоциированными свойствами типов элементов данных property_DET.

BSU обеспечивает ссылку на словарное описание в любом необходимом месте.

Пример 3 — Доставка словаря, доставка обновления, доставка библиотеки, обмен данных компонентов.

Данные, ассоциированные со свойством, можно обменивать как пару (**property_BSU,** <значение>). На рисунке 3 приведена практическая реализация указанного общего механизма.

5.6.2.2 Базовая семантическая единица (basic_semantic_unit)

Basic_semantic_unit — это уникальная идентификация словарного элемента **dictionary_element**.

Пример представления на языке EXPRESS:

```
*)
ENTITY basic_semantic_unit
ABSTRACT SUPERTYPE OF(ONEOF(
      supplier_BSU,
      class_BSU,
      property_BSU,
      data_type_BSU,
      supplier_related_BSU,
      class_related_BSU));
      code: code_type;
      version: version_type;
DERIVE
      dic_identifier: identifier := code + sep_cv + version;
INVERSE
      definition: SET [0:1] OF dictionary_element
            FOR identified_by;
      referenced_by: SET [0:1] OF content_item
            FOR dictionary_definition;
END_ENTITY; -- basic_semantic_unit
(*
```

Определения атрибутов:

code: код, идентифицирующий некоторый словарный элемент.

version: № версии словарного элемента.

dic_identifier: полная идентификация, состоящая из последовательного включения кода и № версии.

definition: ссылка на словарный элемент, идентифицированный указанной BSU. Если определение в некотором контексте обмена отсутствует, то предполагается его присутствие в словаре целевой системы.

referenced_by: элементы, использующие словарный элемент, ассоциированный с указанным BSU.

5.6.2.3 Словарный элемент (dictionary_element)

Dictionary_element — это полное определение данных, отнесенных (в семантическом словаре) к некоторым понятиям. Для каждого понятия используется свой подтип. Словарный элемент **dictionary_element** ассоциирован с базовой семантической единицей **basic_semantic_unit** (BSU), которая уникально идентифицирует данное определение в словаре.

Путем включения атрибута версии в сущность **basic_semantic_unit** формируется частичная идентификация словарного элемента (в качестве примера можно сравнить атрибуты, номер пересмотра **revision** и отметки времени **time_stamps**).

Пример представления на языке EXPRESS:

```
*)
ENTITY dictionary_element
ABSTRACT SUPERTYPE OF(ONEOF(
      supplier_element,
      class_and_property_elements,
      data_type_element));
      identified_by: basic_semantic_unit;
      time_stamps: OPTIONAL dates;
      revision: revision_type;
      administration: OPTIONAL administrative_data;
      is_deprecated: OPTIONAL BOOLEAN;
      is_deprecated_interpretation: OPTIONAL note_type;
WHERE
      WR1: NOT EXISTS (SELF.is_deprecated)
                OR EXISTS (SELF.is_deprecated_interpretation);
END_ENTITY; -- dictionary_element
(*
```

Определения атрибутов:

identified_by: BSU, идентифицирующая словарный элемент.

time_stamps: вспомогательные даты создания и обновления словарного элемента.

revision: № пересмотра словарного элемента.

administration: вспомогательная информация о жизненном цикле словарного элемента **dictionary_element.**

is_deprecated: вспомогательная булевская переменная. Если она равна **true**, то данный словарный элемент **dictionary_element** уже нельзя использовать.

is_deprecated_interpretation: обоснование отказа от использования, а также указание порядка интерпретации значений реализаций больше не используемого элемента и его соответствующей **BSU**.

Примечание 1 — Тип атрибута идентификации **identified_by** может быть повторно определен позже для базовой семантической единицы свойства **property_BSU** и базовой семантической единицы класса **class_BSU.** Тогда он может быть использован для кодирования вместе с атрибутом кода BSU (для свойства и класса соответственно). Он также может быть использован для кодирования атрибута «№ версии» (для свойства и класса соответственно).

Примечание 2 — Атрибут **time_stamps** может быть использован как отправная точка для кодирования (в сущности **dates**) атрибутов свойства и класса «Дата исходного определения», «Дата текущей версии» и «Дата текущего пересмотра» (см. 5.11.3.2).

Примечание 3 — Атрибут пересмотра **revision** используется для кодирования атрибута свойства и класса «№ пересмотра».

Примечание 4 — Атрибут **administration** используется для представления информации, относящейся к управлению конфигурацией и к истории трансляции.

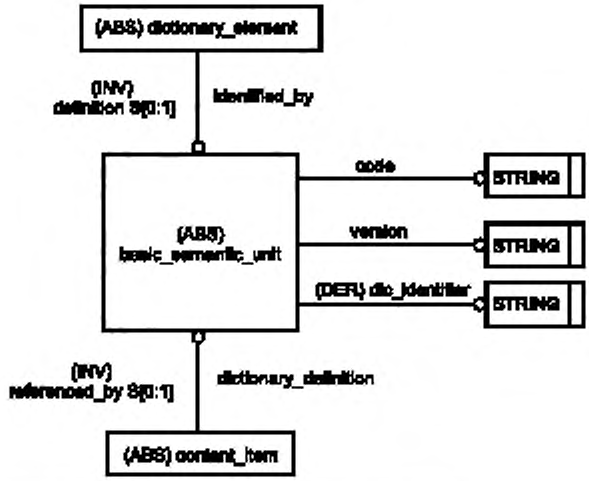
Пояснение к тексту программы:

WR1: если существует элемент **is_deprecated,** то существует и элемент **is_deprecated_ interpretation.**

Дополнительное пояснение:

IP1: значения реализаций элемента **is_deprecated_interpretation** должны быть определены в момент, когда принимается решение об отказе от использования.

На рисунке 4 представлена модель планирования соотношения между базовой семантической единицей и словарным элементом.



(ABS) dictionary_element	Словарный элемент
(INV) definition S[0:1]	Определение ...
identified_by	Идентифицирован ...
code	Код
(ABS) basic_semantic_unit	Базовая семантическая единица
version	№ версии
String	Строка
(INV) referenced_by S[0:1]	Запрашивается ....
dictionary_definition	Определение словаря
(DER) dic_identifier	Идентификатор словаря
(ABS) content_item	Элемент контента

Рисунок 4 — Соотношение между базовой семантической единицей и словарным элементом

5.6.2.4 Элемент контента (Content_item)

Content_item — это блок данных, ссылающийся на свое описание в словаре. Он должен быть описан как подтип.

Пример представления на языке EXPRESS:

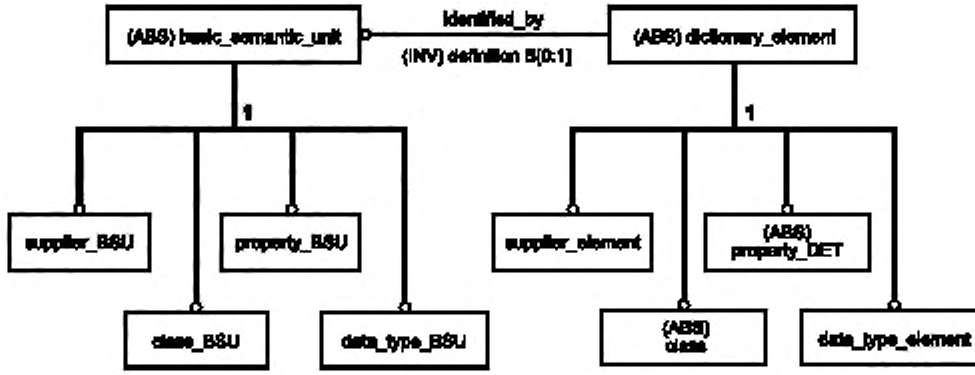
```
*)
ENTITY content_item
ABSTRACT SUPERTYPE;
     dictionary_definition: basic_semantic_unit;
END_ENTITY; -- content_item
(*
```

Определения атрибутов:

dictionary_definition: базовая семантическая единица, используемая для ссылки на определение в словаре.

5.6.3 Краткое описание базовых семантических единиц и словарных элементов

Для каждого вида словарных данных должна быть определена пара подтипов **basic_semantic_unit** и **dictionary_element**. На рисунке 5 (как на модели планирования) показаны базовые семантические единицы (BSU) и словарные элементы. Отметим, что соотношение между BSU и словарным элементом переопределяется для каждого типа данных, так что можно брать только соответствующие пары (на рисунке 5 это не показано).



(ABS) basic_semantic_unit	Базовая семантическая единица
identified_by	Идентифицирован ...
(INV) definition S[0:1]	Определение ...
(ABS) dictionary_element	Словарный элемент
supplier_BSU	BSU поставщика
property_BSU	BSU свойства
supplier_element	Элемент поставщика
(ABS) property_DET	Тип элемента данных свойства
class_BSU	BSU класса
data_type_BSU	BSU типа данных
(ABS) class	Класс
data_type_element	Элемент типа данных

Рисунок 5 — Текущие BSU и словарные элементы

Каждый вид словарных данных рассматривается в одном из следующих разделов:

- для поставщиков см. 5.7;
- для классов см. 5.8;
- для свойств/типов элементов данных см. 5.9;
- для типов данных см. 5.10.

5.6.4 Идентификация словарных элементов: трехуровневая структура

Абсолютная идентификация базовых семантических единиц основана на использовании следующей трехуровневой структуры:

- поставщик (словарных данных);
- определенный поставщиком словарный элемент (любой определенный поставщиком словарный элемент модели; в настоящем стандарте **property_DET** и **data_type_element** определены поставщиком как словарные элементы, при этом существуют положения, распространяющие данный механизм на другие элементы);
- версия определенного поставщиком словарного элемента.

Абсолютная идентификация может быть получена последовательным включением применимых кодов каждого уровня.

Примечание — Структура указанной абсолютной идентификации отличается от структуры, определенной в МЭК 61360-2 (дублируется для удобства в ИСО 13584-42:1998). В предшествующем издании абсолютная идентификация словарного элемента **dictionary_element,** ассоциированная с областью применения названия **name_scope** (включая **property_DET** и **data_type_element**), составлена так: код поставщика + код класса (соответствующего классу **name_scope**) + код словарного элемента + версия словарного элемента. В данной версии настоящего стандарта код класса удален. Таким образом, код словарного элемента является уникальным для одного типа словарного элемента во всех классах, определенных тем же поставщиком. Для существующих ссылочных словарей официальные органы регистрации, официальные органы технической поддержки и группы стандартизации, отвечающие за стандартные словари, должны гарантировать требуемую единственность (например, путем использования новых кодов в качестве приставки для класса **name_scope**).

Данная схема идентификации приемлема, если поставщиков несколько. Если в некоторой области приложений приемлемы данные только от единственного поставщика, то некоторые детали идентификации (которые тогда являются константами) могут отсутствовать. Вместе с тем для выполнения обмена должны быть доступны все уровни во избежание наложения идентификаторов.

Данная схема идентификации формально описана в атрибуте **absolute_id** сущностей **xxx_BSU**, определенных в 5.7—5.12.

5.6.5 Возможности расширения для других типов данных

5.6.5.1 Общие положения

Механизм словарных элементов BSU — очень общий. Он не ограничивается четырьмя видами использованных здесь данных (см. рисунок 5). Данный раздел указывает некоторые возможности, допускающие расширения других видов данных. В зависимости от того, определена область применения идентификатора класса или поставщик, сущность **xxx_related_BSU** описывается как подтип соответствующим образом. Необходимо также переопределить атрибут идентификации **identified_by** сущности **dictionary_element** (см. 5.7.3—5.10 или текущие виды данных).

5.6.5.2 BSU, отнесенная к поставщику (Supplier_related_BSU)

Supplier_related_BSU обеспечивает ассоциацию словарного элемента с поставщиком.

Пример — Для серии стандартов ИСО 13584 — это библиотеки программ.

Пример представления на языке EXPRESS:

```
*)
ENTITY supplier_related_BSU
ABSTRACT SUPERTYPE
SUBTYPE OF(basic_semantic_unit);
END_ENTITY; -- supplier_related_BSU
(*
```

5.6.5.3 BSU, отнесенная к классу (Class_related_BSU)

BSU, отнесенная к классу (**class_related_BSU**), позволяет ассоциировать словарные элементы с классами.

Пример — Для таблиц и документов ИСО 13584.

Пример представления на языке EXPRESS:

```
*)
ENTITY class_related_BSU
ABSTRACT SUPERTYPE
SUBTYPE OF(basic_semantic_unit);
END_ENTITY; -- class_related_BSU
(*
```

5.6.5.4 Соотношение BSU поставщика (Supplier_BSU_relationship)

Соотношение **Supplier_BSU_relationship** позволяет ассоциировать BSU с поставщиком.

Пример представления на языке EXPRESS:

```
*)
ENTITY supplier_BSU_relationship
ABSTRACT SUPERTYPE;
      relating_supplier: supplier_element;
      related_tokens: SET [1:?] OF supplier_related_BSU;
END_ENTITY; -- supplier_BSU_relationship
(*
```

Определения атрибутов:

relating_supplier: элемент **supplier_element**, идентифицирующий поставщика данных.

related_tokens: набор словарных элементов, ассоциированных с поставщиком, идентифицированным атрибутом **relating_supplier**.

5.6.5.5 Соотношение BSU класса (Class_BSU_relationship)

Сущность **class_BSU_relationship** позволяет ассоциировать BSU с классом.

Пример представления на языке EXPRESS:

```
*)
ENTITY class_BSU_relationship
ABSTRACT SUPERTYPE;
      relating_class: class;
      related_tokens: SET [1:?] OF class_related_BSU;
END_ENTITY; -- class_BSU_relationship
(*
```

Определения атрибутов:

relating_class: класс, идентифицирующий словарный элемент.

related_tokens: набор словарных элементов, ассоциированных с классом, идентифицированным атрибутом **relating_class**.

5.7 Поставщик данных

5.7.1 Общие положения

Данный пункт содержит определения для представления данных о самом поставщике. Если поставщиков несколько, то необходимо иметь возможность идентифицировать источник определенного словарного элемента. Рисунок 6 представляет модель планирования данных, ассоциированную с поставщиком, с последующим определением EXPRESS.

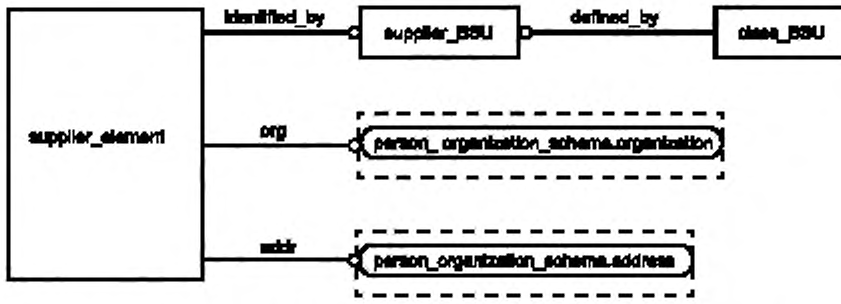


Рисунок 6, лист 1 — Краткое описание поставщика данных и соотношений

identified_by	Идентифицирован ...
supplier_BSU	BSU поставщика
defined_by	Определен ...
class_BSU	BSU класса
supplier_element	Элемент поставщика
org	Организация
person_ organization_schema.organization	Схема коммерческой организации. Организация
addr	Адрес
person_organization_schema.address	Схема коммерческой организации. Адрес

Рисунок 6, лист 2

5.7.2 BSU поставщика (Supplier_BSU)

Сущность **supplier_BSU** задает уникальную идентификацию поставщиков информации.

Пример представления на языке EXPRESS:

```
*)
ENTITY supplier_BSU
SUBTYPE OF(basic_semantic_unit);
      SELF\basic_semantic_unit.code: supplier_code_type;
DERIVE
      SELF\basic_semantic_unit.version: version_type := '1';
      absolute_id: identifier := SELF\basic_semantic_unit.code;
UNIQUE
      UR1: absolute_id;
END_ENTITY; -- supplier_BSU
(*
```

Определения атрибутов:

code: код поставщика, назначенный в соответствии с ИСО 13584-26.

version: номер версии кода поставщика должен быть равен 1.

absolute_id: абсолютная идентификация поставщика.

Пояснение к тексту программы:

UR1: идентификатор поставщика, определенный атрибутом **absolute_id**, является уникальным.

5.7.3 Элемент поставщика (Supplier_element)

Сущность **supplier_element** дает словарное описание поставщиков.

Пример представления на языке EXPRESS:

```
*)
ENTITY supplier_element
SUBTYPE OF(dictionary_element);
      SELF\dictionary_element.identified_by: supplier_BSU;
      org: organization;
      addr: address;
INVERSE
      associated_items: SET [0:?] OF supplier_BSU_relationship
          FOR relating_supplier;
END_ENTITY; -- supplier_element
(*
```

Определения атрибутов:

identified_by: BSU поставщика **supplier_BSU**, используемая для идентификации рассматриваемого элемента **supplier_element**.

org: организационные данные поставщика.

addr: адрес поставщика.

associated_item: разрешает доступ к другим видам данных с помощью механизма BSU.

Пример — Библиотека программ в ИСО 13584-24:2003.

5.8 Данные класса

5.8.1 Общие положения

Данный пункт содержит определения для представления словарных данных классов.

На рисунке 7 в виде модели планирования представлены данные, ассоциированные с классами, и их соотношения с другими словарными элементами.

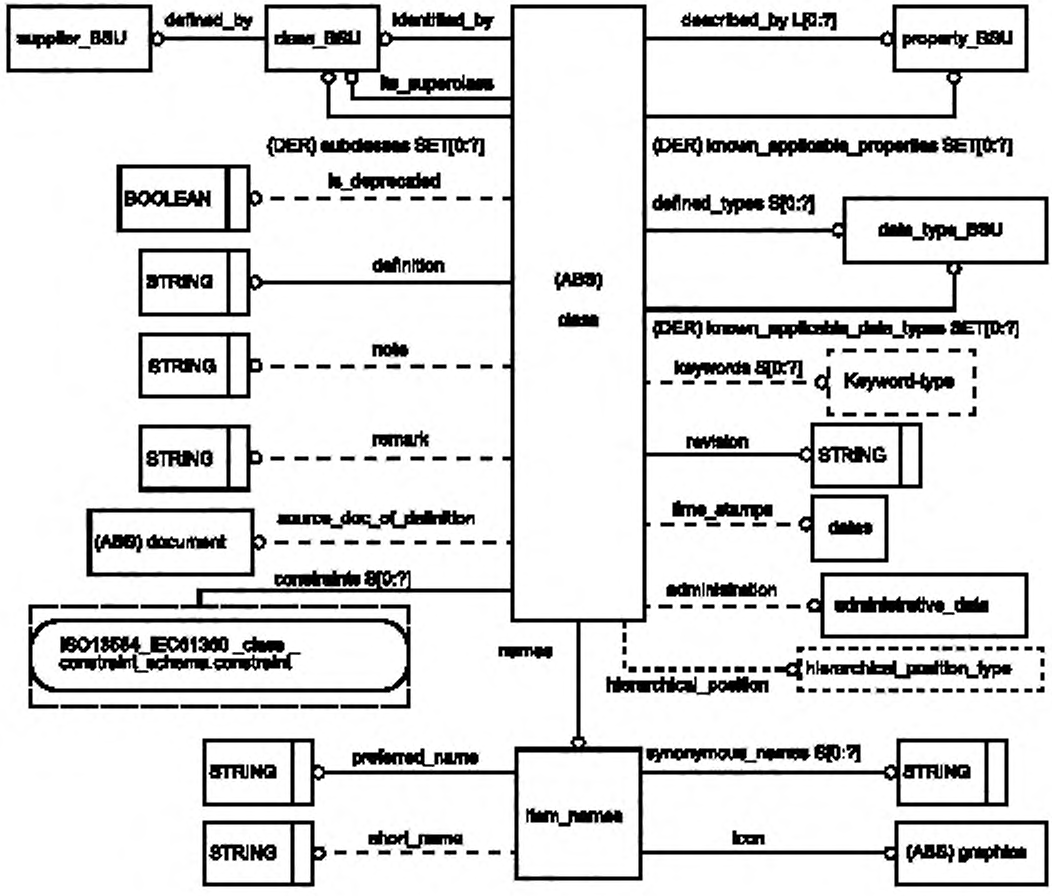


Рисунок 7, лист 1 — Краткое описание данных класса и соотношений

supplier_BSU	BSU поставщика
defined_by	Определен ...
class_BSU	BSU класса
identified_by	Идентифицирован ...
described_by L[0:?]	Описан ...
property_BSU	BSU свойства
its_superclasses	Его суперкласс
(DER) subclass SET[0:?]	Подклассы
(DER) known_applicable_properties SET[0:?]	Известные применимые свойства
boolean	Булевская переменная
is_deprecated	Больше не используется
defined_types S[0:?]	Определенные типы
data_type_BSU	BSU типа данных
string	Строка
definition	Определение
(ABS) class	Класс
(DER) known_applicable_data_types SET[0:?]	Набор известных применимых типов данных
keyword-type	Тип ключевого слова
remark	Заметка
revision	Пересмотр
(ABS) document	Документ
source_doc_of_definition	Исходный документ определения
time_stamps	Временные отметки
dates	Даты
constraints S[0:?]	Ограничения
administration	Администрация
ISO13584_IEC61360 _class _constraint_ schema.constraint	Стандартная схема ограничений класса. Ограничение
names	Названия
hierarchical_position_type	Тип положения в иерархии
hierarchical_position	Положение в иерархии
preferred_name	Предпочтительное имя
short_name	Короткое имя
item_names	Названия элементов
synonymous_names S[0:?]	Синонимичные имена
(ABS) graphics	Графика
icon	Иконка

Рисунок 7, лист 2

Как указано на рисунке 7, с помощью атрибута **its_superclass** классы формируют дерево наследственности. Важно отметить, что во всем документе термины «наследование» и «наследовать» используются в соотношении между классами (определенном в словаре). Язык EXPRESS также включает понятие наследования. Это должно быть четко указано во избежание недоразумений.

Словарные данные классов (см. рисунок 7) распределены по трем уровням наследования:

- **class_and_property_element** определяет данные, являющиеся общими и для классов, и для типов данных свойств **property_DET**;
- рассматриваемый класс позволяет описывать другие виды классов позже.

Пример — Другие подтипы классов (особенно класс функциональных видов functional_view_class, класс функциональных моделей functional_model_class и класс fm_class_view_of) установлены ИСО 13584-24. Они не характеризуют продукты, но помогают обменивать конкретные представления продуктов (например, геометрические представления);

- класс элементов **item_class** и класс категоризаций **categorization_class** — это сущности, содержащие данные различных классов объектов рассматриваемой области приложения.

Примечание 1 — Два подтипа класса элементов **item_class** (класс компонентов **component_class** и класс материалов **material_class)** определены внутри словарной модели первого издания МЭК 61360. Указанные подтипы больше не используются. Они удалены из настоящего стандарта.

Примечание 2 — Следующие изменения гарантируют, что определения класса словаря, удовлетворяющие требованиям первого издания МЭК 61360-2, удовлетворяют и требованиям настоящего стандарта: (1) замена классов **component_class** и **material_class** на класс **item_class** с помощью ссылочного словаря; (2) добавление к каждому новому классу элементов **item_class** атрибута совместного использования реализаций **instance_sharable** со значением *true*; (3) добавление к каждому новому классу элементов **item_class** вспомогательного атрибута **hierarchical_position** без задания его значения; (4) добавление к каждому новому классу элементов **item_class** атрибута ключевого слова **keywords**, значением которого является пустое множество.

Примечание 3 — Другой подтип **item_class,** называемый классом особенностей **feature_class,** определен ИСО 13584-24:2003. Данный подтип также больше не используется. Его использование не допускается в новых практических реализациях настоящего стандарта.

Примечание 4 — Следующие изменения гарантируют, что определения класса словаря, удовлетворяющего требованиям ИСО 13584-25, удовлетворяют также и требованиям настоящего стандарта: (1) замена класса особенностей **feature_class** на класс элементов **item_class** с помощью ссылочного словаря; (2) добавление к каждому новому классу элементов **item_class** атрибута совместного использования реализаций **instance_sharable** со значением *false*; (3) добавление к каждому новому классу элементов **item_class** вспомогательного атрибута **hierarchical_position** без значения; (4) добавление к каждому новому классу элементов **item_class** атрибута ключевого слова **keywords,** значением которого является пустое множество.

5.8.2 Особенности структуры

5.8.2.1 BSU класса (Class_BSU)

Сущность **class_BSU** обеспечивает идентификацию класса.

Пример представления на языке EXPRESS:

```
*)
ENTITY class_BSU
SUBTYPE OF(basic_semantic_unit);
      SELF\basic_semantic_unit.code: class_code_type;
      defined_by: supplier_BSU;
DERIVE
      absolute_id: identifier
           := defined_by.absolute_id + sep_id + dic_identifier;
      known_visible_properties: SET [0:?]OF property_BSU
           := compute_known_visible_properties(SELF);
        known_visible_data_types: SET [0:?]OF data_type_BSU
           := compute_known_visible_data_types(SELF);
INVERSE
      subclasses: SET [0:?] OF class FOR its_superclass;
      added_visible_properties: SET [0:?] OF property_BSU
```

```
        FOR name_scope;
    added_visible_data_types: SET [0:?] OF data_type_BSU
        FOR name_scope;
UNIQUE
    UR1: absolute_id; END_
ENTITY; -- class_BSU
(*
```

Определения атрибутов:

code: код, назначенный данному классу его поставщиком.

defined_by: поставщик, определяющий данный класс и его словарный элемент.

absolute_id: уникальная идентификация данного класса.

known_visible_properties: набор BSU свойств **property_BSU**, ссылающихся на класс как атрибут имени **name_scope** или на любой известный суперкласс данного класса и, следовательно, являющихся видимыми в данном классе (любом его подклассе).

Примечание 1 — Если словарное определение **dictionary_definition** некоторого класса не присутствует в рассматриваемом контексте обмена (а контекст обмена PLIB никогда не предполагается полным), то суперкласс для рассматриваемого класса может быть неизвестен. Следовательно, свойства, определенные как видимые в данном суперклассе, не принадлежат атрибуту **known_visible_properties**. Только для получающей системы все словарные определения **dictionary_definition** BSU должны быть доступными. Следовательно, для получающей системы атрибут **known_visible_properties** содержит все свойства, видимые в данном классе.

known_visible_data_types: набор BSU типов данных **data_type_BSU**, ссылающихся на класс как атрибут **name_scope** или на любой известный суперкласс данного класса и, следовательно, являющихся видимыми в данном классе (любом его подклассе).

Примечание 2 — Если некоторые словарные определения **dictionary_definition** класса не присутствуют в рассматриваемом контексте обмена (а контекст обмена библиотеки PLIB никогда не предполагается полным), то суперкласс для данного класса может быть неизвестен. Следовательно, типы данных **data_types,** определенные как видимые данным суперклассом, не принадлежат атрибуту известного видимого типа данных **known_visible_data_type**. Для получающей системы все словарные определения **dictionary_definition** BSU должны быть доступными. Следовательно, для получающей системы атрибут **known_visible_data_type** содержит все типы данных **data_types,** видимые в данном классе.

subclasses: набор классов, определяющих данный класс как их суперкласс.

added_visible_properties: набор BSU свойств **property_BSU**, ссылающихся на класс как **name_scope** и, следовательно, являющихся видимыми в данном классе (любом его подклассе).

Примечание 3 — Данный атрибут ссылается только на свойства **property_BSU**, принадлежащие рассматриваемому контексту обмена. Для получающей системы они могут уже завершить выполнение других свойств **property_BSU**, ссылающихся на данный класс (контекст обмена библиотеки PLIB никогда не предполагается полным).

Примечание 4 — Атрибут **added_visible_properties** используется для кодирования атрибута «Видимые свойства» класса.

added_visible_data_type: набор BSU типов данных **data_type_BSU**, ссылающихся на класс как и их атрибут **name_scope** и, следовательно, являющихся видимыми в данном классе (любом его подклассе).

Примечание 5 — Данный обратный атрибут ссылается только на **data_type_BSU,** принадлежащий рассматриваемому контексту обмена. Для получающей системы они могут уже завершить выполнение других **data_type_BSU**, ссылающихся на данный класс (контекст обмена библиотеки PLIB никогда не предполагается полным).

Примечание 6 — Атрибут **added_visible_data_type** используется для кодирования атрибута «Видимые типы».

Пояснение к тексту программы:

UR1: последовательность кода поставщика и кода класса является уникальной.

5.8.2.2 Элемент класса и свойства (Class_and_property_element)

Сущность **class_and_property_element** содержит атрибуты, являющиеся общими и для классов, и для BSU свойств **property_DET.**

Пример представления на языке EXPRESS:

```
*)
ENTITY class_and_property_elements
ABSTRACT SUPERTYPE OF(ONEOF(
     property_DET,
     class))
SUBTYPE OF(dictionary_element);
     names: item_names;
     definition: definition_type;
     source_doc_of_definition: OPTIONAL document;
     note: OPTIONAL note_type;
     remark: OPTIONAL remark_type;
END_ENTITY; -- class_and_property_elements
(*
```

Определения атрибутов:

names: названия, описывающие данный словарный элемент.

definition: текст, описывающий данный словарный элемент.

source_doc_of_definition: исходный документ для данного текстового описания.

note: дополнительная информация о любой части словарного элемента, существенной для понимания.

remark: дополнительный текст, поясняющий смысл данного словарного элемента.

Примечание 1 — Атрибут **names** используется как отправная точка при кодировании (в сущности **item_names**) свойства и атрибутов «Предпочтительное имя», «Краткое имя», «Синонимичное имя».

Примечание 2 — Атрибут определения **definition** используется для кодирования атрибута свойства «Определение» и атрибута класса «Определение».

Примечание 3 — Атрибут исходного документа для определения **source_of_doc_definition** используется для кодирования атрибута свойства «Исходный документ определения» и атрибута класса «Исходный документ определения».

Примечание 4 — Атрибут примечания **note** используется для кодирования атрибутов свойств и классов «Примечание».

Примечание 5 — Атрибут заметки **remark** используется для кодирования атрибутов свойств и классов «Заметка».

5.8.2.3 Класс (class)

Сущность **class** — это абстрактный ресурс для всех видов классов.

Пример представления на языке EXPRESS:

```
*)
ENTITY class
ABSTRACT SUPERTYPE OF( ONEOF (item_class, categorization_class))
SUBTYPE OF(class_and_property_elements);
     SELF\dictionary_element.identified_by: class_BSU;
     its_superclass: OPTIONAL class_BSU;
     described_by: LIST [0:?] OF UNIQUE property_BSU;
     defined_types: SET [0:?] OF data_type_BSU;
     constraints: SET [0:?] OF constraint_or_constraint_id;
     hierarchical_position: OPTIONAL hierarchical_position_type;
     keywords: SET [0:?] OF keyword_type;
     sub_class_properties: SET [0:?] OF property_BSU;
     class_constant_values: SET [0:?] OF class_value_assignment;
DERIVE
     subclasses: SET [0:?] OF class :- identified_by.subclasses;
     known_applicable_properties: SET [0:?] OF property_BSU
          :- compute_known_applicable_properties(
```

```
            SELF\dictionary_element.identified_by);
      known_applicable_data_types: SET [0:?] OF data_type_BSU
            := compute_known_applicable_data_types(
            SELF\dictionary_element.identified_by);
      known_property_constraints: SET [0:?] OF property_constraint
            := compute_known_property_constraints(
            [SELF\dictionary_element.identified_by]);
INVERSE
      associated_items: SET [0:?] of class_BSU_relationship
            FOR relating_class;
WHERE
      WR1: acyclic_superclass_relationship(SELF.identified_by, []);
      WR2: NOT all_class_descriptions_reachable(
            SELF\dictionary_element.identified_by)
            OR (list_to_set(SELF.described_by) <=
            SELF\dictionary_element.identified_by
            \class_BSU.known_visible_properties);
      WR3: NOT all_class_descriptions_reachable(
            SELF\dictionary_element.identified_by)
            OR (SELF.defined_types <=
            SELF\dictionary_element.identified_by
            \class_BSU.known_visible_data_types);
      WR5: NOT all_class_descriptions_reachable(
            SELF\dictionary_element.identified_by)
            OR (QUERY (cdp <* described_by
            | (SIZEOF (cdp\basic_semantic_unit.definition)=1)
            AND (('ISO13584_IEC61360_DICTIONARY_SCHEMA'
            +'.DEPENDENT_P_DET') IN TYPEOF
            (cdp\basic_semantic_unit.definition[1]))
            AND NOT
            (cdp\basic_semantic_unit.definition[1].depends_on
            <= known_applicable_properties))=[]);
      WR6: check_datatypes_applicability(SELF);
      WR7: QUERY (cons <* constraints
            | ('ISO13584_IEC61360_CLASS_CONSTRAINT_SCHEMA'
            +'.INTEGRITY_CONSTRAINT' IN TYPEOF (cons))
            AND (SIZEOF (cons\property_constraint.constrained_property
            .definition) =1)
            AND NOT correct_constraint_type(
            cons\integrity_constraint.redefined_domain,
            cons\property_constraint.constrained_property
            .definition[1].domain)) = [];
      WR8: QUERY (cons <* constraints
            | (('ISO13584_IEC61360_CLASS_CONSTRAINT_SCHEMA'
            +'.CONFIGURATION_CONTROL_CONSTRAINT') IN TYPEOF (cons))
            AND NOT correct_precondition (cons, SELF)) = [];
      WR9: NOT all_class_descriptions_reachable(
            SELF\dictionary_element.identified_by)
            OR (QUERY (cons <* constraints
            | (('ISO13584_IEC61360_CLASS_CONSTRAINT_SCHEMA'
            +'.PROPERTY_CONSTRAINT') IN TYPEOF (cons))
            AND NOT
            ((cons\property_constraint.constrained_property
            IN SELF\dictionary_element.identified_by
            \class_BSU.known_visible_properties)
            OR (cons\property_constraint.constrained_property
            IN known_applicable_properties)))=[]);
      WR10: (SIZEOF( QUERY (lab <* keywords
            | ('ISO13584_IEC61360_DICTIONARY_SCHEMA'
```

```
            +'.LABEL_WITH_LANGUAGE') IN TYPEOF (lab)))
            = SIZEOF( keywords))
            OR (SIZEOF (QUERY (lab <* keywords
            : ('ISO13584_IEC61360_DICTIONARY_SCHEMA'
            +'.LABEL_WITH_LANGUAGE') IN TYPEOF (lab)))
            = SIZEOF( keywords));
       WR11: (('ISO13584_IEC61360_ITEM_CLASS_CASE_OF_SCHEMA'
            + '.A_PRIORI_SEMANTIC_RELATIONSHIP')
            IN TYPEOF (SELF)) OR
            ( QUERY(p <* sub_class_properties
            : NOT(p IN SELF.described_by)) = []);
       WR12: NOT all_class_descriptions_reachable(SELF.identified_by) OR
            (QUERY(va <* class_constant_values |
            NOT is_class_valued_property(
            va.super_class_defined_property, SELF.identified_by)) = []);
       WR13: QUERY(val <* SELF.class_constant_values
            · QUERY (v <* class_value_assigned (
            val.super_class_defined_property, SELF.identified_by)
            ; val.assigned_value <> v) <>[]) = [];
END_ENTITY; -- class
(*
```

Определения атрибутов:

identified_by: BSU класса **class_BSU**, задающая данный класс.

its_superclass: ссылка на класс, для которого рассматриваемый класс является подклассом.

described_by: список ссылок на дополнительные свойства, доступные для использования в описании продуктов внутри класса и любого его подкласса.

Примечание 1 — Свойство может также быть применимым в классе, если оно импортировано из другого класса с помощью априорного семантического соотношения **a_priori_semantic_relationship** (см. 8.5 настоящего стандарта). Следовательно, свойства, на которые производится ссылка атрибутом **described_by**, не определяют все применимые свойства класса.

Примечание 2 — Списочный порядок — это порядок представления свойств, предложенный поставщиком.

Примечание 3 — Свойство, являющееся контекстно-зависимым (**context_dependent_P_DET**), может стать применимым в классе, только если все контекстные параметры (**condition_DET**), от которых зависит его значение, также являются применимыми в данном классе.

defined_types: набор ссылок на типы, которые могут быть использованы для различных типов элементов данных свойств **property_DET** с помощью дерева наследственности, нисходящего из данного класса.

Примечание 4 — Тип данных **data_type** также может быть применимым в классе, если этот тип **data_type** импортирован из другого класса с помощью априорного семантического соотношения **a_priori_semantic_relationship** (см. 8.5 настоящего стандарта). Следовательно, типы данных, на которые производится ссылка атрибутом **defined_type**, не определяют все применимые типы данных в классе.

constraints: набор, ограничивающий целевые области значений некоторых свойств класса до некоторых подмножеств унаследованных областей значений.

Примечание 5 — Каждое ограничение атрибута **constraints** должно выполняться для реализаций класса. Таким образом, атрибут **constraints** — это сопряжение ограничений.

hierarchical_position: кодированное представление положения класса в иерархии включения классов, которой он принадлежит; иерархическая позиция **hierarchical_position** класса изменяется, если изменяется структура класса онтологии. Таким образом, данный атрибут не может использоваться как устойчивый идентификатор класса.

Примечание 6 — Данный вид кодированного имени используется, в особенности в иерархии категоризаций продукта для представления структуры включения классов с помощью некоторых конвенций кодирования.

Пример — В спецификации UNSPSC OOH производственные компоненты и поставки имеют иерархическое положение 31000000. Металлические крепежные изделия занимают иерархическое

положение 31160000. Болты занимают иерархическое положение 31161600. По конвенции данное представление иерархического положения допускает, чтобы производственные компоненты и поставки располагались на первом уровне иерархии, металлические крепежные изделия располагались на втором уровне иерархии (и включались в производственные компоненты и поставки), а болты были на третьем уровне иерархии (и включались в металлические крепежные изделия).

keywords: набор ключевых слов (возможно, на нескольких языках), позволяющих отыскивать нужный класс.

sub_class_properties: объявляет свойство как имеющее значение класса, т. е. в нескольких подклассах его единственное значение назначено для всего класса (см. 5.9.6).

class_constant_value: назначения в текущем классе для свойств, имеющих значение класса и объявленных в суперклассах (см. 5.9.6).

subclasses: набор классов, указывающих на данный класс как их суперкласс.

known_applicable_properties: BSU свойства **property_BSU**, на которые производится ссылка некоторым классом или любым его известным суперклассом с помощью атрибута **described_by** и которые, следовательно, являются применимыми в данном классе (любом его подклассе).

Примечание 7 — Если некоторое словарное определение **dictionary_definition** класса отсутствует в рассматриваемом контексте обмена (а контекст обмена библиотеки PLIB никогда не предполагается полным), то суперкласс некоторого класса может быть неизвестен. Следовательно, свойства, определенные как применимые в данном суперклассе, не принадлежат атрибуту известных применимых свойств **known_applicable_properties**. Все словарные определения **dictionary_definition** BSU должны быть доступны только для получающей системы. Следовательно, для получающей системы атрибут **known_applicable_properties** содержит все свойства, являющиеся применимыми в классе, так как на них производится ссылка атрибутом описания **described_by**.

known_applicable_data_type: BSU типа данных **data_type_BSU**, на которые производится ссылка из класса или любого известного суперкласса с помощью атрибута **defined_types** и которые, следовательно, являются применимыми в данном классе (любом его подклассе).

Примечание 8 — Если некоторое словарное определение **dictionary_definition** класса отсутствует в рассматриваемом контексте обмена (а контекст обмена библиотеки PLIB никогда не предполагается полным), то суперкласс некоторого класса может быть неизвестен. Следовательно, типы данных **data_types**, определенные как применимые в данном суперклассе, не принадлежат атрибуту **known_applicable_data_type**. Все словарные определения **dictionary_definition** BSU должны быть доступными только для получающей системы. Следовательно, для получающей системы атрибут **known_applicable_data_types** содержит все типы данных **data_types**, являющиеся применимыми в классе, так как на них производится ссылка атрибутом определения типа **defined_types**.

known_property_constraint: ограничения **constraints** на свойство, на которые производится ссылка некоторым классом или любым его известным представительным суперклассом с помощью атрибута **constraints** или, в случае класса, являющегося подтипом априорного семантического соотношения **a_priori_semantic_relationship**, с помощью атрибута ссылочного ограничения **referenced_constraints**.

associated_item: дает доступ к другим видам данных с помощью BSU механизма.

Пояснения к тексту программы:

WR1: наследственная структура, определенная иерархией классов, не содержит циклы.

WR2: только свойства, являющиеся видимыми в данном классе, могут стать применимыми в данном классе, так как на них производится ссылка атрибутом описания **described_by**.

WR3: только типы данных, являющиеся видимыми в данном классе, могут стать применимыми в данном классе, так как на них производится ссылка атрибутом **defined_types**.

WR4: только свойства, не являющиеся применимыми в классе по наследству, могут стать применимыми в данном классе, так как на них производится ссылка атрибутом описания **described_by**.

WR5: только контекстно-зависимые свойства **(dependent_P_DET)**, все контекстные параметры **(condition_DET)** которых являются применимыми в классе, могут стать применимыми в данном классе, так как на них производится ссылка атрибутом описания **described_by**.

WR6: только типы данных, не являющиеся применимыми в классе по наследству, могут стать применимыми в данном классе, так как на них производится ссылка атрибутом определения типа **defined_types**.

Примечание 9 — Атрибут **its_superclass** используется для кодирования атрибута класса «Суперкласс».

Примечание 10 — Атрибут **described_by** обеспечивает кодирование «Применимых свойств» класса.

Примечание 11 — Атрибут **defined_type** используется для кодирования атрибута «Применимых типов» класса.

WR7: набор ограничений, являющихся ограничениями свойств, должен определять ограничения, совместимые с областью значений свойств, где последние применимы.

WR8: все свойства, на которые производится ссылка в предварительном условии ограничения управления конфигурацией **configuration_control_constraint**, должны быть применимыми в классе.

WR9: все свойства, на которые производится ссылка в атрибуте ограничения **constraint**, должны быть либо видимыми, либо применимыми в классе.

WR10: либо все ключевые слова **keywords** представлены как метки с языком **label_with_languages**, либо все они представлены как просто метки **labels**.

WR11: если класс не является априорным семантическим соотношением **a_priori_semantic_relationship**, то свойства подкласса **sub_class_properties** должны принадлежать списку **described_by**.

Примечание 12 — Свойства подкласса **sub_class_properties** могут быть также импортированы с помощью априорного семантического соотношения **a_priori_semantic_relationship**.

WR12: свойства, на которые производится ссылка в атрибуте постоянных значений класса **class_constant_value**, объявляются как имеющие значение класса в некотором суперклассе текущего класса или в самом текущем классе.

Примечание 13 — Атрибут свойств подкласса **sub_class_properties** сущности **class** используется для кодирования атрибута «Свойств со значением класса».

Примечание 14 — Атрибут постоянных значений класса **class_constant_value** сущности **class** используется для кодирования «Постоянных значений класса».

WR13: если свойству, на которое производится ссылка из области **class_constant_value**, уже назначено значение в суперклассе, то его значение, назначенное в текущем классе, должно быть тем же самым.

Дополнительное пояснение:

IP1: если все представительные суперклассы рассматриваемого класса доступны, то известные ограничения свойства **known_property_constraints** — это ограничения свойств, присоединенных к данному классу либо как видимые, либо как применимые свойства.

5.8.3 Класс элементов (item_class)

Сущность **item_class** позволяет моделировать любой тип сущности области приложения, который может быть отнесен к характеристическому классу, определенному структурой класса и набором свойств. Как реализации продуктов, так и реализации конкретных аспектов продуктов, представленные как особенности, отображаются на **item_class**.

Сущность **item_class** включает атрибут совместного использования реализации **instance_sharable**, указывающий концептуальный статус элемента. Если значение данного атрибута — *true*, то каждая реализация представляет собой независимый элемент. В противном случае — это особенность, т. е. зависимый элемент, являющийся компонентом другого элемента. Вышесказанное не предписывает какой-либо особой практической реализации на уровне представления данных.

Пример — Головка винта — это особенность, описанная рядом свойств. Она может существовать, только если на нее производится ссылка сущностью «винт». Она представляется как класс элементов item_class с атрибутом instance_sharable, равным false.

Примечание 1 — Внутри словарной модели первого издания ИСО 13584-42 и МЭК 61360-2 определено два подтипа класса элементов **item_class**: класс компонентов **component_class** и класс материалов **material_class**. Указанные подтипы больше не используются, они удалены из МЭК 61360.

Примечание 2 — Другой подтип **item_class**, названный классом особенностей **feature_class**, определен в ИСО 13584-24:2003. Данный подтип также больше не используется. Его использование не рекомендуется в новых практических реализациях настоящего стандарта.

Пример представления на языке EXPRESS:

```
*)
ENTITY item_class
SUBTYPE OF(class);
      simplified_drawing: OPTIONAL graphics;
      coded_name: OPTIONAL value_code_type;
```

```
      instance_sharable: OPTIONAL BOOLEAN;
END_ENTITY; -- item_class
(*
```

Определения атрибутов:

simplified_drawing: вспомогательные рисунки (графика), ассоциированные с описанным классом.

Примечание 3 — Атрибут **simplified_drawing** сущности **item_class** используется для кодирования атрибута «Упрощенный рисунок» для классов.

coded_name: может использоваться как постоянное значение класса для характеристики данного класса в области значений свойств **sub_class_properies** соответствующего суперкласса.

