НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

Информационная технология

ОТКРЫТАЯ РАСПРЕДЕЛЕННАЯ ОБРАБОТКА БАЗОВАЯ МОДЕЛЬ

Часть 4

Архитектурная семантика

Издание официальное

БЗ 4—2002/61

ГОССТАНДАРТ РОССИИ
Москва

gost.e-ecolog.ru

Предисловие

1 РАЗРАБОТАН Государственным научно-исследовательским и конструкторско-технологическим институтом «ТЕСТ» Министерства Российской Федерации по связи и информатизации

ВНЕСЕН Министерством Российской Федерации по связи и информатизации

2 УТВЕРЖДЕН И ВВЕДЕН В ДЕЙСТВИЕ Постановлением Госстандарта России от 4 февраля 2004 г. № 51-ст

3 Настоящий стандарт идентичен международному стандарту ИСО/МЭК 10746-4—98 «Информационная технология. Открытая распределенная обработка. Базовая модель. Часть 4. Архитектурная семантика»

4 ВВЕДЕН ВПЕРВЫЕ

© ИПК Издательство стандартов, 2004

Настоящий стандарт не может быть полностью или частично воспроизведен, тиражирован и распространен в качестве официального издания без разрешения Госстандарта России

Содержание

1 Область применения 1
2 Нормативные ссылки. 2
3 Определения 2
3.1 Определения по ИСО/МЭК 8807. 2
3.2 Определения по Рекомендации МСЭ-Т Z.100 2
3.3 Определения Z 3
3.4 Определения по ИСО/МЭК 9074. 3
3.5 Сокращения 3
4 Интерпретация моделирующих понятий 3
4.1 Архитектурная семантика в LOTOS 3
4.2 Архитектурная семантика в ACT ONE. 9
4.3 Архитектурная семантика в SDL 15
4.4 Архитектурная семантика в Z 20
4.5 Архитектурная семантика в ESTELLE 26

НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

Информационная технология

ОТКРЫТАЯ РАСПРЕДЕЛЕННАЯ ОБРАБОТКА БАЗОВАЯ МОДЕЛЬ

Часть 4

Архитектурная семантика

Information technology. Open Distributed Processing. Reference Model. Part 4. Architectural semantics

Дата введения 2005—01—01

1 Область применения

Прогресс в области распределенной обработки привел к необходимости координации стандартов по открытой распределенной обработке (ОРО). Базовая модель ОРО предоставляет необходимый для этого каркас, а также устанавливает архитектуру, в которой могут быть скоординированы поддержка распределения, взаимодействия, совместимости и переносимости.

Базовая модель открытой распределенной обработки (БМ-ОРО) основана на строгих понятиях, полученных из анализа современных разработок распределенной обработки, и, насколько возможно, на методах формального описания для спецификации архитектуры.

БМ-ОРО состоит из:

- ГОСТ Р ИСО/МЭК 10746-1, который содержит обзор основных причин создания ОРО, описывает область действия, обосновывает и объясняет ключевые понятия и очерчивает архитектуру ОРО. Эта часть не является нормативной;
- ГОСТ Р ИСО/МЭК 10746-2, который содержит определения понятий, аналитический каркас и обозначения для нормализованного описания (произвольных) систем распределенной обработки. Уровень подробности достаточен для обеспечения ГОСТ Р ИСО/МЭК 10746-3 и установления требований к новым методам спецификации. Эта часть является нормативной;
- ГОСТ Р ИСО/МЭК 10746-3, который содержит спецификацию обязательных характеристик, позволяющих квалифицировать распределенную обработку как открытую, а также устанавливает ограничения, которым должны соответствовать стандарты ОРО, и использует методы описания по ГОСТ Р ИСО/МЭК 10746-2. Эта часть является нормативной;
- настоящего стандарта, который содержит формализацию моделирующих понятий, определенных в ГОСТ Р ИСО/МЭК 10746-2, разделы 8 и 9, и формализацию языков точек зрения, определенных в ГОСТ Р ИСО/МЭК 10746-3. Формализация достигается путем интерпретации каждого понятия в терминах конструкций различных стандартизованных методов формального описания. Эта часть является нормативной.

Целью настоящего стандарта является установление архитектурной семантики ОРО. Она имеет вид интерпретации основных моделирующих и специфицирующих понятий ГОСТ Р ИСО/МЭК 10746-2 и языков точек зрения ГОСТ Р ИСО/МЭК 10746-3 с использованием различных средств различных языков формальных спецификаций. Архитектурная семантика разработана для четырех языков формальных спецификаций: LOTOS, ESTELLE, SDL и Z. Результатом разработки является формализация архитектуры ОРО. В стандарте показана согласованность с ГОСТ Р ИСО/МЭК 10746-2.

Архитектурная семантика дает дополнительные преимущества за счет:

- содействия точной и единообразной разработке формальных описаний систем ОРО;

Издание официальное

- возможности единообразного и согласованного сравнения формальных описаний одного и того же стандарта на разных языках формальных спецификаций.

Особое внимание в настоящем стандарте обращается на отображение самых основных (но не всех) понятий ГОСТ Р ИСО/МЭК 10746-2. Семантика архитектурных понятий высших уровней устанавливается косвенно, через их определения в терминах основных понятий ОРО.

Примеры использования некоторых языков формальных описаний приведены в ИСО/МЭК ТО 10167.