Примечание 4 — В серии стандартов ИСО 13584 данный атрибут не используется. Он используется только в серии стандартов МЭК 61360.

instance_sharable: если значение равно *false*, то реализации класса элементов **item_class** являются особенностями; если значение не определено или равно *true*, то реализации класса элементов **item_class** являются независимыми элементами.

Примечание 5 — В общей словарной модели ИСО 13584/МЭК 61360 от практической реализации зависит принятие решения: действительно ли несколько реализаций особенностей, моделируемых рассматриваемым набором пар «свойство-значение», представлены несколькими блоками данных EXPRESS или они представлены этим блоком данных в файле обмена данных. Таким образом, реализация класса элементов **item_class** со значением атрибута **instance_sharable** равным *false*, на которую производится ссылка несколькими реализациями **item_class** на уровне модели данных, интерпретируется как несколько реальных реализаций рассматриваемой особенности.

5.8.4 Класс категоризаций (categorization_class)

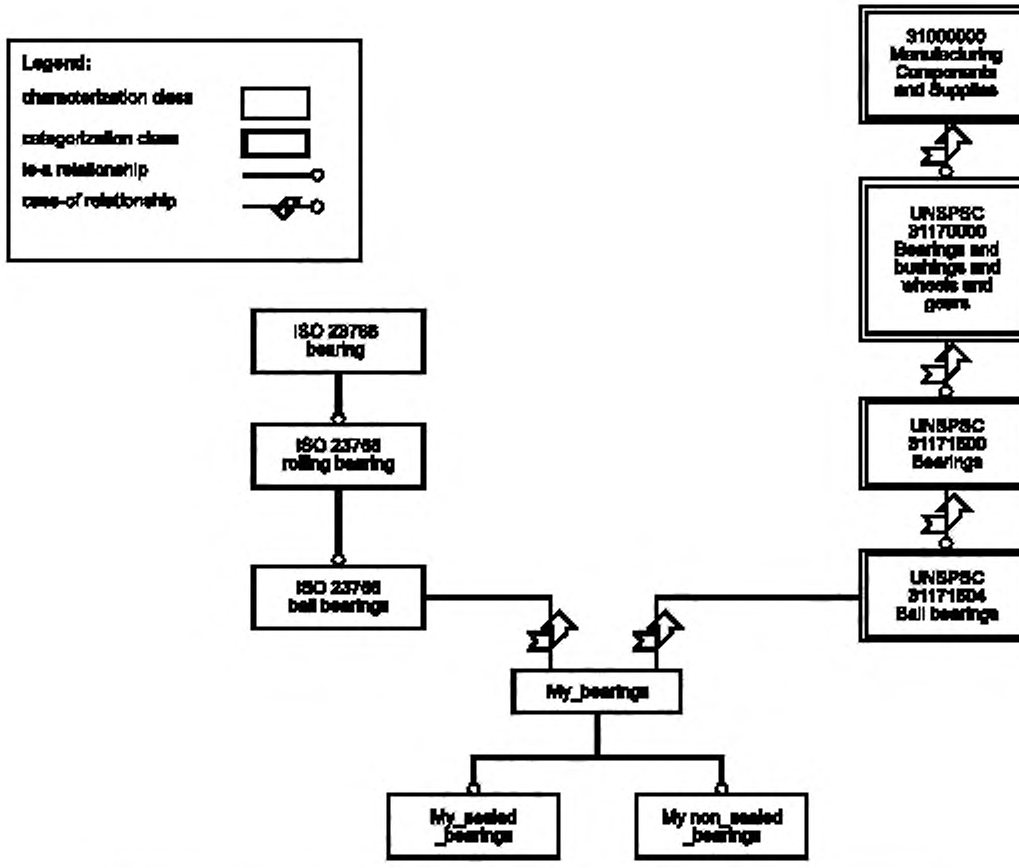
Сущность **categorization_class** позволяет моделировать образование групп из набора объектов, составляющих элемент категоризации.

Пример 1 — Производственные компоненты и поставки и промышленная оптика — это примеры класса категоризации продуктов, определенного UNSPSC.

Ни свойства, ни типы данных, ни ограничения не ассоциируются (как видимые или применимые) с таким классом. Более того, классы категоризаций **categorization_classes** не могут быть отнесены друг к другу с помощью представительного наследственного соотношения. Они могут быть отнесены друг к другу только с помощью условного соотношения между классами. Особый атрибут, называемый **categorization_class_superclass**, позволяет занести классы категоризаций **categorization_classes**, являющиеся суперклассами **categorization_class**, в условную иерархию.

Примечание — С помощью условных соотношений конструктивы ресурса, классы элементов **item_class** могут также быть соединены с классами категоризаций **categorization_class.**

Пример 2 — Следующий пример показывает, как характеристические классы и классы категоризаций могут быть соединены для достижения некоторых конкретных целей. Поставщик шариковых подшипников разрабатывает свою собственную онтологию, что облегчает получение и использование продукта. Для достижения указанной цели он использует стандартные свойства и пытается использовать стандартную классификацию. Он поставляет только шариковые подшипники: некоторые подшипники герметичны, а некоторые — нет. Определенные свойства могут быть ассоциированы только с герметичными подшипниками, а с негерметичными — нет. Однако указанные категории не существуют как классы в стандартной онтологии подшипников. Поэтому поставщик подшипников поступает следующим образом. (1) Он разрабатывает собственную онтологию, включающую три характеристических класса: my_bearing (мои подшипники), my_sealed_bearings (мои герметичные подшипники), my_non_sealed_bearing (мои негерметичные подшипники). Два последних класса соединены с первым представительным наследственным соотношением, и все свойства, назначенные первому, наследуются последними. (2) Чтобы использовать некоторые свойства, определенные ИСО/ТС 23768-1 (находится в разработке), поставщик подшипников указывает, что класс my_bearings — это условный стандартный класс подшипников ball bearing, определенный в ИСО/ТС 23768-1. С помощью данного условного соотношения поставщик может импортировать (в свой класс my_bearings) стандартные свойства: внутренний диаметр, наружный диаметр, класс допуска ИСО. Более того, он создает необходимые свойства, не определенные в стандарте. (3) Для получения сервера, визуализирующего каталог поставщика, представляется небольшой фрагмент классификации UNSPSC, а также условное соотношение между стандартным классом ball_bearings UNSPSC и своим собственным классом my_bearing (см. пример на рисунке 8).



Legend	Легенда
characterization class	Характеристический класс
categorization class	Класс категоризаций
Is-a relationship	Представительное соотношение
case-of relationship	Условное соотношение
31000000 Manufacturing Components and Supplies	Код по классификации ООН 31000000 Производственные компоненты и поставки
UNSPSC 31170000 Bearings and bushings and wheels and gears	Подшипники, втулки, колеса и зубчатые передачи по классификации ООН
ISO 23768 bearing	Подшипник
ISO 23768 rolling bearing	Роликовый подшипник
UNSPSC 31171500 Bearings	Подшипники по классификации ООН
ISO 23768 ball bearing	Шариковый подшипник
My_bearings	Класс «мои подшипники»
UNSPSC 31171504 Ball bearings	Шариковые подшипники по классификации ООН
My_sealed_bearings	Класс «мои герметичные подшипники»
My non_sealed_bearings	Класс «мои негерметичные подшипники»

Рисунок 8 — Пример онтологии поставщика

Пример представления на языке EXPRESS:

```
*)
ENTITY categorization_class
SUBTYPE OF(class);
        categorization_class_superclasses: SET [0:?] of class_BSU;
WHERE
     WR1: QUERY (cl <* SELF. categorization_class_superclasses
          | NOT (('ISO13584_IEC61360_DICTIONARY_SCHEMA'
          +'.CATEGORIZATION_CLASS') IN TYPEOF(cl.definition[1])))
          = [];
     WR2: NOT EXISTS(SELF\class.its_superclass);
     WR3: SIZEOF(SELF\class.described_by) = 0;
     WR4: SIZEOF(SELF\class.defined_types) = 0;
     WR5: SIZEOF(SELF\class.constraints) = 0;
     WR6: SIZEOF(compute_known_visible_properties
          (SELF\dictionary_element.identified_by)) = 0;
     WR7: SIZEOF(SELF\class.sub_class_properties) = 0;
     WR8: SIZEOF(SELF\class.class_constant_values) = 0;
     WR9: SIZEOF(SELF\class.identified_by.known_visible_properties)
          = 0;
     WR10: SIZEOF(SELF\class.identified_by.known_visible_data_types)
          = 0;
     END_ENTITY; -- categorization_class
     (*
```

Определения атрибутов:

categorization_class_superclasses: классы категоризаций **categorization_class**, расположенные на одну ступень выше данного класса категоризаций в условной иерархии классов.

Пояснения к тексту программы:

WR1: только классы категоризаций **categorization_class** могут быть суперклассами для **categorization_class**.

WR2: categorization_class не должен иметь представительного суперкласса.

WR3: свойства не могут быть ассоциированы с **categorization_class**.

WR4: типы данных не могут быть ассоциированы с **categorization_class**.

WR5: ограничения не могут быть ассоциированы с **categorization_class**.

WR6: categorization_class не может быть классом определения свойства для какого-либо свойства.

WR7: свойство подкласса не может быть ассоциировано с **categorization_class**.

WR8: постоянное значение класса не может быть ассоциировано с **categorization_class**.

WR9: видимое свойство не может быть ассоциировано с **categorization_class**.

WR10: видимый тип данных не может быть ассоциирован с **categorization_class**.

5.9 Тип элемента данных/данные о свойствах

5.9.1 Общие положения

Данный пункт содержит определения словарных данных для свойств.

5.9.2 BSU свойства (Property_BSU)

Сущность **property_BSU** дает идентификацию свойства.

Пример представления на языке EXPRESS:

```
*)
ENTITY property_BSU
SUBTYPE OF(basic_semantic_unit);
     SELF\basic_semantic_unit.code: property_code_type;
     name_scope: class_BSU;
```

```
DERIVE
      absolute_id: identifier :=
           name_scope.defined_by.absolute_id
           + sep_id + dic_identifier;
INVERSE
      describes_classes: SET OF class FOR described_by;
UNIQUE
      UR1: absolute_id;
WHERE
      WR1: QUERY(c <* describes_classes |
           NOT(is_subclass(c, name_scope.definition[1])))= [];
END_ENTITY; -- property_BSU
(*
```

Определения атрибутов:

code: данный код необходим для уникальной идентификации свойства во всех онтологиях, определенных рассматриваемым поставщиком с атрибутом **name_scope.defined_by**.

name_scope: ссылка на класс, на котором или ниже которого рассматриваемый элемент свойства доступен для ссылки атрибутом **described_by**.

absolute_id: уникальная идентификация данного свойства.

describes_classes: классы, объявляющие данное свойство доступным для использования в описании продукта.

Пояснения к тексту программы:

WR1: любой класс, на который производится ссылка атрибутом **describes_class** BSU свойства **property_BSU,** либо это класс, на который производится ссылка его атрибутом **name_scope**, либо это подкласс данного класса.

UR1: идентификатор свойства **absolute_id** является уникальным.

Примечание — Атрибут **name_scope** сущности **property_BSU** используется для кодирования атрибута «Класс определений» для свойств.

5.9.3 Типы элементов данных свойств (Property_DET)

Сущность **property_DET** дает словарные описания свойств.

Пример представления на языке EXPRESS:

```
*)
ENTITY property_DET
ABSTRACT SUPERTYPE OF(ONEOF(
      condition_DET, dependent_P_DET, non_dependent_P_DET))
SUBTYPE OF(class_and_property_elements);
      SELF\dictionary_element.identified_by: property_BSU;
      preferred_symbol: OPTIONAL mathematical_string;
      synonymous_symbols: SET [0:?] OF mathematical_string;
      figure: OPTIONAL graphics;
      det_classification: OPTIONAL DET_classification_type;
      domain: data_type;
      formula: OPTIONAL mathematical_string;
DERIVE
      describes_classes: SET [0:?] OF class
           := identified_by.describes_classes;
END_ENTITY; -- property_DET
(*
```

Определения атрибутов:

identified_by: BSU свойства **property_BSU**, идентифицирующая данное свойство.

preferred_symbol: более короткое описание данного свойства.

synonymous_symbols: синоним для более короткого описания свойств.

figure: вспомогательная графика для описания свойства.
det_classification: класс ISO 80000/IEC 80000 (ранее ISO 31) для данного свойства.
domain: ссылка на тип данных **data_type**, ассоциированный со свойством.
formula: математическое выражение, объясняющее свойство.
describes_classes: классы, объявляющие данное свойство доступным для использования в описании продукта.

Примечание 1 — Атрибут **preferred_symbol** используется для кодирования атрибута «Предпочтительный буквенный символ» для свойств.

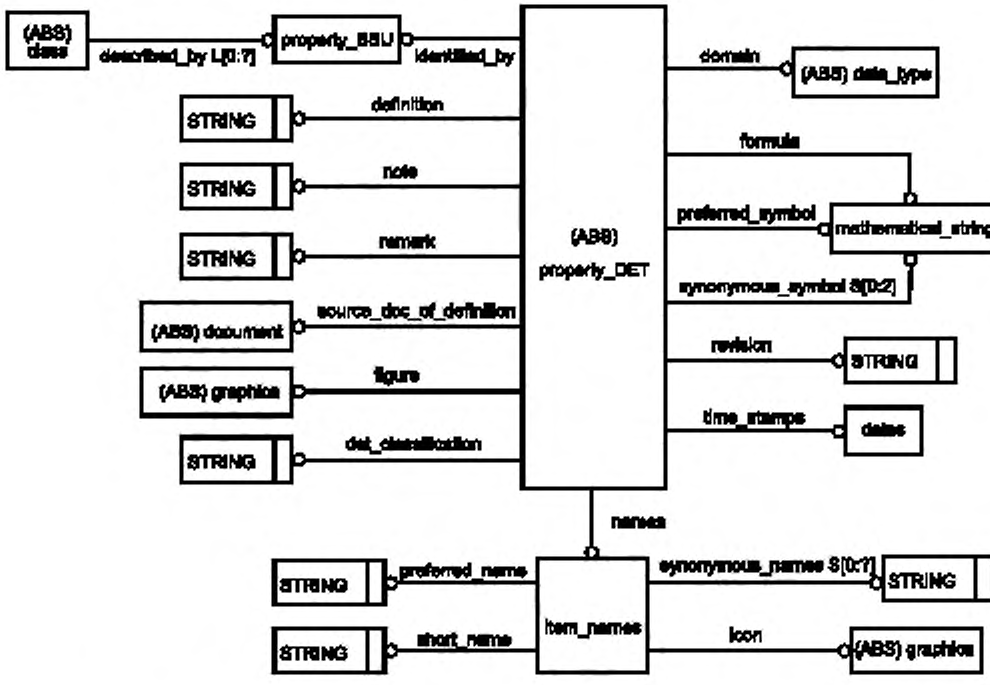
Примечание 2 — Атрибут **synonymous_symbols** используется для кодирования атрибутов «Синонимичных буквенных символов» для свойств.

Примечание 3 — Атрибут **det_classification** используется для кодирования атрибута «Классификация типов свойств».

Примечание 4 — Атрибут области **domain** используется как отправная точка для кодирования атрибута свойства «Тип данных». Сущность **data_type** может быть описана как подтип для различных возможных типов данных.

Примечание 5 — Атрибут **formula** используется для кодирования атрибута «Формула» для свойств.

На рисунке 9 приведена модель планирования данных, ассоциированных с сущностями **property_DETs**.



(ABS) class	Класс
described_by L[0:?]	Описан ...
property_BSU	BSU свойства
identified_by	Идентифицирован ...

Рисунок 9, лист 1 — Краткое описание свойств типов элементов данных и соотношений

domain	Область
(ABS) data_type	Тип данных
string	Строка
definition	Определение
formula	Формула
note	Примечание
remark	Заметка
(ABS) property_DET	Тип элемента данных свойства
preferred_symbol	Предпочтительный символ
mathematical_string	Математическая строка
(ABS) document	Документ
source_doc_of_definition	Исходный документ определения
synonymous_symbol S[0.2]	Синонимичный символ
revision	Пересмотр
(ABS) graphics	Графика
figure	Рисунок
time_stamps	Отметки времени
dates	Даты
det_classification	Классификация типов элементов данных
preferred_name	Предпочтительное имя
names	Имена
synonymous_names S[0:?]	Синонимичные имена
short_name	Краткое имя
item_names	Имена элементов
icon	Иконка

Рисунок 9, лист 2

5.9.4 Условные, зависимые и независимые типы элементов данных

На рисунке 10 даны различные виды типов элементов данных в формате модели планирования.

Отметим, что рисунок 10 упрощен. Вообще говоря, соотношение **«depends_on»** должно применяться со ссылкой на BSU. Но установлено ограничение, по которому ссылочный тип элемента данных свойства **property_DET** должен быть типом элемента данных условия **condition_DET** (см. сущность **dependent_P_DET**).

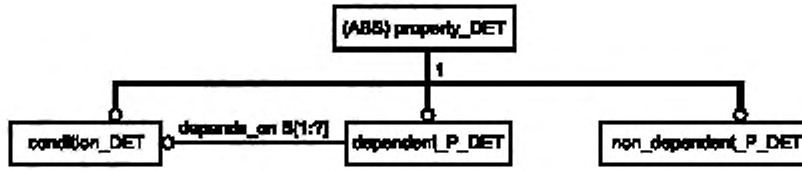


Рисунок 10, лист 1 — Виды типов элементов данных

(ABS) property_DET	Тип элемента данных свойства
condition_DET	Тип элемента данных условия
depends_on S[1:?]	Зависит от ...
dependent_P_DET	Зависимый тип элемента данных свойства
non_dependent_P_DET	Независимый тип элемента данных свойства

Рисунок 10, лист 2

5.9.5 Особенности структуры

5.9.5.1 Тип элемента данных условия (Condition_DET)

Сущность **condition_DET** — это свойство, от которого могут зависеть другие свойства.

Пример представления на языке EXPRESS:

```
*)
ENTITY condition_DET
SUBTYPE OF(property_DET);
END_ENTITY; -- condition_DET
(*
```

5.9.5.2 Зависимый тип элемента данных свойства (Dependent_P_DET)

Сущность **dependent_P_DET** — это свойство, значение которого явно зависит от значений некоторых условий.

Пример — Сопротивление термистора зависит от наружной температуры. Сопротивление термистора должно быть представлено как dependent_P_DET, а наружная температура термистора — это condition_DET.

Пример представления на языке EXPRESS:

```
*)
ENTITY dependent_P_DET
SUBTYPE OF(property_DET);
      depends_on: SET [1:?] OF property_BSU;
WHERE
      WR1: QUERY(p <* depends_on | NOT(definition_available_implies(
           p, ('ISO13584_IEC61360_DICTIONARY_SCHEMA.CONDITION_DET'
           IN TYPEOF(p.definition[1])))))) = [];
END_ENTITY; -- dependent_P_DET
(*
```

Определения атрибутов:

depends_on: набор базовых семантических единиц, идентифицирующих свойства, от которых зависит данное свойство.

Пояснение к тексту программы:

WR1: только условные типы элементов данных **condition_DET** могут использоваться в наборе **depends_on**.

5.9.5.3 Независимые типы элементов данных свойств (Non_dependent_P_DET)

Сущность **non_dependent_P_DET** — это свойства, не зависящие явно от некоторых условий.

Пример представления на языке EXPRESS:

```
*)
ENTITY non_dependent_P_DET
SUBTYPE OF(property_DET);
END_ENTITY; -- non_dependent_P_DET
(*
```

Примечание 1 — Три указанных подтипа (**condition_DET, dependent_P_DET** и **non_dependent_P_DET)** сущности **property_DET** используются для кодирования различных видов свойств (см. раздел 7). Сущность **condition_DET** используется для контекстных параметров. Сущность **dependent_P_DET** используется для контекстно-зависимых характеристик. Сущность **non_dependent_P_DET** используется для характеристики продукта.

Примечание 2 — Атрибут **depends_on** сущности **dependent_P_DET** используется для кодирования атрибута «Условие» для свойств.

5.9.6 Задание значений класса (Class_value_assignment)

Свойства, имеющие значение класса, — это свойства, значение которых не может быть назначено индивидуально для реализации класса. Оно может быть назначено только для всех реализаций, принадлежащих классу. Такие свойства объявляются путем их включения в список свойств подкласса **sub_class_properties** сущности **item_class**. Затем такому свойству может быть задано значение, верное для всех реализаций любого класса элементов **item_class**, являющегося подклассом для класса, где объявлено рассматриваемое свойство, имеющее значение класса, или в самом этом классе. Значение свойства, имеющего значение класса, задается для класса элементов **item_class** с помощью атрибута **class_value_assignment**, на который производится ссылка атрибутом **class_constant_value** данного класса.

Примечание — Свойства, имеющие значение класса, могут относиться к любому типу данных.

Пример представления на языке EXPRESS:

```
*)
ENTITY class_value_assignment;
      super_class_defined_property: property_BSU;
      assigned_value: primitive_value;
WHERE
      WR1: definition_available_implies(super_class_defined_property,
           compatible_data_type_and_value(super_class_defined_property.
           definition[1]\property_DET.domain, assigned_value));
END_ENTITY; -- class_value_assignment
(*
```

Определения атрибутов:

super_class_defined_property: ссылка на свойство (определенное в классе или в любом его суперклассе как принадлежащее набору **sub_class_properties**), которому задается значение **assigned_value**.

assigned_value: значение, заданное свойству, корректному для целого класса, ссылающегося на рассматриваемую реализацию **class_value_assignment** набора **class_constant_value**, и для всех его подклассов.

Пояснение к тексту программы:

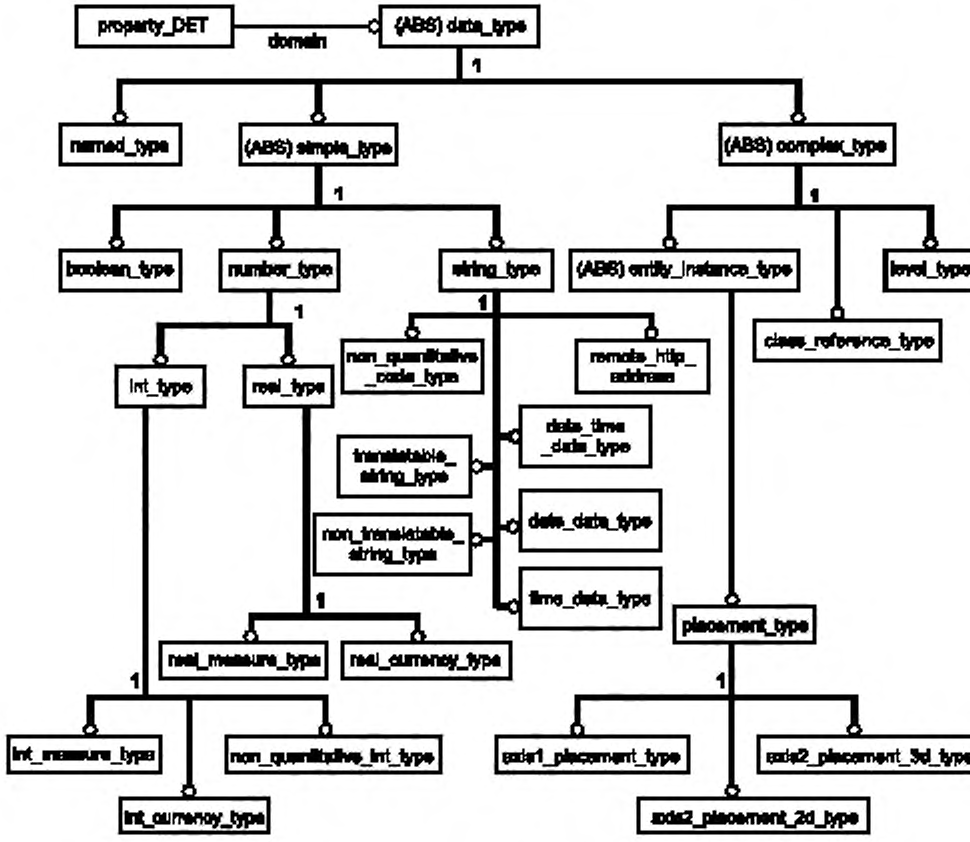
WR1: тип значения, заданного свойству, определенному в суперклассе **super_class_defined_properties**, должен быть совместим с областью значений свойств **super_class_defined_properties.**

5.10 Область данных: система типов

5.10.1 Общие положения

Данный пункт содержит определения представлений типов данных **property_DET**. На рисунке 11 в виде модели планирования показана иерархия сущностей для типов данных.

В отличие от других словарных элементов (поставщики, классы, свойства) идентификация с понятием базовой семантической единицы не является обязательной для типа данных **data_type**, так как она в большинстве случаев прикреплена прямо к типу данных **property_DET** и, таким образом, не требует идентификации. При этом сущности **data_type_BSU** и **data_type_element** могут использоваться для уникальной идентификации там, где это целесообразно. Это обеспечивает повторное использование рассматриваемого определения типа в другом определении **property_DET** даже вне текущего физического файла.



property_DET	Тип элемента данных свойства
domain	Область
(ABS) data_type	Тип данных
named_type	Поименованный тип
(ABS) simple_type	Простой тип
(ABS) complex_type	Комплексный тип
boolean_type	Булевский тип
number_type	Числовой тип
string_type	Строчный тип
(ABS) entity_instance_type	Тип реализации сущности
level_type	Тип уровня
class_reference_type	Тип ссылки на класс
int_type	Целый тип
real_type	Действительный тип
non_quantitative_code_type	Неколичественный кодовый тип

Рисунок 11, лист 1 — Иерархия сущностей для системы типов

remote_http_address	Удаленный http-адрес
translatable_string_type	Тип переводимой строки
date_time_data_type	Тип данных о дате и времени
non_translatable_string_type	Непереводимый строчный тип
date_data_type	Тип данных о дате
time_data_type	Тип данных о времени
placement_type	Тип размещения
real_measure_type	Тип действительной меры
real_currency_type	Тип действительной валюты
int_measure_type	Тип целой меры
non_quantitative_int_type	Неколичественный целый тип
axis1_placement_type	Тип размещения по оси 1
axis2_placement_3d_type	3d-тип размещения по оси 2
int_currency_type	Тип целой валюты
axis2_placement_2d_type	2d-тип размещения по оси 2

Рисунок 11, лист 2

5.10.2 Особенности структуры

5.10.2.1 BSU типа данных (Data_type_BSU)

Сущность **data_type_BSU** позволяет идентифицировать элемент типа данных **data_type_element**.

Пример представления на языке EXPRESS:

```
*)
ENTITY data_type_BSU
SUBTYPE OF(basic_semantic_unit);
     SELF\basic_semantic_unit.code: data_type_code_type;
     name_scope: class_BSU;
DERIVE
     absolute_id: identifier :=
          name_scope.defined_by.absolute_id      (* Supplier*)
          + sep_id + dic_identifier;         (* Data_type *)
INVERSE
     defining_class: SET OF class FOR defined_types;
UNIQUE
     absolute_id;
WHERE
     WR1: is_subclass(defining_class[1], name_scope.definition[1]);
END_ENTITY; -- data_type_BSU
(*
```

Определения атрибутов:

code: дает уникальную идентификацию типа данных для всех онтологий, определенных рассматриваемым поставщиком **name_scope.defined_by**.

name_scope: ссылка на класс, на уровне которого (или ниже которого) рассматриваемый элемент типа данных доступен для ссылки атрибутом **defined_type**.

absolute_id: уникальная идентификация данного свойства.

defining_class: класс, объявляющий данный тип данных **data_type** доступным для использования в описании продукта.

Пояснение к тексту программы:

WR1: класс, используемый в атрибуте **name_scope**, является суперклассом для класса, где данный тип **data_type** определен.

Примечание — Атрибут **name_scope** используется для кодирования ссылки на класс, которому принадлежит соответственный тип данных. Данный атрибут, кроме сущности **data_type_element** (см. ниже), является частью кода атрибута «Видимый тип» класса.

5.10.2.2 Элемент типа данных (Data_type_element)

Сущность **data_type_element** описывает типы словарных элементов. Отметим, что нет необходимости в каждом отдельном случае для заданного типа данных **data_type** иметь BSU и словарный элемент **dictionary_element**, потому что атрибут **property_DET** может ссылаться на тип данных **data_type** прямо. Использование BSU соотношений необходимо, если поставщик ссылается на один и тот же тип в различных физических файлах.

Пример представления на языке EXPRESS:

```
*)
ENTITY data_type_element
SUBTYPE OF(dictionary_element);
      SELF\dictionary_element.identified_by: data_type_BSU;
      names: item_names;
      type_definition: data_type;
END_ENTITY; -- data_type_element
(*
```

Определения атрибутов:

identified_by: BSU, задающий описанный элемент **data_type_element**.

names: имена, дающие описания определенного элемента **data_type_element**.

type_definition: описание типа элемента **data_type_element**.

Примечание — Повторно объявленный атрибут **identified_by** используется для кодирования ссылки на BSU, к которому относится данный элемент типа данных **data_type_element**. Данный элемент, кроме сущности **data_type_BSU** (см. выше), используется для кодирования атрибута класса «Видимые типы».

5.10.3 Система типов

5.10.3.1 Тип данных (Data_type)

Сущность **data_type** служит общим супертипом для сущностей, используемых для указания типа ассоциированного типа элемента данных **DET**.

Пример представления на языке EXPRESS:

```
*)
ENTITY data_type
ABSTRACT SUPERTYPE OF(ONEOF(
      simple_type,
      complex_type,
      named_type));
      constraints: SET [0:?] OF domain_constraint;
WHERE
      WR1: QUERY (cons <* constraints
           |NOT correct_constraint_type(cons, SELF)) = [];
END_ENTITY; -- data_type
(*
```

Определения атрибутов:

constraints: набор ограничений, ограничивающих область значений типа данных.

Примечание — Каждое ограничение области в атрибуте **constraints** должно быть выполнено. Таким образом, атрибут **constraints** есть сопряжение ограничений.

Пояснение к тексту программы:

WR1: набор ограничений области должен определять ограничения, совместимые с областью значений типа данных.

5.10.3.2 Простой тип (Simple_type)

Сущность **simple_type** служит общим супертипом для сущностей, используемых для указания простого ассоциированного типа элемента данных **DET**.

Пример представления на языке EXPRESS:

```
*)
ENTITY simple_type
ABSTRACT SUPERTYPE OF(ONEOF(
      number_type,
      boolean_type,
      string_type))
SUBTYPE OF(data_type);
      value_format: OPTIONAL value_format_type;
END_ENTITY; -- simple_type
(*
```

Определения атрибутов:

value_format: вспомогательное кодирование формата значений свойств.

Примечание 1 — Атрибут **value_format** сущности **simple_type** используется для кодирования атрибута «Формат значения» свойства.

Примечание 2 — Если какое-либо ограничение **string_pattern_constraint** применяется к значению простого типа, то оно имеет преимущество перед форматом значения **value_format.**

5.10.3.3 Числовой тип (Number_type)

Сущность **number_type** задает значения элементов **DET** числового типа.

Пример представления на языке EXPRESS:

```
*)
ENTITY number_type
ABSTRACT SUPERTYPE OF(ONEOF(
      int_type,
      real_type,
      rational_type))
SUBTYPE OF(simple_type);
END_ENTITY; -- number_type
(*
```

5.10.3.4 Целый тип (Int_type)

Сущность **int_type** задает значения элементов **DET** типа **INTEGER**.

Пример представления на языке EXPRESS:

```
*)
ENTITY int_type
SUPERTYPE OF(ONEOF(
      int_measure_type,
      int_currency_type,
      non_quantitative_int_type))
SUBTYPE OF(number_type);
END_ENTITY; -- int_type
(*
```

5.10.3.5 Тип целой меры (Int_measure_type)

Сущность **int_measure_type** задает значения элементов **DET**, являющихся мерами целого типа. Она указывает единицу измерения или идентификатор единицы измерения **(unit_id)**, выражающей значения, обмениваемые как отдельные целые. Она может также устанавливать альтернативные единицы измерения или идентификаторы альтернативных единиц измерения, допустимые для использования, когда каждое значение явно ассоциировано со своей единицей измерения.

Примечание 1 — Использование либо единицы **unit**, либо идентификатора единицы **unit_id** обязательно. В случае, когда присутствуют оба, **unit** имеет преимущество.

Примечание 2 — Если присутствуют и атрибуты **alternative_unit**, и атрибуты **alternative_unit_ids**, и они имеют одинаковый размер, то атрибут **alternative_unit** имеет преимущество.

Примечание 3 — Идентификатор словарной единицы **dic_unit_identifier**, используемый и в атрибуте **unit_id,** и в атрибуте **alternative_unit_ids**, является идентификатором единицы измерения, который может быть разрешен в атрибуте **dic_unit** из сервера ИСО/ТС 29002-20.

Примечание 4 — Каждая словарная единица измерения **dic_unit**, определенная в атрибуте **alternative_unit**, и каждая словарная единица измерения **dic_unit**, идентифицированная в атрибуте **alternative_unit_ids**, должна быть ассоциирована со строчным представлением **string_representation**. Его текстовое представление **text_representation** может быть использовано для характеристики альтернативной единицы измерения, используемой на уровне реализации.

Пример представления на языке EXPRESS:

```
*)
ENTITY int_measure_type
SUBTYPE OF(int_type);
      unit: OPTIONAL dic_unit;
      alternative_units: OPTIONAL LIST [1:?] OF dic_unit;
      unit_id: OPTIONAL dic_unit_identifier;
      alternative_unit_ids: OPTIONAL LIST [1:?] OF dic_unit_identifier;
WHERE
      WR1: EXISTS(unit) OR EXISTS(unit_id);
      WR2: NOT EXISTS(alternative_units) OR
           NOT EXISTS(alternative_unit_ids) OR
           (SIZEOF(alternative_units) = SIZEOF(alternative_unit_ids));
      WR3: NOT EXISTS(alternative_units)
           OR (QUERY (un <* SELF.alternative_units
           |NOT EXISTS (un.string_representation))
           = []);
END_ENTITY; -- int_measure_type
(*
```

Определения атрибутов:

unit: единица измерения по умолчанию, ассоциированная со значением целочисленной меры **int_measure_type**.

alternative_unit: список прочих единиц измерения, которые могут быть использованы для выражения значений целочисленной меры **int_measure_type**.

Примечание 5 — Списочный порядок гарантирует, что атрибуты **alternative_unit** и **alternative_unit_id**, если оба существуют, определяют ту же единицу в том же порядке.

unit_id: идентификатор единицы измерения по умолчанию, ассоциированный с описанной мерой.

Примечание 6 — Атрибут **unit** и атрибут **unit_id** используются для кодирования атрибута «Единица измерения» свойства. Если имеются оба атрибута, то **unit** имеет преимущество.

Примечание 7 — Если значение свойства с областью **int_measure_type** обменивается как отдельное целое, то данное значение выражается с помощью атрибутов **unit** или **unit_id**.

alternative_unit_ids: список идентификаторов прочих единиц измерения, используемых для выражения значения в целочисленной мере **int_measure_type**.

Примечание 8 — Если значение свойства с областью **int_measure_type** оценивается либо атрибутом **unit**, либо атрибутом **alternative_unit**, либо идентифицируется атрибутом **alternative_unit_id**, то это значение не может быть представлено как отдельное целое. Оно должно быть представлено парой «значение, единица измерения».

Пояснения к тексту программы:

WR1: должен существовать хотя бы один из двух атрибутов **unit** и **unit_id**.

WR2: если существуют и атрибут **alternative_unit**, и атрибут **alternative_unit_id**, то они должны иметь одинаковую длину.

WR3: каждая словарная единица измерения **dic_unit**, представленная в альтернативных единицах измерения **alternative_unit**, должна иметь строчное представление **string_representation.**

Дополнительные пояснения:

IP1: идентификатор **dic_unit_identifier**, используемый и в атрибуте **unit_id**, и в атрибуте **alternative_unit_ids**, должен быть разрешен в **dic_unit** из существующего сервера ISO/TS 29002-20.

IP2: если имеются и атрибут **unit**, и атрибут **unit_id**, то они должны определять одну и ту же единицу измерения.

IP3: если имеются и атрибут **alternative_unit**, и атрибут **alternative_unit_ids**, то они должны определять тот же список единиц измерения в том же порядке.

IP4: если атрибут **alternative_unit_ids** существует, то все единицы, которые данный атрибут идентифицирует, должны быть разрешены в словарную единицу **dic_unit**, имеющую строчное представление **string_representation**.

5.10.3.6 Целый валютный тип (Int_currency_type)

Сущность **int_currency_type** задает значение элементам **DET**, которые являются целыми валютами.

Пример представления на языке EXPRESS:

```
*)
ENTITY int_currency_type
SUBTYPE OF(int_type);
      currency: OPTIONAL currency_code;
END_ENTITY; -- int_currency_type
(*
```

Определения атрибутов:

currency: ассоциированный код валюты, описанный в соответствии с ИСО 4217. Если он отсутствует, то валютный код нужно обменивать вместе с соответствующими данными (значениями).

5.10.3.7 Неколичественный целый тип (Non_quantitative_int_type)

Сущность **non_quantitative_int_type** — это тип нумерации, где элементы нумерации представлены целым значением (см. также сущность неколичественный кодовый тип **non_quantitative_code_type** и рисунок 12).

Пример представления на языке EXPRESS:

```
*)
ENTITY non_quantitative_int_type
SUBTYPE OF(int_type);
      domain: value_domain;
WHERE
      WR1: QUERY(v <* domain.its_values |
           'ISO13584_IEC61360_DICTIONARY_SCHEMA.VALUE_CODE_TYPE' IN
           TYPEOF(v.value_code)) = [];
END_ENTITY; -- non_quantitative_int_type
(*
```

Определения атрибутов:

domain: множество перенумерованных значений, описанных в сущности **value_domain**.

Пояснение к тексту программы:

WR1: значения, ассоциированные со списком **domain.its_value**, не должны содержать тип **value_code_type**.

5.10.3.8 Действительный тип (Real_type)

Сущность **real_type** задает значения элементов **DET** типа **REAL**.

Пример представления на языке EXPRESS:

```
*)
ENTITY real_type
SUPERTYPE OF(ONEOF(
      real_measure_type,
      real_currency_type))
SUBTYPE OF(number_type);
END_ENTITY; -- real_type
(*
```

5.10.3.9 Тип действительной меры (Real_measure_type)

Сущность **real_measure_type** задает значения элементам **DET** с мерой типа **REAL**. Она указывает единицы измерения или идентификаторы единиц измерения, в которых выражаются значения, обмениваемые как отдельные действительные величины. Данная сущность может также устанавливать альтернативные единицы измерения или идентификаторы альтернативных единиц измерения, допустимые для использования, когда каждое значение явно ассоциировано со своей единицей измерения.

Примечание 1 — Наличие либо атрибута **unit**, либо атрибута **unit_id** обязательно. Если они имеются оба, то атрибут **unit** имеет преимущество.

Примечание 2 — Если имеются и атрибут **alternative_unit**, и атрибут **alternative_unit_ids**, то они имеют одинаковый размер, а атрибут **alternative_unit** имеет преимущество.

Примечание 3 — Идентификаторы **dic_unit_identifier**, используемые и в атрибуте **unit_id**, и в атрибуте **alternative_unit_id**, являются идентификаторами единиц измерения, которые могут быть разрешены в словарной единице **dic_unit** из сервера ИСО/ТС 29002-20.

Примечание 4 — Каждая словарная единица **dic_unit**, определенная в атрибуте **alternative_unit**, и каждая словарная единица **dic_unit**, идентифицированная в атрибуте **alternative_unit_ids**, должна быть ассоциирована со строчным представлением **string_representation**, чье текстовое представление **text_representation** может быть использовано для характеристики альтернативной единицы измерения, используемой на уровне реализации.

Пример представления на языке EXPRESS:

```
*)
ENTITY real_measure_type
SUBTYPE OF(real_type);
      unit: OPTIONAL dic_unit;
      alternative_units: OPTIONAL LIST [1:?] OF dic_unit;
      unit_id : OPTIONAL dic_unit_identifier;
      alternative_unit_ids: OPTIONAL LIST [1:?] OF dic_unit_identifier;
WHERE
      WR1: EXISTS(unit) OR EXISTS(unit_id);
      WR2: NOT EXISTS(alternative_units)
           OR NOT EXISTS(alternative_unit_ids)
           OR (SIZEOF(alternative_units) =
           SIZEOF(alternative_unit_ids));
      WR3: NOT EXISTS(alternative_units)
           OR (QUERY (un <* SELF.alternative_units
           |NOT EXISTS (un.string_representation))
           = []);
END_ENTITY; -- real_measure_type
(*
```

Определения атрибутов:

unit: единица измерения по умолчанию, ассоциированная со значением **real_measure_type**.

alternative_unit: список прочих единиц измерения, которые могут использоваться для выражения значений **real_measure_type**.

Примечание 5 — Используемый списочный порядок гарантирует, что и атрибут **alternative_unit,** и атрибут **alternative_unit_ids,** если они существуют, определяют ту же единицу в том же порядке.

unit_id: идентификатор единицы измерения по умолчанию, ассоциированный с описанной мерой.

Примечание 6 — И атрибут **unit**, и атрибут **unit_id** используются для кодирования атрибута «Единица измерения». Если они имеются оба, то атрибут **unit** имеет преимущество.

Примечание 7 — Если значение свойства с областью **real_measure_type** обменивается как отдельная действительная величина, то данное значение выражается в единицах меры **unit** или **unit_id**.

alternative_unit_ids: список идентификаторов прочих единиц измерения, используемых для выражения значений **real_measure_type**.

Примечание 8 — Если значение свойства с областью **real_measure_type** оценивается либо с помощью атрибута **unit**, либо с помощью атрибута **alternative_unit,** либо идентифицируется атрибутом **alternative_unit_ids**, то его значение не может быть представлено как отдельное действительное значение. Оно может быть представлено парой «значение, единица измерения».

Пояснения к тексту программы:

WR1: должен существовать хотя бы один из двух атрибутов **unit** или **unit_id**.

WR2: если и атрибут **alternative_unit,** и атрибут **alternative_unit_ids** существуют, то они должны иметь одинаковую длину.

WR3: каждая словарная единица **dic_unit** с атрибутом **alternative_unit** должна иметь строчное представление **string_representation.**

Дополнительные пояснения:

IP1: идентификаторы **dic_unit_identifier**, используемые и в идентификаторе единицы измерения **unit_id**, и в альтернативных идентификаторах **alternative_unit_id**, должны быть разрешимы в словарных единицах измерения **dic_unit** из существующего сервера ИСО/ТС 29002-20.

IP2: если существуют и атрибут **unit**, и атрибут **unit_id**, то они должны определять одинаковые единицы измерения.

IP3: если существуют и атрибуты **alternative_unit**, и атрибуты **alternative_unit_ids**, то они должны определять тот же список единиц измерения в том же порядке.

IP4: если существует атрибут **alternative_unit_ids**, то все единицы измерения, которые данный атрибут задает, должны разрешаться в словарной единице измерения **dic_unit**, имеющей строчное представление **string_representation**.

5.10.3.10 Действительный валютный тип (Real_currency_type)

Сущность **real_currency_type** определяет действительные валюты.

Пример представления на языке EXPRESS:

```
*)
ENTITY real_currency_type
SUBTYPE OF(real_type);
      currency: OPTIONAL currency_code;
END_ENTITY; -- real_currency_type
(*
```

Определения атрибутов:

currency: ассоциированный код валюты, описанный в соответствии с ИСО 4217. Если данный код отсутствует, то валютный код обменивается вместе с соответствующими данными (значениями).

5.10.3.11 Рациональный тип (Rational_type)

Сущность **rational_type** задает значения элементов **DET** с типом **rational**.

Примечание — В ИСО 13584-32 рациональные значения представлены тремя целыми элементами расширяемого языка разметки XML: целая часть, числитель и знаменатель. В ИСО 13584-24:2003 рациональные значения представлены массивом из трех целых: целая часть, числитель и знаменатель.

Пример представления на языке EXPRESS:

```
*)
ENTITY rational_type
SUPERTYPE OF(
     rational_measure_type)
SUBTYPE OF(number_type);
END_ENTITY; -- rational_type
(*
```

5.10.3.12 Тип рациональной меры (Rational_measure_type)

Сущность **rational_measure_type** задает значения элементов **DET** с типом меры **RATIONAL**.

Пример — Диаметр винта 4 1/8 дюйма.

Сущность **rational_measure_type** указывает единицу измерения или идентификатор единицы измерения, выражающей обмениваемые значения как рациональные. Данная сущность может также устанавливать альтернативные единицы измерения, а также идентификаторы альтернативной единицы измерения, допустимые для использования, когда каждое значение явно ассоциировано со своей единицей измерения.

Примечание 1 — Присутствие либо атрибута **unit,** либо **unit_id** обязательно. Если присутствуют оба атрибута, то атрибут **unit** имеет преимущество.

Примечание 2 — Если присутствуют и атрибут **alternative_unit,** и атрибут **alternative_unit_ids**, то они имеют одинаковый размер, и атрибут **alternative_unit** имеет преимущество.

Примечание 3 — Идентификаторы словарной единицы измерения **dic_unit_identifier,** используемые и в атрибуте **unit_id,** и в атрибуте **alternative_unit_ids,** — это идентификаторы единицы измерения, разрешаемые в **dic_unit** из сервера ИСО/ТС 29002-20.

Примечание 4 — Каждая единица измерения **dic_unit**, определенная в атрибуте **alternative_unit**, и каждая единица измерения **dic_unit,** идентифицированная в атрибуте **alternative_unit_ids,** ассоциирована со строчным представлением **string_representation.** Ее текстовое представление **text_representation** может быть использовано для характеристики альтернативной единицы измерения, используемой на уровне реализации.

Пример представления на языке EXPRESS:

```
*)
ENTITY rational_measure_type
SUBTYPE OF(rational_type);
     unit: OPTIONAL dic_unit;
     alternative_units: OPTIONAL LIST [1:?] OF dic_unit;
     unit_id: OPTIONAL dic_unit_identifier;
     alternative_unit_ids: OPTIONAL LIST [1:?] OF dic_unit_identifier;
WHERE
     WR1: EXISTS(unit) OR EXISTS(unit_id);
     WR2: NOT EXISTS(alternative_units)
          OR NOT EXISTS(alternative_unit_ids)
          OR (SIZEOF(alternative_units) =
          SIZEOF(alternative_unit_ids));
     WR3: NOT EXISTS(alternative_units) OR (QUERY (un <*
          SELF.alternative_units |
          NOT EXISTS (un.string_representation)) = []);
END_ENTITY; -- rational_measure_type
(*
```

Определения атрибутов:

unit: единица измерения по умолчанию, ассоциированная со значением типа рациональной меры **rational_measure_type**.

alternative_unit: список прочих единиц измерения, которые могут быть использованы для выражения значений типа рациональной меры **rational_measure_type**.

Примечание 5 — Данный списочный порядок гарантирует, что если существуют и атрибут **alternative_unit,** и атрибут **alternative_unit_ids,** то определяются те же единицы измерения и в том же порядке.

unit_id: идентификатор единиц измерения по умолчанию, ассоциированный с описанной мерой.

Примечание 6 — И атрибут **unit**, и атрибут **unit_id** используются для кодирования атрибута «Единица измерения» свойств. Если существуют оба, то атрибут **unit** имеет преимущество.

Примечание 7 — Если значение свойства с областью **rational_measure_type** обменивается как отдельное рациональное число, то данное значение выражается в единицах измерения **unit** или **unit_id**.

alternative_unit_ids: список идентификаторов прочих единиц измерения, используемых для выражения значений типа рациональной меры **rational_measure_type.**

Примечание 8 — Если значение свойства с областью **rational_measure_type** оценивается в единицах измерения либо определенных атрибутом **alternative_unit,** либо идентифицированных атрибутом **alternative_unit_ids**, то это значение не может быть представлено как отдельное рациональное число. Оно должно быть представлено парой «значение, единица измерения».

Пояснения к тексту программы:

WR1: должен существовать хотя бы один из атрибутов **unit** или **unit_id**.

WR2: если существуют и атрибут **alternative_unit**, и атрибут **alternative_unit_ids**, то они должны иметь одинаковую длину.

WR3: каждая словарная единица измерения **dic_unit** в атрибуте **alternative_unit** должна иметь строчное представление **string_representation**.

Дополнительные пояснения:

IP1: идентификатор **dic_unit_identifier**, используемый и в атрибуте **unit_id**, и в атрибуте **alternative_unit_ids**, должен разрешаться в **dic_unit** из существующего сервера ИСО/ТС 29002-20.

IP2: если существуют и атрибут **unit**, и атрибут **unit_id**, то они должны определять одинаковые единицы измерения.

IP3: если существуют и атрибут **alternative_unit**, и атрибут **alternative_unit_ids**, то они должны определять тот же список единиц измерения в том же порядке.

IP4: если существует атрибут **alternative_unit_ids**, то все единицы измерения, которые идентифицирует данный атрибут, должны разрешаться в **dic_unit**, имеющей строчное представление **string_representation**.

5.10.3.13 Булевский тип (boolean_type)

Сущность **boolean_type** задает значения элементов **DET** булевского типа.

Пример представления на языке EXPRESS:

```
*)
ENTITY boolean_type
SUBTYPE OF(simple_type);
END_ENTITY; -- boolean_type
(*
```

5.10.3.14 Строчный тип (String_type)

Сущность **string_type** задает значения элементов **DET** строчного типа.

Пример представления на языке EXPRESS:

```
*)
ENTITY string_type
SUPERTYPE OF ( ONEOF (
      translatable_string_type,
      non_translatable_string_type,
      URI_type,
      non_quantitative_code_type,
      date_time_data_type,
      date_data_type,
```

```
      time_data_type))
SUBTYPE OF(simple_type);
END_ENTITY; -- string_type
 (*
```

5.10.3.15 Переводимый строчный тип (Translatable_string_type)

Сущность **translatable_string_type** задает значения элементов **DET** строчного типа, которые могут быть представлены различными строками на различных языках.

Примечание 1 — Значения таких свойств не могут быть использованы для идентификации продуктов.

Примечание 2 — Значения таких свойств могут быть либо простыми строчными значениями **string_value,** если глобальное назначение **global_language_assignment** определяет текущий язык, либо переведенными строчными значениями **translated_string_value,** где каждое значение строки ассоциировано с языком.

Примечание 3 — Два значения одного свойства с типом данных **data_type,** равным **translatable_string_type,** можно сравнивать на элемент равенства, только если такое соответствующее свойство, как исходный язык **source_language,** определено как часть его административных данных **administrative_data,** и если указанные значения доступны в данном исходном языке **source_language.** При этом не предполагается, что на языке, отличном от исходного языка **source_language,** тот же смысл представлен той же строкой.

Пример представления на языке EXPRESS:

```
*)
ENTITY translatable_string_type
SUBTYPE OF(string_type);
END_ENTITY; -- translatable_string_type
(*
```

5.10.3.16 Непереводимый строчный тип (Non_translatable_string_type)

Сущность **non_translatable_string_type** задает значения элементов **DET** строчного типа, представленных неизменным способом на любом языке.

Примечание — Значения таких свойств могут использоваться для идентификации продуктов.

Пример представления на языке EXPRESS:

```
*)
ENTITY non_translatable_string_type
SUBTYPE OF(string_type);
END_ENTITY; -- non_translatable_string_type
(*
```

5.10.3.17 Тип универсального идентификатора ресурсов (URI_type)

Сущность **URI_type** задает значения элементов **DET** строчного типа, содержащих универсальный идентификатор ресурсов **URI**.

Примечание — Сущность **URI_type** прежде всего идентифицирует универсальный идентификатор ресурсов **URL**.

Пример представления на языке EXPRESS:

```
*)
ENTITY URI_type
SUBTYPE OF(string_type);
END_ENTITY; -- URI_type
(*
```

5.10.3.18 Date_time_data_type

Сущность **date_time_data_type** задает значения элементов **DET** строчного типа. Она содержит особый момент времени, указанный в соответствии с конкретным представлением по ИСО 8601.

Примечание 1 — Только подмножество лексических представлений, допускаемое ИСО 8601, можно использовать для задания значений **date_time_data_type.** См. пояснение IP1 ниже.

Примечание 2 — Вышеуказанные ограничения ИСО 8601 на рассматриваемое представление соответствуют требованиям схемы XML.

Пример представления на языке EXPRESS:

```
*)
ENTITY date_time_data_type
SUBTYPE OF(string_type);
END_ENTITY; -- date_time_data_type
(*
```

Дополнительное пояснение:

IP1: значение свойства с типом данных **date_time_data_type** должно соответствовать нижеследующему лексическому представлению, являющемуся подмножеством лексического представления, определенного ИСО 8601. Данное лексическое представление — это определенный ИСО 8601 расширенный формат CCYY-MM-DDThh:mm:ss, где «CC» представляет век (первому веку соответствует «00»), «YY» — год, «MM» — месяц и «DD» — день. Вспомогательный знак «–» (минус) спереди указывает отрицательное число. Если минус опущен, то предполагается «+» (плюс). Буква «T» — это разделитель даты и времени. Обозначения «hh», «mm», «ss» — это час, минута и секунда соответственно. Дополнительные цифры могут использоваться для указания долей секунды при необходимости. При этом используется формат ss.ss... с любым количеством цифр после десятичной точки. Дробная часть секунды указывается по особому требованию. Прочие части рассматриваемой лексической формы присутствуют всегда. Чтобы указать значение года, превышающее 9999, нужны дополнительные цифры слева. Спереди ставят нули, если значение года требует менее четырех цифр. В противном случае нули не используются. Например, год 0000 запрещен. Поле CCYY должно иметь по крайней мере четыре цифры. Поля MM, DD, SS, hh, mm и ss представляются двумя цифрами каждое (не считая долей секунды). Предшествующие нули используются, если в рассматриваемом поле не хватает значащих цифр. За указанным представлением может сразу идти символ «Z», указывающий координированное универсальное время (UTC). Для указания временного пояса, т. е. сдвига между местным временем и координированным универсальным временем, используется знак «+» или «–». Далее указывается временной сдвиг (по отношению к UTC) в формате hh:mm (примечание — указание минут обязательно). В ИСО 8601 даны требования к записи значений в различных полях. Если указывается временной пояс, то наличие и часов, и минут обязательно.

Пример — Чтобы указать время 1:20 после полудня 31 мая 1999 года по восточному календарю, отстающее на 5 часов от координированного универсального времени (UTC), нужно записать: 1999-05-31T13:20:00-05:00.

5.10.3.19 Тип данных о дате (Date_data_type)

Сущность **date_data_type** задает значения элементов **DET** строчного типа. Она содержит особую календарную дату, установленную в соответствии с конкретным представлением по ИСО 8601.

Примечание 1 — Только подмножество лексических представлений, соответствующих ИСО 8601, может быть использовано как значение сущности **date_data_type.** См. пояснение IP1 ниже.

Примечание 2 — Вышеуказанное ограничение ИСО 8601 на представления совпадает с ограничением, установленным схемой XML.

Пример представления на языке EXPRESS:

```
*)
ENTITY date_data_type
SUBTYPE OF(string_type);
END_ENTITY; -- date_data_type
(*
```

Дополнительное пояснение:

IP1: значение свойства с типом данных **date_data_type** должно соответствовать нижеследующему лексическому представлению, являющемуся подмножеством лексических представлений, установлен-

ных ИСО 8601. Лексическое представление для типа данных **date_data_type** является сокращенным (усеченным справа) лексическим представлением для типа данных **date_time_data_type**: CCYY-MM-DF. Усечение слева недопустимо. Для типа данных **date_time_data_type** может быть использован вспомогательный идентификатор часового пояса. Если значение года выходит из интервала 0001-9999, то в левой части данного представления добавляются цифры и знак «–».

Пример — Представление 1999-05-31 — это представление типа date_data_type для 31 мая 1999 года.

5.10.3.20 Тип времени суток (Time_data_type)

Сущность **time_data_type** задает значения элементов **DET** строчного типа. Она содержит указание особого момента времени в соответствии с требованиями ИСО 8601. Это значение **time_data_type** дает момент времени, повторяющийся каждый день.

Примечание 1 — Только подмножество лексических представлений, определенных ИСО 8601, может использоваться для значения **time_data_type.** См. пояснение IP1 ниже.

Примечание 2 — Вышеуказанное ограничение ИСО 8601 совпадает с ограничением, установленным схемой XML.

Примечание 3 — Рассматриваемое лексическое представление допускает использование вспомогательного идентификатора часового пояса. Поэтому значения **time_data_type** частично упорядочены, чтобы исключить возможность произвольно задавать порядок следования двух значений, одно из которых имеет часовой пояс, а другое — нет.

Пример представления на языке EXPRESS:

```
*)
ENTITY time_data_type
SUBTYPE OF(string_type);
END_ENTITY; -- time_data_type
(*
```

Дополнительное пояснение:

IP1: значение свойства с типом данных **time_data_type** должно соответствовать нижеследующему лексическому представлению, являющемуся подмножеством лексических представлений, определенных ИСО 8601. Лексическое представление типа данных **time_data_type** — это усеченное слева лексическое представление типа **date_time_data_type** hh:mm:ss.sss со вспомогательным (по выбору) идентификатором часового пояса.

Пример — Представление 13:20:00-05:00 — это представление суточного времени time_data_type, соответствующее 1.20 после полудня для восточного стандартного времени, которое отстает на 5 часов от координированного универсального времени (UTC).

5.10.3.21 Неколичественный кодовый тип (Non_quantitative_code_type)

Сущность **non_quantitative_code_type** — это тип нумерации, когда элементы нумерации представлены значением строки (см. также сущность неколичественный целый тип **non_quantitative_int_type** и рисунок 12).

Пример представления на языке EXPRESS:

```
*)
ENTITY non_quantitative_code_type
SUBTYPE OF(string_type);
      domain: value_domain;
WHERE
      WR1: QUERY(v <* domain.its_values

            NOT('ISO13584_IEC61360_DICTIONARY_SCHEMA.VALUE_CODE_TYPE' IN
            TYPEOF(v.value_code))) = []; END_
ENTITY; -- non_quantitative_code_type
(*
```

Определения атрибутов:

domain: набор перенумерованных значений, описанных сущностью **value_domain**.

Пояснение к тексту программы:

WR1: значения, ассоциированные со списком **domain.its_value,** должны содержать только элементы типа **value_code_type**.

5.10.3.22 Комплексный тип (Complex_type)

Сущность **complex_type** дает определения типов, значения которых представлены как реализации языка EXPRESS.

Пример представления на языке EXPRESS:

```
*)
ENTITY complex_type
ABSTRACT SUPERTYPE OF(ONEOF(
      level_type,
      class_reference_type,
      entity_instance_type))
SUBTYPE OF(data_type);
END_ENTITY; -- complex_type
(*
```

5.10.3.23 Тип уровня (Level_type)

Тип данных **level_type** — это комплексный тип, указывающий, что значение свойства содержит от одного до четырех действительных или целых значений. Каждое из них описывается конкретным индикатором, указывающим смысл данного значения.

Примечание — Значения реализаций **level_type** содержат значения только индикаторов, установленных атрибутом **levels**. Если некоторые из указанных значений недоступны, то они представляются нулями **null_value**.

Пример — Если атрибут level_type указывает, что только минимальные и типовые значения могут быть целыми, то рассматриваемая реализация содержит целые значения (или нулевые значения null_value) только для минимальных и типовых значений реализаций level_type.

Пример представления на языке EXPRESS:

```
*)
ENTITY level_type
SUBTYPE OF(complex_type);
      levels: LIST [1:4] OF UNIQUE level;
      value_type: simple_type;
WHERE
      WR1: ('ISO13584_IEC61360_DICTIONARY_SCHEMA.INT_MEASURE_TYPE'
           IN TYPEOF(value_type))
           OR ('ISO13584_IEC61360_DICTIONARY_SCHEMA.REAL_MEASURE_TYPE'
           IN TYPEOF(value_type));
      WR2: NOT EXISTS(SELF.levels[2]) OR

            (SELF.levels[1] < SELF.levels[2]);
      WR3: NOT EXISTS(SELF.levels[2]) OR NOT EXISTS(SELF.levels[3]) OR
           (SELF.levels[2] < SELF.levels[3]);
      WR4: NOT EXISTS(SELF.levels[3]) OR NOT EXISTS(SELF.levels[4]) OR
           (SELF.levels[3] < SELF.levels[4]);
END_ENTITY; -- level_type
(*
```

Определения атрибутов:

levels: список уникальных индикаторов, указывающих, какие из заданных значений должны быть ассоциированы со свойством.

value_type: тип данных указанных значений атрибута **level_type**.

Пояснения к тексту программы:

WR1: тип собственного значения **SELF.value_type** — это либо тип целой меры **int_measure_type,** либо тип действительной меры **real_measure_type**.

WR2: порядок первого и второго уровней **level**, если оба существуют, должен соответствовать порядку нумерации типа **level**.

WR3: порядок второго и третьего уровней **level**, если оба существуют, должен соответствовать порядку нумерации типа **level**.

WR4: порядок третьего и четвертого уровней **level**, если оба существуют, должен соответствовать порядку нумерации типа **level**.

5.10.3.24 Уровень (level)

Сущность **level** определяет индикаторы, используемые для задания значений атрибута **level_type.**

Используемые индикаторы:

- **minimum:** наименьшее значение рассматриваемой величины, установленное для заданного набора рабочих условий, в которых компонент, устройство или оборудование являются работоспособными и функционируют в соответствии с установленными требованиями;
- **nominal:** значение величины, используемое для обозначения и идентификации компонента, устройства, оборудования или системы;
- **typical:** обычно встречающееся значение величины, используемое для задания целей, установленных для указанного набора условий эксплуатации компонента, устройства, оборудования или системы;
- **maximum:** наибольшее значение величины, установленное для заданного набора условий эксплуатации, в которых компонент, устройство или оборудование являются работоспособными и функционируют в соответствии с установленными требованиями.

Примечание 1 — Номинальное значение обычно округляют.

Пример — Автомобильный аккумулятор с номинальным напряжением 12 В имеет 6 элементов с типовым значением напряжения 2,2 В каждый. Типовое напряжение этой батареи равно 13,5 В. Напряжение может достигать максимального значения 14,5 В. Аккумулятор считается полностью разряженным, если напряжение падает ниже минимума 12,5 В.

Примечание 2 — Значения, установленные для определенного уровня свойства, определены в словаре.

Примечание 3 — Рекомендуется ограничивать тип уровня теми типами элементов данных **DET,** для которых множественность значений одной и той же характеристики является востребованным и общепринятым делом (например, для электронных компонентов промышленных установок).

Пример представления на языке EXPRESS:

```
*)
TYPE level = ENUMERATION OF(
     min,    (* the minimal value of the physical quantity *)
     nom,    (* the nominal value of the physical quantity *)
     typ,    (* the typical value of the physical quantity *)
     max);   (* the maximal value of the physical quantity *)
END_TYPE; -- level
(*
```

5.10.3.25 Тип ссылки на класс (Class_reference_type)

Сущность **class_reference_type** задает значения элементов **DET**, являющиеся реализациями класса. Она используется, как правило, для описания сборок или материалов (деталей), из которых состоит компонент.

Пример представления на языке EXPRESS:

```
*)
ENTITY class_reference_type
SUBTYPE OF(complex_type);
     domain: class_BSU; END_ENTITY;
-- class_reference_type
(*
```

Определения атрибутов:

domain: BSU класса **class_BSU**, ссылающаяся на класс, представляющий описанный тип.

5.10.3.26 Тип реализации сущности (Entity_instance_type)

Сущность **entity_instance_type** задает значения элементов **DET**, представляемых реализациями некоторых типов данных сущностей языка EXPRESS. Атрибут имени типа **type_names** позволяет выбрать допустимый тип данных. Указанные типы данных — это строки, образующие набор. Данный атрибут, вместе с функцией типа TYPEOF языка EXPRESS, примененный к некоторому значению, позволяет выполнить строгую проверку типа и выявить полиморфизм. Данная сущность описывается как подтип для некоторых типов данных, допустимых для использования в словарных схемах.

Примечание — Если некоторая сущность языка EXPRESS — это значение некоторого элемента **DET**, тип данных которого — это тип реализации сущности **entity_instance_type**, то возможно скорректировать тип, применив функцию типа TYPEOF языка EXPRESS для данного значения **DET** и сравнив результаты данного действия со строками, содержащимися в атрибуте набора имен типов **type_names**.

Пример представления на языке EXPRESS:

```
*)
ENTITY entity_instance_type
SUBTYPE OF(complex_type);
      type_name: SET OF STRING;
END_ENTITY; -- entity_instance_type
(*
```

Определения атрибутов:

type_names: набор строк, дающих описание (в формате функции типа TYPEOF языка EXPRESS) имен типов данных сущностей языка EXPRESS, принадлежащих результату применения функции типа TYPEOF языка EXPRESS, примененной к значению, ссылающемуся на рассматриваемую сущность, как на ее тип данных.

5.10.3.27 Тип размещения (Placement_type)

Сущность **placement_type** используется для значений элементов **DET**, являющихся реализациями типа данных сущности **placement**.

Примечание 1 — Сущности размещения **placement** импортируются из ИСО 10303-42. В соответствии с ИСО 10303-42 реализация **placement** может существовать, только если она относится к реализации контекста геометрического представления **geometric_representation_context** (в некоторой реализации представления). Следовательно, если некоторые свойства класса имеют реализации сущности **placement** как их значения, то данный класс содержит контекст геометрического представления **geometric_representation_context** (который определяет контекст указанных размещений) и само представление (которое соединяет указанные размещения **placements** с их контекстом). Ни **geometric_representation_context**, ни **representation** не импортируются в настоящий стандарт. Сущности **placement** не могут использоваться, если используются только схемы ИСО 13584-42. Указанные сущности вводятся как ресурсы для других частей ИСО 13584.

Примечание 2 — Сущности **Placement**, в частности, используются в ИСО 13584-32 (OntoML) и в ИСО 13584-25.

Пример представления на языке EXPRESS:

```
*)
ENTITY placement_type
SUPERTYPE OF(ONEOF(
      axis1_placement_type,
      axis2_placement_2d_type,
      axis2_placement_3d_type))
SUBTYPE OF(entity_instance_type);
WHERE
      WR1: 'GEOMETRY_SCHEMA.PLACEMENT'
            IN SELF\entity_instance_type.type_name;
END_ENTITY; -- placement_type
(*
```

Пояснение к тексту программы:

WR1: строка **'GEOMETRY_schema.PLACEMENT'** должна содержаться в наборе, определенном атрибутом **SELF\entity_instance_type.type_names**.

5.10.3.28 Тип размещения на оси 1 (Axis1_placement_type)

Сущность **axis1_placement_type** задает значения элементов **DET**, являющихся реализациями типа данных сущности **axis1_placement** (см. ИСО 10303-42).

Пример представления на языке EXPRESS:

```
*)
ENTITY axis1_placement_type
SUBTYPE OF(placement_type);
WHERE
     WR1: 'GEOMETRY_SCHEMA.AXIS1_PLACEMENT' IN
          SELF\entity_instance_type.type_name;
END_ENTITY; -- axis1_placement_type
(*
```

Пояснение к тексту программы:

WR1: строка **'GEOMETRY_schema.AXIS1_PLACEMENT'** должна содержаться в наборе, определенном атрибутом **SELF\entity_instance_type.type_names**.

5.10.3.29 Тип 2d-размещения на оси 2 (Axis2_placement_2d_type)

Сущность **axis2_placement_2d_type** задает значения элементов **DET**, являющихся реализациями типа данных сущности **axis2_placement_2d** (см. ИСО 10303-42).

Пример представления на языке EXPRESS:

```
*)
ENTITY axis2_placement_2d_type
SUBTYPE OF(placement_type); WHERE
     WR1: 'GEOMETRY_SCHEMA.AXIS2_PLACEMENT_2D'
          IN SELF\entity_instance_type.type_name;
END_ENTITY; -- axis2_placement_2d_type
(*
```

Пояснение к тексту программы:

WR1: строка **'GEOMETRY_schema.AXIS2_PLACEMENT_2D'** должна содержаться в наборе, определенном для атрибута **SELF\entity_instance_type.type_names**.

5.10.3.30 Тип 3d-размещения на оси 2 (Axis2_placement_3d_type)

Сущность **axis2_placement_3d_type** задает значения элементов **DET**, являющихся реализациями типа данных сущности **axis2_placement_3d** (см. ИСО 10303-42).

Пример представления на языке EXPRESS:

```
*)
ENTITY axis2_placement_3d_type
SUBTYPE OF(placement_type); WHERE
     WR1: 'GEOMETRY_SCHEMA.AXIS2_PLACEMENT_3D'
          IN SELF\entity_instance_type.type_name;
END_ENTITY; -- axis2_placement_3d_type
(*
```

Пояснение к тексту программы:

WR1: строка **'GEOMETRY_schema.AXIS2_PLACEMENT_3D'** должна содержаться в наборе, определенном для атрибута **SELF\entity_instance_type.type_names**.

5.10.3.31 Поименованный тип (Named_type)

Сущность **named_type** позволяет ссылаться на другие типы с помощью BSU механизма.

Пример представления на языке EXPRESS:

```
*)
ENTITY named_type
SUBTYPE OF(data_type );
     referred_type: data_type_BSU;
END_ENTITY; -- named_type
(*
```

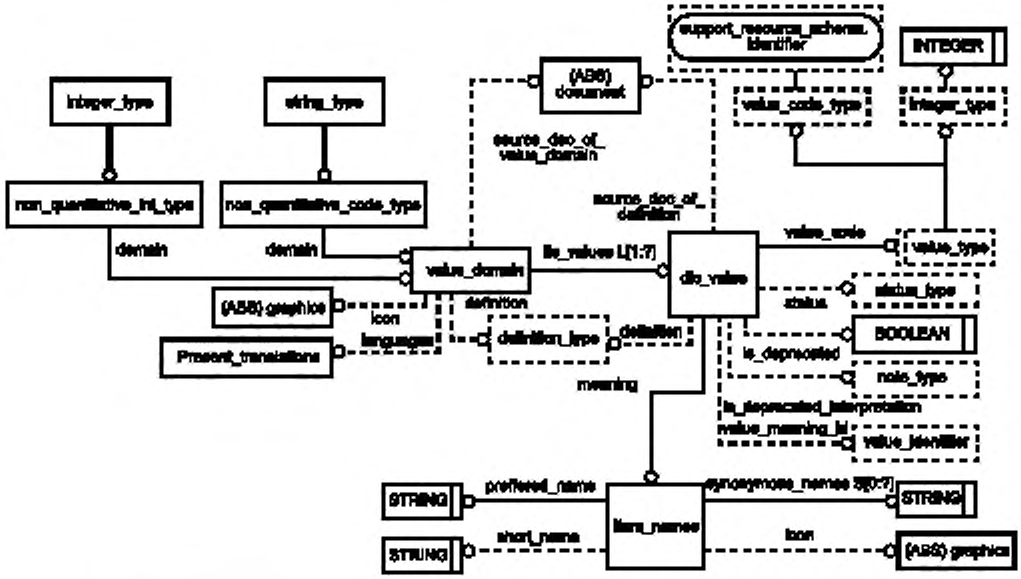
Определения атрибутов:

referred_type: BSU, идентифицирующая тип данных **data_type**, на который ссылается рассматриваемая сущность.

5.10.4 Значения

Данный пункт содержит определения неколичественных типов элементов данных (см. сущности **non_quantitative_int_type** и **non_quantitative_code_type**).

На рисунке 12 представлены (как модель планирования) основные данные, ассоциированные с неколичественными типами элементов данных.



support_resource_schema.identifier	Идентификатор схемы ресурса поддержки
integer	Целое
integer_type	Целый тип
string_type	Строчный тип
(ABS) document	Документ
source_doc_of_value_domain	Исходный документ для области значений
value_code_type	Тип кода значения
integer_type	Целый тип
non_quantitative_int_type	Неколичественный целый тип

Рисунок 12, лист 1 — Краткое описание неколичественных типов элементов данных

non_quantitative_code_type	Неколичественный кодовый тип
source_doc_of_definition	Исходный документ определения
domain	Область
value_domain	Область значений
its_values L[1:?]	Его значения
dic_value	Словарное значение
value_code	Код значения
value_type	Тип значения
(ABS) graphics	Графика
icon	Иконка
definition	Определение
status	Статус
status_type	Тип статуса
present_translations	Настоящие переводы
languages	Языки
definition_type	Тип определения
is_deprecated	Больше не используется
boolean	Булевская величина
meaning	Смысл
is_deprecated_interpretation	Интерпретация величин, не рекомендуемых для использования
note_type	Тип примечания
value_meaning_id	Идентификатор значения величины
value_identifier	Идентификатор значения
string	Строка
preffered_name	Предпочтительное имя
short_name	Краткое имя
item_names	Названия элементов
synonymous_names S[0:?]	Синонимичные имена

Рисунок 12, лист 2

5.10.5 Особенности структуры

5.10.5.1 Область значений (Value_domain)

Сущность **value_domain** описывает набор допустимых значений неколичественного типа элемента данных.

Пример представления на языке EXPRESS:

```
*)
ENTITY value_domain;
      its_values: LIST [1:?] OF dic_value;
      source_doc_of_value_domain: OPTIONAL document;
      languages: OPTIONAL present_translations;
      terms: LIST [0:?] OF item_names;
```

```
        definition: OPTIONAL definition_type;
        icon: OPTIONAL graphics;
    WHERE
        WR1: NOT EXISTS(languages) OR (QUERY(v <* its_values |
            languages :<>: v.meaning.languages) = []);
        WR2: codes_are_unique(its_values);
        WR3: EXISTS(languages) OR (QUERY(v <* its_values |
            EXISTS(v.meaning.languages)) = []);
        WR4: EXISTS(languages) OR (QUERY(v <* its_values |
            EXISTS(v.definition.languages)) = []);
    END_ENTITY; -- value_domain
    (*
```

Определения атрибутов:

its_value: список нумерации значений, содержащихся в описанной области.

source_doc_of_value_domain: документ, описывающий область, ассоциированную с описанной сущностью **value_domain**.

languages: вспомогательные языки, на которые выполнены переводы.

terms: список названий элементов **item_names**, дающий для IEC 61360 связь со словарем терминов.

definition: вспомогательный текст, описывающий область значений **value_domain**.

icon: вспомогательная иконка **icon**, дающая графическое описание, ассоциированное с областью значений **value_domain**.

Пояснения к тексту программы:

WR1: если смысл значения представлен более чем на одном языке, то набор используемых языков должен быть одним и тем же для всего набора значений.

WR2: код значения должен быть уникальным внутри указанного типа данных.

WR3: если языки не определены, то смыслу значения язык не назначается.

WR4: если языки не определены, то определению значения язык не назначается.

5.10.5.2 Тип значения (Value_type)

Каждое значение неколичественного элемента данных ассоциируется с кодом, характеризующим это значение. Тип значения **value_type** может быть либо целым **integer_type**, либо кодовым **value_code_type**.

Пример представления на языке EXPRESS:

```
    *)
    TYPE integer_type = INTEGER;
    END_TYPE; -- integer_type

    TYPE value_type = SELECT(value_code_type, integer_type);
    END_TYPE; -- value_type
    (*
```

5.10.5.3 Словарное значение (Dic_value)

Сущность **dic_value** — это одно из значений сущности **value_domain**.

Пример представления на языке EXPRESS:

```
    *)
    ENTITY dic_value;
        value_code: value_type;
        meaning: item_names;
        source_doc_of_definition: OPTIONAL document;
        definition: OPTIONAL definition_type;
        status: OPTIONAL status_type;
        is_deprecated: OPTIONAL BOOLEAN;
        is_deprecated_interpretation: OPTIONAL note_type;
        value_meaning_id: OPTIONAL dic_value_identifier;
```

```
WHERE
     WR1: NOT EXISTS (SELF. is_deprecated)
          OR EXISTS (SELF. is_deprecated_interpretation);
END_ENTITY; -- dic_value
(*
```

Определения атрибутов:

value_code: код, ассоциированный с описанным значением. Это может быть либо кодовый тип **value_code_type**, либо целый тип **integer_type**.

meaning: смысл, ассоциированный с данным значением. Он определяется названием.

source_doc_of_definition: вспомогательный исходный документ, определяющий рассматриваемое значение.

definition: вспомогательный текст, описывающий словарное значение

dic_value. status: тип статуса **status_type**, определяющий состояние жизненного цикла словарного значения **dic_value**.

Примечание 1 — Допустимые значения **status_type** не стандартизованы. Они определены для каждого отдельного словаря его поставщиком информации.

Пример 1 — Набор допустимых значений статуса элементов, представленный для стандартизации в агентство технической поддержки стандартов ИСО, определен директивами ИСО.

Пример 2 — Набор допустимых значений статуса элементов в стандартной базе данных МЭК определен директивами МЭК.

Примечание 2 — Если словарные значения **dic_value** еще не подготовлены и не внесены, то может оказаться полезным использование варианта **dic_value**.

Пример 3 — Для эксперимента перед утверждением.

Примечание 3 — Если атрибут **status** для словарного значения **dic_value** отсутствует и если данное словарное значение **dic_value** не отменено в соответствии с возможными атрибутом **is_deprecated**, то это рассматриваемое **dic_value** имеет тот же статус стандартизации, что и целый словарь. В частности, если словарь стандартизован, то данное **dic_value** является частью текущего издания стандарта.

is_deprecated: булевская переменная, указывающая (если имеет значение *true*), что рассматриваемое словарное значение **dic_value** использовать уже нельзя.

Примечание 4 — Если атрибут **is_deprecated** значения не имеет, то **dic_value** еще не отменено.

Примечание 5 — Больше не используемые словарные значения **dic_value** сохраняются в области значений **value_domain** по соображениям обеспечения совместимости.

is_deprecated_interpretation: дает обоснование отказа от использования, устанавливает порядок интерпретации значений реализаций больше не используемых элементов.

value_meaning_id: идентификатор словарного значения **dic_value_identifier,** являющийся глобальным идентификатором **dic_value** независимо от области значений **value_domain,** в которую он включен.

Примечание 6 — Данный идентификатор допускает повторное использование рассматриваемого определения словарного значения **dic_value** в различных областях.

Пояснение к тексту программы:

WR1: если атрибут **is_deprecated** существует, то должен существовать и атрибут **is_deprecated_interpretation**.

Дополнительное пояснение к тексту программы:

IP1: значения реализаций элементов **is_deprecated_interpretation** должны быть определены в момент, когда принимается решение об отказе от использования.

5.10.5.4 Административные данные (Administrative_data)

Сущность **administrative_data** содержит информацию о жизненном цикле словарного элемента.

Пример представления на языке EXPRESS:

```
*)
ENTITY administrative_data;
     status: OPTIONAL status_type;
```

```
        translation: LIST[0:?] of translation_data;
        source_language: language_code;
    INVERSE
        administrated_element: dictionary_element FOR administration;
    WHERE
        WR1: one_language_per_translation(SELF);
        WR2: SIZEOF(QUERY (trans <* SELF.translation |
             trans.language = source_language)) = 0;
    END_ENTITY; -- administrative_data
    (*
```

Определения атрибутов:

status: тип статуса **status_type**, определяющий состояние жизненного цикла словарного элемента.

Примечание 1 — Допустимые значения типа статуса **status_type** не стандартизованы. Они определены для каждого отдельного словаря его поставщиком информации.

Пример 1 — Набор допустимых значений статуса элементов, предложенный для стандартизации агентством технической поддержки стандарта ИСО, определен директивами ИСО.

Пример 2 — Набор допустимых значений статуса элементов в базе данных стандарта МЭК определен директивами МЭК.

Примечание 2 — Для словарных элементов **dictionary_element**, еще не готовых для практического использования, могут оказаться полезными их предварительные варианты.

Пример 3 — Для экспериментальных целей перед утверждением.

Примечание 3 — Если атрибут **status** отсутствует и если данный словарный элемент **dictionary_element** больше не используется (см. атрибут **is_deprecated**), то **dictionary_element** имеет тот же статус стандартизации, как и целый словарь. В частности, если словарь стандартизован, то данный **dictionary_element** является частью текущего издания стандарта.

translation: описание надежных трансляторов с различных языков.

source_language: язык, на котором изначально определен словарный элемент **dictionary_element**, на который нужно ссылаться при невязках трансляции.

Примечание 4 — Словарь может содержать словарные элементы **dictionary_element** с различными исходными языками **source_language**, например, потому что они импортированы из различных словарей. Поставщик словарных данных должен гарантировать, что информация об указанных элементах доступна на рассматриваемом языке или на других языках.

administrated_element: словарный элемент **dictionary_element**, данные жизненного цикла которого имеются в административных данных **administrative_data**.

Пояснения к тексту программы:

WR1: языки перевода, ассоциированные с административными данными **administrative_data**, являются уникальными.

WR2: исходный язык **source_language** не присутствует на языке трансляции, ассоциированном с административными данными **administrative_data**.

5.10.5.5 Данные перевода (Translation_data)

Сущность **translation_data** записывает информацию о возможных переводах словарного элемента.

Пример представления на языке EXPRESS:

```
*)
ENTITY translation_data;
    language: language_code;
    responsible_translator: supplier_BSU;
    translation_revision: revision_type;
    date_of_current_translation_revision: OPTIONAL date_type;
INVERSE
    belongs_to: administrative_data FOR translation;
END_ENTITY; -- translation_data
(*
```

Определения атрибутов:

language: язык, на который переведен словарный элемент.

Примечание 1 — В случае невязки между исходным определением словарного элемента и некоторыми его переводами фактический смысл словарного элемента дается на исходном языке перевода.

responsible_translator: организация, ответственная за трансляцию на язык элемента.

translation_revision: № пересмотра соответствующего перевода.

Примечание 2 — Изменение № версии или изменение № пересмотра словарного элемента не всегда требует каких-либо изменений его перевода. Если изменений перевода, обусловленных изменением № версии или изменением № пересмотра словарного элемента, нет, то соответствующий № пересмотра перевода **translation_revision** не изменяется. Однако любые изменения перевода подразумевают изменение соответствующего **translation_revision.**

date_of_current_translation: дата последнего пересмотра перевода.

belongs_to: административные данные **administrative_data**, ссылающиеся на запись данных перевода **translation_data**.

5.10.6 Расширения определений единиц измерения по ИСО 10303-41

5.10.6.1 Общие положения

Данный подраздел определяет ресурсы описания единиц измерения в словаре. Здесь даны расширения ресурсов, определенных в ИСО 10303-41.

5.10.6.2 Единица, не относящаяся к системе СИ (Non_si_unit)

Сущность **non_si_unit** расширяет модель единицы измерения ИСО 10303-41. Она устанавливает представления единиц, не относящихся к системе СИ, которые не являются ни контекстно-зависимыми, ни производными (см. ИСО 10303-41).

Пример представления на языке EXPRESS:

```
*)
ENTITY non_si_unit
SUBTYPE OF(named_unit);
     names: label; END_
ENTITY; -- non_si_unit
(*
```

Определения атрибутов:

names: метки **label,** используемые для именования описанных единиц измерения.

5.10.6.3 Правило подтверждения выбора (Assert_ONEOF rule)

Правило **assert_ONEOF** подтверждает, что выбор производится из следующих подтипов поименованных единиц измерения **named_unit**: единицы системы СИ **si_unit**, контекстно-зависимые единицы **context_dependent_unit,** производные единицы **conversion_based_unit** и единицы, не относящиеся к системе СИ **non_si_unit**.

Пример представления на языке EXPRESS:

```
*)
RULE assert_ONEOF FOR(named_unit);
WHERE
     QUERY(u <* named_unit :
          ('ISO13584_IEC61360_DICTIONARY_SCHEMA.NON_SI_UNIT'
          IN TYPEOF(u)) AND
          ('MEASURE_SCHEMA.SI_UNIT' IN TYPEOF(u))
          OR ('ISO13584_IEC61360_DICTIONARY_SCHEMA.NON_SI_UNIT'
          IN TYPEOF(u)) AND
          ('MEASURE_SCHEMA.CONTEXT_DEPENDENT_UNIT' IN TYPEOF(u))
          OR ('ISO13584_IEC61360_DICTIONARY_SCHEMA.NON_SI_UNIT'
          IN TYPEOF(u)) AND
          ('MEASURE_SCHEMA.CONVERSION_BASED_UNIT' IN TYPEOF(u))
          ) = [];
END_RULE; -- assert_ONEOF
(*
```

5.10.6.4 Словарная единица измерения (Dic_unit)

Базовое представление единиц измерения производится в структурированной форме в соответствии с ИСО 10303-41. Но так как одной из целей хранения единиц измерения в словаре является удобство пользователя, только одного структурированного представления недостаточно. Оно должно быть дополнено строчным представлением.

Используемые определения имеют варианты:

- можно использовать функцию строки для единицы измерения **string_for_unit** (см. 5.12). Для рассматриваемого структурированного представления единицы измерения она возвращает строчное представление, соответствующее представлению, используемому в приложении В МЭК 61360-1: 2009;
- строчное представление может быть дано в обычной текстовой форме (например, сущность математическая строка **mathematical_string**, атрибут текстового представления **text_representation**);
- представление MathML приводит расширенное представление единицы измерения, включая верхние индексы, нижние индексы и т. д. (например, сущность математическая строка **mathematical_string**, атрибут представления **MathML_representation**).

Сущность **dic_unit** описывает единицу измерения, включаемую в словарь.

Пример представления на языке EXPRESS:

```
*)
ENTITY dic_unit;
      structured_representation: unit;
      string_representation: OPTIONAL mathematical_string;
END_ENTITY; -- dic_unit
(*
```

Определения атрибутов:

structured_representation: структурированное представление по ИСО 10303-41, включая расширение, определенное в 5.10.6.

string_representation: функцию **string_for_unit** можно использовать для расчета строчного представления из **structured_representation**, когда строчное представление **string_representation** отсутствует.

Примечание — Атрибут структурированного представления **structured_representation** сущности **dic_unit** используется для кодирования атрибута свойства «Единица измерения».

5.11 Определения базового типа и сущности

5.11.1 Определение базового типа

Данный подраздел содержит определения базового типа и сущности, используемые в основной части модели. Нижеследующий раздел содержит определения базового типа и сущности, расположенные по английскому алфавиту.

5.11.2 Особенности структуры

5.11.2.1 Тип кода класса (Class_code_type)

Сущность **class_code_type** задает допустимые значения кода класса.

Пример представления на языке EXPRESS:

```
*)
TYPE class_code_type = code_type;
WHERE
      WR1: LENGTH(SELF) <= class_code_len;
END_TYPE; -- class_code_type
(*
```

Пояснения к тексту программы:

WR1: длина значений, соответствующих **class_code_type**, должна быть меньше или равна длине кода класса **class_code_len** (т. е. 35).

5.11.2.2 Тип кода (code_type)

Сущность **code_type** задает допустимые значения типа кода, используемого для идентификации.

Примечание — Если код также предназначен для обмена в соответствии с ИСО/ТС 29002-5, то рекомендуется выполнить требования, определенные данным стандартом. Для задания кода можно использовать только «безопасные символы». Безопасные символы включают: буквы верхнего регистра, цифры, двоеточия, десятичную точку, подчеркивание. В некоторых случаях допускается использование символа «–» (минус).

Пример представления на языке EXPRESS:

```
*)
TYPE code_type = identifier;
WHERE
     WR1: NOT (SELF LIKE '*#*');
     WR2: NOT (SELF LIKE '* *');
     WR3: NOT (SELF = '');
END_TYPE; -- code_type
(*
```

Пояснения к тексту программы:

WR1: символ «#» не должен содержаться в значении **code_type**. Символ «#» используется для последовательного соединения идентификаторов (см.: CONSTANT **sep_id**) или кода и версии (см.: CONSTANT **sep_cv**).

WR2: пробелы не допускаются (во избежание проблем с предшествующими и последующими пропусками при последовательном соединении кодов).

WR3: значение **code_type** не должно быть пустой строкой.

5.11.2.3 Тип валюты (Currency_code)

5.11.2.3.1 Общие положения

Сущность **currency_code** задает значения, допустимые для кода валюты. Указанные значения соответствуют ИСО 4217.

Пример — «CHF» — код швейцарского франка, «CNY» — код китайского юаня, «JPY» — код японской йены, «SUR» — код рубля СССР, «USD» — код доллара США, «EUR» — код евро.

Пример представления на языке EXPRESS:

```
*)
TYPE currency_code = identifier;
WHERE
     WR1: LENGTH(SELF) = 3;
END_TYPE; -- currency_code
(*
```

Пояснение к тексту программы:

WR1: длина кода валюты **currency_code** равна 3.

5.11.2.3.2 Тип кода типа данных (Data_type_code_type)

Сущность **data_type_code_type** задает значения, допустимые для кода типа данных.

Пример представления на языке EXPRESS:

```
*)
TYPE data_type_code_type = code_type;
WHERE
     WR1: LENGTH(SELF) = data_type_code_len;
END_TYPE; -- data_type_code_type
(*
```

Пояснение к тексту программы:

WR1: длина типа кода типа данных **data_type_code_type** должна быть равна значению **data_type_code_len** (т. е. 35).

5.11.2.3.3 Тип даты (Date_type)

Сущность **date_type** задает значения, допустимые для даты. Указанные значения соответствуют ИСО 8601.

Пример — «1994-03-21».

Пример представления на языке EXPRESS:

```
*)
TYPE date_type = STRING(10) FIXED;
WHERE
     WR1: SELF LIKE '####-##-##';
END_TYPE; -- date_type
(*
```

5.11.2.4 Тип определения (Definition_type)

Сущность **definition_type** задает значения, допустимые для определения типа.

Пример представления на языке EXPRESS:

```
*)
TYPE definition_type = translatable_text;
END_TYPE; -- definition_type
(*
```

5.11.2.5 Тип классификации типа элемента данных (DET_classification_type)

Сущность **DET_classification_type** задает значения, допустимые для классификации типов элементов данных **DET**. Указанные значения используются для классификации **DET** в соответствии с ИСО 80000/МЭК 80000 (ранее ИСО 31).

Пример представления на языке EXPRESS:

```
*)
TYPE DET_classification_type = identifier;
WHERE
     WR1: LENGTH(SELF) = DET_classification_len;
END_TYPE; -- DET_classification_type
(*
```

Пояснение к тексту программы:

WR1: длина значения **DET_classification_type** должна быть равна значению **DET_classification_len** (т. е. 3).

5.11.2.6 Тип примечания (Note_type)

Сущность **note_type** задает значения, допустимые для примечаний.

Пример представления на языке EXPRESS:

```
*)
TYPE note_type = translatable_text;
END_TYPE; -- note_type
(*
```

5.11.2.7 Тип предпочтительного имени (Pref_name_type)

Сущность **pref_name_type** задает значения, допустимые для предпочтительного имени.

Пример представления на языке EXPRESS:

```
*)
TYPE pref_name_type = translatable_label;
WHERE
     WR1: check_label_length(SELF, pref_name_len);
END_TYPE; -- pref_name_type
(*
```

Пояснение к тексту программы:

WR1: длина значения **pref_name_type** не должна превышать длину **pref_name_len** (т. е. 255).

5.11.2.8 Тип кода свойства (Property_code_type)

Сущность **property_code_type** задает значения, допустимые для кода свойства.

Пример представления на языке EXPRESS:

```
*)
TYPE property_code_type = code_type;
WHERE
     WR1: LENGTH(SELF) <= property_code_len;
END_TYPE; -- property_code_type
(*
```

Пояснение к тексту программы:

WR1: длина значения **property_code_type** не должна превышать длину **property_code_len** (т. е. 35).

5.11.2.9 Тип заметки (Remark_type)

Сущность **remark_type** задает значения, допустимые для заметки.

Пример представления на языке EXPRESS:

```
*)
TYPE remark_type = translatable_text;
END_TYPE; -- remark_type
(*
```

5.11.2.10 Тип иерархического положения (Hierarchical_position_type)

Сущность **hierarchical_position_type** задает значения, допустимые для иерархического положения.

Пример представления на языке EXPRESS:

```
*)
TYPE hierarchical_position_type = identifier;
END_TYPE; -- hierarchical_position_type
(*
```

Примечание — Представление иерархического положения атрибутом **hierarchical_position_type** основано на использовании некоторых конвенций кодирования. В настоящем стандарте конвенции кодирования не рассматриваются.

5.11.2.11 Тип пересмотра (Revision_type)

Сущность **revision_type** задает значения, допустимые для № пересмотра.

Примечание — Для новой версии № пересмотра равен «0».

Пример представления на языке EXPRESS:

```
*)
TYPE revision_type = code_type;
WHERE
     WR1: LENGTH(SELF) <= revision_len;
     WR2: EXISTS(VALUE(SELF)) AND ('INTEGER' IN TYPEOF(VALUE(SELF)))
          AND (VALUE(SELF) >= 0);
END_TYPE; -- revision_type
(*
```

Пояснения к тексту программы:

WR1: длина значения **revision_type** не должна превышать длину **revision_len** (т. е. 3).

WR2: значение **revision_type** должно содержать только цифры, его целое значение может быть только натуральным.

5.11.2.12 Тип короткого имени (Short_name_type)

Сущность **short_name_type** задает значения, допустимые для короткого имени.

Пример представления на языке EXPRESS:

```
*)
TYPE short_name_type = translatable_label;
WHERE
     WR1: check_label_length(SELF, short_name_len);
END_TYPE; -- short_name_type
(*
```

Пояснение к тексту программы:

WR1: длина значения **short_name_type** не должна превышать длину **short_name_len** (т. е. 30).

5.11.2.13 Тип кода поставщика (Supplier_code_type)

Сущность **supplier_code_type** задает значения, допустимые для кода поставщика.

Примечание — Если код поставщика также предназначен для обмена в соответствии с ИСО/ТС 29002-5, то различные части кода поставщика в соответствии с ИСО 6523 (ICD, OI, OPI, OPIS и AI) разделяются символом «–».

Пример представления на языке EXPRESS:

```
*)
TYPE supplier_code_type = code_type;
WHERE
     WR1: LENGTH(SELF) <= supplier_code_len;
END_TYPE; -- supplier_code_type
(*
```

Пояснение к тексту программы:

WR1: длина значения **supplier_code_type** не должна превышать длину **supplier_code_len** (т. е. 149).

5.11.2.14 Тип синонимичного имени (Syn_name_type)

Сущность **syn_name_type** задает значения, допустимые для синонимичного имени.

Пример представления на языке EXPRESS:

```
*)
TYPE syn_name_type = SELECT(label_with_language, label);
WHERE
     WR1: check_syn_length(SELF, syn_name_len);
END_TYPE; -- syn_name_type
(*
```

Пояснение к тексту программы:

WR1: длина значения **syn_name_type** не должна превышать длину **syn_name_len** (т. е. 255).

5.11.2.15 Тип ключевого слова (Keyword_type)

Сущность **keyword_type** задает значения, допустимые для ключевых слов.

Пример представления на языке EXPRESS:

```
*)
TYPE keyword_type = SELECT(label_with_language, label);
END_TYPE; -- keyword_type
(*
```

5.11.2.16 Тип глобальной идентификации ISO_29002_IRDI_type

Сущность **ISO_29002_IRDI_type** — это глобальный идентификатор, указывающий на администрируемый элемент в реестре. Структура данного идентификатора удовлетворяет синтаксическим требованиям к идентификаторам, определенным в ИСО/ТС 29002-5.

Примечание 1 — Сущность **ISO_29002_IRDI_type** может быть использована для любого вида информации, рассмотренной в ИСО/ТС 29002-5 и ассоциированной с идентификатором IRDI. Ниже рассмотрены три специальных случая, нашедшие применение в стандартной словарной схеме **ISO13584_IEC61360_dictionary_schema:** идентификатор ограничения **constraint_identifier,** идентификатор словарной единицы измерения **dic_unit_identifier** и идентификатор словарного значения **dic_value_identifier.**

Пример представления на языке EXPRESS:

```
*)
TYPE ISO_29002_IRDI_type = identifier;
WHERE
     WR1: LENGTH (SELF) <= 290;
END_TYPE; -- syn_name_type
(*
```

Пояснение к тексту программы:

WR1: в соответствии с ИСО/ТС 29002-5 длина идентификатора не должна превышать 290.

Дополнительное пояснение:

IP1: идентификатор должен удовлетворять требованиям, установленным в ИСО/ТС 29002-5 для международного идентификатора регистрации данных (IRDI).

Примечание 2 — В соответствии с ИСО/ТС 29002-5 IRDI состоит либо из строки, не содержащей символ «#» (идентификатор организации), либо из трех подстрок, не содержащих символ «#» внутри и разделенных символом «#» для указания администрированных элементов.

Примечание 3 — Случаи, когда IRDI не используется для идентификации организации:

- первая подстрока, называемая регистрационным идентификатором полномочий (RAI), указывает организацию, несущую ответственность за администрирование рассматриваемого элемента;
- вторая подстрока, называемая идентификатором данных (DI), содержит как категоризацию администрируемого элемента, представленную двумя символами и последующим знаком минус («–») в соответствии с ИСО/ТС 29002-5 (например, класс, свойство, единица измерения), так и идентификатор, назначенный для администрируемого элемента регистрационным идентификатором полномочий RAI;
- третья подстрока соответствует идентификатору версии (VI) IRDI.

5.11.2.17 Идентификатор ограничения (Constraint_identifier)

Идентификатор **constraint_identifier** — это идентификатор типа **ISO_29002_IRDI_type**, определяющий глобальное указание на ограничение, представленное как сущность **constraint**. Структура данного идентификатора удовлетворяет синтаксическим требованиям, определенным в ИСО/ТС 29002-5.

Примечание — Идентификатор ограничения **constraint_identifier** может быть ассоциирован с услугой разрешения по ИСО/ТС 29002-20. Данная услуга дает возможность получить формальное определение ограничения, идентифицированного как сущность **constraint_identifier** в соответствии с моделью ограничения EXPRESS в рамках синтаксиса, определенного ИСО 13584-32 (OntoML) и, возможно, ИСО 10303-21.

Пример представления на языке EXPRESS:

```
*)
TYPE constraint_identifier = ISO_29002_IRDI_type;
END_TYPE; -- constraint_identifier
(*
```

Дополнительное пояснение:

IP1: часть идентификатора, идущая после второго символа «#» (идентификатор данных), должна начинаться с «04-», чтобы указать на ограничение в соответствии с ИСО/ТС 29002-5.

5.11.2.18 Идентификатор словарной единицы измерения (Dic_unit_identifier)

Идентификатор **dic_unit_identifier** — это идентификатор **ISO_29002_IRDI_type**, определяющий единицу измерения, представление которой в формате **dic_unit** можно загрузить с сервера ИСО/ТС 29002-20. Структура данного идентификатора удовлетворяет синтаксическим требованиям ИСО/ТС 29002-5.

Пример представления на языке EXPRESS:

```
*)
TYPE dic_unit_identifier = ISO_29002_IRDI_type;
END_TYPE; -- dic_unit_identifier
(*
```

Дополнительные пояснения:

IP1: идентификатор **dic_unit_identifier** должен быть ассоциирован с услугой разрешения по ИСО/ТС 29002. Данная услуга предоставляет формальное определение единицы измерения, идентифицированной сущностью **dic_unit_identifier,** соответствующей требованиям модели **dic_unit** языка EXPRESS в синтаксисе, определенном ИСО 13584-32 (OntoML) и, возможно, ИСО 10303-21.

IP2: часть идентификатора, идущая после второго символа «#» (идентификатор данных), должна начинаться с «05-», чтобы указать на единицу измерения по ИСО/ТС 29002-5.

Примечание — Идентификатор словарной единицы измерения **dic_unit_identifier** составляет международный идентификатор регистрации данных (IRDI) в соответствии с ИСО/МЭК 11179-5.

5.11.2.19 Идентификатор словарного значения (Dic_value_identifier)

Идентификатор **dic_value_identifier** — это стандартный идентификатор типа **ISO_29002_IRDI_type**, определяющий глобальный идентификатор значения свойства, представленный как сущность **dic_value**. Структура данного идентификатора удовлетворяет синтаксическим требованиям ИСО/ТС 29002-5.

Примечание 1 — Назначение идентификатора словарного значения **dic_value_identifier** допускает повторное использование рассматриваемого определения **dic_value** в нескольких областях значений **value_domain.**

Примечание 2 — Идентификатор **dic_value_identifier** может быть ассоциирован с услугой разрешения по ИСО/ТС 29002. Данная услуга дает возможность получить формальное определение значения, идентифицированного сущностью **dic_value_identifier** в соответствии с моделью **dic_value** языка EXPRESS в синтаксисе, определенном ИСО 13584-32 (OntoML) и, возможно, ИСО 10303-21.

Пример представления на языке EXPRESS:

```
*)
TYPE dic_value_identifier = ISO_29002_IRDI_type;
END_TYPE; -- dic_value_identifier
(*
```

Дополнительное пояснение к тексту программы:

IP1: часть идентификатора, идущая после второго символа «#» (идентификатор данных), должна начинаться с «07-» для указания значения свойства в соответствии с ИСО/ТС 29002-5.

Примечание 3 — Идентификатор словарного значения **dic_value_identifier** составляет международный идентификатор регистрации данных (IRDI) в соответствии с ИСО/МЭК 11179-5.

5.11.2.20 Тип кода значения (Value_code_type)

Сущность **value_code_type** задает значения, допустимые для кода значения.

Пример представления на языке EXPRESS:

```
*)
TYPE value_code_type = identifier;
WHERE
     WR1: LENGTH(SELF) <= value_code_len;
END_TYPE; -- value_code_type
(*
```

Пояснение к тексту программы:

WR1: длина значения **value_code_type** не должна превышать длину **value_code_len** (т. е. 35).

5.11.2.21 Тип формата значения (Value_format_type)

Сущность **value_format_type** задает значения, допустимые для формата значения. Указанные значения определяются в соответствии с приложением D.

Пример представления на языке EXPRESS:

```
*)
TYPE value_format_type = identifier;
WHERE
     WR1: LENGTH(SELF) <= value_format_len;
END_TYPE; -- value_format_type
(*
```

Пояснение к тексту программы:

WR1: длина значения **value_format_type** не должна превышать длину **value_format_len** (т. е. 80).

5.11.2.22 Тип версии (Version_type)

Сущность **version_type** задает значения, допустимые для номера версии.

Пример представления на языке EXPRESS:

```
*)
TYPE version_type = code_type;
WHERE
     WR1: LENGTH(SELF) <= version_len;
     WR2: EXISTS(VALUE(SELF)) AND ('INTEGER' IN TYPEOF(VALUE(SELF)))
          AND (VALUE(SELF) >= 0);
END_TYPE; -- version_type
(*
```

Пояснения к тексту программы:

WR1: длина значения **version_type** не должна превышать длину **version_len** (т. е. 10).

WR2: значение **version_type** должно содержать только цифры.

5.11.2.23 Тип статуса (Status_type)

Сущность **status_type** задает значения, допустимые для статуса. Допустимые значения **status_type** не стандартизованы. Они должны быть определены для каждого отдельного словаря поставщиком словарных данных.

Пример 1 — Набор допустимых значений статуса элементов, предлагаемый для стандартизации агентству технической поддержки стандарта ИСО, определен директивами ИСО.

Пример 2 — Набор допустимых значений статуса элементов в базе данных стандарта МЭК определен директивами МЭК.

Примечание — Статус может быть ассоциирован со словарным элементом **dictionary_element** или со словарным значением **dic_value.**

Пример представления на языке EXPRESS:

```
*)
TYPE status_type = identifier;
END_TYPE; -- status_type
(*
```

5.11.2.24 Тип кода словаря (Dictionary_code_type)

Сущность **dictionary_code_type** — это тип кода **code_type**, задающий словарь.

Пример представления на языке EXPRESS:

```
*)
TYPE dictionary_code_type = identifier;
```

```
WHERE
     WR1: LENGTH(SELF) <= dictionary_code_len;
END_TYPE; -- dictionary_code_type
(*
```

Пояснение к тексту программы:

WR1: длина значения **dictionary_code_type** не должна превышать длину **dictionary_code_len** (т. е. 131).

5.11.3 Определения базовой сущности

5.11.3.1 Общие положения

Данный подпункт содержит определения базовой сущности, расположенные по английскому алфавиту.

5.11.3.2 Даты (dates)

Сущность **dates** описывает три даты, ассоциированные соответственно с первым устойчивым описанием, текущим номером версии и текущим номером пересмотра для данного описания.

Примечание — По каждому отдельному правилу управления словарем поставщик информации словаря несет ответственность за выбор момента времени первого устойчивого описания элемента.

Пример представления на языке EXPRESS:

```
*)
ENTITY dates;
     date_of_original_definition: date_type;
     date_of_current_version: date_type;
     date_of_current_revision: OPTIONAL date_type;
END_ENTITY; -- dates
(*
```

Определения атрибутов:

date_of_initial_definition: дата, ассоциированная с первой устойчивой версией элемента.

date_of_current_version: дата, ассоциированная с текущей версией.

date_of_current_revision: дата, ассоциированная с последним пересмотром.

5.11.3.3 Документ (document)

Сущность **document** — это абстрактный ресурс, заменяющий документ. Словарная схема обеспечивает только обмен идентификации документа (см. ниже). Сущность **document** может также быть описана как подтип с сущностями, использующими средства обмена данных документа.

Пример — Путем ссылки на внешний файл и на точную спецификацию формата файла.

Пример представления на языке EXPRESS:

```
*)
ENTITY document
ABSTRACT SUPERTYPE;
END_ENTITY; -- document
(*
```

5.11.3.4 Графика (graphics)

Сущность **graphics** — это абстрактный ресурс, описываемый как подтип, содержащий сущности, применяющие средства обмена графическими данными.

Пример — Ссылка на внешний файл и на точное задание формата файла.

Пример представления на языке EXPRESS:

```
*)
ENTITY graphics
ABSTRACT SUPERTYPE;
```

```
END_ENTITY; -- graphics
(*
```

5.11.3.5 Внешняя графика (External_graphics)

Сущность **external_graphics** обеспечивает обмен графических данных с помощью внешних файлов, на которые производится ссылка сущностью **graphic_file**.

Пример представления на языке EXPRESS:

```
*)
ENTITY external_graphics
SUBTYPE OF (graphics);
      representation: graphic_files;
END_ENTITY; -- external_graphics
(*
```

Определения атрибутов:

representation: представление графики с помощью внешних файлов.

5.11.3.6 Графический файл (Graphic_file)

Графический файл **graphic_file** — это внешний элемент **external_item,** содержанием которого является рисунок.

Примечание 1 — Сущность **external_item**, определенная в ИСО 13584-24:2003, — это элемент, содержание которого может быть составлено из библиотечных внешних файлов. Она ссылается на протокол внешнего файла **external_file_protocol,** указывающий порядок обработки библиотечного внешнего файла, и на внешний контент **external_content,** указывающий библиотечные внешние файлы, представляющие нужное содержание.

Примечание 2 — Протокол **external_file_protocol** и контент **external_content** определены в ИСО 13584-24:2003.

Примечание 3 — Только на внешний контент **external_content,** содержащий **http_file,** и только на гипертекстовые протоколы **http_protocol** и протоколы внешних файлов **external_file_protocols** производится ссылка стандартной словарной схемой **ISO13584_IEC61360_dictionary_schema.** Указанные сущности могут быть использованы в контексте настоящего стандарта.

Примечание 4 — Графические файлы **graphic_file** могут зависеть от языка. Это определяется следующими подтипами внешнего контента **external_content**: непереводимый внешний контент **not_translatable_external_content,** непереведенный внешний контент **not_translated_external_content,** переведенный внешний контент **translated_external_content.**

Примечание 5 — Сущности **http_file, http_protocol, not_translatable_external_content, not_translated_external_content** и **translated_external_content** определены в ИСО 13584-24:2003. На них производится ссылка стандартной словарной схемой **ISO13584_IEC61360_dictionary_schema.**

Пример представления на языке EXPRESS:

```
*)
ENTITY graphic_files
SUBTYPE OF (external_item);
END_ENTITY; -- graphic_files
(*
```

5.11.3.7 Идентифицированный документ (Identified_document)

Сущность **identified_document** описывает документ, идентифицированный своей меткой.

Пример представления на языке EXPRESS:

```
*)
ENTITY identified_document
SUBTYPE OF(document);
      document_identifier: translatable_label;
WHERE
      WR1: check_label_length(SELF.document_identifier,source_doc_len);
```

```
END_ENTITY; -- identified_document
(*
```

Определения атрибутов:

document_identifier: метка описанного документа.

Пояснение к тексту программы:

WR1: длина значения **document_identifier** не должна превышать длину **source_doc_len** (т. е. 255).

5.11.3.8 Названия элементов (Item_names)

Сущность **item_names** задает имена, которые можно ассоциировать с данным описанием. Она определяет предпочтительное имя, набор синонимичных имен, краткое имя и языки для прочих имен. Данная сущность может быть ассоциирована с иконкой.

Пример представления на языке EXPRESS:

```
*)
ENTITY item_names;
      preferred_name: pref_name_type;
      synonymous_names: SET OF syn_name_type;
      short_name: OPTIONAL short_name_type;
      languages: OPTIONAL present_translations;
      icon: OPTIONAL graphics;
WHERE
      WR1: NOT(EXISTS(languages )) OR (
           ('ISO13584_IEC61360_LANGUAGE_RESOURCE_SCHEMA'
           + '.TRANSLATED_LABEL' IN TYPEOF(preferred_name))
           AND (languages :=:
           preferred_name\translated_label.languages)
           AND (NOT(EXISTS(short_name)) OR
           ('ISO13584_IEC61360_LANGUAGE_RESOURCE_SCHEMA'
           + '.TRANSLATED_LABEL' IN TYPEOF(short_name))
           AND (languages :=: short_name\translated_label.languages))
           AND (QUERY(s <* synonymous_names |
           NOT('ISO13584_IEC61360_DICTIONARY_SCHEMA' +
           '.LABEL_WITH_LANGUAGE' IN TYPEOF(s))) = []));
      WR2: NOT EXISTS(languages) OR (QUERY(s <* synonymous_names |
           EXISTS(s.language) AND NOT(s.language IN
           QUERY(l <* languages.language_codes | TRUE
           ))) = []);
      WR3: EXISTS(languages) OR
           (('SUPPORT_RESOURCE_SCHEMA.LABEL' IN
           TYPEOF(preferred_name))

           AND (NOT(EXISTS(short_name)) OR
           ('SUPPORT_RESOURCE_SCHEMA.LABEL' IN
           TYPEOF(short_name)))
           AND (QUERY(s <* synonymous_names |
           'ISO13584_IEC61360_DICTIONARY_SCHEMA.LABEL_WITH_LANGUAGE' IN
           TYPEOF(s)) = []));
END_ENTITY; -- item_names
(*
```

Определения атрибутов:

preferred_name: имя, предпочтительное для использования.

synonymous_names: набор синонимичных имен.

short_names: аббревиатура предпочтительного имени.

languages: список языков по выбору, на которых даются другие имена.

icon: вспомогательная иконка для графического представления описания, ассоциированного с атрибутом **item_names**.

Пояснения к тексту программы:

WR1: если предпочтительные и короткие имена даются более чем на одном языке, то все атрибуты **languages** переведенных меток **translated_labels** должны содержать реализации переводов **present_translations** как в атрибуте **languages** рассматриваемой реализации **item_names**.

WR2: если синонимичные имена даются на более чем одном языке, то можно использовать только языки, указанные в реализациях переводов **present_translations** и в атрибутах **languages** рассматриваемой реализации **item_names**.

WR3: если язык не указан, то атрибуты **preferred_names**, **short_names** и **synonymous_names** можно не переводить.

Примечание 1 — Атрибуты **preferred_names**, **synonymous_names** и **short_names** используются для кодирования атрибутов «Предпочтительное имя», «Синонимичное имя» и «Краткое имя» свойств и классов соответственно.

Примечание 2 — Атрибут **languages** используется для определения последовательности переводов (если это необходимо для атрибутов).

5.11.3.9 Метка с языком (Label_with_language)

Сущность **label_with_language** обеспечивает ресурс, ассоциирующий метку с языком.

Пример представления на языке EXPRESS:

```
*)
ENTITY label_with_language; l:
      label;
      language: language_code; END_
ENTITY; -- label_with_language
(*
```

Определения атрибутов:

l: метка, ассоциированная с языком.

language: код помеченного языка.

5.11.3.10 Математическая строка (Mathematical_string)

Сущность **mathematical_string** обеспечивает ресурсы, определяющие представление математических строк. Она допускает представление в формате MathML.

Пример представления на языке EXPRESS:

```
*)
ENTITY mathematical_string;
      text_representation: text;
      MathML_representation: OPTIONAL text;
END_ENTITY; -- mathematical_string
(*
```

Определения атрибутов:

text_representation: «линейная» форма математической строки, использующая ИСО 843 (при необходимости).

MathML_representation: текст в формате MathML, размеченный в соответствии с требованиями XML DTD (определения типа документа) для MathML. Текст MathML обрабатывается как отдельная строка во время обмена (см. ИСО 10303-21).

5.12 Определения функций

5.12.1 Общие положения

Данный подраздел содержит функции, на которые производится ссылка из разделов по месту для подтверждения непротиворечивости данных и которые предоставляют ресурсы для разработки приложений.

5.12.2 Функция ациклического соотношения суперклассов (Acyclic_superclass_relationship)

Функция **acyclic_superclass_relationship** проверяет отсутствие цикла в соотношении суперклассов. Кардинальное число атрибута **its_superclass** в классе сущностей гарантирует, что существует дерево наследственности, а ациклических графов нет. Данная функция не проверяет тот факт, что

реализации класса не ссылаются (в атрибуте **its_superclass**) на другую реализацию, которая, в сущности, является подклассом.

Пример представления на языке EXPRESS:

```
*)
FUNCTION acyclic_superclass_relationship(
     current: class_BSU; visited: SET OF class): LOGICAL;

IF SIZEOF(current.definition) = 1 THEN
     IF current.definition[1] IN visited THEN
          RETURN(FALSE);
     (* wrong: current declares a subclass as its superclass *)
     ELSE
          IF EXISTS(current.definition[1]\class.its_superclass)
          THEN
               RETURN(acyclic_superclass_relationship(
                    current.definition[1]\class.its_superclass,
                    visited + current.definition[1]));
          ELSE


               RETURN(TRUE);
          END_IF;
     END_IF;
ELSE
     RETURN(UNKNOWN);
END_IF;
END_FUNCTION; -- acyclic_superclass_relationship
(*
```

5.12.3 Функция проверки длины (Check_syn_length)

Функция **check_syn_length** проверяет тот факт, что длина атрибута **s** не превышает длину, определенную атрибутом **s_length.**

Пример представления на языке EXPRESS:

```
*)
FUNCTION check_syn_length(s: syn_name_type; s_length: INTEGER):BOOLEAN;

IF 'ISO13584_IEC61360_DICTIONARY_SCHEMA.LABEL_WITH_LANGUAGE'
     IN TYPEOF(s)
THEN
     RETURN(LENGTH(s.1) <= s_length);
ELSE
     RETURN(LENGTH(s) <= s_length);
END_IF;
END_FUNCTION; -- check_syn_length
(*
```

5.12.4 Функция проверки уникальности кода (Codes_are_unique)

Функция **codes_are_unique** возвращает значение TRUE, если коды значений **value_codes** уникальны внутри рассматриваемого списка значений.

Пример представления на языке EXPRESS:

```
*)
FUNCTION codes_are_unique(values: LIST OF dic_value): BOOLEAN;
LOCAL
```

```
      ls: SET OF STRING := [];
      li: SET OF INTEGER := [];
END_LOCAL;

IF('ISO13584_IEC61360_DICTIONARY_SCHEMA.VALUE_CODE_TYPE' IN
      TYPEOF(values[1].value_code))
THEN
      REPEAT i := 1 TO SIZEOF(values);
           ls := ls + values[i].value_code;
      END_REPEAT;

      RETURN(SIZEOF(values) = SIZEOF(ls));
ELSE
      IF('ISO13584_IEC61360_DICTIONARY_SCHEMA.INTEGER_TYPE' IN
           TYPEOF(values[1].value_code))
      THEN


           REPEAT i := 1 TO SIZEOF(values);
                li := li + values[i].value_code;
           END_REPEAT;

           RETURN(SIZEOF(values) = SIZEOF(li));
      ELSE
           RETURN(?);
      END_IF;
END_IF;

END_FUNCTION; -- codes_are_unique
(*
```

5.12.5 Функция проверки наличия определения (Definition_available_implies)

Функция **definition_available_implies** проверяет, действительно ли существует определение, соответствующее рассматриваемому параметру **BSU**. Если данное определение существует, то возвращается значение параметра **expression**.

Пример представления на языке EXPRESS:

```
*)
FUNCTION definition_available_implies(
      BSU: basic_semantic_unit;
      expression: LOGICAL): LOGICAL;

RETURN(NOT(SIZEOF(BSU.definition) = 1) OR expression);

END_FUNCTION; -- definition_available_implies
(*
```

5.12.6 Функция проверки подкласса (Is_subclass)

Функция **is_subclass** возвращает значение TRUE, если подкласс является либо суперклассом, либо подклассом суперкласса. Если некоторые словарные определения **dictionary_definition** класса недоступны, то функция возвращает значение UNKNOWN.

Пример представления на языке EXPRESS:

```
*)
FUNCTION is_subclass(sub, super: class): LOGICAL;
      IF (NOT EXISTS(sub)) OR (NOT EXISTS(super)) THEN
```

```
            RETURN(UNKNOWN);
        END_IF;

        IF sub = super
        THEN
            RETURN(TRUE);
        END_IF;

        IF NOT EXISTS(sub.its_superclass)
        THEN
            (* end of chain reached, didn't meet super so far *)
            RETURN(FALSE);
        END_IF;
        IF SIZEOF(sub.its_superclass.definition) = 1
        THEN
        (* definition available *)
            IF (sub.its_superclass.definition[1] = super)
            THEN
                RETURN(TRUE);
            ELSE
                RETURN(is_subclass(sub.its_superclass.definition[1],
                super));
            END_IF;
        ELSE
            RETURN(UNKNOWN);
        END_IF;

    END_FUNCTION; -- is_subclass
    (*
```

5.12.7 Функция строчного представления производной единицы измерения (String_for_derived_unit)

Функция **string_for_derived_unit** возвращает строчное представление производной единицы измерения **derived_unit** (в соответствии с ИСО 10303-41), рассматриваемой как параметр. Элементы производной единицы измерения отличаются по знаку показателя степени. Если существуют элементы обоих знаков, то обозначение «/» используется для отделения положительных значений от отрицательных. Если имеются только отрицательные показатели степени, то используется обозначение «u-e». Точка «.» (десятичный код № 46 в соответствии с ИСО/МЭК 8859-1, см. раздел ИСО 10303-21) используется для отделения индивидуальных элементов.

Пример представления на языке EXPRESS:

```
    *)
    FUNCTION string_for_derived_unit(u: derived_unit): STRING;

        FUNCTION string_for_derived_unit_element(
            u: derived_unit_element; neg_exp: BOOLEAN
            (* print negative exponents with power -1 *)): STRING;
            (* returns a STRING representation of the
                derived_unit_element (according to ISO 10303-41)
                passed as parameter *)

        LOCAL
            result: STRING;
        END_LOCAL;

        result := string_for_named_unit(u.unit);
        IF (u.exponent <> 0)
```

```
      THEN
           IF (u.exponent > 0) OR NOT neg_exp
           THEN
                result := result + '**' + FORMAT(
                     ABS(u.exponent), '2I')[2];
           ELSE
                result := result + '**' + FORMAT(u.exponent, '2I')[2];
           END_IF;
      END_IF;
           RETURN(result);
      END_FUNCTION; -- string_for_derived_unit_element
LOCAL
      pos, neg: SET OF derived_unit_element;
      us: STRING;
END_LOCAL;

(* separate unit elements according to the sign of the exponents: *)
pos := QUERY(ue <* u.elements | ue.exponent > 0);
neg := QUERY(ue <* u.elements | ue.exponent < 0);
us := '';
IF SIZEOF(pos) > 0 THEN
      (* there are unit elements with positive sign *)
      REPEAT i := LOINDEX(pos) TO HIINDEX(pos);
           us := us + string_for_derived_unit_element(pos[i], FALSE);
           IF i <> HIINDEX(pos)
           THEN
                us := us + '.';
           END_IF;
      END_REPEAT;

      IF SIZEOF(neg) > 0
      THEN
           (* there are unit elements with negative sign, use '/'
                notation: *)
           us := us + '/';

           IF SIZEOF(neg) > 1
           THEN
                us := us + '(';
           END_IF;

           REPEAT i := LOINDEX(neg) TO HIINDEX(neg);
                us := us + string_for_derived_unit_element(
                     neg[i], FALSE);
                IF i <> HIINDEX(neg)
                THEN
                     us := us + '.';
                END_IF;
           END_REPEAT;

           IF SIZEOF(neg) > 1
           THEN
                us := us + ')';
           END_IF;
      END_IF;
```

```
ELSE
      (* only negative signs, use u-e notation *)
      IF SIZEOF(neg) > 0 THEN
           REPEAT i := LOINDEX(neg) TO HIINDEX(neg);
                us := us + string_for_derived_unit_element
                     ( neg[i], TRUE);
                IF i <> HIINDEX(neg)
                THEN
                     us := us + '.';
                END_IF;
           END_REPEAT;
      END_IF;
END_IF;

RETURN(us);

END_FUNCTION; -- string_for_derived_unit
(*
```

5.12.8 Функция строчного представления поименованных единиц измерения (String_for_named_unit)

Функция **string_for_named_unit** возвращает строчное представление поименованной единицы измерения **named_unit** (в соответствии с ИСО 10303-41 и расширением в 5.10.6.2), рассматриваемой как параметр.

Пример представления на языке EXPRESS:

```
*)
FUNCTION string_for_named_unit(u: named_unit): STRING;

IF 'MEASURE_SCHEMA.SI_UNIT' IN TYPEOF(u) THEN
      RETURN(string_for_SI_unit(u));
ELSE
      IF 'MEASURE_SCHEMA.CONTEXT_DEPENDENT_UNIT' IN TYPEOF(u)
      THEN
           RETURN(u\context_dependent_unit.name);
      ELSE
           IF 'MEASURE_SCHEMA.CONVERSION_BASED_UNIT' IN TYPEOF(u)
           THEN
                RETURN(u\conversion_based_unit.name);
           ELSE
                IF 'ISO13584_IEC61360_DICTIONARY_SCHEMA'
                    +'.NON_SI_UNIT' IN TYPEOF(u)
                THEN
                     RETURN(u\non_si_unit.name);
                ELSE
                     RETURN('name_unknown');
                END_IF;
           END_IF;
      END_IF;
END_IF;

END_FUNCTION; -- string_for_named_unit
(*
```

5.12.9 Функция строчного представления единицы системы СИ (String_for_SI_unit)

Функция **string_for_SI_unit** возвращает строчное представление единицы системы СИ **si_unit** (в соответствии с ИСО 10303-41), рассматриваемой как параметр.

Пример представления на языке EXPRESS:

```
*)

FUNCTION string_for_SI_unit(unit: si_unit): STRING;

LOCAL
     prefix_string, unit_string: STRING;
END_LOCAL;

IF EXISTS(unit.prefix) THEN
     CASE unit.prefix OF
          exa       : prefix_string := 'E';
          peta      : prefix_string := 'P';
          tera      : prefix_string := 'T';
          giga      : prefix_string := 'G';
          mega      : prefix_string := 'M';
          kilo      : prefix_string := 'k';
          hecto     : prefix_string := 'h';
          deca      : prefix_string := 'da';
          deci      : prefix_string := 'd';
          centi     : prefix_string := 'c';
          milli     : prefix_string := 'm';
          micro     : prefix_string := 'u';
          nano      : prefix_string := 'n';
          pico      : prefix_string := 'p';
          femto     : prefix_string := 'f';
          atto      : prefix_string := 'a';
     END_CASE;
ELSE
     prefix_string := '';
END_IF;

CASE unit.name OF
     metre               : unit_string:= 'm';
     gram                : unit_string := 'g';
     second              : unit_string := 's';
     ampere              : unit_string := 'A';
     kelvin              : unit_string := 'K';
     mole                : unit_string := 'mol';
     candela             : unit_string := 'cd';
     radian              : unit_string := 'rad';
     steradian           : unit_string := 'sr';
     hertz               : unit_string := 'Hz';
     newton              : unit_string := 'N';
     pascal              : unit_string := 'Pa';
     joule               : unit_string := 'J';
     watt                : unit_string := 'W';
     coulomb             : unit_string := 'C';
     volt                : unit_string := 'V';
     farad               : unit_string := 'F';
     ohm                 : unit_string := 'Ohm';
     siemens             : unit_string := 'S';
     weber               : unit_string := 'Wb';
     tesla               : unit_string := 'T';
```

```
        henry               : unit_string := 'H';
        degree_Celsius      : unit_string := 'Cel';
        lumen               : unit_string := 'lm';
        lux                 : unit_string := 'lx';
        becquerel           : unit_string := 'Bq';
        gray                : unit_string := 'Gy';
        sievert             : unit_string := 'Sv';
    END_CASE;

    RETURN(prefix_string + unit_string);

    END_FUNCTION; -- string_for_SI_unit
    (*
```

5.12.10 Функция строчного представления единицы измерения (String_for_unit)

Функция **string_for_unit** возвращает строчное представление единицы измерения **unit** (в соответствии с ИСО 10303-41), рассматриваемой как параметр.

Примечание — Функция **string_for_unit** не вызывается кодами языка EXPRESS. Это функция-утилита, позволяющая вычислить строчное представление по атрибуту **structured_representation** словарной единицы измерения **dic_unit,** если строчное представление **string_representation** отсутствует. Данное строчное представление соответствует варианту, рассмотренному в приложении В МЭК 61360-1:2009.

Пример представления на языке EXPRESS:

```
*)
FUNCTION string_for_unit(u: unit): STRING;
     IF 'MEASURE_SCHEMA.DERIVED_UNIT' IN TYPEOF(u)
     THEN
          RETURN(string_for_derived_unit(u));
     ELSE (* 'MEASURE_SCHEMA.NAMED_UNIT' IN TYPEOF(u) holds true *)
          RETURN(string_for_named_unit(u));
     END_IF;
END_FUNCTION; -- string_for_unit
(*
```

5.12.11 Функция обеспечения доступа ко всем описаниям класса (All_class_descriptions_reachable)

Функция **all_class_descriptions_reachable** проверяет тот факт, что если словарные элементы **dictionary_element** дают описание класса и на них производится ссылка BSU некоторого класса **class_ BSU** и всех его суперклассов, то они могут быть вычислены в дереве наследственности, определенном иерархией классов.

Пример представления на языке EXPRESS:

```
*)
FUNCTION all_class_descriptions_reachable(cl: class_BSU): BOOLEAN;

IF NOT EXISTS(cl)
THEN
     RETURN(?);
END_IF;

IF SIZEOF(cl.definition) = 0
THEN
     RETURN(FALSE);
END_IF;

IF NOT(EXISTS(cl.definition[1]\class.its_superclass))
```

```
THEN
      RETURN(TRUE);
ELSE

      RETURN(all_class_descriptions_reachable(
           cl.definition[1]\class.its_superclass));
END_IF;

END_FUNCTION; -- all_class_descriptions_reachable
(*
```

5.12.12 Функция вычисления известных видимых свойств (Compute_known_visible_properties)

Функция **compute_known_visible_properties** вычисляет набор свойств, являющихся видимыми в данном классе. Если определение недоступно, то функция возвращает только те видимые свойства, которые могут быть вычислены.

Примечание — Если словарные определения **dictionary_definition** некоторого класса не присутствуют в рассматриваемом контексте обмена (контекст обмена библиотеки PLIB никогда не предполагается полным), то суперкласс некоторого класса может быть неизвестен. Следовательно, свойства, определенные как видимые в данном суперклассе, не могут быть вычислены функцией **compute_known_visible_properties**. Только для получающей системы все словарные определения **dictionary_definition** базовых семантических единиц (BSU) являются доступными. Следовательно, для получающей системы функция **compute_known_visible_properties** вычисляет все свойства, являющиеся видимыми в классе, путем ссылки на него (или на любой его суперкласс) с помощью атрибута **name_scope**.

Пример представления на языке EXPRESS:

```
*)
FUNCTION compute_known_visible_properties(cl: class_BSU):
      SET OF property_BSU;
LOCAL
      s: SET OF property_BSU := [];
END_LOCAL;

s := s + USEDIN(cl, 'ISO13584_IEC61360_DICTIONARY_SCHEMA' +
      '.PROPERTY_BSU.NAME_SCOPE');
IF SIZEOF(cl.definition) = 0
THEN
      RETURN(s);
ELSE
      IF EXISTS(cl.definition[1]\class.its_superclass) THEN
           s := s + compute_known_visible_properties(
                cl.definition[1]\class.its_superclass);
      END_IF;

      RETURN(s);
END_IF;

END_FUNCTION; -- compute_known_visible_properties
(*
```

5.12.13 Функция вычисления известного видимого типа данных (Compute_known_visible_data_type)

Функция **compute_known_visible_data_type** вычисляет набор типов данных **data_types**, являющихся видимыми в данном классе. Если определение недоступно, то функция возвращает только видимые типы данных **data_types**, которые могут быть вычислены.

Примечание — Если словарные определения **dictionary_definition** некоторых классов не присутствуют в рассматриваемом контексте обмена (контекст обмена библиотеки PLIB никогда не предполагается полным), то суперкласс некоторого класса может быть неизвестен. Следовательно, типы данных **data_types**, определенные

как видимые в данном суперклассе, не могут быть вычислены функцией **compute_known_visible_data_type**. Только для получающей системы все словарные определения **dictionary_definition** базовых семантических элементов (BSU) являются доступными. Следовательно, для получающей системы функция **compute_known_visible_data_type** вычисляет все типы данных **data_types**, являющиеся видимыми в классе, путем ссылки на данный класс (или любой его суперкласс) с помощью атрибута **name_scope**.

Пример представления на языке EXPRESS:

```
*)
FUNCTION compute_known_visible_data_types(cl: class_BSU):
     SET OF data_type_BSU;
LOCAL
     s: SET OF data_type_BSU :=[ ];
END_LOCAL;

s := s + USEDIN(cl, 'ISO13584_IEC61360_DICTIONARY_SCHEMA' +
     '.DATA_TYPE_BSU.NAME_SCOPE');

IF SIZEOF(cl.definition) = 0
THEN
     RETURN(s);
ELSE
     IF EXISTS(cl.definition[1]\class.its_superclass)
     THEN
          s := s + compute_known_visible_data_types(
               cl.definition[1]\class.its_superclass);
     END_IF;

     RETURN(s);
END_IF;

END_FUNCTION; -- compute_known_visible_data_types
(*
```

5.12.14 Функция вычисления известных применимых свойств (Compute_known_applicable_properties)

Функция **compute_known_applicable_properties** вычисляет набор свойств, являющихся применимыми в данном классе. Если определение недоступно, то функция возвращает только те применимые свойства, которые могут быть вычислены.

Примечание — Если словарное определение **dictionary_definition** некоторого класса не присутствует в рассматриваемом контексте обмена (контекст обмена библиотеки PLIB никогда не предполагается полным), то суперкласс некоторого класса может быть неизвестен. Следовательно, свойство, определенное как применимое в данном суперклассе, не может быть вычислено функцией **compute_known_applicable_properties**. Только для получающей системы все словарные определения **dictionary_definition** базового семантического элемента (BSU) являются доступными. Следовательно, для получающей системы функция **compute_known_applicable_properties** вычисляет все свойства, являющиеся применимыми в данном классе, так как на них производится ссылка атрибутом **described_by**, или они импортируются с помощью априорного семантического соотношения **a_priori_semantic_relationship**.

Пример представления на языке EXPRESS:

```
*)
FUNCTION compute_known_applicable_properties(cl: class_BSU):
     SET OF property_BSU;

LOCAL
     s: SET OF property_BSU := [];
END_LOCAL;
```

```
IF SIZEOF(cl.definition)=0
THEN
     RETURN(s);
ELSE
     REPEAT i := 1 TO SIZEOF(cl.definition[1]\class.described_by);

          s := s + cl.definition[1]\class.described_by[i];
     END_REPEAT;

     IF (('ISO13584_IEC61360_ITEM_CLASS_CASE_OF_SCHEMA.'
          + 'A_PRIORI_SEMANTIC_RELATIONSHIP')
          IN TYPEOF (cl.definition[1]))
     THEN
          s                    :=                    s                    +
               cl.definition[1]\a_priori_semantic_relationship.referenc
          ed_properties;
     END_IF;

     IF EXISTS(cl.definition[1]\class.its_superclass)
     THEN
          s := s + compute_known_applicable_properties(
               cl.definition[1]\class.its_superclass);
     END_IF;

     RETURN(s);
END_IF;
END_FUNCTION; -- compute_known_applicable_properties
(*
```

5.12.15 Функция вычисления известного применимого типа данных (Compute_known_applicable_data_types)

Функция **compute_known_applicable_data_types** вычисляет набор типов данных **data_type**, являющихся применимыми в данном классе. Если определение недоступно, то функция возвращает только те применимые типы данных **data_types**, которые могут быть вычислены.

Примечание — Если словарное определение **dictionary_definition** некоторого класса не присутствует в рассматриваемом контексте обмена (контекст обмена библиотеки PLIB никогда не предполагается полным), то суперкласс некоторого класса может быть неизвестен. Следовательно, типы данных **data_types**, определенные как применимые в данном суперклассе, не могут быть вычислены функцией **compute_known_applicable_data_type**. Только для получающей системы все словарные определения **dictionary_definition** базовых семантических единиц (BSU) являются доступными. Следовательно, для получающей системы функция **compute_known_applicable_data_types** вычисляет все типы данных **data_types**, являющиеся применимыми в классе, так как на них производится ссылка атрибутом **defined_type**, или они импортируются с помощью априорного семантического соотношения **a_priori_semantic_relationship**.

Пример представления на языке EXPRESS:

```
*)
FUNCTION compute_known_applicable_data_types(cl: class_BSU):
     SET OF data_type_BSU;
LOCAL
     s: SET OF data_type_BSU := [];
END_LOCAL;

IF SIZEOF(cl.definition) = 0
THEN
     RETURN(s);
```

```
ELSE
     REPEAT i := 1 TO SIZEOF(cl.definition[1]\class.defined_types);
          s := s + cl.definition[1]\class.defined_types[i];
     END_REPEAT;

     IF (('ISO13584_IEC61360_ITEM_CLASS_CASE_OF_SCHEMA.'
          + 'A_PRIORI_SEMANTIC_RELATIONSHIP')
          IN TYPEOF (cl.definition[1]))
     THEN
          s                    :=                    s                    +
              cl.definition[1]\a_priori_semantic_relationship.referenc
          ed_data_types;
     END_IF;

     IF EXISTS(cl.definition[1]\class.its_superclass)
     THEN
          s := s + compute_known_applicable_data_types(
              cl.definition[1]\class.its_superclass);
     END_IF;

     RETURN(s);
END_IF;

END_FUNCTION; -- compute_known_applicable_data_types
(*
```

5.12.16 Функция создания набора из элементов списка (List_to_set)

Функция **list_to_set** создает набор из элементов списка с именем **l**. Тип элементов набора совпадает с типом элементов в исходном списке.

Пример представления на языке EXPRESS:

```
*)
FUNCTION list_to_set(l: LIST [0:?] OF GENERIC:type_elem): SET
     OF GENERIC: type_elem;

LOCAL
     s: SET OF GENERIC: type_elem := [];
END_LOCAL;

REPEAT i := 1 TO SIZEOF(l);
     s := s + l[i];
END_REPEAT;

RETURN(s);
END_FUNCTION; -- list_to_set
(*
```

5.12.17 Функция проверки применимости свойств (Check_property_applicability)

Функция **check_property_applicability** проверяет тот факт, что только свойства, не являющиеся применимыми в классе по наследству, могут стать применимыми в данном классе, так как на них производится ссылка атрибутом **described_by**.

Пример представления на языке EXPRESS:

```
*)
FUNCTION check_properties_applicability(cl: class): LOGICAL;
```

```
LOCAL
     inter: SET OF property_bsu := [];
END_LOCAL;

IF EXISTS(cl.its_superclass)
THEN
     IF (SIZEOF(cl.its_superclass.definition)=1)
     THEN
          inter := (list_to_set(cl.described_by) *
               cl.its_superclass.definition[1]\class.
               known_applicable_properties);
          RETURN(inter = []);
     ELSE
          RETURN(UNKNOWN);
     END_IF;
ELSE
     RETURN(TRUE);
END_IF;

END_FUNCTION; -- check_properties_applicability (
*
```

5.12.18 Функция проверки применимости типов данных (Check_datatypes_applicability)

Функция **check_datatypes_applicability** проверяет тот факт, что только типы данных, не являющиеся применимыми в классе по наследству, могут стать применимыми в данном классе, так как на них производится ссылка атрибутом **defined_type**.

Пример представления на языке EXPRESS:

```
*)
FUNCTION check_datatypes_applicability(cl: class): LOGICAL;
LOCAL
     inter: SET OF data_type_bsu := [];
END_LOCAL;

IF EXISTS(cl.its_superclass)
THEN
     IF (SIZEOF(cl.its_superclass.definition) = 1)
     THEN
          inter := cl.defined_types *
               cl.its_superclass.definition[1]\class.
               known_applicable_data_types;
          RETURN(inter = []);
     ELSE
          RETURN(UNKNOWN);
     END_IF;
ELSE
     RETURN(TRUE);
END_IF;

END_FUNCTION; -- check_datatypes_applicability

(*
```

5.12.19 Функция проверки уникальности языка перевода (One_language_per_translation)

Функция **one_language_per_translation** проверяет тот факт, что язык перевода административных данных **administrative_data** является уникальным.

Пример представления на языке EXPRESS:

```
*)
FUNCTION one_language_per_translation (adm: administrative_data)
                        : LOGICAL;
     LOCAL
          count: INTEGER; lang:
          language_code;
     END_LOCAL;

     REPEAT i := 1 TO SIZEOF (adm.translation);
          lang := adm.translation[i].language;
          count := 0;
          REPEAT j :=1 TO SIZEOF (adm.translation);
               IF lang = adm.translation[j].language
               THEN
                    count := count+1;
               END_IF;
          END_REPEAT;
          IF count >1
          THEN RETURN (FALSE);
          END_IF;
     END_REPEAT;
     RETURN(TRUE);

END_FUNCTION; -- one_language_per_translation
(*
```

5.12.20 Функция вычисления целых значений неколичественного типа (Allowed_value_integer_type)

Функция **allowed_value_integer_type** вычисляет набор целых **integer_type** значений, допустимых для атрибута **non_quantitative_int_type**. Если рассматриваемый параметр не определен, то функция возвращает неопределенное значение.

Пример представления на языке EXPRESS:

```
*)
FUNCTION allowed_values_integer_types (nqit: non_quantitative_int_type)
                        : SET OF integer_type;
LOCAL
     s : SET OF integer_type :=[];
END_LOCAL;

REPEAT i:=1 TO SIZEOF (nqit.domain.its_values);
     s := s + nqit.domain.its_values[i].value_code;
END_REPEAT;
RETURN(s);

END_FUNCTION; -- allowed_values_integer_types
(*
```

5.12.21 Функция идентификации свойств со значением класса (Is_class_valued_properties)

Функция **is_class_valued_properties** возвращает значение TRUE, если свойство **prop** определено как свойство со значением класса в классе **cl** с помощью атрибута **sub_class_properties** в классе **cl** или в любом его суперклассе. Если словарные определения **dictionary_definition** некоторых классов недоступны (при вычислении всех суперклассов для класса **cl**), то функция возвращает UNKNOWN.

Пример представления на языке EXPRESS:

```
*)
FUNCTION is_class_valued_property(
      prop: property_BSU; cl: class_BSU): LOGICAL;
      IF (SIZEOF(cl.definition) = 0)
      THEN
           RETURN (UNKNOWN);
      ELSE
           IF NOT (('ISO13584_IEC61360_DICTIONARY_SCHEMA'
               +'.ITEM_CLASS') IN TYPEOF(cl.definition[1]))
           THEN
                RETURN (FALSE);
           END_IF;
           IF prop IN cl.definition[1].sub_class_properties
           THEN RETURN (TRUE);
           END_IF;
           IF NOT EXISTS(cl.definition[1].its_superclass)
           THEN
           (* end of chain reached, didn't meet super so far *)
                RETURN(FALSE);
           END_IF;
           RETURN(is_class_valued_property(prop,
                     cl.definition[1].its_superclass));
      END_IF;

END_FUNCTION; -- is_class_valued_property
(*
```

5.12.22 Функция определения значения свойства (Class_value_assigned)

Функция **class_value_assigned** возвращает набор значений свойства **prop**, назначенного для класса **cl**, с помощью атрибута постоянного значения класса **class_constant_value** в классе **cl** или в любом его суперклассе. Если в различных суперклассах назначены различные значения, то функция возвращает набор всех этих назначенных значений. Если словарные определения **dictionary_definition** некоторых классов недоступны при вычислении всех суперклассов для класса **cl**, то возвращаются только вычисленные значения.

Пример представления на языке EXPRESS:

```
*)
FUNCTION class_value_assigned(prop: property_BSU;
      cl: class_BSU) : SET OF primitive_value;
      LOCAL
           val:SET OF primitive_value :=[];
           cva : SET OF class_value_assignment :=[];
      END_LOCAL;
      IF (SIZEOF(cl.definition) = 0)
      THEN
           RETURN (val);

      END_IF;
      IF NOT (('ISO13584_IEC61360_DICTIONARY_SCHEMA'
           +'.ITEM_CLASS') IN TYPEOF(cl.definition[1]))
      THEN
           RETURN (val);
      END_IF;
```

```
        IF EXISTS(cl.definition[1])
        THEN
             cva:= QUERY
                  (a <* cl.definition[1].class_constant_values
                  | a.super_class_defined_property = prop);
             REPEAT i :=1 TO SIZEOF (cva);
                  val := val + cva[i].assigned_value;
             END_REPEAT;
             IF NOT EXISTS(cl.definition[1].its_superclass)
             THEN
                  RETURN (val);
             ELSE RETURN (val + class_value_assigned
                  (prop,cl.definition[1].its_superclass));
             END_IF;
        END_IF;
   END_FUNCTION; -- class_value_assigned


   END_SCHEMA; -- ISO13584_IEC61360_dictionary_schema
   (*
```

6 Стандартная схема языкового ресурса (ISO13584_IEC61360_language_resource_schema)

6.1 Краткое описание

Следующая схема обеспечивает ресурсы для представления строк на различных языках. Данная языковая схема извлечена из имеющейся словарной схемы и может быть использована в других схематиках. В значительной степени она основана на схеме поддержки ресурсов **support_resource_schema** ИСО 10303-41 и может рассматриваться как ее расширение. В соответствии с данной схемой только один особый язык можно использовать в рассматриваемом контексте обмена (в физическом файле) без дополнительных усложнений, которые становятся необходимыми при использовании нескольких языков. См. графические пояснения на рисунке 13.

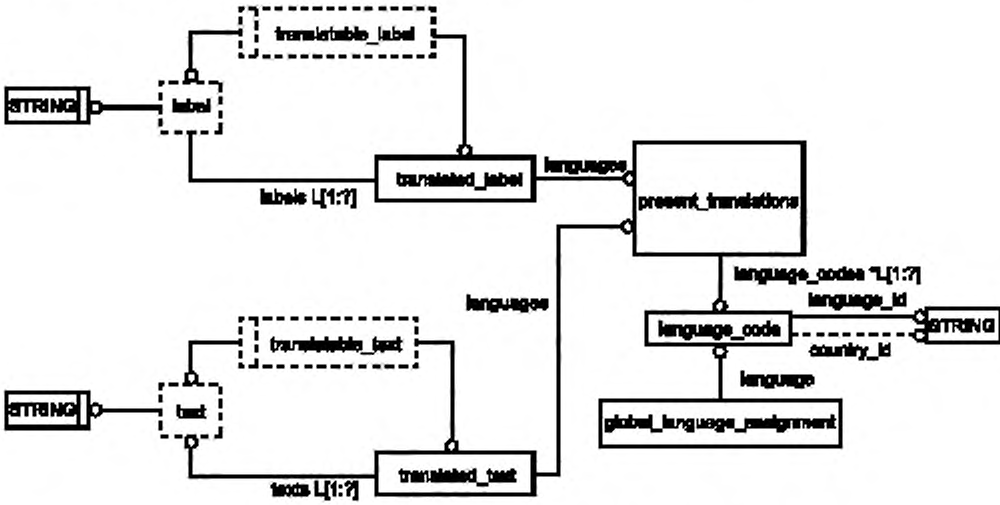


Рисунок 13, лист 1 — Стандартная схема языкового ресурса ISO13584_IEC61360_language_resource_schema и схема ресурса поддержки support_resource_schema

translatable_label	Переводимая метка
string	Строка
label	Метка
labels L[1:?]	Массив меток
translated_label	Переведенная метка
languages	Языки
present_translations	Доступные переводы
language_codes *L[1:?]	Массив языковых кодов
translatable_text	Переводимый текст
text	Текст
texts L[1:?]	Массив текстов
translated_text	Переведенный текст
language_id	Идентификатор языка
language_code	Код языка
country_id	Идентификатор страны
global_language_assignment	Назначение глобального языка

Рисунок 13, лист 2

Пример представления на языке EXPRESS:

```
*)
SCHEMA ISO13584_IEC61360_language_resource_schema;

REFERENCE FROM support_resource_schema(identifier, label, text);

(*
```

Примечание — Вышеуказанная схема ресурса поддержки **support_resource_schema** описана в ИСО 10303-41.

6.2 Определения типов и сущностей стандартной схемы языкового ресурса ISO13584_IEC61360_language_resource_schema

6.2.1 Общие положения

Данный подраздел содержит определения типов и сущностей языка EXPRESS для стандартной языковой схемы **ISO13584_IEC61360_language_resource_schema**.

6.2.2 Код языка (Language_code)

Сущность **language_code** дает возможность идентифицировать язык в соответствии с ИСО 639-1. Существует два кода языка:

- код языка в соответствии с ИСО 639-1 и ИСО 639-2, а также по выбору;
- код страны в соответствии с ИСО 3166-1, указывающий, в какой стране на данном языке говорят.

Пример представления на языке EXPRESS:

```
*)
ENTITY language_code; language_id:
      identifier;
      country_id: OPTIONAL identifier;
```

```
WHERE
     WR1: (LENGTH (language_id) = 2) OR (LENGTH (language_id) = 3);
     WR2: LENGTH (country_id) = 2;
END_ENTITY; -- language_code
(*
```

Определения атрибутов:

language_id: код, указывающий язык в соответствии с ИСО 639-1 и ИСО 639-2.

country_id: код, указывающий страну, где на данном языке говорят, в соответствии с ИСО 3166-1.

Пояснения к тексту программы:

WR1: длина кода **language_id** равна 2 или 3.

WR2: длина кода **country_id** равна 2.

6.2.3 Назначение глобального языка (Global_language_assignment)

Сущность **global_language_assignment** указывает язык для переводимой метки **translatable_label** и переводимого текста **translatable_text**, если и метка, и текст выбраны соответственно (т. е. без явного указания языка, как это сделано в атрибутах переведенная метка **translated_label** и переведенный текст **translated_text**).

Пример представления на языке EXPRESS:

```
*)
ENTITY global_language_assignment;
     language: language_code;
END_ENTITY; -- global_language_assignment
(*
```

Определения атрибутов:

language: код назначенного языка.

6.2.4 Доступные переводы (Present_translations)

Сущность **present_translations** служит для указания языков, использованных для получения переведенной метки **translated_label** и переведенного языка **translated_text**.

Пример представления на языке EXPRESS:

```
*)
ENTITY present_translations;
     language_codes: LIST [1:?] OF UNIQUE language_code;
UNIQUE
     UR1: language_codes;
END_ENTITY; -- present_translations
(*
```

Определения атрибутов:

language_codes: список уникальных кодов, соответствующих языкам, на которых сделан перевод.

Пояснение к тексту программы:

UR1: для каждого списка кодов языков **language_codes** существует уникальная реализация доступных переводов **present_translations**.

6.2.5 Переводимая метка (Translatable_label)

Сущность **translatable_label** определяет тип значения, которое может быть меткой или переведенной меткой **translated_labels**.

Пример представления на языке EXPRESS:

```
*)
TYPE translatable_label = SELECT(label, translated_label);
END_TYPE; -- translatable_label
(*
```

6.2.6 Переведенная метка (Translated_label)

Сущность **translated_label** определяет метки, которые уже переведены, и соответствующий язык перевода.

Пример представления на языке EXPRESS:

```
*)
ENTITY translated_label;
      labels: LIST [1:?] OF label;
      languages: present_translations;
WHERE
      WR1: SIZEOF(labels) = SIZEOF(languages.language_codes);
END_ENTITY; -- translated_label
(*
```

Определения атрибутов:

labels: список меток **labels**, которые уже переведены.

language: список языков, на которые переведены рассматриваемые метки.

Пояснение к тексту программы:

WR1: количество меток **labels**, содержащихся в списке **labels**, равно количеству языков, определяемому в атрибуте **languages.language_codes**.

Дополнительное пояснение:

IP1: содержание массива **labels[i]** на языке, идентифицированном массивом атрибутов **languages. language_codes[i]**.

6.2.7 Переводимый текст (Translatable_text)

Сущность **translatable_text** определяет тип значений, который может быть текстом или переведенным текстом.

Пример представления на языке EXPRESS:

```
*)
TYPE translatable_text = SELECT(text, translated_text);
END_TYPE; -- translatable_text
(*
```

6.2.8 Переведенный текст (Translated_text)

Сущность **translated_text** определяет переведенные тексты и соответствующие языки перевода.

Пример представления на языке EXPRESS:

```
*)
ENTITY translated_text;
      texts: LIST [1:?] OF text; languages:
      present_translations;
WHERE
      WR1: SIZEOF(texts) = SIZEOF(languages.language_codes);
END_ENTITY; -- translated_text
(*
```

Определения атрибутов:

texts: список переведенных текстов.

languages: список языков, на которые переведен каждый текст.

Пояснение к тексту программы:

WR1: количество текстов, содержащихся в списке текстов, равно количеству языков, определенных атрибутом **languages.language_codes**.

Дополнительное пояснение:

IP1: содержание массивов **texts[i]** на языке, идентифицированном атрибутом **languages. language_codes[i].**

6.3 Определения функций стандартной схемы языкового ресурса ISO13584_IEC61360_language_resource_schema

6.3.1 Общие положения

Данный подраздел содержит описание функции, на которую производится ссылка в разделах по месту для подтверждения непротиворечивости данных.

6.3.2 Функция проверки длины метки (Check_label_length)

Функция **check_label_length** проверяет тот факт, что ни одна из меток массива **l** не превышает длину, указанную атрибутом **l_length.**

Пример представления на языке EXPRESS:

```
*)
FUNCTION check_label_length(l: translatable_label; l_
     length: INTEGER): BOOLEAN;

IF 'ISO13584_IEC61360_LANGUAGE_RESOURCE_SCHEMA.TRANSLATED_LABEL'

     IN TYPEOF(l)
THEN
     REPEAT i :=1 TO SIZEOF(l.labels);
          IF LENGTH(l.labels[i]) > l_length
          THEN
               RETURN(FALSE);
          END_IF;
     END_REPEAT;

     RETURN(TRUE);

ELSE (* the argument l is a single string *)
     RETURN(LENGTH(l) <= l_length);
END_IF;
END_FUNCTION; -- check_label_length
(*
```

6.4 Определение правила стандартной схемы языкового ресурса ISO13584_IEC61360_language_resource_schema

Правило **single_language_assignment** подтверждает, что только один язык может быть назначен для использования в переводимых метках **translatable_label** и переводимых текстах **translatable_text.**

Пример представления на языке EXPRESS:

```
*)
RULE single_language_assignment FOR(global_language_assignment);
WHERE
     SIZEOF(global_language_assignment) <= 1;
END_RULE; -- single_language_assignment


END_SCHEMA; -- ISO13584_IEC61360_language_resource_schema
(*
```

7 Стандартная схема ограничений класса ISO13584_IEC61360_class_constraint_schema

7.1 Общие положения

Данный раздел определяет требования к схеме ограничений класса **class_constraint_schema.** Нижеследующее объявление языка EXPRESS представляет блок **ISO13584_IEC61360_class_constraint_schema** и задает необходимые внешние ссылки.

Пример представления на языке EXPRESS:

```
*)
SCHEMA ISO13584_IEC61360_class_constraint_schema;

REFERENCE FROM ISO13584_IEC61360_dictionary_schema (
      class_BSU,
      property_BSU,
      definition_available_implies,
      is_subclass,
      data_type,
      simple_type,
      complex_type,
      named_type,
      allowed_values_integer_types);

REFERENCE FROM ISO13584_extended_dictionary_schema
      (data_type_typeof,
      data_type_class_of,
      data_type_type_name);

REFERENCE FROM ISO13584_instance_resource_schema
      (Boolean_value,
      compatible_class_and_class,
      complex_value,
      dic_class_instance,
      entity_instance_value,
      int_level_spec_value,
      integer_value,
      level_spec_value,
      number_value,
      primitive_value,
      rational_value,
      real_level_spec_value,
      real_value,
      right_values_for_level_spec,
      same_translations,
      simple_value,
      string_value,
      translatable_string_value,
      translated_string_value,
      property_or_data_type_BSU);

REFERENCE FROM ISO13584_aggregate_value_schema
      (aggregate_entity_instance_value,
      list_value,
      set_value,
      bag_value,
      array_value,
      set_with_subset_constraint_value,
      compatible_complete_types_and_value);
(*
```

Примечание — Схематики, на которые производятся ссылки выше, описаны в следующих документах:
ISO13584_IEC61360_dictionary_schema МЭК 61360-2
(дублируется для удобства в 4.5 и далее)

ISO13584_extended_dictionary_schema	ИСО 13584-24:2003
ISO13584_instance_resource_schema	ИСО 13584-24:2003
ISO13584_aggregate_value_schema	ИСО 13584-25

7.2 Введение в стандартную схему ограничений класса ISO13584_IEC61360_class_constraint_schema

Схема **ISO13584_IEC61360_class_constraint_schema** задает конструктивы языка EXPRESS, позволяющие переопределить путем задания ограничений область значений данного свойства, если оно применяется в подклассе характеристического класса, где данное свойство определено как видимое. Данное ограничение должно явно описывать только те ограничения области значений, которые следуют из структуры класса.

Пример — В ИСО 13584-511 класс болтов/винтов с метрической резьбой определен следующим образом: «крепежный элемент с головкой, наружной резьбой, цилиндрическим телом, на котором резьба нарезана частично или полностью, и головкой с приспособлениями для завинчивания». Данный класс среди прочих имеет два свойства: тип головки и свойства головки. Область значений свойства тип головки — это неколичественный тип данных, включающий главным образом следующие значения: шестигранная_головка, восьмигранная_головка и круглая_головка. Свойство свойства головки является особенностью. Это означает, что она имеет тип данных item_class. Областью значений является класс головка, определяющий любой вид головки. Класс головка имеет несколько подклассов: шестигранная головка, ассоциированная со всеми свойствами, позволяющими описывать шестигранную головку (например, размер под ключ), и круглая головка, ассоциированная со всеми свойствами, позволяющими описывать круглую головку (например, диаметр головки).

Класс *болтов/винтов с метрической резьбой* имеет подкласс, называемый *винты с шестигранной головкой* и определяемый следующим образом: «крепежный элемент с метрической наружной резьбой, шестигранной головкой и резьбой, нарезанной до головки». Данный класс унаследует свойства: *тип головки* и *свойства головки*. Из определения подкласса *винтов с шестигранной головкой* ясно, что свойство *тип головки* может принимать только значение *шестигранная_головка*. Свойство *свойства головки* может быть только реализацией класса особенностей *шестигранная головка*. Однако указанные ограничения являются неявными: они просто фиксируются неформально в определении сущности.

Таким образом, указанные ограничения не являются компьютерными. Рассматриваемые ограничения, определенные стандартной схемой ограничений класса **ISO13584_IEC61360_class_constraint_schema**, позволяют сделать следующие два ограничения явными путем ассоциирования их с классом *винтов с шестигранной головкой*: (1) ограничение нумерации **enumeration_constraint** для свойства *тип головки* (допускающее только код *шестигранная_головка*) и (2) **subclass_constraint** для свойства *свойства головки* (допускающее только класс особенностей *шестигранная головка*).

Ограничения наследуются. Если свойство, область значений которого имеет одно ограничение в классе С, требует задания другого ограничения в подклассе С, то оба ограничения применяются совместно. Таким образом, реальная область значений в подклассе С — это пересечение двух областей, определенных двумя ограничениями. Предложенный механизм аналогичен механизму повторного определения типа на языке EXPRESS.

Данная схема позволяет выразить ограничения, применимые к типам данных системы типов стандартной словарной схемы **ISO13584_IEC61360_dictionary_schema**. По правилу для данных сущностей, ссылающихся на ограничение, каждое ограничение применимо к типу данных, к которому оно относится.

7.3 Определения сущностей стандартной схемы ограничений класса ISO13584_IEC61360_class_constraint_schema

7.3.1 Общие положения

Данный раздел определяет сущности стандартной схемы ограничений класса **ISO13584_IEC61360_class_constraint_schema**.

7.3.2 Ограничение (Constraint)

Сущность **constraint** позволяет дать определение ограничения.

Пример представления на языке EXPRESS:

```
*)
ENTITY constraint
ABSTRACT SUPERTYPE OF ( ONEOF (
                    property_constraint,
```

```
                    class_constraint));
      constraint_id: OPTIONAL constraint_identifier;
END_ENTITY; -- constraint
(*
```

Определения атрибутов:

constraint_id: атрибут **constraint_identifier**, задающий ограничения.

7.3.3 Ограничение свойства (Property_constraint)

Сущность **property_constraint** — это ограничение набора реализаций класса путем задания одного условия для области значений некоторого свойства данного класса.

Пример представления на языке EXPRESS:

```
*)
ENTITY property_constraint
ABSTRACT SUPERTYPE OF ( ONEOF (
                    integrity_constraint,
                    context_restriction_constraint))
SUBTYPE OF (constraint);
     constrained_property: property_BSU;
END_ENTITY; -- property_constraint
(*
```

Определения атрибутов:

constrained_property: базовая семантическая единица свойства **property_BSU**, для которой применяется ограничение.

7.3.4 Ограничение класса (Class_constraint)

Сущность **class_constraint** — это ограничение, накладывающее условие на допустимый набор реализаций класса. При этом рассматриваются несколько свойств или глобальных ограничений.

Пример представления на языке EXPRESS:

```
*)
ENTITY class_constraint
ABSTRACT SUPERTYPE OF (configuration_control_constraint)
SUBTYPE OF (constraint);
END_ENTITY; -- class_constraint
(*
```

7.3.5 Ограничение управления конфигурацией (Configuration_control_constraint)

Сущность **configuration_control_constraint** накладывает условие на набор реализаций, на которые производится ссылка. На эти реализации некоторая другая ссылающаяся реализация может ссылаться прямо или косвенно с помощью цепочки свойств. Ссылающаяся реализация — это любая реализация класса, ссылающаяся на ограничение управления конфигурацией **configuration_control_constraint** с помощью атрибута **constraint** или наследующая его в классе, обладающем данным свойством. Сущность **configuration_control_constraint** определяет вспомогательное предварительное условие **precondition** наложения ограничения на ссылающуюся реализацию. Она определяет также выходное условие **postcondition**, задающее допустимые наборы значений некоторых свойств реализаций класса, на которые производится ссылка.

Пример — Болтовое соединение состоит из нижеследующего набора крепежных элементов: один крепежный элемент с наружной резьбой, некоторое количество шайб и одна или несколько гаек. Существуют различные виды резьбы, включая резьбу винта-самореза, резьбу винта по дереву, метрическую наружную резьбу, метрическую внутреннюю резьбу, дюймовую внутреннюю резьбу и дюймовую наружную резьбу. Предположим, что нужно дать описание метрического болтового соединения. Нужно гарантировать, что какая бы ни была точностная структура соединения, и крепежный элемент с наружной резьбой, и все гайки данного соединения имеют метрическую резьбу. Для этого в классе метрических болтовых соединений задается ограничение управления конфигурацией configuration_control_constraint, гарантирующее, что любой крепежный элемент (соответствующий ISO 13584-511), на который производится ссылка некоторой реализацией данного класса (или любого его подкласса), либо должен принадлежать классам, в которых не задано значение свойства тип резьбы (например,

шайба), либо должен принадлежать классам, значения которых принадлежат паре (метрическая наружная резьба, метрическая внутренняя резьба).

Примечание 1 — Как предварительное условие **precondition**, так и выходное условие **postcondition** могут ограничивать только те свойства, данные которых имеют неколичественный кодовый тип **non_quantitative_code_type**. Такие свойства могут быть заданы для значений либо на уровне реализации, либо на уровне класса, если они объявлены как свойства со значением класса, т. е. свойства подкласса **sub_class_properties** в некотором классе.

Примечание 2 — Свойства, на которые производится ссылка в предварительном условии **precondition**, должны быть применимы в классе, ссылающемся на ограничение управления конфигурацией **configuration_control_constraint**.

Примечание 3 — В ограничении управления конфигурацией **configuration_control_constraint** используются фильтры **filters**. Они накладывают предварительное условие на ссылающуюся реализацию и задают ограничения на реализации, на которые производится ссылка.

Пример представления на языке EXPRESS:

```
*)
ENTITY configuration_control_constraint
SUBTYPE OF (class_constraint);
      precondition: SET [0:?] OF filter;
      postcondition: SET [1:?] OF filter;
END_ENTITY; -- configuration_control_constraint
(*
```

Определения атрибутов:

precondition: фильтры **filters** ссылающейся реализации для наложения ограничения.

Примечание 4 — Если набор фильтров пуст, то ограничение накладывается на любую ссылающуюся реализацию.

postcondition: фильтры **filters** реализации, на которую ссылаются, для получения разрешения на данную ссылку.

7.3.6 Фильтр (Filter)

Сущность **filter** — это ограничение нумерации **enumeration_constraint**, ограничивающее допустимую область свойства, тип данных которого либо неколичественный кодовый **non_quantitative_code_type**, либо неколичественный целый **non_quantitative_int_type**.

Пример представления на языке EXPRESS:

```
*)
ENTITY filter;
      referenced_property: property_BSU;
      domain: enumeration_constraint;
WHERE
      WR1: definition_available_implies (
            referenced_property,
            (('ISO13584_IEC61360_DICTIONARY_SCHEMA'
            +'.NON_QUANTITATIVE_CODE_TYPE') IN TYPEOF(
            referenced_property.
            definition[1]\property_DET.domain))
            OR
            (('ISO13584_IEC61360_DICTIONARY_SCHEMA'
            +'.NON_QUANTITATIVE_INT_TYPE') IN TYPEOF(
            referenced_property.
            definition[1]\property_DET.domain)));
      WR2: definition_available_implies (
            referenced_property,
            correct_constraint_type(domain,
            referenced_property.definition[1].domain));
END_ENTITY; -- filter
(*
```

Определения атрибутов:

referenced_property: свойство, область значений которого ограничена фильтром **filter**.

domain: ограничение нумерации **enumeration_constraint**, ограничивающее область значений свойства, на которое производится ссылка.

Пояснения к тексту программы:

WR1: тип данных ссылочного свойства **referenced_property** — это либо неколичественный кодовый **non_quantitative_code_type**, либо неколичественный целый **non_quantitative_int_type**.

WR2: сущность **domain** должна определять область значений, которая может ограничивать исходную область значений свойства.

7.3.7 Ограничение целостности (Integrity_constraint)

Сущность **integrity_constraint** — это особое ограничение свойства, которое объявляет явно, что для некоторого конкретного класса (как результат определения данного класса и всех его подклассов) только ограничение области значений, идентифицированное типом данных, допустимо для свойства.

Пример — В ссылочном словаре, определенном для крепежных элементов в ИСО 13584-511, болт/винт с метрической резьбой имеет свойство свойства головки, которое может приобретать в качестве значения член любого подкласса рассматриваемого класса особенностей головки. Если данный болт/винт с метрической резьбой также является членом подкласса винтов с шестигранной головкой, то свойство свойства головки может только быть членом класса особенностей шестигранной головки. В противном случае рассматриваемый болт/винт с метрической резьбой не может быть членом подкласса винтов с шестигранной головкой.

Примечание — В рассмотренном выше примере ограничение целостности никак не изменяет смысл свойства свойства головки, унаследованное от *болта/винта с метрической резьбой*, переходящее к *винту с шестигранной головкой*. Рассматриваемое ограничение явно указывает на тот факт, что в контексте подкласса *винтов с шестигранной головкой* допустимым остается только подмножество значений для данного свойства в контексте класса *болтов/винтов с метрической резьбой*.

Пример представления на языке EXPRESS:

```
*)
ENTITY integrity_constraint
SUBTYPE OF (property_constraint);
      redefined_domain: domain_constraint;
WHERE
      WR1: definition_available_implies (constrained_property,
           correct_constraint_type(redefined_domain,
           constrained_property.definition[1].domain));
END_ENTITY; -- integrity_constraint
(*
```

Определения атрибутов:

redefined_domain: ограничение, накладываемое на область значений ограниченного свойства.

Пояснение к тексту программы:

WR1: повторно определенная область **redefined_domain** должна задавать область значений, ограничивающую исходную область значений свойства.

7.3.8 Ограничение на условия контекста (Context_restriction_constraint)

Сущность **context_restriction_constraint** — это ограничение свойства **property_constraint**, накладывающее условие на допустимую область значений контекстных параметров, от которых зависит рассматриваемое контекстно-зависимое свойство.

Пример представления на языке EXPRESS:

```
*)
ENTITY context_restriction_constraint
SUBTYPE OF (property_constraint);
      context_parameter_constraints: SET [1:?] OF property_constraint;
WHERE
      WR1: definition_available_implies(constrained_property,
           QUERY (cp <*SELF.context_parameter_constraints
```

```
        : NOT (cp.constrained_property IN
        constrained_property.definition[1].depends_on))=[];
    WR2: QUERY (cp <*SELF.context_parameter_constraints
        : NOT (('ISO13584_IEC61360_CLASS_CONSTRAINT_SCHEMA'
        +'.INTEGRITY_CONSTRAINT') IN TYPEOF (cp))) =[];
    WR3:definition_available_implies(constrained_property,
        'ISO13584_IEC61360_DICTIONARY_SCHEMA.DEPENDENT_P_DET'
        IN TYPEOF(constrained_property.definition[1]));
END_ENTITY; -- context_restriction_constraint
(*
```

Определения атрибутов:

context_parameter_constraints: ограничение, применимое в области значений контекстных параметров.

Пояснения к тексту программы:

WR1: набор свойств, область значений которого ограничена свойством **context_parameter_constraint**, должен являться набором контекстных параметров, от которых зависит рассматриваемое ограниченное свойство.

WR2: все ограничения контекстных параметров **context_parameter_constraint** должны быть ограничениями целостности **integrity_constraints**.

WR3: ограниченное свойство **constrained_property** должно быть контекстно-зависимым типом данных **property dependent_P_DET**.

7.3.9 Ограничение области (Domain_constraint)

Сущность **domain_constraint** задает условие, ограничивающее область значений типа данных.

Пример представления на языке EXPRESS:

```
*)
ENTITY domain_constraint
ABSTRACT SUPERTYPE OF(ONEOF(
      subclass_constraint,
      entity_subtype_constraint,
      enumeration_constraint,
      range_constraint,
      string_size_constraint,
      string_pattern_constraint,

      cardinality_constraint
      ));
END_ENTITY; -- domain_constraint
(*
```

7.3.10 Ограничение подкласса (Subclass_constraint)

Сущность **subclass_constraint** ограничивает область значений типа ссылки на класс **class_reference_type** для одного или нескольких подклассов рассматриваемого класса, определяющего исходную область.

Пример представления на языке EXPRESS:

```
*)
ENTITY subclass_constraint
SUBTYPE OF(domain_constraint);
      subclasses: SET [1:?] OF class_BSU;
END_ENTITY; -- subclass_constraint
(*
```

Определения атрибутов:

subclasses: базовые семантические единицы класса **class_BSUs,** переопределяющие данный класс, которому должны принадлежать значения ограниченного свойства **constrained_property**.

7.3.11 Ограничение подтипа сущности (Entity_subtype_constraint)

Сущность **entity_subtype_constraint** ограничивает область значений типа реализации сущности **entity_instance_type** до подтипа СУЩНОСТИ, определяющей ее исходную область.

Пример представления на языке EXPRESS:

```
*)
ENTITY entity_subtype_constraint
SUBTYPE OF(domain_constraint);
      subtype_names: SET[1:?] OF STRING;
END_ENTITY; -- entity_subtype_constraint
(*
```

Определения атрибутов:

subtype_names: набор строк, описывающих (в формате функции типа TYPEOF языка EXPRESS) имена типов данных сущности языка EXPRESS, принадлежащих результату действия функции типа TYPEOF языка EXPRESS, примененной к значению, ссылающемуся на повторно определенное ограниченное свойство **constrained_properties**.

7.3.12 Ограничение нумерации (Enumeration_constraint)

Сущность **enumeration_constraint** ограничивает область значений типа данных до списка значений, определенных в расширении. Порядок, определенный списком, — это рекомендуемый порядок представления информации. Конкретное описание по желанию может быть ассоциировано с каждым значением подмножества с помощью неколичественного целого типа **non_quantitative_int_type**, *i*-е значение которого описывает смысл *i*-го значения подмножества.

Для подтипов **number_type**, ассоциированных со словарными единицами измерения **dic_unit** и альтернативными единицами измерения и, возможно, с идентификаторами словарных единиц измерения **dic_unit_identifier**, а также с идентификаторами альтернативных единиц измерения, данное ограничение применяется к значению, соответствующему словарной единице **dic_unit** или одному идентификатору словарной единицы **dic_unit_identifier**. Если существуют и сама единица, и ее идентификатор, то они соответствуют одной и той же единице измерения.

Для подтипов **number_type**, ассоциированных с валютой, ограничения накладываются на валюту, указанную в определении их типов данных. Если никакая валюта в определении типов данных не указана, то рассматриваемое ограничение не используется.

Для значений, принадлежащих переводимому строчному типу **translatable_string_type,** ограничения накладываются на строку, представленную на исходном языке, на котором определена область значений свойства. Данный исходный язык может быть определен атрибутом **source_language** административных данных **administrative_data** свойства. Если данный атрибут не существует, то указанный исходный язык считается известным пользователю словаря.

Если какое-либо другое ограничение нумерации **enumeration_constraint** накладывается на свойство, ранее уже ассоциированное с некоторым ограничением нумерации **enumeration_constraint** в некотором суперклассе, то применяются оба ограничения. Таким образом, допустимый набор значений — это пересечение двух подмножеств. Порядок представления информации и возможный смысл ограничения, ассоциированного с каждым значением для рассматриваемого ограничения нумерации **enumeration_constraint**, иллюстрируется примерами ниже.

Пример 1 — Если (в классе C1) данное свойство ассоциировано с ограничением нумерации enumeration_constraint, атрибут subset которого равен {1, 3, 5, 7}, то в классе C1 и в любом его подклассе свойство P1 может принимать только одно из четырех следующих значений: 1, 3, 5, 7.

Пример 2 — Если тип данных свойства P1 — это целочисленный массив LIST [1:4], и если в классе C1 данное свойство ассоциировано с ограничением нумерации enumeration_constraint, атрибут subset которого { {1}, {3, 5}, {7}, {1, 3, 7} }, то в классе C1 и в любом его подклассе свойство P1 может принимать только одно из четырех следующих значений: {1}, {3, 5}, {7}, {1, 3, 7}.

Пример представления на языке EXPRESS:

```
*)
ENTITY enumeration_constraint
SUBTYPE OF (domain_constraint);
```

```
        subset: LIST [1:?] OF UNIQUE primitive_value;
        value_meaning: OPTIONAL non_quantitative_int_type;
    WHERE
        WR1: (NOT(EXISTS(SELF.value_meaning)))
             OR
             (integer_values_in_range(1, SIZEOF(SELF.subset))
                  <= allowed_values_integer_types(SELF.value_meaning));
    END_ENTITY; -- enumeration_constraint
    (*
```

Определения атрибутов:

subset: список, описывающий подмножество значений, допустимых для свойства, идентифицированного атрибутом **constrained_properties**.

value_meaning: набор словарных значений **dic_value**, определяющих смысл каждого значения, принадлежащего списку **subset**.

Пояснение к тексту программы:

WR1: если существует неколичественный целый тип **non_quantitative_int_type** для смысла значения **value_codes**, то набор кодов **value_codes** его словарных значений **dic_value** должен принадлежать подмножеству 1.. SIZE_OF.

7.3.13 Ограничение диапазона (Range_constraint)

Сущность **range_constraint** ограничивает область значений упорядоченного типа подмножеством значений, определенных некоторым диапазоном.

Примечание 1 — Строка не рассматривается как упорядоченный тип и не может быть ограничена условием **range_constraint.**

Для подтипов **number_type**, ассоциированных со словарными единицами измерения **dic_unit**, альтернативными единицами измерения, а также, возможно, с идентификаторами словарных единиц измерения **dic_unit_identifier** и идентификаторами альтернативных единиц измерения, рассматриваемое ограничение накладывается на значение, соответствующее словарной единице **dic_unit** или одному инедтификатору словарной единицы **dic_unit_identifier.** Если существуют и словарная единица, и идентификатор, то они должны соответствовать одной и той же единице измерения.

Для подтипов **number_type,** ассоциированных с валютой, ограничение накладывается на валюту, указанную в определении их типов данных. Если никакая валюта в определении типа данных не указана, то ограничение не используется.

Примечание 2 — Для неколичественного целого типа данных **non_quantitative_int_type** ограничения накладываются на код значения **value_code.**

Пример представления на языке EXPRESS:

```
*)
ENTITY range_constraint
SUBTYPE OF (domain_constraint);
     min_value, max_value: OPTIONAL NUMBER; min_inclusive,
     max_inclusive: OPTIONAL BOOLEAN;
WHERE
     WR1: min_value <= max_value;
     WR2: TYPEOF(min_value) = TYPEOF(max_value);
     WR3: NOT EXISTS (min_value) OR EXISTS (min_inclusive);
     WR4: NOT EXISTS (max_value) OR EXISTS (max_inclusive);
END_ENTITY; -- range_constraint
(*
```

Определения атрибутов:

min_value: число, определяющее нижнюю границу диапазона значений; если это число не задано, то нижней границы нет.

max_value: число, определяющее верхнюю границу диапазона значений; если это число не задано, то верхней границы нет.

min_inclusive: указывает, действительно ли минимальное значение **min_value** принадлежит рассматриваемому диапазону; если такое значение не задано, то нижней границы нет.

max_inclusive: указывает действительно ли максимальное значение **max_value** принадлежит рассматриваемому диапазону; если такое значение не задано, то верхней границы нет.

Пояснения к тексту программы:

WR1: min_value должно быть меньше или равно **max_value**.

WR2: min_value и **max_value** должны иметь один и тот же тип данных.

WR3: если **min_value** задано, то **min_inclusive** также должно быть задано.

WR4: если **max_value** задано, то **max_inclusive** также должно быть задано.

7.3.14 Ограничение на длину строки (String_size_constraint)

Сущность **string_size_constraint** ограничивает длину строки, допустимую типом строки или ее подтипом.

Примечание 1 — Областью значений свойства **string_type** может быть строчный тип **string_type**, непереводимый строчный тип **non_translatable_string_type**, переводимый строчный тип **translatable_string_type**, тип универсального идентификатора ресурса **URI_type**, неколичественный кодовый тип **non_quantitative_code_type**, тип данных о дате **date_data_type**, тип данных о времени **time_data_type** или тип данных о дате и времени **date_time_data_type**.

Примечание 2 — Для неколичественного кодового типа **non_quantitative_code_type** ограничение накладывается на код.

Для значений, принадлежащих переводимому строчному типу **translatable_string_type**, ограничение накладывается на строку, представленную на исходном языке, на котором дано определение области значений свойств. Исходный язык может быть определен атрибутом **source_language** административных данных **administrative_data** свойства. Если данный атрибут не существует, то рассматриваемый исходный язык предполагается известным пользователю словаря.

Пример представления на языке EXPRESS:

```
*)
ENTITY string_size_constraint
SUBTYPE OF (domain_constraint);
     min_length: OPTIONAL INTEGER;
     max_length: OPTIONAL INTEGER;
WHERE
     WR1: (min_length >= 0) AND (max_length >= min_length);
END_ENTITY; -- string_size_constraint
(*
```

Определения атрибутов:

min_length: минимальная длина строки, допустимая в качестве значения свойства, идентифицированного атрибутом **constrained_properties**.

max_length: максимальная длина строки, допустимая в качестве значения свойства, идентифицированного атрибутом **constrained_properties**.

Примечание 3 — Если значение **min_length** не существует, то берется 0. Если значение **max_length** не существует, то берется бесконечность.

Пояснение к тексту программы:

WR1: значения **min_length** и **max_length** задают корректные границы.

7.3.15 Ограничение на шаблон строки (String_pattern_constraint)

Сущность **string_pattern_constraint** ограничивает область значений строчного типа **string_type** или любого его подтипа до значений, заданных конкретным шаблоном. Синтаксис шаблона определен регулярным выражением языка XML и ассоциированными алгоритмами сравнения, определенными в схеме XML, часть 2: «Рекомендованные типы данных».

Примечание 1 — Область значений свойства **string_type** — это строчный тип **string_type**, непереводимый строчный тип **non_translatable_string_type**, переводимый строчный тип **translatable_string_type**, тип универсального идентификатора ресурса **URI_type**, неколичественный кодовый тип **non_quantitative_code_type**, тип данных даты **date_data_type**, тип данных времени **time_data_type** или тип данных даты и времени **date_time_data_type**.

Для строчного типа **string_type**, непереводимого строчного типа **non_translatable_string_type**, типа универсального идентификатора ресурса **URI_type**, неколичественного кодового типа **non_quantitative_code_type**, типа данных о дате **date_data_type**, типа данных о времени **time_data_type** или типа данных о времени и дате **date_time_data_type** ограничение накладывается на (уникальную) строку, значения которой — это значение указанного типа данных. Для неколичественного кодового типа **non_quantitative_code_type** ограничение накладывается на код.

Для значений, принадлежащих переводимому строчному типу **translatable_string_type**, ограничение накладывается на строку, представленную на исходном языке, на котором определена область значений свойства. Исходный язык может быть определен атрибутом **source_language** административных данных **administrative_data** свойства. Если данный атрибут не существует, то данный исходный язык предполагается известным пользователю словаря.

Примечание 2 — Для неколичественного кодового типа **non_quantitative_code_type,** типа данных о дате **date_data_type,** типа данных о времени **time_data_type** или типа данных о времени и дате **date_time_data_type** шаблон должен соответствовать неформальным требованиям, определенным соответствующими типами данных.

Пример представления на языке EXPRESS:

```
*)
ENTITY string_pattern_constraint
SUBTYPE OF (domain_constraint);
     pattern: STRING;
END_ENTITY; -- string_pattern_constraint
(*
```

Определения атрибутов:

pattern: шаблон значений строки, допустимых в качестве значений свойства, идентифицированного атрибутом **constrained_property**.

Дополнительное пояснение к тексту программы:

IP1: синтаксис шаблона **pattern** должен удовлетворять требованиям синтаксиса регулярных выражений языка XML и ассоциированных алгоритмов сравнения, определенных схемой XML, часть 2: «Рекомендованные типы данных».

Пример — Шаблон схемы XML «[0-9]{4}\-[0-9]{2}\-[0-9]{2}» соответствует выражению языка структурированных запросов SQL «[0-9][0-9][0-9][0-9]-[0-9][0-9]-[0-9][0-9]». Он допускает сравнение строк типа «2009-05-3».

7.3.16 Cardinality_constraint

Сущность **cardinality_constraint** ограничивает кардинальное число комплексного типа данных.

Примечание 1 — Результирующий диапазон значений кардинального числа — это пересечение уже существующих диапазонов значений кардинального числа и диапазона, определенного атрибутом **cardinality_constraint.**

Примечание 2 — Ограничения **cardinality_constraint** недопустимы для типа данных **array_type.**

Пример представления на языке EXPRESS:

```
*)
ENTITY cardinality_constraint
SUBTYPE OF (domain_constraint);
     bound_1: OPTIONAL INTEGER;
     bound_2: OPTIONAL INTEGER;
WHERE
     WR1: (bound_1 >= 0) AND (bound_2 >= bound_1);
END_ENTITY; -- cardinality_constraint
(*
```

Определения атрибутов:

bound_1: нижняя граница кардинального числа.

bound_2: верхняя граница кардинального числа.

Примечание 3 — Если граница **bound_1** не задана, то минимальное кардинальное число равно 0. Если граница **bound_2** не задана, то максимальное кардинальное число не ограничено.

Пояснение к тексту программы:
WR1: атрибуты **bound_1** и **bound_2** задают корректные границы.

7.4 Определения типа стандартной схемы ограничений класса ISO13584_IEC61360_class_constraint_schema

7.4.1 Общие положения

Данный подраздел определяет типы стандартной схемы ограничений класса **ISO13584_IEC61360_class_constraint_schema**.

7.4.2 Ограничение и идентификатор ограничения (Constraint_or_constraint_id)

Сущность **constraint_or_constraint_id** задает либо само ограничение **constraint**, либо идентификатор ограничения **constraint_identifier**.

Пример представления на языке EXPRESS:

```
*)
TYPE constraint_or_constraint_id =
     SELECT (constraint, constraint_identifier);
END_TYPE; -- constraint_or_constraint_id
(*
```

7.5 Определения функций стандартной схемы ограничений класса ISO13584_IEC61360_class_constraint_schema

7.5.1 Общие положения

Данный подраздел определяет функции стандартной схемы ограничений класса **ISO13584_IEC61360_class_constraint_schema**.

7.5.2 Функция определения целого значения в диапазоне (Integer_value_in_range)

Функция **integer_value_in_range** вычисляет целое значение, принадлежащее диапазону целых значений, определенному своими нижней и верхней границами. Если границы диапазона не определены, то значение функции также не определено.

Пример представления на языке EXPRESS:

```
*)
FUNCTION integer_values_in_range(
     low_bound, high_bound: INTEGER): SET OF INTEGER;
LOCAL
     i: INTEGER;
     result: SET OF INTEGER:= [];
END_LOCAL;
     IF EXISTS (low_bound) AND EXISTS (high_bound)
     THEN
          REPEAT i := low_bound TO high_bound;
               result := result + [i];
          END_REPEAT;
          RETURN(result);
     ELSE
          RETURN(?);
     END_IF;
END_FUNCTION; -- integer_values_in_range
(*
```

7.5.3 Функция проверки предварительного условия (Correct_precondition)

Функция **correct_precondition** проверяет тот факт, что предварительное условие ограничения управления конфигурацией **configuration_control_constraint**, определенное атрибутом **cons**, исполь-

зует только свойства, применимые в классе **cl**. Функция возвращает логическое значение. Это значение неизвестно, если полный набор применимых свойств в классе не может быть вычислен.

Пример представления на языке EXPRESS:

```
*)
FUNCTION correct_precondition(
     cons: configuration_control_constraint; cl:class): LOGICAL;
LOCAL
     prop: SET OF property_BSU:= [];
END_LOCAL;
     REPEAT i := 1 to SIZEOF (cons.precondition);
          prop := prop + cons.precondition[i].referenced_property;
     END_REPEAT;

     IF prop <= cl.known_applicable_properties
     THEN RETURN (TRUE);
     ELSE
          IF all_class_descriptions_reachable(cl.identified_by)
          THEN RETURN (FALSE);
          ELSE RETURN (UNKNOWN);
          END_IF;
     END_IF;
END_FUNCTION; -- correct_precondition
(*
```

7.5.4 Функция проверки корректности типа ограничения (Correct_constraint_type)

Функция **correct_constraint_type** проверяет тот факт, что ограничение на область **domain_constraint**, определенное атрибутом **cons**, совместимо с типом данных **data_type**, определенным атрибутом **typ**. Функция возвращает логическое значение. Это значение неизвестно, если тип **domain_constraint**, определенный атрибутом **cons**, не является одним из подтипов, определенных стандартной схемой ограничений класса **ISO13584_IEC61360_class_constraint_schema**.

Пример представления на языке EXPRESS:

```
*)
FUNCTION correct_constraint_type(
     cons: domain_constraint; typ:data_type): LOGICAL;

(*case subclass constraint*)

IF ('ISO13584_IEC61360_CLASS_CONSTRAINT_SCHEMA'
     +'SUBCLASS_CONSTRAINT') IN TYPEOF(cons)

THEN
     (*the data type shall be class_reference_type*)
     IF NOT
          ('ISO13584_IEC61360_DICTIONARY_SCHEMA.CLASS_REFERENCE_TYPE'
          IN TYPEOF (typ))
     THEN RETURN(FALSE);
     END_IF;

     (*the cons.subclasses shall consist of subclasses for the class
     that defined the initial domain of typ.*)
     IF NOT (QUERY (sc <* cons.subclasses
               definition_available_implies
          (sc,definition_available_implies
```

```
          (typ\class_reference_type.domain,is_subclass(sc.definition[1]
          , typ\class_reference_type.domain.definition[1])))= false)
          = [])
     THEN RETURN(FALSE);
     END_IF;

     RETURN (TRUE);
END_IF;
(*case entity subtype constraint*)

IF (('ISO13584_IEC61360_CLASS_CONSTRAINT_SCHEMA'
     +'ENTITY_SUBTYPE_CONSTRAINT') IN TYPEOF (CONS))
THEN

(* the data type is a class_reference_type*)
     IF NOT (('ISO13584_IEC61360_DICTIONARY_SCHEMA'
     +'.ENTITY_INSTANCE_TYPE') IN TYPEOF (typ))
     THEN RETURN(FALSE);
     END_IF;
(* the subtype_name shall define a subtype for the entity_instance_type
of the constrained *)
     IF NOT (cons\entity_subtype_constraint.subtype_names
          >= typ\entity_instance_type.type_name)
     THEN RETURN(FALSE);
     END_IF;

RETURN (TRUE);
END_IF;

(*case enumeration_constraint *)

IF ('ISO13584_IEC61360_CLASS_CONSTRAINT_SCHEMA'
     +'.ENUMERATION_CONSTRAINT') IN TYPEOF (CONS)
THEN

(* all the values belonging to the subset of values shall be compatible
with the typ data type *)
     IF (QUERY (val<*cons.subset |
          NOT compatible_data_type_and_value ( typ, val))<> [])
     THEN RETURN(FALSE);
     END_IF;

RETURN (TRUE);
END_IF;

(*case range_constraint *)

IF ('ISO13584_IEC61360_CLASS_CONSTRAINT_SCHEMA.RANGE_CONSTRAINT'
       IN TYPEOF (CONS))
THEN

(*if the data type is an integer_type then min_value and max_value
shall be INTEGERs.*)
     IF ('ISO13584_IEC61360_DICTIONARY_SCHEMA.INTEGER_TYPE'
          IN TYPEOF (typ)) AND
```

```
            NOT ('INTEGER' IN TYPEOF (cons.min_value))
      THEN RETURN(FALSE);
      END_IF;

(*if the data type is a rational_type then min_value and max_value
shall be rational.*)
      IF ('ISO13584_IEC61360_DICTIONARY_SCHEMA.RATIONAL_TYPE'
          IN TYPEOF (typ)) AND
          NOT    ('ISO13584_INSTANCE_RESOURCE_SCHEMA.RATIONAL_VALUE'    IN
TYPEOF (cons.min_value))
      THEN RETURN(FALSE);
      END_IF;

(*if the data type is a real_type then min_value and max_value shall be
REALs.*)
      IF ('ISO13584_IEC61360_DICTIONARY_SCHEMA.REAL_TYPE'
          IN TYPEOF (typ)) AND NOT ('REAL' IN TYPEOF (cons.min_value))
      THEN RETURN(FALSE);
      END_IF;

(*all values of the range shall belong to the allowed values defined by
the type.*)
      IF (('ISO13584_IEC61360_DICTIONARY_SCHEMA'
          + '.NON_QUANTITATIVE_INT_TYPE') IN TYPEOF (typ))
      AND NOT
          (integer_values_in_range(cons.min_value, cons.max_value)
          <= allowed_values_integer_types (typ))
      THEN RETURN(FALSE);
      END_IF;

RETURN (TRUE);
END_IF;

(*case entity string_size_constraint*)

IF ('ISO13584_IEC61360_CLASS_CONSTRAINT_SCHEMA'
      +'.STRING_SIZE_CONSTRAINT') IN TYPEOF (CONS)
THEN

(* the data type shall be a string_type or any of its subtypes *)
IF NOT    ('ISO13584_IEC61360_DICTIONARY_SCHEMA.STRING_TYPE'
          IN TYPEOF (typ))
THEN RETURN(FALSE);
END_IF;

RETURN (TRUE);
END_IF;

(*case entity string_pattern_constraint *)

IF ('ISO13584_IEC61360_CLASS_CONSTRAINT_SCHEMA'
      +'.STRING_PATTERN_CONSTRAINT') IN TYPEOF (CONS)
THEN
(* the data type shall be a string_type or any of its subtypes *)
```

```
      IF NOT ('ISO13584_IEC61360_DICTIONARY_SCHEMA.STRING_TYPE'
           IN TYPEOF (typ))
      THEN RETURN(FALSE);
      END_IF;
RETURN (TRUE);
END_IF;

(*case entity cardinality_constraint *)

IF ('ISO13584_IEC61360_CLASS_CONSTRAINT_SCHEMA'
      +'.CARDINALITY_CONSTRAINT') IN TYPEOF (CONS)
THEN

(* the data type shall be an aggregate type but not an array*)
      IF (NOT(
      ('ISO13584_IEC61360_DICTIONARY_AGGREGATE_EXTENSION_SCHEMA'
      + '.ENTITY_INSTANCE_TYPE_FOR_AGGREGATE')
       IN TYPEOF(typ)))
      THEN
           RETURN(FALSE);
      END_IF;

      IF ('ISO13584_IEC61360_DICTIONARY_AGGREGATE_EXTENSION_SCHEMA'
           + '.ARRAY_TYPE' IN TYPEOF(typ.type_structure))
      THEN
           RETURN(FALSE);
      END_IF;
      RETURN (TRUE);
END_IF;

RETURN (UNKNOWN);

END_FUNCTION; -- correct_constraint_type
(*
```

7.5.5 Функция проверки совместимости типа данных и значения (Compatible_data_type_and_value)

Функция **compatible_data_type_and_value** проверяет тот факт, что тип значения **val** атрибута **primitive_value** совместим с типом, определенным атрибутом **dom**. Функция возвращает логическое значение, равное TRUE, если типы совместимы. Это значение равно FALSE, если типы несовместимы. Данная функция возвращает значение UNKNOWN, когда тип данных **val** соответствует типу неуправляемой реализации **uncontrolled_instance_value** (см. ИСО 13584-24:2003) или если рассматриваемый тип не относится к типам, определенным стандартной схемой ресурса реализаций **ISO13584_instance_resource_schema**.

Примечание — Значение переменной **val** может существовать и может не существовать.

Пример представления на языке EXPRESS:

```
*)
FUNCTION compatible_data_type_and_value(dom: data_type;
      val: primitive_value): LOGICAL;
LOCAL
      temp: class_BSU;
      set_string: SET OF STRING := [];
      set_integer: SET OF INTEGER := [];
      code_type: non_quantitative_code_type;
```

```
        int_type: non_quantitative_int_type;
    END_LOCAL;

    (* The following express statements deal with simple types *)

    IF ('ISO13584_INSTANCE_RESOURCE_SCHEMA.INTEGER_VALUE' IN TYPEOF(val))
    THEN
        IF ('ISO13584_IEC61360_DICTIONARY_SCHEMA.' +
            'NON_QUANTITATIVE_INT_TYPE' IN TYPEOF (dom))
        THEN
            set_integer := [];
            int_type := dom;
            REPEAT j := 1 TO SIZEOF(int_type.domain.its_values);
                set_integer := set_integer +
                    int_type.domain.its_values[j].value_code;
            END_REPEAT;

            RETURN(val IN set_integer);

        ELSE
            RETURN(('ISO13584_IEC61360_DICTIONARY_SCHEMA.INT_TYPE'
                IN TYPEOF (dom)) OR
                (('ISO13584_IEC61360_DICTIONARY_SCHEMA.NUMBER_TYPE'
                IN TYPEOF (dom))
                AND NOT(('ISO13584_IEC61360_DICTIONARY_SCHEMA.REAL_TYPE'
                IN TYPEOF (dom))
                OR ('ISO13584_IEC61360_DICTIONARY_SCHEMA.RATIONAL_TYPE'
                IN TYPEOF (dom)))));

        END_IF;
    END_IF;

    IF ('ISO13584_INSTANCE_RESOURCE_SCHEMA.REAL_VALUE' IN TYPEOF(val))
    THEN
        RETURN(('ISO13584_IEC61360_DICTIONARY_SCHEMA.REAL_TYPE'
            IN TYPEOF (dom)) OR
            (('ISO13584_IEC61360_DICTIONARY_SCHEMA.NUMBER_TYPE'
            IN TYPEOF (dom))
            AND NOT(('ISO13584_IEC61360_DICTIONARY_SCHEMA.INT_TYPE'
            IN TYPEOF (dom))
            OR ('ISO13584_IEC61360_DICTIONARY_SCHEMA.RATIONAL_TYPE'
                IN TYPEOF (dom)))));
    END_IF;

    IF ('ISO13584_INSTANCE_RESOURCE_SCHEMA.RATIONAL_VALUE' IN TYPEOF(val))
    THEN
        RETURN(('ISO13584_IEC61360_DICTIONARY_SCHEMA.RATIONAL_TYPE'
            IN TYPEOF (dom)) OR
            (('ISO13584_IEC61360_DICTIONARY_SCHEMA.NUMBER_TYPE'
            IN TYPEOF (dom))
            AND NOT(('ISO13584_IEC61360_DICTIONARY_SCHEMA.INT_TYPE'
            IN TYPEOF (dom))
            OR ('ISO13584_IEC61360_DICTIONARY_SCHEMA.REAL_TYPE'
                IN TYPEOF (dom)))));
    END_IF;

    IF ('ISO13584_INSTANCE_RESOURCE_SCHEMA.STRING_VALUE'
        IN TYPEOF(val))
```

```
THEN
     IF (('ISO13584_IEC61360_DICTIONARY_SCHEMA' +
          '.NON_QUANTITATIVE_CODE_TYPE') IN TYPEOF (dom))
     THEN
          set_string := [];
          code_type := dom;
          REPEAT j := 1 TO SIZEOF(code_type.domain.its_values);
               set_string := set_string +
                    code_type.domain.its_values[j].value_code;
          END_REPEAT;

               RETURN(val IN set_string);

     ELSE
          RETURN('ISO13584_IEC61360_DICTIONARY_SCHEMA' +
               '.STRING_TYPE' IN TYPEOF (dom));

     END_IF;
END_IF;

IF ('ISO13584_INSTANCE_RESOURCE_SCHEMA.TRANSLATED_STRING_VALUE'
     IN TYPEOF(val))
THEN
          RETURN('ISO13584_IEC61360_DICTIONARY_SCHEMA' +
               '.TRANSLATABLE_STRING_TYPE' IN TYPEOF (dom));
END_IF;

(* The following express statements deal with complex types *)

IF 'ISO13584_INSTANCE_RESOURCE_SCHEMA.DIC_CLASS_INSTANCE'
     IN TYPEOF(val)
THEN
     IF ('ISO13584_IEC61360_DICTIONARY_SCHEMA.CLASS_REFERENCE_TYPE'
          IN TYPEOF (dom))
     THEN
          temp := dom.domain;
          RETURN(compatible_class_and_class(temp,
               val\dic_class_instance.class_def));
     ELSE
          RETURN(FALSE);
     END_IF;
END_IF;

IF 'ISO13584_INSTANCE_RESOURCE_SCHEMA.LEVEL_SPEC_VALUE' IN TYPEOF(val)
THEN
     IF ('ISO13584_IEC61360_DICTIONARY_SCHEMA.LEVEL_TYPE'
          IN TYPEOF (dom))
     THEN
          RETURN(compatible_level_type_and_instance(
               dom.levels,
               TYPEOF(dom.value_type),
               val));
     ELSE
          RETURN(FALSE);
     END_IF;
END_IF;
```

```
(* The following express statements deal with aggregate types *)

IF 'ISO13584_AGGREGATE_VALUE_SCHEMA.AGGREGATE_ENTITY_INSTANCE_VALUE'
IN TYPEOF(val) THEN

     IF (NOT(
     'ISO13584_IEC61360_DICTIONARY_SCHEMA.ENTITY_INSTANCE_TYPE' IN
     TYPEOF(dom)))
     THEN
          RETURN(FALSE);
     END_IF;

     IF (NOT(
          'ISO13584_IEC61360_DICTIONARY_AGGREGATE_EXTENSION_SCHEMA'
     + '.AGGREGATE_TYPE' IN dom.type_name))
     THEN
          RETURN(FALSE);
          END_IF;

     RETURN(compatible_aggregate_type_and_value(dom, val));

END_IF;

IF 'ISO13584_INSTANCE_RESOURCE_SCHEMA.ENTITY_INSTANCE_VALUE'
     IN TYPEOF(val)
THEN
     IF 'ISO13584_INSTANCE_RESOURCE_SCHEMA' +
          '.UNCONTROLLED_ENTITY_INSTANCE_VALUE'
          IN TYPEOF(val)
     THEN
          RETURN(UNKNOWN);
     END_IF;
     IF ('ISO13584_IEC61360_DICTIONARY_SCHEMA.ENTITY_INSTANCE_TYPE'
          IN TYPEOF (dom))
          AND (dom.type_name <- TYPEOF(val))
     THEN
          RETURN(TRUE);
     ELSE
          RETURN(FALSE);
     END_IF;
END_IF;

RETURN(UNKNOWN);

END_FUNCTION; -- compatible_data_type_and_value
(*
```

7.6 Определение правил стандартной схемы ограничений класса ISO13584_IEC61360_class_constraint_schema

7.6.1 Общие положения

Данный подраздел определяет правила стандартной схемы ограничений класса **ISO13584_IEC61360_class_constraint_schema**.

7.6.2 Уникальный идентификатор ограничения (Unique_constraint_id)

Правило **unique_constraint_id** подтверждает, что два идентификатора ограничения **constraint_identifier**, ассоциированные с двумя различными ограничениями, имеют различные значения.

Пример представления на языке EXPRESS:

```
*)
RULE unique_constraint_id FOR(constraint);
WHERE
     QUERY(c1 <* constraint |
          SIZEOF(QUERY(c2 <* constraint |
          c1.constraint_id = c2.constraint_id))>1) = [];
END_RULE; -- unique_constraint_id
(*

*)
END_SCHEMA; -- ISO13584_IEC61360_class_constraint_schema

(*
```

8 Стандартная условная схема класса элементов ISO13584_IEC61360_item_class_case_of_schema

8.1 Краткое описание

Данный раздел определяет требования к условной схеме класса элементов **ISO13584_IEC61360_item_class_case_of_schema**. Нижеследующая декларация языка EXPRESS представляет блок схемы **ISO13584_IEC61360_item_class_case_of_schema** и задает необходимые внешние ссылки.

Пример представления на языке EXPRESS:

```
*)
SCHEMA ISO13584_IEC61360_item_class_case_of_schema;

REFERENCE FROM ISO13584_IEC61360_dictionary_schema
     (all_class_descriptions_reachable,
     class,
     class_BSU,
     data_type_BSU,
     item_class,
     property_BSU);

REFERENCE FROM ISO13584_IEC61360_class_constraint_schema

     (constraint,
     integrity_constraint,
     context_restriction_constraint,
     property_constraint,
     domain_constraint);

REFERENCE FROM ISO13584_extended_dictionary_schema
     (document_BSU,
     table_BSU,
     visible_properties,
     applicable_properties,
     visible_types,
     applicable_types,
     data_type_named_type);

(*
```

Примечание — Схематики, на которые производятся ссылки выше, можно найти в следующих документах:

ISO13584_IEC61360_dictionary_schema	МЭК 61360-2
(схема дублирована для удобства в настоящем стандарте).	
ISO13584_IEC61360_class_constraint_schema	МЭК 61360-2
(схема дублирована для удобства в настоящем стандарте).	
ISO13584_extended_dictionary_schema	ИСО 13584-24:2003

8.2 Введение в стандартную условную схему класса элементов ISO13584_IEC61360_item_class_case_of_schema

Полная общая словарная модель ИСО/МЭК поделена между несколькими документами по принципу блочности. Модель ядра онтологии продукта определена в МЭК 61360-2 и дублирована в настоящем стандарте. Ресурс расширения данной модели, включая представление реализации, представление документа, функциональную модель, функциональные виды и представление таблицы, определен в ИСО 13584-24:2003. Различные стандартные уровни практической реализации полной стандартной модели ИСО/МЭК, называемые классами соответствия, определены в ИСО 13584-25 и дублируются (для справочных целей) в МЭК 61360-5. Первый уровень точно соответствует содержанию настоящего стандарта главным образом в части ресурса для агрегатно-структурированных значений, определенных в ИСО 13584-25. Прочие классы соответствия включают главным образом ресурсы из ИСО 13584-24:2003.

Определение класса элементов как другого условного класса элементов все чаще используется в приложениях, основанных на общей словарной модели ИСО 13584/МЭК 61360. Более того, данная версия настоящего стандарта требует изменения информационной модели данного понятия. Поэтому принято решение перенести соответствующую сущность языка EXPRESS, называемую условный класс элементов **item_class_case_of**, и ее суперкласс, называемый априорным семантическим соотношением **a_priori_semantic_relationship**, из ИСО 13584-24:2003 в настоящий стандарт. Указанные сущности включены в новую схему, называемую стандартной условной схемой класса элементов **ISO13584_IEC61360_item_class_case_of_schema**. ИСО 13584-24:2003 и ИСО 13584-25 обновляются путем внесения технических поправок в установленном порядке.

8.3 Определения сущностей стандартной условной схемы класса элементов ISO13584_IEC61360_item_class_case_of_schema

8.3.1 Априорное семантическое соотношение

Априорное семантическое соотношение **a_priori_semantic_relationship** — это абстрактный класс, определенный на базе других классов. Данный класс может импортировать свойства, типы данных, таблицы и документы, содержащиеся в указанных классах. Он также импортирует все ограничения, накладывающие условия на область импортированных свойств в классе, из которого они импортированы. Данный абстрактный ресурс разбивается классами на подтипы. Если класс специализирует некоторое априорное семантическое соотношение **a_priori_semantic_relationship**, то свойства, типы данных, таблицы или документы, определения которых импортированы по наследству с помощью **a_priori_semantic_relationship,** становятся применимыми в классе, который их импортирует. В частности, свойства и типы данных, импортированные таким образом, могут использоваться для описаний реализаций класса. Факт наличия аспекта (продукта), соответствующего каждому импортированному свойству, является необходимым критерием членства в классе.

Примечание 1 — Все импортированные свойства и типы данных становятся непосредственно применимыми без обеспечения их видимости. Таким образом, они не возвращаются функцией вычисления известных видимых свойств **compute_known_visible_properties** или функцией вычисления известных видимых типов данных **compute_known_visible_data_type**.

Примечание 2 — Соотношение наследственности — это хорошо известный пример установления семантического соотношения между классами, моделируемыми в соответствии с имеющейся объектно-ориентированной парадигмой. Все свойства и прочие ресурсы, определеные в классе, обычно задаются неявно во всех своих подклассах. Данное соотношение используется в стандартах серии ИСО 13584, где все свойства, типы данных, таблицы или документы, видимые (и, соответственно, применимые) в некоторых классах, являются неявно видимыми (и, соответственно, применимыми) во всех своих подклассах. Как обычно, в ИСО 13584 данное наследование является неявным (т. е. не объявлено с помощью априорного семантического соотношения **a_priori_semantic_relationship**) и глобальным (т. е. все свойства и типы данных унаследованы всеми его подклассами). Априорное семантическое соотношение **a_priori_semantic_relationship** дает возможность определять прочие семантические соотношения, используемые в области приложений ИСО 13584, и, в частности, условные соотношения, допускающие явное и частичное импортирование свойств и прочих ресурсов, определенных в классе.

Пример представления на языке EXPRESS:

```
*)
ENTITY a_priori_semantic_relationship
ABSTRACT SUPERTYPE
SUBTYPE OF(class);
      referenced_classes: SET [1:?] OF class_BSU;
      referenced_properties: LIST [0:?] OF property_BSU;
      referenced_data_types: SET [0:?] OF data_type_BSU;
      referenced_tables: SET [0:?] OF table_BSU;
      referenced_documents: SET [0:?] OF document_BSU;
      referenced_constraints: SET [0:?] OF constraint_or_constraint_id;
WHERE
      WR1: QUERY (cons <* SELF.referenced_constraints
           | NOT(('ISO13584_IEC61360_DICTIONARY_SCHEMA' +
           '.ISO_29002_IRDI_type') IN TYPEOF(cons))
           AND NOT (('ISO13584_IEC61360_CLASS_CONSTRAINT_SCHEMA'
           +'.PROPERTY_CONSTRAINT') IN TYPEOF (cons)))
           - [];
      WR2: QUERY (cons <* SELF.referenced_constraints
           | (('ISO13584_IEC61360_CLASS_CONSTRAINT_SCHEMA'
           +'.PROPERTY_CONSTRAINT') IN TYPEOF (cons))
           AND NOT (cons\property_constraint.constrained_property
           IN SELF.referenced_properties))
           - [];
      WR3: compute_known_referenced_property_constraints(SELF)
           <- SELF.referenced_constraints;
      WR4: QUERY(prop <* SELF.referenced_properties
           | QUERY(cl <* SELF.referenced_classes
           | visible_properties(cl, [prop])
           OR applicable_properties(cl, [prop]))
           - [:) - [];
      WR5: QUERY(typ <* SELF.referenced_data_types
           | QUERY(cl <* SELF.referenced_classes
           | visible_types(cl, [typ])
           OR applicable_types(cl, [typ]))
           - [:) - [];
END_ENTITY; -- a_priori_semantic_relationship
(*
```

Определения атрибутов:

referenced_classes: классы, откуда импортируются свойства, типы данных, таблицы и документы.

Примечание 3 — Класс, из которого импортируются свойства, типы данных, таблицы и документы, не может быть выведен по идентификации импортированных свойств, типов данных, таблиц и документов потому, что они сами могут быть импортированы из класса, где они унаследованы или импортированы. Например, по МЭК 61360-DB «входное напряжение» — это свойство, видимое на корневом уровне классификации Международной электротехнической комиссии (МЭК). Если поставщик класса импортирует свойство «входное напряжение» из класса «транзисторы» МЭК, то: (1) поставщик класса определяет транзистор, (2) указанные транзисторы описаны с помощью свойства «входное напряжение».

referenced_properties: свойства, определения которых импортированы с помощью сущности **a_priori_semantic_relationship**.

Примечание 4 — Списочный порядок определяет порядок по умолчанию для представления импортированных свойств во время процесса обеспечения доступа пользователя к различным подтипам априорного семантического соотношения **a_priori_semantic_relationship.**

referenced_data_type: типы данных, определения которых импортированы с помощью сущности **a_priori_semantic_relationship**.

referenced_tables: таблицы, определения которых импортированы с помощью сущности **a_priori_semantic_relationship**.

Примечание 5 — Подробные описания ресурсов и правил пользования таблицами даны в ИСО 13584-24:2003. Они не используются ни в настоящем стандарте, ни в интегрированных моделях, задокументированных в ИСО 13584-32 (OntoML) и ИСО 13584-25.

referenced_document: документы, определения которых импортированы с помощью сущности **a_priori_semantic_relationship**.

Примечание 6 — Составные части описания ресурсов и правил пользования документами определены в ИСО 13584-24:2003. Они используются в интегрированных моделях, задокументированных в ИСО 13584-32 (OntoML) и ИСО 13584-25.

referenced_constraint: ссылочные ограничения свойств **property_constraints**, накладываемые на различные импортированные свойства.

Примечание 7 — В отличие от других ссылочных сущностей ссылочные ограничения **referenced_constraints** не могут быть выбраны при разработке априорного семантического соотношения **a_priori_semantic_relationship**. Указанные ограничения — это все ограничения для всех свойств, определенных атрибутом **referenced_properties** в любом классе атрибута **referenced_class**.

Пояснения к тексту программы:

WR1: все ссылочные ограничения **referenced_constraints**, не относящиеся к IRDI, должны быть ограничениями свойств **property_constraints**.

WR2: все ссылочные ограничения **referenced_constraints** должны ограничивать свойства, импортированные с помощью атрибута **referenced_properties**.

WR3: все ограничения свойств **property_constraints**, ограничивающие одно из ссылочных свойств **referenced_properties** в любом ссылочном классе **referenced_classes**, должны быть импортированы с помощью атрибута **referenced_constraints**.

WR4: импортированные свойства, определенные атрибутом **referenced_properties**, должны быть видимыми или применимыми в одном из классов, принадлежащих атрибуту **referenced_classes**.

WR5: импортированные типы, определенные атрибутом **referenced_data_types**, должны быть видимыми или применимыми в одном из классов, принадлежащих атрибуту **referenced_classes**.

Дополнительные пояснения:

IP1: все ограничения, представленные идентификаторами **constraint_identifiers** в наборе **referenced_constraints**, должны соответствовать требованиям ограничений свойств **property_constraints**, наложенных на одно из ссылочных свойств **referenced_properties** в одном из рассматриваемых ссылочных классов **referenced_classes**. Такие ограничения не могут быть представлены (в том же наборе ссылочных ограничений **referenced_constraint**) одновременно и как ограничение свойства **property_constraint**, и как идентификатор ограничения **constraint_identifier**.

Примечание 8 — Ограничение, представленное как ограничение свойства **property_constraint** в одном из ссылочных классов **referenced_classes,** может быть представлено в наборе ссылочных ограничений **referenced_constraints** либо как ограничение свойства **property_constraint,** либо как идентификатор ограничения **constraint_identifier.**

IP2: все ограничения, представленные идентификатором ограничения **constraint_identifier** в одном из рассматриваемых ссылочных классов **referenced_classes** (с соответствующим ограничением свойства **property_constraint**, наложенным на одно из свойств, импортированных с помощью атрибута **referenced_properties**), должны быть представлены идентификатором ограничения **constraint_identifier** в наборе ссылочных ограничений **referenced_constraints**.

Примечание 9 — Эти два неформальных правила гарантируют, что рассматриваемый набор ссылочных ограничений **referenced_constraints** является объединением наборов ограничений свойств **property_constraint,** определенных в различных ссылочных классах **referenced_classes** с ограниченными свойствами **constrained_property,** принадлежащими набору ссылочных свойств **referenced_properties,** даже если контекст обмена не содержит определений всех классов, привлеченных к априорному семантическому соотношению **a_priori_semantic_relationship,** и если некоторые ограничения представлены только их идентификаторами **constraint_identifier.**

8.3.2 Условный класс элементов (Item_class_case_of)

Сущность **item_class_case_of** дает описание класса элементов, определенного как условная комбинация некоторых других классов элементов.

Примечание 1 — Сущность **item_class_case_of** определяет априорное семантическое соотношение.

Пример представления на языке EXPRESS:

```
*)
ENTITY item_class_case_of
SUBTYPE OF(item_class, a_priori_semantic_relationship);
      is_case_of: SET [1:?] OF class_BSU;
      imported_properties: LIST [0:?] OF property_BSU;
      imported_types: SET [0:?]OF data_type_BSU;
      imported_tables: SET [0:?] OF table_BSU;
      imported_documents: SET [0:?] OF document_BSU;
      imported_constraints: SET [0:?] OF constraint_or_constraint_id;
DERIVE
      SELF\a_priori_semantic_relationship.referenced_classes:
           SET [1:?] OF class_BSU := SELF.is_case_of;
      SELF\a_priori_semantic_relationship.referenced_properties:
           LIST [0:?] OF property_BSU := SELF.imported_properties;
      SELF\a_priori_semantic_relationship.referenced_data_types:
           SET [0:?] OF data_type_BSU := SELF.imported_types;
      SELF\a_priori_semantic_relationship.referenced_tables:
           SET [0:?] OF table_BSU := SELF.imported_tables;
      SELF\a_priori_semantic_relationship.referenced_documents:
           SET [0:?] OF document_BSU := SELF.imported_documents;
      SELF\a_priori_semantic_relationship.referenced_constraints:
           SET [0:?] OF property_constraint
           := SELF.imported_constraints;
WHERE
      WR1: superclass_of_item_is_item(SELF);
      WR2: check_is_case_of_referenced_classes_definition(SELF);
      WR3: QUERY(p <* SELF\class.sub_class_properties
           | NOT((p IN SELF.described_by)
           OR (p IN SELF.imported_properties))) = [];
      WR4: QUERY(p <* SELF\class.sub_class_properties
           | (p IN SELF.imported_properties)
           AND (QUERY(cl<*SELF.is_case_of
           | all_class_descriptions_reachable(cl) AND
           (p IN compute_known_applicable_properties(cl)) AND
           (NOT is_class_valued_property(p, cl)))<>[]))
           =[];
      WR5: QUERY(ccv <* SELF\class.class_constant_values
           | (ccv.super_class_defined_property
           IN SELF.imported_properties)
           AND (QUERY(cl<*SELF.is_case_of
           | all_class_descriptions_reachable(cl) AND
           (ccv.super_class_defined_property
            IN compute_known_applicable_properties(cl)) AND
           (QUERY (v<*class_value_assigned(
           ccv.super_class_defined_property, cl)
           |v<> ccv.assigned_value) <> [])<>[]))
           =[];
      WR6: QUERY(prop <* imported_properties
           | (QUERY(cl<*SELF.is_case_of
           | is_class_valued_property(prop, cl)) <>[])
           AND NOT is_class_valued_property(prop, SELF.identified_by))
           =[];
      WR7: QUERY(ccv <* SELF\class.class_constant_values
           | QUERY(cl<*SELF.is_case_of
```

```
            : (class_value_assigned
            (ccv.super_class_defined_property, cl) <> [])
            AND (QUERY(v <* class_value_assigned
            (ccv.super_class_defined_property, cl)
            : v <> ccv. assigned_value)<>[])) <> [])
            -[];
END_ENTITY; -- item_class_case_of
(*
```

Определения атрибутов:

is_case_of: классы элементов **item_class(es)**, из которых комбинируется рассматриваемый класс элементов **item_class**.

imported_properties: список свойств, импортированных из классов элементов **item_class(es)**, из которых комбинируется рассматриваемый класс элементов **item_class**.

imported_types: набор типов данных, импортированных из классов элементов **item_class(es)**, из которых комбинируется рассматриваемый класс элементов **item_class**.

imported_tables: набор базовых семантических единиц таблиц **table_BSUs**, импортированных из классов элементов **item_class(es)**, из которых комбинируется рассматриваемый **item_class**.

imported_documents: набор базовых семантических единиц документов **document_BSUs**, импортированных из классов элементов **item_class(es)**, из которых комбинируется рассматриваемый **item_class**.

imported_constraints: набор ограничений свойств **property_constraint** или идентификаторов ограничений **constraint_id**, импортированных из классов элементов **item_class(es)**, из которых комбинируется рассматриваемый **item_class**.

Примечание 2 — В отличие от других импортированных сущностей, импортированные ограничения **imported_constraints** не могут быть выбраны при разработке условного класса элементов **item_class_case_of**. Указанные ограничения — это все ограничения, наложенные на область значений любого из свойств, определенного импортированными свойствами **imported_properties** в классах атрибута **is_case_of**, из которых они импортированы. Это определено областью применения априорного семантического соотношения **a_priori_semantic_relationship**.

Пояснения к тексту программы:

WR1: суперклассом для условного класса элементов **item_class_case_of** должен быть класс элементов **item_class**.

WR2: условный класс элементов **item_class_case_of** должен быть комбинацией классов элементов **item_class(es)**.

WR3: свойства подкласса **sub_class_properties** должны принадлежать либо списку описаний **described_by**, либо списку импортированных свойств **imported_properties**.

WR4: все свойства со значением класса, объявленные с помощью свойств подкласса **sub_class_properties** (являющихся импортированными свойствами **imported_properties**), должны быть свойствами со значением класса во всех комбинируемых классах, где они применимы.

WR5: значения, заданные импортированному свойству с помощью атрибута значения константы класса **class_constant_value**, не должны различаться более чем на возможное значение, заданное для того же самого свойства в рассматриваемых ссылочных классах.

WR6: все импортированные свойства **imported_properties**, являющиеся свойствами со значением класса, являющегося условной комбинацией других классов, должны быть свойствами со значением класса в рассматриваемом классе.

WR7: всем импортированным свойствам **imported_properties** со значением константы класса **class_constant_value** из условной комбинации других классов должно быть задано одно и то же значение константы класса **class_constant_value** в рассматриваемом классе.

8.4 Определения функций стандартной схемы условного класса элементов ISO13584_IEC61360_item_class_case_of_schema

8.4.1 Общие положения

Данный подраздел содержит описания функций, на которые производится ссылка в разделах по месту для подтверждения непротиворечивости данных или для создания ресурсов приложений.

8.4.2 Функция вычисления ограничения известного свойства (Compute_known_property_constraints)

Функция **compute_known_property_constraints** вычисляет набор ограничений свойств **property_constraints**, применимых для свойств набора классов. Ограничения, представленные их идентификаторами, не вычисляются. Если определение класса недоступно, то функция возвращает только те ограничения свойств **property_constraints**, которые могут быть вычислены.

Примечание — Если словарное определение **dictionary_definition** класса недоступно в рассматриваемом контексте обмена (контекст обмена библиотеки PLIB никогда не предполагается полным), то собственный суперкласс может быть неизвестен. Следовательно, ограничения, определенные данным суперклассом, не могут быть вычислены функцией **compute_known_property_constraint**. И наоборот, если все представительные суперклассы рассматриваемого класса доступны в том же контексте обмена, то все ограничения, применимые в данном классе, могут быть вычислены за один обход представительного дерева наследственности, даже если некоторые из указанных суперклассов импортируют свойства с помощью априорного семантического соотношения **a_priori_semantic_relationship**, такого как условный класс элементов **item_class_case_of**.

Пример представления на языке EXPRESS:

```
*)
FUNCTION compute_known_property_constraints(classes: SET OF class_BSU):
     SET OF property_constraint;

LOCAL
     s: SET OF property_constraint := [];
END_LOCAL;

REPEAT nb := 1 TO SIZEOF (classes);
     IF SIZEOF(classes[nb].definition)=1
     THEN
          REPEAT i := 1 TO
               SIZEOF(classes[nb].definition[1]\class.constraints);
               IF (('ISO13584_IEC61360_CLASS_CONSTRAINT_SCHEMA'
               +'.PROPERTY_CONSTRAINT')

               IN TYPEOF
                    (classes[nb].definition[1]\class.constraints[i]))
               THEN
               s := s + classes[nb].definition[1]\class.constraints[i];
               END_IF;
          END_REPEAT;

          IF (('ISO13584_IEC61360_ITEM_CLASS_CASE_OF_SCHEMA.'
               + 'A_PRIORI_SEMANTIC_RELATIONSHIP')
               IN TYPEOF (classes[nb].definition[1]))

          THEN
               REPEAT i := 1 TO
               SIZEOF(classes[nb].definition[1]
               \a_priori_semantic_relationship
               .referenced_constraints);

                    IF (('ISO13584_IEC61360_CLASS_CONSTRAINT_SCHEMA'
                         +'.PROPERTY_CONSTRAINT') IN TYPEOF
                         (classes[nb].definition[1]
                         \a_priori_semantic_relationship
                         .referenced_constraints[i]))
```

```
                    THEN
                        s := s + classes[nb].definition[1]
                        \a_priori_semantic_relationship
                        .referenced_constraints [i];
                    END_IF;
                END_REPEAT;
            END_IF;

            IF EXISTS(classes[nb].definition[1]\class.its_superclass)
            THEN
                s := s + compute_known_property_constraints(
                    [classes[nb].definition[1]\class.its_superclass]);
            END_IF;

        END_IF;
    END_REPEAT;
    RETURN(s);

    END_FUNCTION; -- compute_known_property_constraints
    (*
```

8.4.3 Функция вычисления известного ссылочного ограничения свойства (Compute_known_referenced_property_constraint)

Функция **compute_known_referenced_property_constraint** вычисляет все ограничения свойства **property_constraint**, импортируемые априорным семантическим соотношением (**ap**) **a_priori_semantic_relationship**, путем вычисления всех ограничений, применимых к свойству, импортированному с помощью атрибута ссылочных свойств **referenced_properties** (из **ap**), определенному или унаследованному в любом ссылочном классе (**ap**) **referenced_class**, словарное определение **dictionary_definition** которого доступно в рассматриваемом контексте обмена.

Примечание 1 — В априорном семантическом соотношении **a_priori_semantic_relationship** все ограничения свойств **property_constraints**, определенные (унаследованные) классами, на которые производится ссылка атрибутом **referenced_class**, применяемом к свойству, импортированному с помощью атрибута **referenced_properties** априорного семантического соотношения **a_priori_semantic_relationship**, должны быть импортированы с помощью своего атрибута ссылочного ограничения **referenced_constraints**.

Примечание 2 — Если словарное определение **dictionary_definition** класса, принадлежащее атрибуту **referenced_class** сущности **ap**, не является доступным в рассматриваемом контексте обмена как сущность **ap** (контекст обмена библиотеки PLIB никогда не предполагается полным), то ограничения, принадлежащие данному классу, не могут быть вычислены. Таким образом, результат функции **compute_known_referenced_property_constraint** может быть только подмножеством ограничений, импортируемых сущностью **ap**.

Примечание 3 — Если словарные определения **dictionary_definition** всех ссылочных классов **referenced_classes** сущности **ap** доступны в рассматриваемом контексте обмена и если никакие ограничения не представлены единственным идентификатором ограничений **constraint_identifier**, то функция **compute_known_referenced_property_constraint** возвращает точно все ограничения, импортируемые сущностью **ap**.

Пример представления на языке EXPRESS:

```
*)
FUNCTION compute_known_referenced_property_constraints(
          ap: a_priori_semantic_relationship):
          SET OF property_constraint;

LOCAL
     s: SET OF property_constraint := []; -- result
     prop: SET OF property_BSU :=
          list_to_set(ap.referenced_properties); --imported properties
     cl: SET OF class_BSU :=
          ap.referenced_classes; --source of importation
```

```
      cons: SET OF property_constraint
           := compute_known_property_constraints(cl);
           -- all those property_constraints existing in the various
           -- classes from cl that may be computed in the current
           -- exchange context.
END_LOCAL;

      REPEAT n_cons := 1 TO SIZEOF(cons);
           IF cons[n_cons].constrained_property IN prop
           THEN
                s := s + cons[n_cons];
           END_IF;
      END_REPEAT;
RETURN(s);

END_FUNCTION; -- compute_known_referenced_property_constraints
(*
```

8.4.4 Функция проверки суперкласса элементов (Superclass_of_item_is_item)

Функция **superclass_of_item_is_item** проверяет тот факт, что суперкласс класса элементов (сущность **cl**) **item_class,** если он существует, также является классом элементов **item_class**.

Если класс, ассоциированный с базовой семантической единицей класса **class_BSU**, не может быть вычислен, то функция возвращает значение UNKNOWN.

Пример представления на языке EXPRESS:

```
*)
FUNCTION superclass_of_item_is_item(cl: item_class): LOGICAL;

IF NOT EXISTS(cl\class.its_superclass)
THEN
      RETURN(TRUE);
END_IF;
IF SIZEOF(cl\class.its_superclass.definition) = 0
THEN
      RETURN(UNKNOWN);
END_IF;

RETURN(('ISO13584_IEC61360_DICTIONARY_SCHEMA.ITEM_CLASS')
      IN TYPEOF(cl\class.its_superclass.definition[1]));

END_FUNCTION; -- superclass_of_item_is_item
(*
```

8.4.5 Функция проверки определения условного ссылочного класса (Check_is_case_of_referenced_class_definition)

Функция **check_is_case_of_referenced_class_definition** возвращает значение TRUE, если тип условного класса элементов **item_class_case_of** из набора словарных определений ссылочных классов **is_case_of** совместим с рассматриваемой реализацией условного класса элементов (**cl**) **item_class_case_of**. В противном случае функция возвращает значение FALSE.

Пример представления на языке EXPRESS:

```
*)
FUNCTION check_is_case_of_referenced_classes_definition(
      cl: item_class_case_of): BOOLEAN;
LOCAL
      class_def_ok: BOOLEAN := TRUE;
```

```
END_LOCAL;

REPEAT i := 1 TO SIZEOF(cl.is_case_of);
     IF (SIZEOF(cl.is_case_of[i].definition) = 1)
     THEN
          IF (NOT('ISO13584_IEC61360_DICTIONARY_SCHEMA' +
               '.ITEM_CLASS'
               IN TYPEOF(cl.is_case_of[i].definition[1])))
          THEN
               class_def_ok := FALSE;
          END_IF;

     END_IF;
END_REPEAT;

RETURN(class_def_ok);

END_FUNCTION; -- check_is_case_of_referenced_classes_definition
(*
```

8.5 Определения правил стандартной схемы условных классов элементов ISO13584_IEC61360_item_class_case_of_schema

8.5.1 Общие положения

Данный подраздел определяет правило пользования стандартной схемой **ISO13584_IEC61360_item_class_case_of_schema**.

8.5.2 Правило видимости и применимости импортированного свойства (Imported_property_are_visible_or_applicable_rule)

Правило **imported_property_are_visible_or_applicable_rule** проверяет тот факт, что если свойство импортировано классом с помощью априорного семантического соотношения **a_priori_semantic_relationship**, то данное свойство является видимым или применимым в классе, из которого оно импортировано.

Примечание — Применимые свойства включают свойства, импортированные с помощью семантического соотношения. Данное правило дает возможность импортировать свойства из класса, куда они уже были импортированы ранее.

Пример представления на языке EXPRESS:

```
*)
RULE imported_properties_are_visible_or_applicable_rule FOR
     ( a_priori_semantic_relationship, property_DET);
WHERE
     WR1: QUERY(rel <* a_priori_semantic_relationship
          | QUERY(prop <* rel.referenced_properties
          | QUERY(cl <* rel.referenced_classes
          | NOT visible_properties(cl, [prop])
          AND NOT applicable_properties(cl, [prop]))
          = rel.referenced_classes) = [])
          = a_priori_semantic_relationship;
END_RULE; -- imported_properties_are_visible_or_applicable_rule
(*
```

8.5.3 Правило видимости и применимости импортированных типов данных (Imported_data_type_are_visible_or_applicable_rule)

Правило **imported_data_type_are_visible_or_applicable_rule** проверяет тот факт, что если тип данных импортирован классом с помощью априорного семантического соотношения **a_priori_semantic_relationship**, то этот тип данных является видимым или применимым в классе, из которого он импортирован.

Примечание — Применимые типы данных включают типы данных, импортированные с помощью семантического соотношения. Данное правило дает возможность импортировать типы данных из класса, куда они уже были импортированы ранее.

Пример представления на языке EXPRESS:

```
*)
RULE imported_data_types_are_visible_or_applicable_rule FOR(
     a_priori_semantic_relationship, data_type_element);
WHERE
     WR1: QUERY(rel <* a_priori_semantic_relationship
          | QUERY(typ <* rel.referenced_data_types
          | QUERY(cl <* rel.referenced_classes
          | NOT visible_types(cl, [typ])
          AND NOT applicable_types(cl, [typ]))
          - rel.referenced_classes) - [])
          = a_priori_semantic_relationship;
END_RULE; -- imported_data_types_are_visible_or_applicable_rule
(*
```

8.5.4 Правило использования поименованного типа (Allowed_named_type_usage_rule)

Правило **allowed_named_type_usage_rule** касается использования поименованного типа. Правило требует, чтобы только типы, применимые в классе, могли быть использованы для описания области значений свойств, объявленных в классе, с помощью атрибута **described_by**.

Пример представления на языке EXPRESS:

```
*)
RULE allowed_named_type_usage_rule FOR(class);
LOCAL
     named_type_usage_allowed: LOGICAL := TRUE;
     is_app: LOGICAL;
     prop: property_bsu;
     cl: class;
     dtnt: SET[0:1] OF data_type_bsu := [];
END_LOCAL;

REPEAT i := 1 TO SIZEOF(class);
      cl := class[i];
      REPEAT j := 1 TO SIZEOF(class[i].described_by);
           prop := cl.described_by[j];
           dtnt := data_type_named_type(prop);

           IF (SIZEOF(dtnt) = 1) THEN
                is_app := applicable_types(cl.identified_by, dtnt);
                IF (NOT is_app) THEN
                     named_type_usage_allowed := FALSE;
                END_IF;
           END_IF;
     END_REPEAT;
END_REPEAT;

WHERE
     WR1: named_type_usage_allowed;
END_RULE; -- allowed_named_type_usage_rule
     (*

*)
END_SCHEMA; -- ISO13584_IEC61360_item_class_case_of_schema
(*
```

Приложение А
(справочное)

Пример физического файла

А.1 Цель рассмотрения примера

В настоящем приложении приведены некоторые фрагменты физического файла, используемого для обмена данными в соответствии с МЭК 61360-DB. В приложении приведен пример использования модели языка EXPRESS (см. раздел 5 «Стандартная словарная схема (**ISO13584_IEC61360_dictionary_schema**)» в соответствии с ИСО 10303-21 для обмена соответствующих данных.

А.2 Заголовок файла

```
*/
ISO-10303-21;
HEADER;
FILE_DESCRIPTION(('Example physical file'), '2;1');
FILE_NAME('example.spf', '2007-07-18', ('IEC SC3D WG2'), (),
'Version 1', '', '');
FILE_SCHEMA(('example_schema'));
ENDSEC;
DATA;
/*
```

А.3 Поставщик данных

```
*/
#1=SUPPLIER_BSU('112/2///61360_4_1', *); /*according to ISO 13584-26*/
#2=SUPPLIER_ELEMENT(#1, #3, '01', $, $, $, #4, #5);
#3=DATES('1994-09-16', '1994-09-16', $);
#4=ORGANIZATION('IEC',    'IEC  Maintenance  Agency',  'The  IEC  Maintenance
Agency');
#5=ADDRESS('to be determined', $, $, $, $, $, $, $, $, $, $, $);
#10=SUPPLIER_BSU('112/3///_00', *);   /* ISO/IEC ICS */
/*
```

А.4 Данные корневого класса

Корневой класс, удовлетворяющий требованиям AAA000 МЭК, задает область имен в соответствии с МЭК 61360-DB. Данный класс использует два дерева: одно — для материалов, другое — для компонентов. Корневой класс определен как класс элементов **item_class**.

```
*/
#90=CLASS_BSU('00', '001', #10);
#100=CLASS_BSU('AAA000', '001', #1);
#101=ITEM_CLASS(#100, #3, '01', $, $, $, #102, TEXT('IEC root class that
provides a name scope corresponding to IEC 61360-DB. It covers two trees,
one for materials, one for components'), $, $, $, #90, (#110), (), (), $,
(), (#110), (), $, $, $);
#102=ITEM_NAMES(LABEL('IEC root class'), (), LABEL('IEC root'), $, $);
#110=PROPERTY_BSU('AAE000', '001', #200);
#111=NON_DEPENDENT_P_DET(#110, #3, '01', $, $,$, #112, TEXT('the type of
tree: material or component'), $, $, $, $, (), $, $, #113, $);
#112=ITEM_NAMES(LABEL('type of tree'), (), LABEL('tree type'), $, $);
#113=NON_QUANTITATIVE_CODE_TYPE((), 'A..8', #114);
#114=VALUE_DOMAIN((#120,#122), $, $, (), $, $);
#120=DIC_VALUE(VALUE_CODE_TYPE('MATERIAL'), #121, $, $, $, $, $, $);
```

```
#121=ITEM_NAMES(LABEL('material tree'), (), LABEL('mat tree'), $, $);
#122=DIC_VALUE(VALUE_CODE_TYPE('COMPONS'), #123, $, $, $, $, $, $);
#123=ITEM_NAMES(LABEL('component tree'), (), LABEL('comp tree'), $, $);
/*
```

А.5 Данные о материалах

```
*/
#200=CLASS_BSU('AAA218', '001', #1);
#201=ITEM_CLASS(#200, #3, '01', $, $, $, #202, TEXT('root class of the
materials tree'), $, $, $, #100, (#210,#230), (), (), $, (), (#210),
(#205), $, 'MATERIAL', $);
#202=ITEM_NAMES(LABEL('materials root class'), (), LABEL('materials root'),
$, $);
#205=CLASS_VALUE_ASSIGNMENT(#110,STRING_VALUE('MATERIAL'));
#210=PROPERTY_BSU('AAF311', '005', #100);
#211=NON_DEPENDENT_P_DET(#210, #3, '01', $, $, $, #212, TEXT('code of the
type of material'), $, $, $, $, (), $, 'A57', #213, $);
#212=ITEM_NAMES(LABEL('material type'), (), LABEL('material type'), $, $);
#213=NON_QUANTITATIVE_CODE_TYPE((), 'M..3', #214);
#214=VALUE_DOMAIN((#220,#222,#224,#226), $, $, (), $, $);
#220=DIC_VALUE(VALUE_CODE_TYPE('ACO'), #221, $, $, $, $, $, $);
#221=ITEM_NAMES(LABEL('acoustical'), (), LABEL('acoustical'), $, $);
#222=DIC_VALUE(VALUE_CODE_TYPE('MG'), #223, $, $, $, $, $, $);
#223=ITEM_NAMES(LABEL('magnetic'), (), LABEL('magnetical'), $, $);
#224=DIC_VALUE(VALUE_CODE_TYPE('OP'), #225, $, $, $, $, $, $);
#225=ITEM_NAMES(LABEL('optical'), (), LABEL('optical'), $, $);
#226=DIC_VALUE(VALUE_CODE_TYPE('TH'), #227, $, $, $, $, $, $);
#227=ITEM_NAMES(LABEL('thermal-electric'), (), LABEL('th-electric'), $, $);
#230=PROPERTY_BSU('AAF286', '005', #100);
#231=NON_DEPENDENT_P_DET(#230, #3, '01', $, $,$, #232, TEXT('The nominal
density (in kg/m**3) of a material.'), $, $, $, #233, (), $, 'K02', #234,
$);
#232=ITEM_NAMES(LABEL('density'), (), LABEL('density'), $, $);
#233=MATHEMATICAL_STRING('$r_d', '&rho;<sub>d</sub>');
#234=REAL_MEASURE_TYPE((), 'NR3..3.3ES2', #235, $, $, $);
#235=DIC_UNIT(#236, $);
#236=DERIVED_UNIT((#239,#237));
#237=DERIVED_UNIT_ELEMENT(#238, 1.);
#238=SI_UNIT(*,.KILO.,.GRAM.);
#239=DERIVED_UNIT_ELEMENT(#240, -3.);
#240=SI_UNIT(*, $,.METRE.);
/*
```

А.6 Данные компонентов

```
*/
#300=CLASS_BSU('EEE000', '001', #1);
#301=ITEM_CLASS(#300, #3, '01', $, $, $, #302, TEXT('root class of the
components tree'), $, $, $, #100, (#310,#330,#350), (), (), $, (), (#310), (#305),
$, 'COMPONS', $);
```

```
#302=ITEM_NAMES(LABEL('components root class'), (), LABEL('components
root'), $, $);
#305=CLASS_VALUE_ASSIGNMENT(#110, STRING_VALUE('COMPONS'));
#310=PROPERTY_BSU('AAE001', '005', #100);
#311=NON_DEPENDENT_P_DET(#310, #3, '01', $, $, $, #312, TEXT('Code of the
main functional class to which a component belongs'), $, $, $, $, (), $,
'A52', #313, $);
#312=ITEM_NAMES(LABEL('main class of component'), (), LABEL('main class'),
$, $);
#313=NON_QUANTITATIVE_CODE_TYPE((), 'M..3', #314);
#314=VALUE_DOMAIN((#320,#322,#324,#326), $, $, (), $, $);
#320=DIC_VALUE(VALUE_CODE_TYPE('EE'), #321, $, $, $, $, $, $);
#321=ITEM_NAMES(LABEL('EE (electric / electronic)'), (), LABEL('EE'), $,
$);
#322=DIC_VALUE(VALUE_CODE_TYPE('EM'), #323, $, $, $, $, $, $);
#323=ITEM_NAMES(LABEL('electromechanical'), (), LABEL('electromech'), $,
$);
#324=DIC_VALUE(VALUE_CODE_TYPE('ME'), #325, $, $, $, $, $, $);
#325=ITEM_NAMES(LABEL('mechanical'), (), LABEL('mechanical'), $, $);
#326=DIC_VALUE(VALUE_CODE_TYPE('MP'), #327, $, $, $, $, $, $);
#327=ITEM_NAMES(LABEL('magnetic part'), (), LABEL('magnetic'), $, $);
#330=PROPERTY_BSU('AAF267', '005', #100);
#331=NON_DEPENDENT_P_DET(#330, #3, '01', $, $,$, #332, TEXT('The nominal
distance (in m) between the inside of the two tapes used for taped products
with axial leads'), $, $, $, #333, (), $, 'T03', #334, $);
#332=ITEM_NAMES(LABEL('inner tape spacing'), (), LABEL('inner tape spac'),
$, $);
#333=MATHEMATICAL_STRING('b_tape', 'b<sub>tape</sub>');
#334=LEVEL_TYPE((), (.NOM.), #335);
#335=REAL_MEASURE_TYPE((), 'NR3..3.3ES2', #336, $, $, $);
#336=DIC_UNIT(#337, $);
#337=SI_UNIT(*, $,.METRE.);
#350=PROPERTY_BSU('AAE022', '005', #100);
#351=NON_DEPENDENT_P_DET(#350, #3, '01', $, $, $,#352, TEXT('The value as
specified by level (mlNoMax) of the outside diameter (in m) of a component
with a body of circular cross-section'), $, $, $, #353, (), $, 'T03', #354,
$);
#352=ITEM_NAMES(LABEL('outside diameter'), (), LABEL('outside diam'), $,
$);
#353=MATHEMATICAL_STRING('d_out', 'd<sub>out</sub>');
#354=LEVEL_TYPE((), (.MIN.,.NOM.,.MAX.), #355);
#355=REAL_MEASURE_TYPE((), 'NR3..3.3ES2', #356, $, $, $);
#356=DIC_UNIT(#357, #358);
#357=SI_UNIT(*, $,.METRE.);
#358=MATHEMATICAL_STRING('m', 'm');
/*
```

A.7 Данные электрических/электронных компонентов

```
*/
#400=CLASS_BSU('EEE001', '001', #1);
```

```
#401=ITEM_CLASS(#400, #3, '01', $, $,$, #402, TEXT('electric / electronic
components'), $, $, $, #300, (#410,#470), (), (), $, (), (#410), (#405), $, 'EE', $);
#402=ITEM_NAMES(LABEL('EE components'), (), LABEL('EE components'), $, $);
#405=CLASS_VALUE_ASSIGNMENT(#310, STRING_VALUE('EE'));
#410=PROPERTY_BSU('AAE002', '005', #100);
#411=NON_DEPENDENT_P_DET(#410, #3, '01', $, $,$, #412, TEXT('Code of the
category to which an electric/electronic component belongs.'), $, $, $, $,
(), $, 'A52', #413, $);
#412=ITEM_NAMES(LABEL('category EE component'), (), LABEL('categ EE comp'),
$, $);
#413=NON_QUANTITATIVE_CODE_TYPE((), 'M..3', #414);
#414=VALUE_DOMAIN((#420,#422,#424,#426,#428
,#430,#432,#434,#436,#438
,#440), $, $, (), $, $);
#420=DIC_VALUE(VALUE_CODE_TYPE('AMP'), #421, $, $, $, $, $, $);
#421=ITEM_NAMES(LABEL('amplifier'), (), LABEL('amplifier'), $, $);
#422=DIC_VALUE(VALUE_CODE_TYPE('ANT'), #423, $, $, $, $, $, $);
#423=ITEM_NAMES(LABEL('antenna (aerial)'), (), LABEL('antenna (aer)'), $,
$);
#424=DIC_VALUE(VALUE_CODE_TYPE('BAT'), #425, $, $, $, $, $, $);
#425=ITEM_NAMES(LABEL('battery'), (), LABEL('battery'), $, $);
#426=DIC_VALUE(VALUE_CODE_TYPE('CAP'), #427, $, $, $, $, $, $);
#427=ITEM_NAMES(LABEL('capacitor'), (), LABEL('capacitor'), $, $);
#428=DIC_VALUE(VALUE_CODE_TYPE('CND'), #429, $, $, $, $, $, $);
#429=ITEM_NAMES(LABEL('conductor'), (), LABEL('conductor'), $, $);
#430=DIC_VALUE(VALUE_CODE_TYPE('DEL'), #431, $, $, $, $, $, $);
#431=ITEM_NAMES(LABEL('delay line'), (), LABEL('delay line'), $, $);
#432=DIC_VALUE(VALUE_CODE_TYPE('DID'), #433, $, $, $, $, $, $);
#433=ITEM_NAMES(LABEL('diode device'), (), LABEL('diode device'), $, $);
#434=DIC_VALUE(VALUE_CODE_TYPE('FIL'), #435, $, $, $, $, $, $);
#435=ITEM_NAMES(LABEL('filter'), (), LABEL('filter'), $, $);
#436=DIC_VALUE(VALUE_CODE_TYPE('IC'), #437, $, $, $, $, $, $);
#437=ITEM_NAMES(LABEL('integrated circuit'), (), LABEL('IC'), $, $);
#438=DIC_VALUE(VALUE_CODE_TYPE('IND'), #439, $, $, $, $, $, $);
#439=ITEM_NAMES(LABEL('inductor'), (), LABEL('inductor'), $, $);
#440=DIC_VALUE(VALUE_CODE_TYPE('LAM'), #441, $, $, $, $, $, $);
#441=ITEM_NAMES(LABEL('lamp'), (), LABEL('lamp'), $, $);
#470=PROPERTY_BSU('AAE754', '005', #100);
#471=NON_DEPENDENT_P_DET(#470, #3, '01', $, $, $,#472, TEXT('The number of
electrical terminals of an electric/electronic or electromechanical
component'), $, $, $, #473, (), $, 'Q56', #474, $);
#472=ITEM_NAMES(LABEL('number of terminals'), (LABEL('number of pins')),
LABEL('nr of terminals'), $, $);
#473=MATHEMATICAL_STRING('N_term', 'N<sub>term</sub>');
#474=INT_TYPE((), 'NR1..4');


ENDSEC;
END-ISO-10303-21;
```

Приложение В
(справочное)

Диаграмма EXPRESS-G

Данное приложение содержит диаграммы EXPRESS-G (см. разделы 6—8). Понятие диаграммы EXPRESS-G определено в приложении ИСО 10303-11:2004.

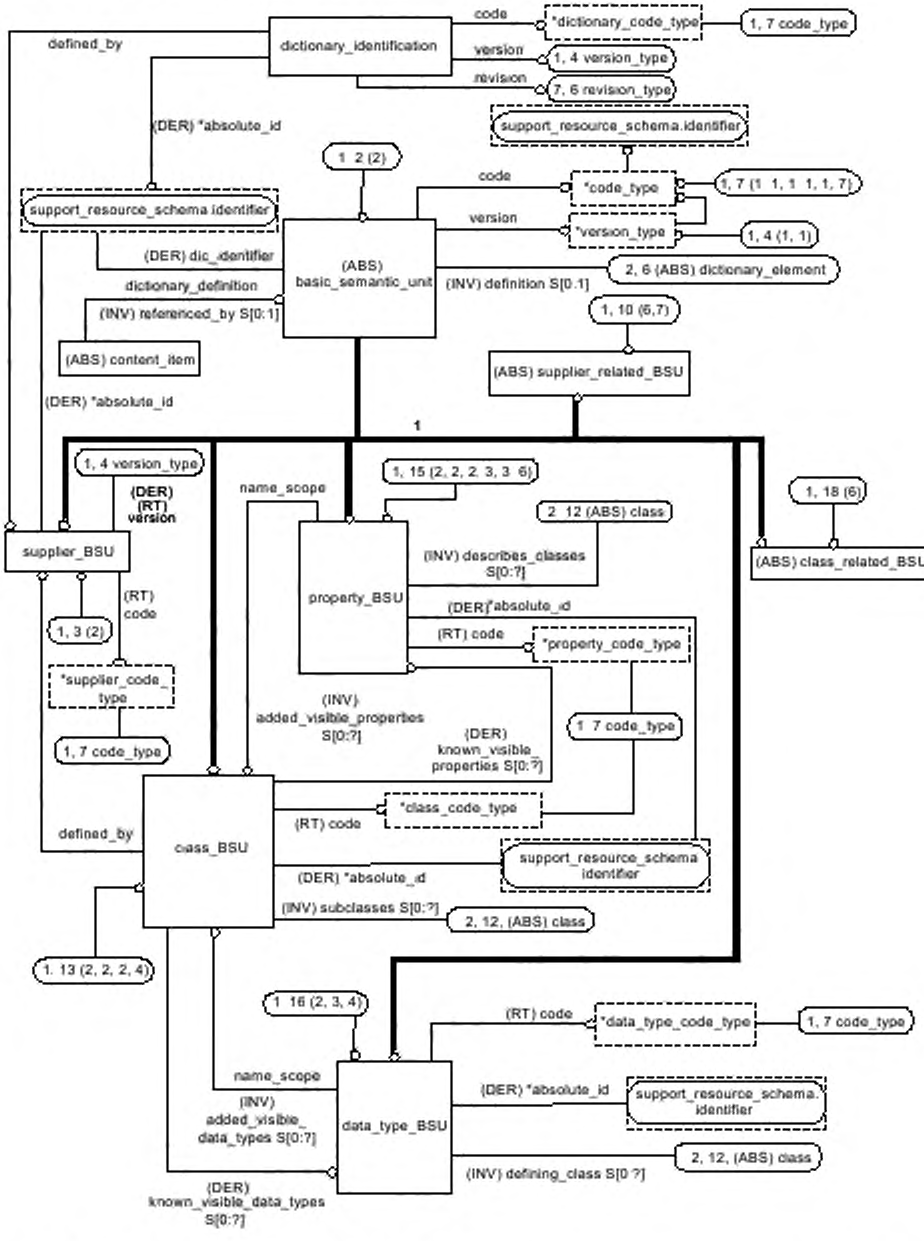


Рисунок В.1 — Диаграмма EXPRESS-G № 1 (из семи) для стандартной словарной схемы **ISO13584_IEC61360_dictionary_schema**

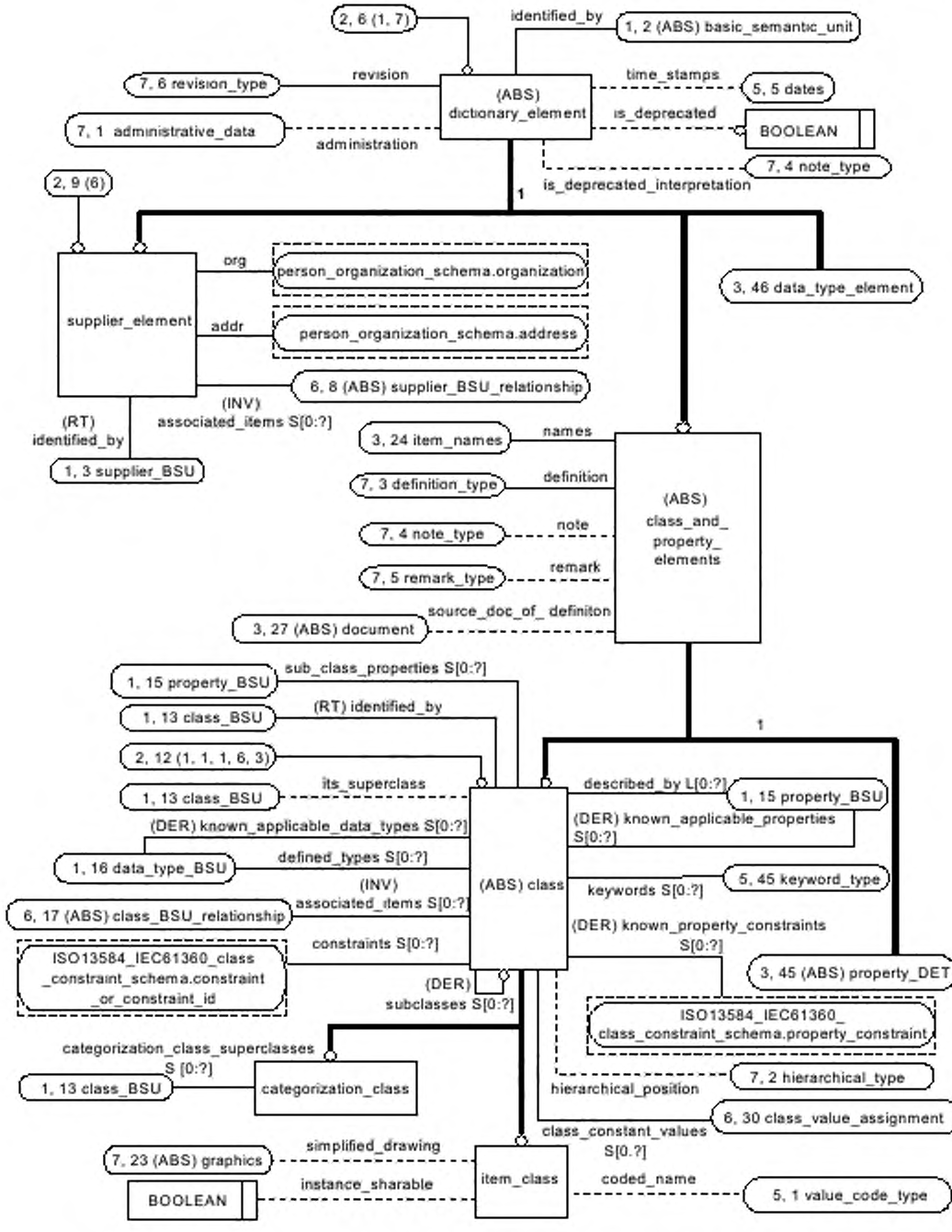


Рисунок В.2 — Диаграмма EXPRESS-G № 2 (из семи) для стандартной словарной схемы **ISO13584_IEC61360_dictionary_schema**

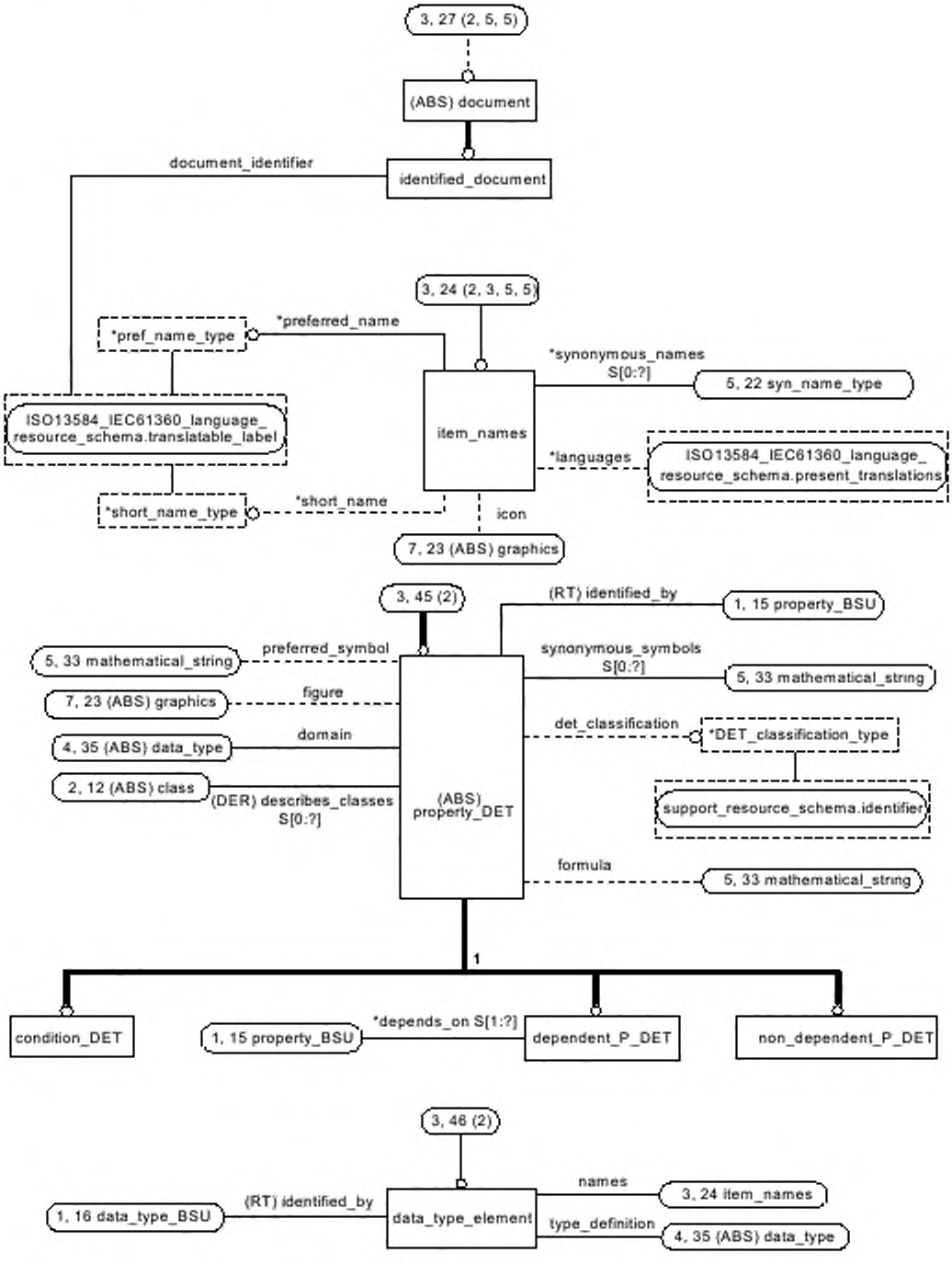


Рисунок В.3 — Диаграмма EXPRESS-G № 3 (из семи) для стандартной словарной схемы **ISO13584_IEC61360_dictionary_schema**

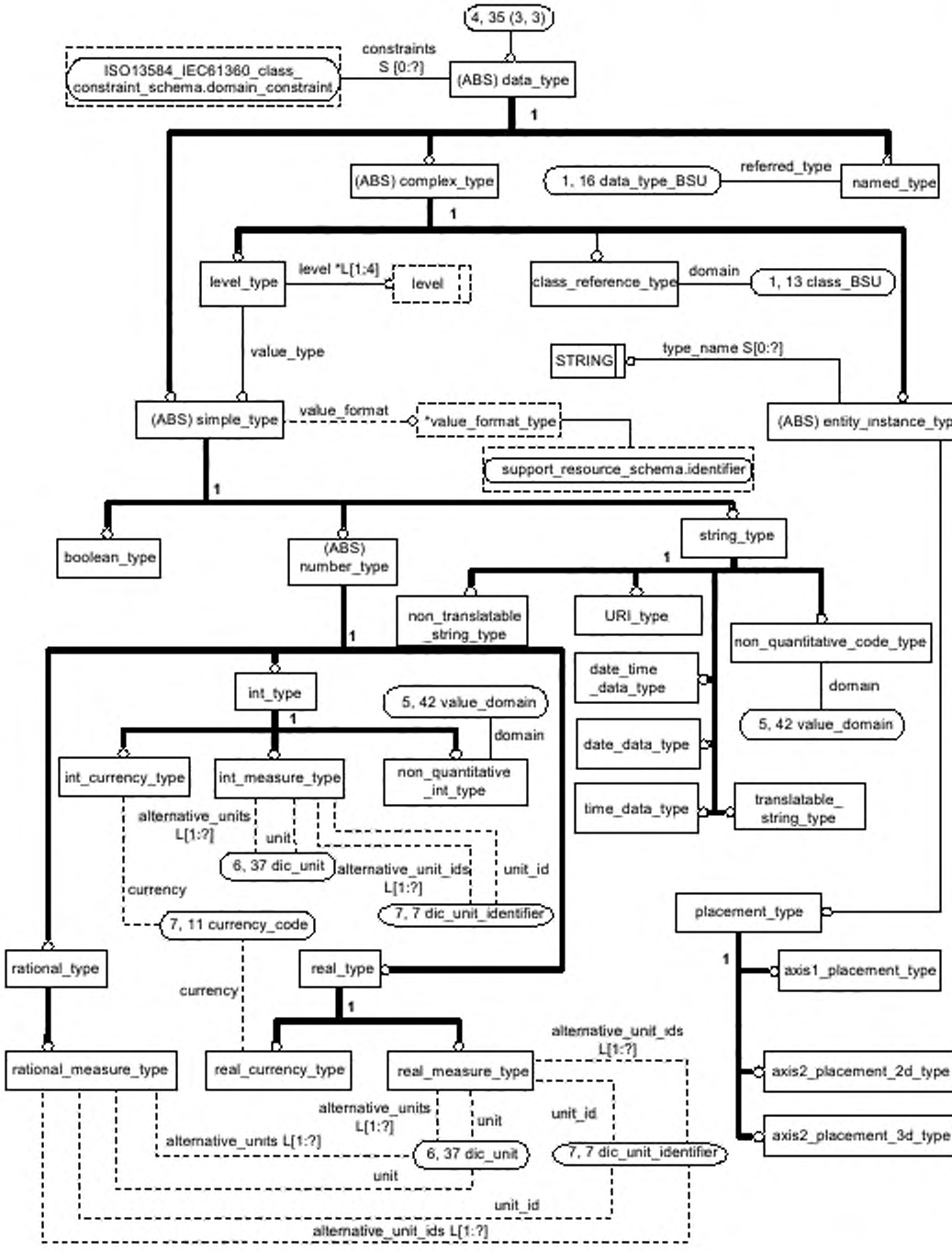


Рисунок В.4 — Диаграмма EXPRESS-G № 4 (из семи) для стандартной словарной схемы **ISO13584_IEC61360_dictionary_schema**

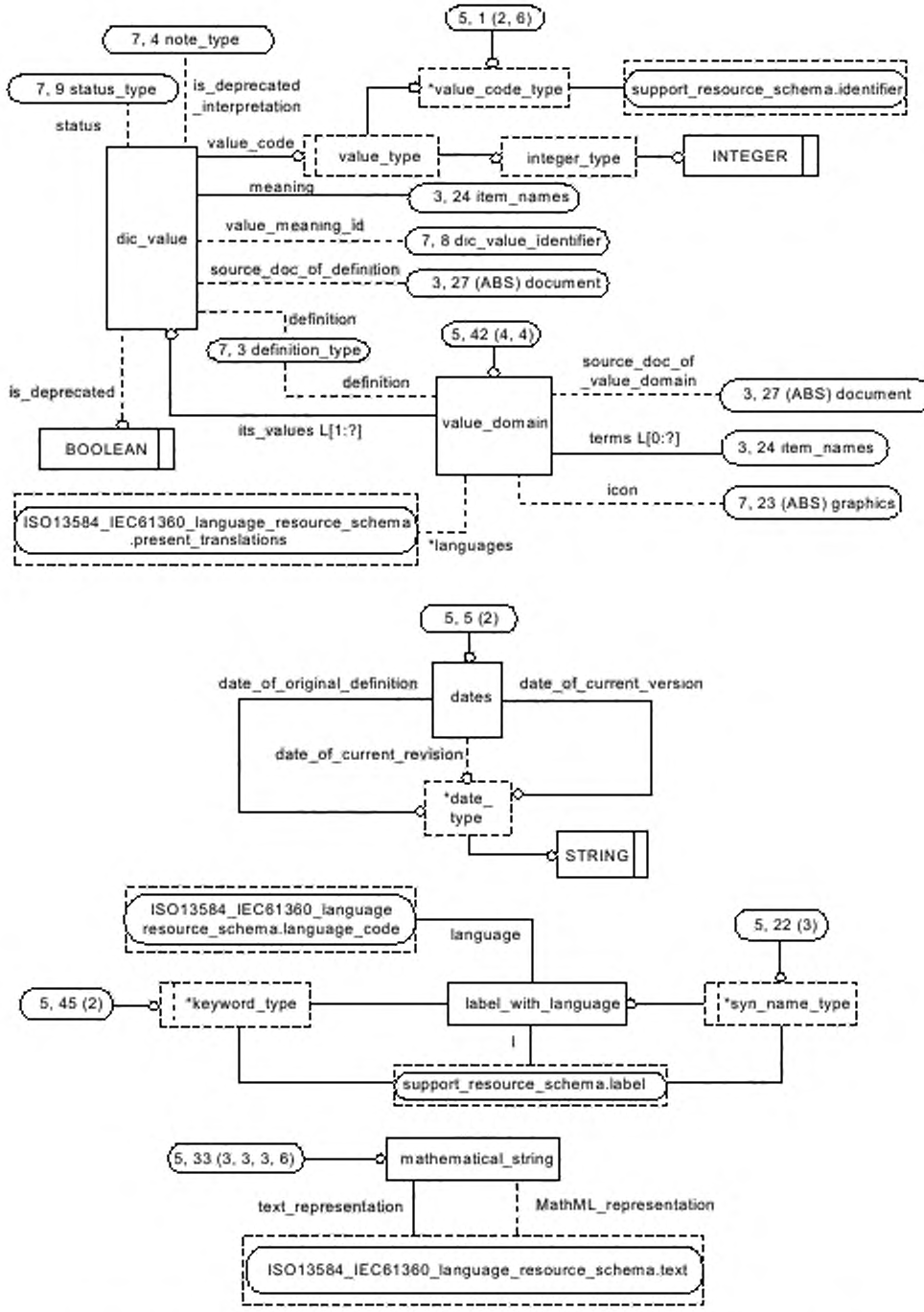


Рисунок В.5 — Диаграмма EXPRESS-G № 5 (из семи) для стандартной словарной схемы **ISO13584_IEC61360_dictionary_schema**

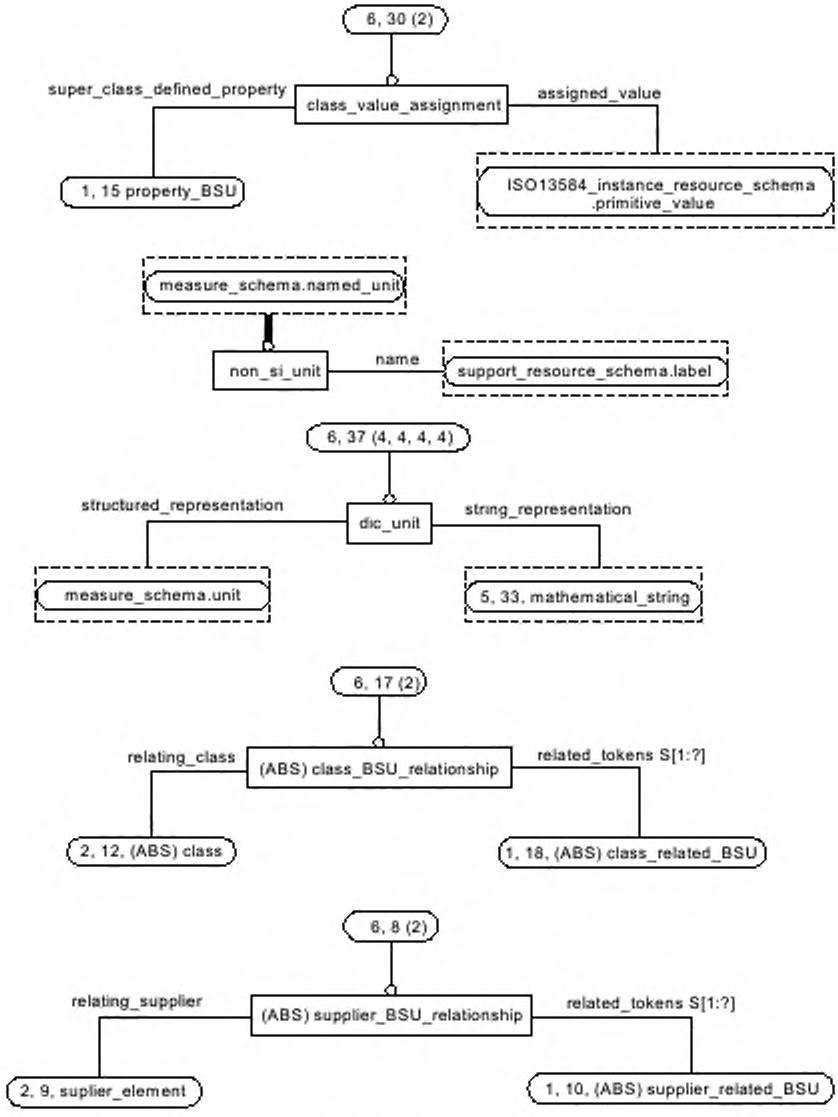


Рисунок В.6 — Диаграмма EXPRESS-G № 6 (из семи) для стандартной словарной схемы **ISO13584_IEC61360_dictionary_schema**

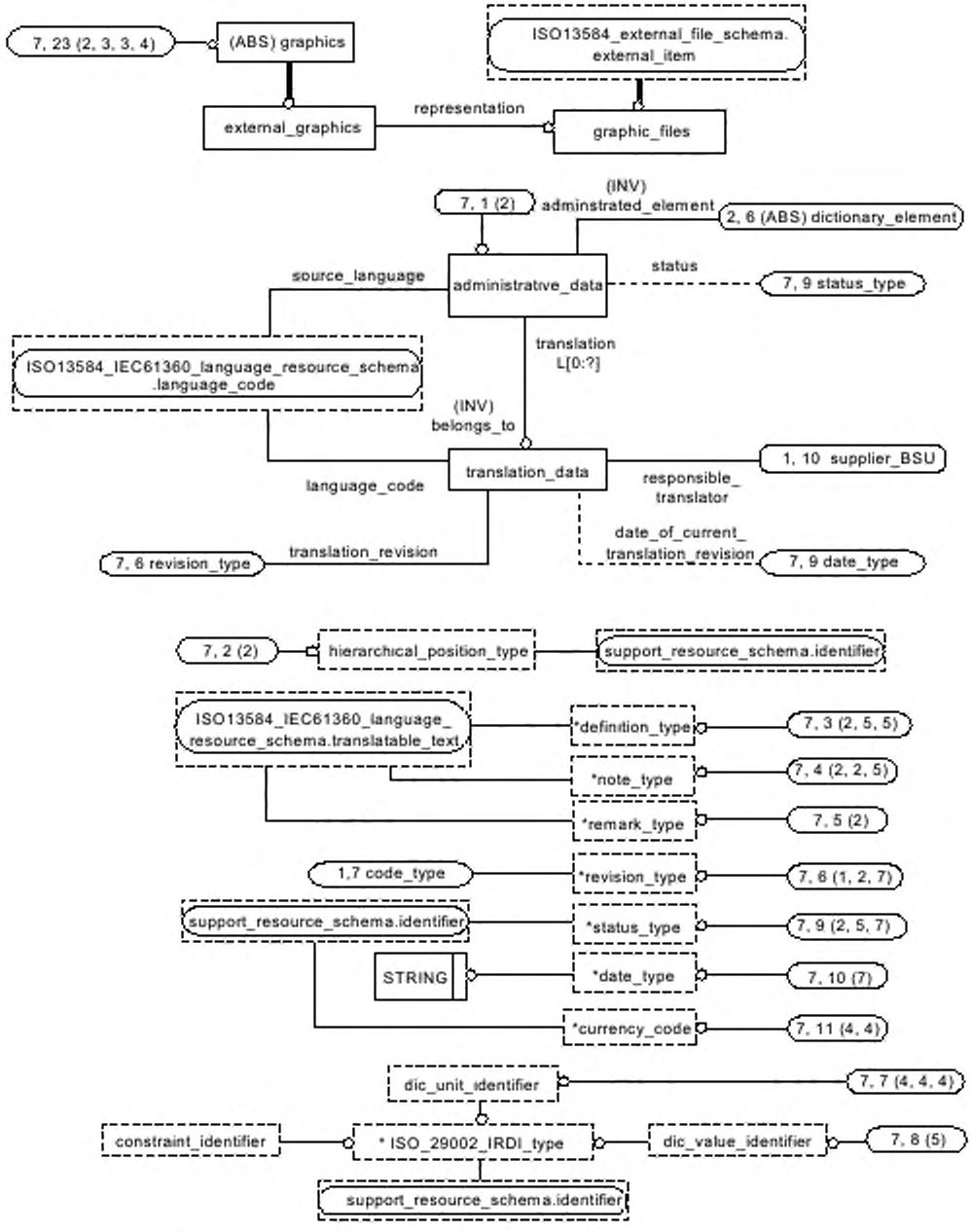


Рисунок В.7 — Диаграмма EXPRESS-G № 7 (из семи) для стандартной словарной схемы **ISO13584_IEC61360_dictionary_schema**

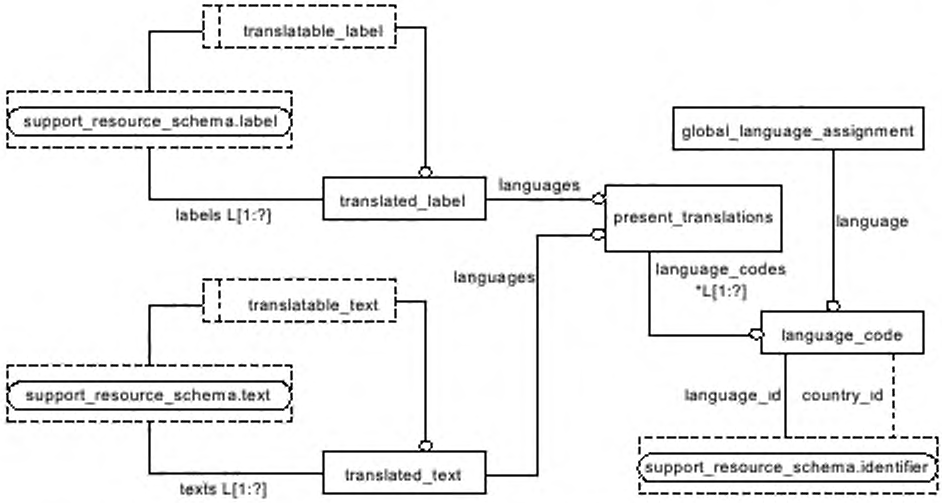


Рисунок В.8 — Диаграмма EXPRESS-G № 1 (единственная) для стандартной схемы языкового ресурса **ISO13584_IEC61360_language_resource_schema**

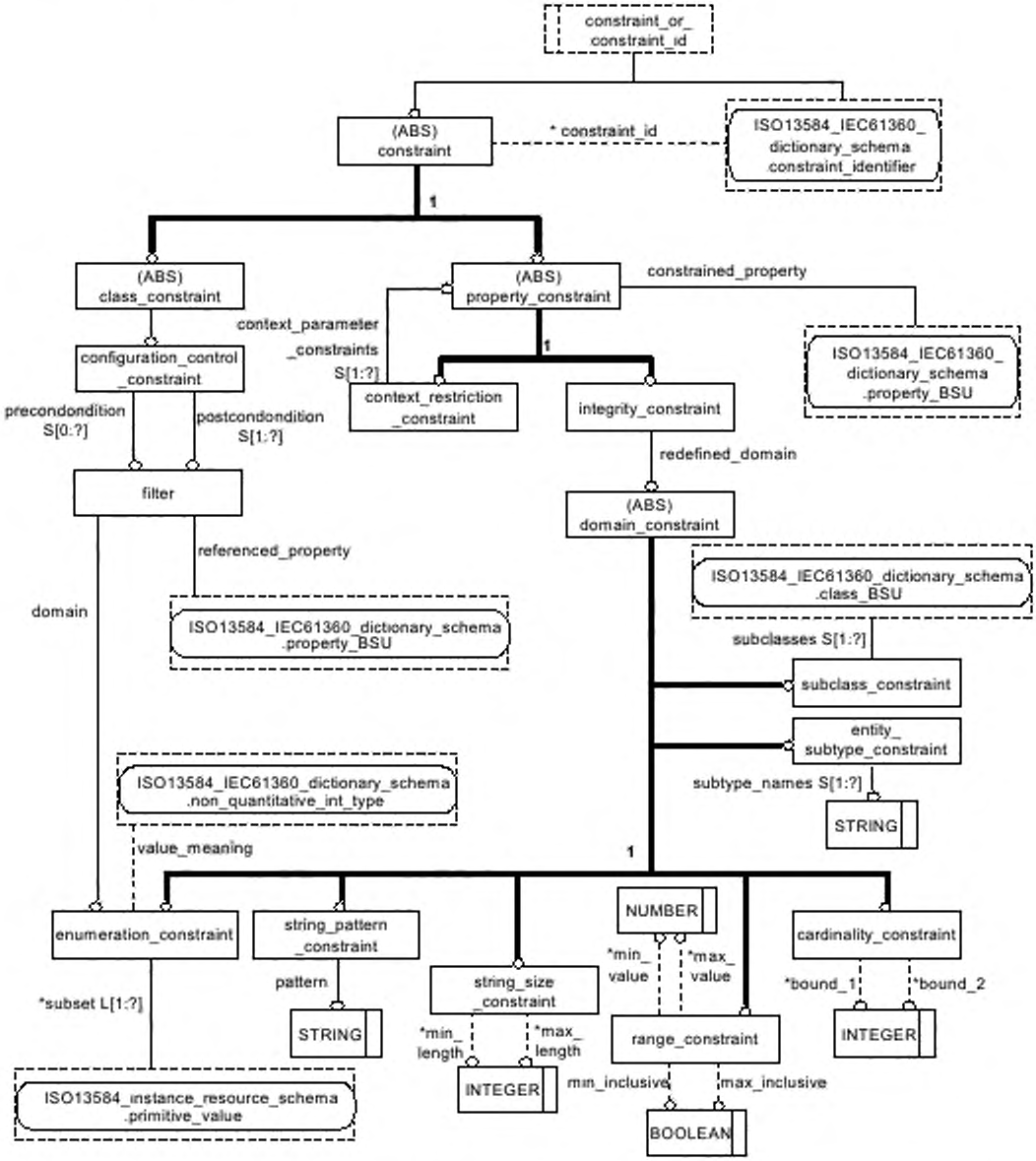


Рисунок В.9 — Диаграмма EXPRESS-G № 1 (единственная) для стандартной схемы ограничений
ISO13584_IEC61360_constraint_schema

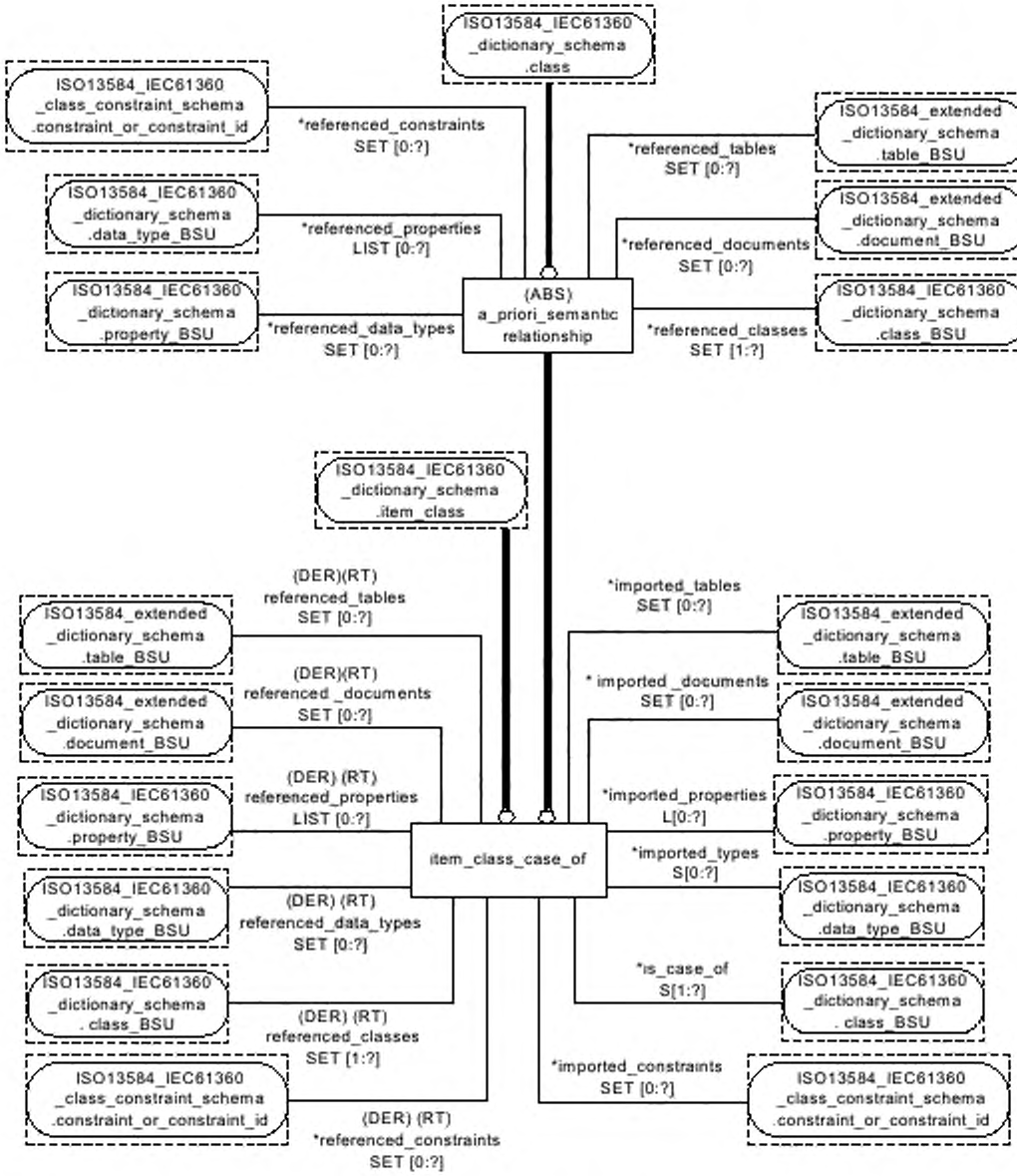


Рисунок В.10 — Диаграмма EXPRESS-G № 1 (единственная)
для стандартной схемы условного класса продуктов
ISO13584_IEC61360_item_class_case_of_schema

Приложение С
(справочное)

Частные словари

Модель данных EXPRESS, опубликованная в настоящем стандарте и дублированная (для удобства) в ИСО 13584-42, допускает описание словарей, составленных из классов, свойств и типов данных. Данная модель обеспечивает их уникальную идентификацию с помощью механизма базовой семантической единицы (BSU). Рассматриваемая модель данных допускает описание иерархий классов, организованных в соответствии со структурой дерева, с помощью простого механизма наследственности. Указанную модель данных используют только отдельные автономные словари.

Использование рассматриваемой модели данных способствует разработке нескольких словарей. Во время разработки словаря может возникнуть необходимость ссылки на конкретный класс, свойство или тип данных, определенные ранее в другом словаре. Возможность импортирования свойств и типов данных в разрабатываемый словарь обеспечивается некоторым условным соотношением, которое может быть использовано совместно с полной общей словарной моделью ИСО 13584/МЭК 61360, задокументированной как в ИСО 13584-25, так и в МЭК 61360-5. Более того, данное соотношение допускает импортирование внешне определенных свойств и типов данных, обеспечивающих поддержку частных словарей и исключение дублирования словарей. При этом каждый словарь может определять свою собственную структуру класса.

Полная общая словарная модель ИСО 13584/МЭК 61360 предлагает два механизма:

- сущность языка EXPRESS **a_priori_case_of_semantic_relationship** (априорное условное семантическое соотношение) допускает прямое использование свойств или типов данных, определенных во внешнем словаре (словарях), без их повторного описания;
- сущность языка EXPRESS **a_posteriori_case_of_relationship** (апостериорное условное соотношение) допускает (если некоторые свойства или типы данных уже определены в разрабатываемом словаре) их отображение на соответствующие свойства или типы данных, определенные во внешнем словаре.

Указанные механизмы позволяют разрабатывать словари, ссылающиеся на элементы данных, определенные в других словарях, без изменения их семантического смысла. Более того, условные соотношения заносятся в словари и используются для автоматической интеграции словарей, основанных на указанных стандартных словарях.

Данный механизм также может быть использован при разработке словаря конечного пользователя. Как правило, конечному пользователю словаря не нужна вся структура класса, определенная в стандартных словарях. Ему часто достаточно иметь возможность обмениваться информацией с другими пользователями, словари которых основаны на том же стандартном словаре (словарях). Если конечный пользователь определяет свою собственную иерархию, отображает каждый свой класс (с помощью условного соотношения) на соответствующий стандартный класс и если пользователь импортирует все существующие стандартные свойства, необходимые в текущем контексте, добавляет свои собственные особые свойства, то он может приспособить свой собственный словарь для обмена стандартной информацией с другими пользователями. Рассматриваемый подход к разработке частных словарей конечных пользователей — это суть предложения, содержащегося в настоящем стандарте.

Приложение D
(справочное)

Спецификация формата значения

D.1 Общие положения

Настоящий стандарт и ИСО 13584-42 приводят конкретный синтаксис формата строк и численных значений, ассоциируемых со свойством.

Пример 1 — Формат NR1 3 устанавливает, что допустимы только целые значения, состоящие из трех цифр.

Примечание 1 — Формат значения типа данных **data_type**, включая булевский тип **boolean_type**, не устанавливается.

Примечание 2 — В настоящем стандарте определение формата значения свойства не обязательно.

Синтаксис допустимого формата определен в данном приложении с помощью подмножества расширенной формы Бэкуса — Наура (EBNF), определенного в ИСО/МЭК 14977.

Пример 2 — Синтаксис формата NR1 3 — это буквы 'NR1' ' ' '3'.

Смысл каждого варианта синтаксиса (символов, используемых для представления значения) не может быть задан с помощью формы EBNF. Смысл каждой части формата символов, допустимых для представления значений, устанавливается отдельно для каждой части формата.

Пример 3 — Синтаксис формата NR1 3 имеет следующий смысл: NR1 означает, что можно представить только целое значение. Пробел означает, что можно использовать только фиксированное число символов. Цифра 3 означает допустимое количество цифр в записи.

D.2 Обозначения

Таблица D.1 содержит подмножество синтаксического метаязыка EBNF, определенного в ИСО/МЭК 14977 и использованного в настоящем стандарте. Данный язык устанавливает формат значений свойств.

Таблица D.1 — Синтаксический метаязык EBNF по ИСО/МЭК 14977

Представление	Названия символов по ИСО/МЭК 10646-1	Символ метаязыка и его роль
' '	апостроф	Одинарная кавычка представляет терминал языка. Терминал не должен содержать апостроф. Пример: 'Hello'
"..."	кавычки	Двойная кавычка представляет терминал языка. Терминал не должен содержать кавычки. Пример: «Машина Джона»
( )	левая скобка, правая скобка	Символы начала/окончания группы. Содержание рассматривается как один символ
[ ]	левая квадратная скобка, правая квадратная скобка	Символ начала/окончания опции. Содержание может быть и может не быть.
{ }	левая фигурная скобка, правая фигурная скобка	Символ начала/окончания повтора. Содержание может повторяться от 0 до *n* раз
–	тире-минус	Символ замены
,	запятая	Символ последовательного соединения
=	знак равенства	Символ определения. Синтаксическое правило: определяет символ слева через формулу справа
|	вертикальная линия	Альтернативный разделитель
;	точка с запятой	Символ терминатора. Окончание синтаксического правила

С помощью введенных обозначений синтаксис рассматриваемого подмножества метаязыка EBNF (используемый в настоящем стандарте для задания формата значений свойств) характеризуется следующей грамматикой (описание символов метаидентификаторов, букв и цифр отсутствует):

```
syntax = syntaxrule,{ syntaxrule };
syntaxrule = metaidentifier, '=', definitionslist, ';';
definitionslist = singledefinition, { ' ', singledefinition };
singledefinition = term, { ',', term };
term = primary, [ '-', primary ];
primary = optionalsequence | repeatedsequence | groupedsequence |
          metaidentifier | terminal | empty;
optionalsequence = '[' definitionslist ']';
repeatedsequence = '{' definitionslist '}';
groupedsequence = '(' definitionslist ')';
metaidentifier = letter, { letter };
terminal = "'", {character - "'"}, "'"
           | '"', {character - '"'},{character - '"'}, '"';
empty =;
```

Знак равенства «=» указывает синтаксическое правило. Метаидентификатор слева можно заменить комбинацией элементов справа. Любые пробелы между элементами информации не несут, если только они не находятся внутри терминала. Синтаксическое правило закачивается символом «;».

Использование метаидентификатора внутри списка определений означает, что нетерминальный символ находится слева от другого синтаксического правила. Метаидентификатор составлен из букв и цифр. Первый символ — буква. Если термин содержит и первичное выражение перед знаком «минус», и первичное выражение за знаком «минус», то только последовательность символов, представленная первым первичным выражением и не представленная вторым первичным выражением, представляется термином.

Пример 1 — Обозначение:
"'", символ-'"',"'"
означает любой символ (кроме апострофа), вставленный между двумя апострофами.

Терминал означает символ, который не может быть расширен далее с помощью синтаксического правила и который появится в окончательном результате. Терминал может быть представлен двумя способами: либо это набор символов без апострофа (вставленный между двумя апострофами), либо это набор символов без кавычек (вставленный между двумя кавычками).

Пример 2 — Предположим, что мы хотим установить (с помощью данной грамматики) цену продукта в евро, €. Цена — это положительное число не более чем с двумя цифрами после запятой (центы). Определим три мета-идентификатора, ассоциированных с тремя синтаксическими правилами:

```
digit = '0' | '1' | '2'   '3' | '4'   '5' | '6'   '7' | '8' | '9';
cents = [ '.', digit [, digit ] ];
euros = digit {, digit } cents;
```

По указанным синтаксическим правилам: 012, 4323.3, 3.56 — это примеры незапрещенных представлений цены в евро. Аналогично 12...10 — это примеры недопустимых представлений цены в евро.

D.3 Типы формата значений данных

Грамматика, определенная в данном приложении, определяет 9 различных типов форматов значения: 4 количественных и 5 неколичественных форматов значения.

В следующем разделе определяются метаидентификаторы, используемые для описания указанных форматов. В разделе D.5 определяется синтаксическое правило для 4-х метаидентификаторов, представляющих 4 количественных формата значения, вместе с их смыслом на уровне значений. В разделе D.6 определяются метаидентификаторы для 5-ти неколичественных форматов значения вместе с их смыслом на уровне значений.

D.4 Метаидентификаторы, используемые для определения форматов

Метаидентификаторы, используемые в грамматике, определяющей различные форматы значений:
dot = '.'; (точка);
decimalMark = '.'; (десятичная точка);
exponentIndicator = 'E'; (показатель степени);

numeratorIndicator = 'N'; (показатель числителя);
denominatorIndicator = 'D'; (показатель знаменателя);
leadingDigit = T | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'; (первая цифра);
lengthOfExponent (длина показателя) = **leadingDigit** (первая цифра), {**trailingDigit**} (последующая цифра);
lengthOfIntegerPart (длина целой части) = (**leadingDigit, {trailingDigit}**);
lengthOfNumerator (длина числителя) = **leadingDigit, {trailingDigit};**
lengthOfDenominator (длина знаменателя) = **leadingDigit**, {**trailingDigit**}; **lengthOfFractionalPart** (длина дробной части) = (**leadingDigit, {trailingDigit**}) | '0'; **lengthOfIntegralPart** (длина целой части) = (**leadingDigit**, **{trailingDigit})**| '0';
lengthOfNumber (длина числа) = **leadingDigit, {trailingDigit**};
trailingDigit (последующая цифра) = '0' | T I '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9';
signedExponent (показатель степени со знаком) = 'S';
signedNumber (число со знаком) = **space** (пробел),'S';
space (пробел) = ' ';
variableLengthIndicator (индикатор длины переменной) = '..';
decimalMark (десятичный знак): разделитель между целой и дробной частями числа в формате NR2 или NR3;
leadingDigit: первая цифра числа, включающая одну или несколько цифр;
trailingDigit: одна из последующих цифр, образующая число (кроме первой).

Примечание — Если число состоит только из одной цифры, то в нем последующих цифр нет.

D.5 Количественный формат значения

D.5.1 Общие положения

В следующих четырех подразделах определены четыре синтаксических правила для количественного формата значения и их смысл, необходимый для представления значения. Они могут использоваться для свойств с нижеследующими типами данных:

- числовой тип **number_type** или любой его подтип;
- тип уровня **level_type** с типом значения **value_type** либо действительной меры **real_measure_type,** либо целой меры **int_measure_type;**
- списочный тип **list_type,** тип множества **set_type,** упакованный тип **bag_type,** тип массива **array_type,** или множества с подмножеством ограничений **set_with_subset_constraint_type,** для которых тип значения **value_type —** это тип числа **number_type** или любой его подтип.

Примечание 1 — Указанные выше типы **list_type, set_type, bag_type, array_type** и **set_with_subset_constraint_type** определены в ИСО 13584-25.

Примечание 2 — Для неколичественного целого типа **non_quantitative_int_type** формат значения используется для задания кода.

Примечание 3 — Значение данного атрибута должно быть совместимо с типом данных свойства: оно не должно изменять указанный тип данных или должно игнорироваться.

Пример — Формат значения NR2 не совместим с целым типом int_type, так как целые значения не имеют дробной части.

D.5.2 Формат значения NR1

Синтаксис значений NR1 указывает формат целого значения свойства.
Синтаксическое правило:

```
NR1Value    =    'NR1',    ((signedNumber,    variableLengthIndicator)    |
(signedNumber,    space)          variableLengthIndicator      |    space),
lengthOfNumber;
```

Смысл компонентов формата NR1:
- ' NR1 ': значение должно быть целым.

Примечание 1 — Значение NR1 не должно содержать пробелов.

- lengthOfNumber: количество цифр числа.

Примечание 2 — Если указан индикатор длины переменной variableLengthIndicator, то фактическое количество цифр может быть меньше.

- signedNumber: если указано число со знаком signedNumber, то оно должно быть либо положительным, либо отрицательным, либо нулем. Положительное число может иметь знак «+». Отрицательное число имеет знак «–». Нуль не может иметь знак «–».

- variableLengthIndicator: если задан индикатор длины переменной, то количество цифр соответствующего числа не должно превышать заданной длины, т. е. lengthOfNumber.

D.5.3 Формат значения NR2

Синтаксис значения NR2 указывает формат действительного значения свойства, не требующего указания показателя степени.

Синтаксическое правило:

```
NR2Value = 'NR2', ((signedNumber, variableLengthIndicator) |
(signedNumber, space) | variableLengthIndicator | space),
lengthOfIntegralPart, decimalMark, lengthOfFractionalPart;
```

Смысл компонент формата значений NR2 для отображения:

" 'NR2 ': значение должно быть действительным.

Примечание 1 — Численные значения NR2 не должны содержать пробелов.

- lengthOfFractionalPart: число цифр дробной части числа.

Примечание 2 — Если указан индикатор длины переменной variableLengthIndicator, то фактическое количество цифр дробной части числа может быть меньше.

Примечание 3 — Длина дробной части числа lengthOfFractionalPart неявно указывает рекомендованную точность числа. Фактическая точность числа, из которого выведено значение, может быть выше той, что указана здесь.

- lengthOf IntegralPart: количество цифр целой части числа.

Примечание 4 — Если указан индикатор длины переменной variableLengthIndicator, то фактическое количество цифр целой части числа может быть меньше.

- signedNumber: если указано число со знаком signedNumber, то оно должно быть либо положительным, либо отрицательным, либо нулем. У положительного числа спереди может быть плюс «+». У отрицательного числа спереди должен быть минус «–». Перед нулем минус «–» недопустим.
- variableLengthIndicator: если задан индикатор длины переменной variableLengthIndicator, то либо целая часть, либо дробная часть числа, либо обе эти части должны содержать цифры, количество которых не превышает длину, заданную параметрами lengthOfIntegralPart или lengthOfFractionalPart. Число должно содержать по крайней мере одну цифру.

D.5.4 Формат значения NR3

Синтаксис значения NR3 указывает формат действительного значения свойства, представленного в степенной форме.

Синтаксическое правило:

```
NR3Value = 'NR3', ((signedNumber, variableLengthIndicator) |
(signedNumber, space) | variableLengthIndicator | space),
lengthOfIntegralPart, decimalMark, lengthOfFractionalPart,
exponentIndicator, [signedExponent], lengthOfExponent;
```

Смысл компонент формата значений NR3 для отображения:

" 'NR3': значение должно быть действительным с основанием степени 10.

Примечание 1 — Число должно содержать по крайней мере одну цифру и десятичный знак в мантиссе. Показатель степени имеет по крайней мере одну цифру.

Примечание 2 — Значения формата NR3 не имеют пробелов.

- exponentIndicator: разделитель между мантиссой и показателем степени в формате NR3.
- lengthOfExponent: число цифр показателя степени.

Примечание 3 — Если указан индикатор длины переменной variableLengthIndicator, то фактическое количество цифр показателя степени может быть меньше.

Примечание 4 — Случайные плюсы и минусы или десятичные знаки не считаются параметрами lengthOfNumber, lengthOfIntegralPart, lengthOfFractionalPart или lengthOfExponent.

- lengthOfFractionalPart: количество цифр дробной части мантиссы.

Примечание 5 — Если указан индикатор длины переменной variableLengthIndicator, то фактическое количество цифр дробной части числа может быть меньше.

Примечание 6 — Параметр длины дробной части числа lengthOfFractionalPart неявно указывает рекомендованную точность числа. Фактическая точность числа, из которого выводится данное значение, может быть выше, чем значение, указанное здесь.

- lengthOfIntegralPart: количество цифр целой части мантиссы.

Примечание 7 — Если указан индикатор длины переменной variableLengthIndicator, то фактическое количество цифр целой части числа может быть меньше.

- signedExponent: если указан показатель степени со знаком signedExponent, то он может быть положительным, отрицательным или нулем. Положительная степень может быть со знаком «+». Отрицательная степень должна быть со знаком «–». Нуль не может быть со знаком «–».
- variableLengthIndicator: если указан индикатор длины переменной variableLengthIndicator, то количество цифр числа (показателя степени) не может превышать значения, указанного в спецификации, т. е. lengthOfIntegralPart, lengthOfFractionalPart или lengthOfExponent. В мантиссе и в показателе степени должна быть по крайней мере одна цифра.

D.5.5 Формат значения NR4

Синтаксис значения NR4 указывает формат рационального значения свойства, имеющего целую часть и, возможно, дробную часть со знаменателем и числителем.

Синтаксическое правило:

```
NR4Value = 'NR4', ((signedNumber, variableLengthIndicator)    (signedNumber, space)
|   variableLengthIndicator   |   space),   lengthOfIntegerPart,   numeratorIndicator,
lengthOfNumerator, denominatorIndicator, lengthOfDenominator;
```

Смысл компонент формата значений NR4 для отображения:

- 'NR4': значение должно быть рациональным числом, представленным либо как целое, либо как дробь, состоящая из числителя и знаменателя, либо как целое и дробь.

Пример — 12 1/2 и 12 ¼ — это значения, представленные в формате NR4.

Примечание 1 — В целой части или в числителе и в знаменателе должна быть по крайней мере одна цифра. Если одна часть дроби содержит цифру, то другая часть дроби также должна содержать цифру. Все три части числа могут содержать цифры.

Примечание 2 — Формат NR4 не допускает пробелов.

- numeratorIndicator: разделитель между целой частью и дробной частью в формате NR4.
- lengthOfNumerator: количество цифр числителя.

Примечание 3 — Если указан индикатор длины переменной variableLengthIndicator, то фактическое количество цифр числителя может быть меньше.

Примечание 4 — Если значение рационального числа полно представлено своей целой частью, то ни числителя, ни знаменателя дроби может не быть.

- denominatorIndicator: разделитель между числителем и знаменателем дроби в формате NR4.
- lengthOfDenominator: количество цифр знаменателя.

Примечание 5 — Если указан индикатор длины переменной variableLengthIndicator, то фактическое количество цифр знаменателя может быть меньше.

Примечание 6 — Если значение рационального числа полно представлено его целой частью, то ни числителя, ни знаменателя дроби может не быть.

- lengthOfIntegerPart: количество цифр целой части рационального числа.

Примечание 7 — Если указан индикатор длины переменной variableLengthIndicator, то фактическое количество цифр целой части может быть меньше.

- variableLengthIndicator: если указан индикатор длины переменной variableLengthIndicator, то количество цифр во всех трех частях рационального числа не должно превышать значения, указанного в спецификации, т. е. lengthOfIntegralPart, lengthOfNumerator или lengthOfDenominator. В целой части числа или в обеих частях дробной части числа должна быть по крайней мере одна цифра.

D.6 Неколичественные форматы значений

D.6.1 Общие положения

В следующих пяти подразделах определены пять синтаксических правил для неколичественных форматов значений и рассмотрен их смысл. Они используются для свойств со следующими типами данных:

- строчный тип **string_type** или любой его подтип;
- списочный тип **list_type,** тип множества **set_type,** упакованный тип **bag_type,** тип массива **array_type** или тип множества с подмножеством ограничений **set_with_subset_constraint_type**, для которого тип значения **value_type** — это строчный тип **string_type** или любой его подтип.

Примечание 1 — Типы **list_type, set_type, bag_type, array_type** и **set_with_subset_constraint_type** определены в ИСО 13584-25.

Примечание 2 — Для переводимого строчного типа **translatable_string_type** рассматриваемый формат значения применяется для представления строк на заданном языке.

Примечание 3 — Для неколичественного кодового типа **non_quantitative_code_type** формат значения применяется к коду.

Неколичественное значение представляется символьной строкой. Длина строки либо устанавливается прямо (путем прямого указания верхнего предела для количества символов), либо путем указания, что общее число символов может быть любым целым кратным некоторой заданной длине.

Синтаксическое правило:

```
factor = leadingDigit, {trailingDigit};
numberOfCharacters = (leadingDigit, {trailingDigit})|( '(nx', factor,')');
```

Смысл компонент формата значений factor для отображения:

- factor: если factor указан, то количество символов numberOfCharacters должно быть любым целым кратным значению, указанному в factor и не должно содержать значение «нуль».
- numberOfCharacters: определяет максимальное количество символов в строке.

D.6.2 Формат алфавитного значения

«Формат алфавитного значения (A)» определяет формат значения строки, содержащей буквы. Таким образом, содержание строки составляется из символов строки 00, клетки 20, клетки 40 7E и клетки C0 FF базовой многоязыковой плоскости (БМП) (плоскость 00 группы 00), описанной в ИСО/МЭК 10646-1.

Примечание 1 — Из-за потенциальных проблем интеграции содержания значений внутри компонентов одной системы или нескольких систем используемые символы по возможности ограничиваются множеством G0 по ИСО/МЭК 10646-1 и/или строкой 00 и столбцами 002-007 по ИСО/МЭК 10646-1.

Примечание 2 — Для альтернативных языков, используемых для перевода, требуется доступ к некоторым символам или идеограммам особого набора символов (соответствующего языка) в соответствии со стандартом Unicode. В большинстве случаев между символами исходного языка и символами целевого языка имеет место соотношение 1:1.

Пример — Идеограмма CJK (Chinese-Japanese-Korean, т. е. Китай-Япония-Корея).

Синтаксическое правило:

```
AValue = 'A', (space | variableLengthIndicator), numberOfCharacters;
```

Смысл компонент формата A — значений для отображения:

- 'A': значение должно быть строкой, несколькими подстроками или буквами.
- variableLengthIndicator: если указан индикатор длины переменной variableLengthIndicator, то строка может содержать меньше символов, чем указано параметром numberOfCharacters. В строке должен быть по крайней мере один символ.

D.6.3 Формат значения со смешанными символами

«Смешанный формат значения (M)» определяет строку, содержащую любой символ, установленный в разделе D.7.

Примечание — Для альтернативных языков, используемых для перевода, требуется доступ к некоторым символам или идеограммам особого набора символов (соответствующего языка) в соответствии со стандартом Unicode.

Пример — Идеограмма CJK (Chinese-Japanese-Korean, т. е. Китай-Япония-Корея).

Синтаксическое правило:

```
MValue = 'M', (space | variableLengthIndicator), numberOfCharacters;
```

Смысл компонент формата M — значений для отображения:

- 'M': значение должно быть строкой или несколькими подстроками.
- variableLengthIndicator: если указан индикатор длины переменной variableLengthIndicator, то строка может содержать меньше символов, чем указано параметром numberOfCharacters. Строка должна содержать как минимум один символ.

D.6.4 Формат численного значения

«Формат численного значения (N)» определяет строку, содержащую только цифры. Таким образом, необходимое значение составляется из строки 00, клетки 2B, клетки 2D, клеток 30—39 и клетки 45 базовой многоязыковой плоскости (БМП) (плоскость 00 группы 00) в соответствии с ИСО/МЭК 10646-1.

Примечание — Для альтернативных языков, используемых для перевода, необходим доступ к символам или идеограммам особого набора символов соответствующего языка в соответствии со стандартом Unicode.

Пример — Таблица D.2 устанавливает соответствие европейских цифр «0»—«9» и арабских цифр.

Таблица D.2 — Соответствие европейских цифр и арабских цифр

Европейские цифры	9	8	7	6	5	4	3	2	1	0
Арабские цифры	٩	٨	٧	٦	٥	٤	٣	٢	١	٠

Синтаксическое правило:

```
NValue = 'N', (space | variableLengthIndicator), numberOfCharacters;
```

Смысл компонент формата N — значений для отображения:

- 'N': значение должно быть строкой, несколькими подстроками или цифрами.
- variableLengthIndicator: если указан индикатор длины переменной variableLengthIndicator, то строка может содержать меньше символов, чем указано параметром numberOfCharacters. Строка должна содержать по крайней мере один символ.

D.6.5 Смешанный алфавитно-цифровой формат

«Смешанный алфавитно-цифровой формат (X)» определяет строку, содержащую буквенно-цифровые символы, т. е. любую комбинацию символов значения A-формата и N-формата.

Примечание — Для альтернативных языков, используемых для перевода, необходим доступ к символам и идеограммам особого набора символов соответствующего языка в соответствии со стандартом Unicode.

Синтаксическое правило:

```
XValue = 'X', (space | variableLengthIndicator), numberOfCharacters;
```

Смысл компонент формата X — значений для отображения:

- 'X': значение должно быть строкой, несколькими подстроками или буквенно-цифровым выражением, т. е. любой комбинацией букв и цифр.
- variableLengthIndicator: если указан индикатор длины переменной variableLengthIndicator, то строка может содержать меньше символов, чем указано параметром numberOfCharacters. Строка должна содержать по крайней мере один символ.

D.6.6 Формат бинарного значения

«Формат бинарного значения (B)» определяет значение строки, содержащей бинарные символы, т. е. «0» или «1». Таким образом, строка составляется из символов строки 00, клеток 30 или 31 базовой многоязыковой плоскости (БМП) (плоскость 00 группы 00) в соответствии с ИСО/МЭК 10646-1.

Примечание — Для альтернативных языков, используемых для перевода, необходим доступ к символам и идеограммам особого набора символов соответствующего языка в соответствии со стандартом Unicode.

Синтаксическое правило:

```
BValue = 'B', (space | variableLengthIndicator), numberOfCharacters;
```

Смысл компонент формата B — значений для отображения:

- 'B': значение должно быть строкой или несколькими подстроками, содержащими бинарные значения, т. е. символы «0» или «1» или их последовательности.
- variableLengthIndicator: если указан индикатор длины переменной variableLengthIndicator, то строка может содержать меньше символов, чем указано параметром numberOfCharacters. Строка должна содержать по крайней мере один символ.

D.7 Пример задания значения

Таблица D.3 содержит некоторые примеры задания значений, выраженных числами и строками в соответствии с указанными форматами.

Таблица D.3 — Пример задания значения

Формат значения	Пример задания значения
NR1 3	123; 001; 000;
NR1..3	123; 87; 5
NR1S3	+123; +000;
NR1S..3	–123; +1; 0; –12;
NR2 3.3	123.300; 000.400; 000.420;
NR2..3.3	321.233; 1.234; 23.56; 9.783;.72; 324.
NR2S 3.3	–123.123; +123.300;
NR2S..3.3	–123.123; +12.3; 0.1; +.4; –3.; 0.;.0;
NR3 3.3E4	123.123E0004; 003.000E1000;
NR3 3.3ES4	123.123E+0004; 123.123E0004; 123.000E–0001
NR3..3.3E4	123.123E0004;.123E0001; 5.E1234
NR3S 3.3ES4	+ 123.123E+0004; 123.000E–0001;
NR3S..3.3ES4	–123.123E+0004; +1.00E–01 ;.0E0; +3.E–1; –.2E–1000;
NR4 3N2D2	001 02/03; 012 00/01; 123 03/04; 000 01/04
NR4..3N2D2	1 $^{1}/_{2}$; 12; 123 $^{3}/_{4}$; $^{1}/_{4}$
NR4S 3N2D2	+001 02/03; –012 00/01; 123 03/04; –000 01/04
NR4S..3N2D2	–1 $^{1}/_{2}$; 12; +123 $^{3}/_{4}$; $^{1}/_{4}$
A19	My name is Reinhard, abcdefghijklmnopqrs
A..3	Abc; de; G
X..5	A23RN1; B1; ca
M..10	A23RN1; B1; ca. 256 U–m;
N (nx5)	12345; 1234512345; 222223333344444;
N..(nx5)	1234; 12345; 34512345; 1234512345; 23333344444; 222223333344444; –3; 5E2;
B 1	0; 1;
B3	011; 101;

Символы по ИСО/МЭК 10646-1

В смешанном формате значений (M) используются следующие символы (см. D.5.3):

- все символы из строки 00 базовой многоязыковой плоскости (БМП) (плоскости 00 группы 00) в соответствии с ИСО/МЭК 10646-1;
- символы из других строк базовой многоязыковой плоскости в соответствии с ИСО/МЭК 10646-1 (см. таблицу D.4).

Для альтернативных языков, используемых для перевода, необходим доступ к полному набору символов, определенному стандартом Unicode.

Примечание 1 — Из-за потенциальных проблем с интерпретацией значений внутри компонентов одной системы или нескольких систем, рекомендуется по возможности брать символы только из набора G0, определенного ИСО/МЭК 10646-1, и/или из строки 00 и столбцов 002—007 ИСО/МЭК 10646-1.

Примечание 2 — Международный стандарт Unicode опубликован консорциумом Unicode. Адрес: P.O. Box 391476, Mountain View, CA 94039-1476, U.S.A., gost.e-ecolog.ru

Таблица D.4 — Символы из других рядов основной многоязыковой плоскости по ИСО/МЭК 10646-1

Символ	Название	Ряд	Клетка
	CARON («птичка» над буквой как у «Й»)	02	C7
≡	IDENTICAL TO (идентичность множеств)	22	61
∧	LOGICAL AND (логическая операция «И»)	22	27
∨	LOGICAL OR (логическая операция «ИЛИ»)	22	28
∩	INTERSECTION (пересечение множеств)	22	29
∪	UNION (операция «включающее ИЛИ»)	22	2A
⊂	SUBSET OF (включать)	22	82
⊃	SUBSET OF (быть включенным)	22	83
⇐	LEFTWARDS DOUBLE ARROW (из второго следует первое)	21	D0
⇒	RIGHTWARDS DOUBLE ARROW (из первого следует второе)	21	D2
∴	THEREFORE (следовательно)	22	34
∵	BECAUSE (потому что)	22	35
∈	ELEMENT OF (является элементом чего-либо)	22	08
∋	CONTAINS AS MEMBER (включать как элемент)	22	0B
⊆	SUBSET OR EQUAL TO (само множество включается в другое множество как подкласс)	22	86
⊇	SUPERSET OR EQUAL TO (множество включает другое множество как подкласс)	22	87
∫	INTEGRAL (интеграл)	22	2B
∮	CONTOUR INTEGRAL (интеграл по замкнутому контуру)	22	2E
∞	INFINITY (бесконечность)	22	1E
∇	NABLA (дифференциальный набла-оператор)	22	07
∂	PARTIAL DIFFERENTIAL (частная производная)	22	02
~	TILDE OPERATOR (тильда-оператор: разность между)	22	3C
≈	ALMOST EQUAL TO (приближенное равенство)	22	48
≃	ASYMPTOTICALLY EQUAL TO (асимптотическое равенство)	22	43
≅	APPROXIMATELY EQUAL TO (подобие)	22	45
≤	LESS THEN OT EQUAL TO (меньше или равно)	22	64
≠	NOT EQUAL TO (не равно)	22	60
≥	GREATER TAN OR EQUAL TO (больше или равно)	22	65
⇔	LEFT RIGHT DOUBLE ARROW (если и только если)	21	D4
¬	NOT SIGN (знак «не»: не больше, не меньше)	00	AC
∀	FOR ALL (всех)	22	00
∃	THERE EXISTS (существует)	22	03

Продолжение таблицы D.4

Символ	Название	Ряд	Клетка
ℵ	HEBREW LETTER ALEF (буква «алеф» древнееврейского языка)	05	D0
□	WHITE SQUARE (оператор Д'Аламбера)	25	A1
∥	PARALLEL (параллельность)	22	25
Γ	GREEK CAPITAL LETTER GAMMA (греческая прописная буква «гамма»)	03	93
Δ	GREEK CAPITAL LETTER DELTA (греческая прописная буква «дельта»)	03	94
⊥	UPTACK (ортогональность)	22	A5
∠	ANGLE (угол)	22	20
∟	RIGHT ANGLE WITH ARC (правый угол с дугой)	22	BE
θ	GREEK CAPITAL LETTER THETA (греческая прописная буква «тета»)	03	98
〈	LEFT POINTING ANGLE BRACKET (левая угловая скобка)	23	29
〉	RIGHT POINTING ANLE BRACKET (правая угловая скобка)	23	2A
Λ	GREEK CAPITAL LETTER LAMBDA (греческая прописная буква «ламбда»)	03	9B
′	PRIME (знак штриха «прим»)	20	32
″	DOUBLE PRIME (знак двойного штриха «прим»)	20	33
Ξ	GREEK CAPITAL LETTER XI (греческая прописная буква «кси»)	03	9E
∓	MINUS-OR-PLUS SIGN (знак минус-плюс)	22	13
Π	GREEK CAPITAL LETTER PI (греческая прописная буква «пи»)	03	A0
^{2}	SUPERSCRIPT TWO (верхний индекс 2)	00	B2
Σ	GREEK CAPITAL LETTER SIGMA (греческая прописная буква «сигма»)	03	A3
×	MULTIPLICATION SIGN (умножение)	00	D7
^{3}	SUPERSCRIPT THREE (верхний индекс 3)	00	B3
ϒ	GREEK CAPITAL LETTER UPSILON (греческая прописная буква «ипсилон»)	03	A5
Φ	GREEK CAPITAL LETTER PHI (греческая прописная буква «фи»)	03	A6
·	MIDDLE DOT (точка-разделитель)	00	B7
Ψ	GREEK CAPITAL LETTER PSI (греческая прописная буква «пси»)	03	A8
Ω	GREEK CAPITAL LETTER OMEGA (греческая прописная буква «омега»)	03	A9
Ø	EMPTY SET (пустое множество)	22	05
⇀	RIGHTWARDS HARPOON WITH BARB UPWARDS («правый гарпун с зазубриной сверху» — черта над вектором)	21	C0
√	SQUARE ROOT (квадратный корень — радикал)	22	1A
f	LATIN SMALL LETTER F WITH HOOK (латинская строчная буква «f с крючком» — обозначение функции)	01	92
∝	PROPORTIONAL (пропорциональность)	22	1D
+	PLUS-MINUS SIGN (знак плюс-минус)	00	B1

Окончание таблицы D.4

Символ	Название	Ряд	Клетка
°	DEGREE SIGN (градус)	00	B0
α	GREEK SMALL LETTER ALPHA (греческая строчная буква «альфа»)	03	B1
β	GREEK SMALL LETTER BETA (греческая строчная буква «бета»)	03	B2
γ	GREEK SMALL LETTER GAMMA (греческая строчная буква «гамма»)	03	B3
δ	GREEK SMALL LETTER DELTA (греческая строчная буква «дельта»)	03	B4
ε	GREEK SMALL LETTER EPSILON (греческая строчная буква «эпсилон»)	03	B5
ζ	GREEK SMALL LETTER ZETA (греческая строчная буква «дзета»)	03	B6
η	GREEK SMALL LETTER ETA (греческая строчная буква «эта»)	03	B7
θ	GREEK SMALL LETTER THETA (греческая строчная буква «тэта»)	03	B8
ι	GREEK SMALL LETTER IOTA (греческая строчная буква «иота»)	03	B9
κ	GREEK SMALL LETTER KAPPA (греческая строчная буква «каппа»)	03	BA
λ	GREEK SMALL LETTER LAMBDA (греческая строчная буква «ламбда»)	03	BB
μ	GREEK SMALL LETTER MU (греческая строчная буква «мю»)	03	BC
ν	GREEK SMALL LETTER NU (греческая строчная буква «ню»)	03	BD
ξ	GREEK SMALL LETTER XI (греческая строчная буква «кси»)	03	BE
‰	PER MILLE SIGN (на тысячу, промилле)	20	30
π	GREEK SMALL LETTER PI (греческая строчная буква «пи»)	03	C0
ρ	GREEK SMALL LETTER RHO (греческая строчная буква «ро»)	03	C1
σ	GREEK SMALL LETTER SIGMA (греческая строчная буква «сигма»)	03	C3
÷	DIVISION SIGN (деление)	03	F7
τ	GREEK SMALL LETTER TAU (греческая строчная буква «тау»)	03	C4
υ	GREEK SMALL LETTER UPSILON (греческая строчная буква «ипсилон»)	03	C5
φ	GREEK SMALL LETTER PHI (греческая строчная буква «фи»)	03	C6
χ	GREEK SMALL LETTER CHI (греческая строчная буква «хи»)	03	C7
ψ	GREEK SMALL LETTER PSI (греческая строчная буква «пси»)	03	C8
ω	GREEK SMALL LETTER OMEGA (греческая строчная буква «омега»)	03	C9
†	DAGGER (знак ссылки)	20	20
←	LEFTWARDS ARROW (стрелка влево)	21	90
↑	UPWARDS ARROW (стрелка вверх)	21	91
→	RIGHTWARDS ARROW (стрелка вправо)	21	92
↓	DOWNWARDS ARROW (стрелка вниз)	21	93
¯	OVERLINE (черта над символом)	20	3E

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
МЭК 61360-1:2009	—	*
МЭК 61360-4-DB	—	*
ИСО/МЭК 8859-1:1998	—	*
ИСО/МЭК 10646:2012	—	*
ИСО/МЭК 14977:1996	—	*
ИСО 639 (все части)	—	*
ИСО 843:1997	—	*
ИСО 3166-1:2006	—	*
ИСО 4217:2008	—	*
ИСО 8601:2004	—	*
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		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 10303-41:2005	—	*
ИСО 13584-26:2000	IDT	ГОСТ Р ИСО 13584-26—2006 «Системы автоматизации производства и их интеграция. Библиотека деталей. Часть 26. Логический ресурс. Идентификация поставщика информации»
ИСО 13584-42:2010	IDT	ГОСТ Р ИСО 13584-42—2012 «Системы промышленной автоматизации и интеграция. Библиотека деталей. Часть 42. Методология описания. Методология структурирования семейств деталей»

* Соответствующий национальный стандарт отсутствует. До его утверждения рекомендуется использовать перевод на русский язык данного международного стандарта. Перевод данного международного стандарта находится в Федеральном информационном фонде технических регламентов и стандартов.

Примечание — В настоящей таблице использовано следующее условное обозначение степени соответствия стандартов:

IDT — идентичные стандарты.

УДК 681.5:006.354 ОКС 31.020 Т58

Ключевые слова: автоматизированные промышленные системы, управление производством, словарная схема EXPRESS, базовая семантическая единица, схема языкового ресурса, представление и обмен данными, модель EXPRESS

Редактор *Е.А. Черепко*
Технический редактор *В.Н. Прусакова*
Корректор *Е.Р. Ароян*
Компьютерная верстка *Ю.В. Половой*

Сдано в набор 09.11.2015. Подписано в печать 25.02.2016. Формат 60 × 84 1/8. Гарнитура Ариал.
Усл. печ. л. 18,14. Уч.-изд. л. 16,50. Тираж 30 экз. Зак. 616.

Набрано в ИД «Юриспруденция», 115419, Москва, ул. Орджоникидзе, 11.
gost.e-ecolog.ru gost.e-ecolog.ru
Издано и отпечатано во
ФГУП «СТАНДАРТИНФОРМ», 123995 Москва, Гранатный пер., 4.
gost.e-ecolog.ru gost.e-ecolog.ru

gost.e-ecolog.ru