В последующих разделах понятия нумеруются в соответствии со схемой, использованной в ГОСТ Р ИСО/МЭК 10746-2.

Настоящий стандарт устанавливает архитектурную семантику ОРО, необходимую:

- для обеспечения формализации моделирующих понятий ОРО;
- для содействия точной и единообразной разработке формальных описаний в стандартах по распределенным системам;
- для связи между моделирующими понятиями ОРО и семантическими моделями языков спецификаций LOTOS, ESTELLE, SDL и Z;
- для возможности единообразного и согласованного сравнения формальных описаний одного и того же стандарта на разных языках формальных спецификаций, которые использовались для разработки архитектурной семантики.

Настоящий стандарт является нормативным.

2 Нормативные ссылки

В настоящем стандарте использованы ссылки на следующие стандарты:

ГОСТ Р ИСО/МЭК 10746-1—2004 Информационная технология. Открытая распределенная обработка. Базовая модель. Часть 1. Основные положения

ГОСТ Р ИСО/МЭК 10746-2—2000 Информационная технология. Открытая распределенная обработка. Базовая модель. Часть 2. Модель

ГОСТ Р ИСО/МЭК 10746-3—2001 Информационная технология. Открытая распределенная обработка. Базовая модель. Часть 3. Архитектура

ИСО/МЭК 8807—89*) Системы обработки информации. Открытая распределенная обработка. LOTOS — метод формального описания, основанный на временном упорядочении наблюдаемого поведения

ИСО/МЭК 9074—97*) Информационная технология. Открытая распределенная обработка. ESTELLE — метод формального описания, основанный на расширенной модели переходов между состояниями

ИСО/МЭК ТО 10167—91*) Информационная технология. Открытая распределенная обработка. Руководство по применению ESTELLE, LOTOS и SDL

Рекомендация МСЭ-Т Z.100 (1999) Язык спецификации и описаний (SDL)

Рекомендация МСЭ-Т Z.105 (1999) Комбинация SDL с модулями ACH.1

Рекомендация МСЭ-Т Z.120 (1999) Диаграмма последовательности сообщений (MSC)

3 Определения

3.1 Определения по ИСО/МЭК 8807

В настоящем стандарте применены следующие термины:

обозначение действия, актуализация параметров, выражение поведения, выбор, соответствие, запрещение, разрешение, обогащение, уравнение, событие, расширение, список формальных ворот, список формальных параметров, ворота, сокрытие ворот, предохранитель, реализация, перекрытие, внутреннее событие, наблюдаемое событие, операция, параллельная композиция, определение параметризованного типа, определение процесса, уменьшение, предикат выбора, сортировка, синхронизация, определение типа, список значений параметров.

3.2 Определения по Рекомендации МСЭ-Т Z.100

В настоящем стандарте применены следующие термины:

предложение действия, блок (тип), вызов, канал, параметр содержимого, непрерывный сигнал, разрешающее условие, экспорт, экспортированная процедура, экспортированная переменная, конечный,

*) Международные стандарты — во ВНИИКИ Госстандарта России.

ворота, импорт, импортированная переменная, ввод, вывод, процедура, процесс (тип), предоставленный, переопределенный, удаленная процедура, переустановка, возврат, обнаруженная переменная, услуга (тип), множество, сигнал, сигнальный путь, стоп, система (тип), задача, время, таймер, переход, вид, видимая переменная, виртуальный.

3.3 Определения Z

В настоящем стандарте применены следующие термины:

аксиоматическое описание, сопряжение, уточнение данных, инвариант, уточнение операции, перезапись, постусловие, предусловие, схема (операции, состояний, кадрирования), вычисление схемы, композиция схем.

3.4 Определения по ИСО/МЭК 9074

В настоящем стандарте применены следующие термины:

деятельность, предложение присваивания, присоединение, канал, определение канала, соединение, состояние контроля, раздел-DELAY, отсоединение, рассоединение, экспортированная переменная, внешняя точка взаимодействия, раздел-FROM, функция, реализация, взаимодействие, точка взаимодействия, определение тела модуля, определение заголовка модуля, экземпляр модуля, вывод, родительский экземпляр, примитивная процедура, процедура, раздел-PROVIDED, освобождение, роль, раздел-TO, переход, блок переходов, раздел переходов, раздел-WHEN.

3.5 Сокращения

В настоящем стандарте применены следующие сокращения:

АСН.1 — абстрактная синтаксическая нотация версии 1;

БМ-ОРО — базовая модель ОРО;

ДЗП — диаграмма записи последовательностей;

ИСО — Международная организация по стандартизации;

ОРО — открытая распределенная обработка;

ПК — подкомитет;

РГ — рабочая группа;

ЯФС — язык формальных спецификаций;

SDL — Specification and Description Language (язык спецификаций и описаний).

4 Интерпретация моделирующих понятий

4.1 Архитектурная семантика в LOTOS

LOTOS является стандартизованным (ИСО/МЭК 8807) языком формальных спецификаций (ЯФС). Методические материалы приведены в указанном стандарте.

В настоящем разделе объясняется, как основные моделирующие понятия могут быть выражены на LOTOS. Нужно отметить, что в LOTOS существует два основных способа моделирования понятий, установленных в ГОСТ Р ИСО/МЭК 10746-2. Один из них основан на алгебраической части языка, другой — на типе данных ACT ONE. Так как формализация понятий с помощью ACT ONE применима и для SDL, то эта формализация приведена в отдельном подразделе (см. 4.2).

Во избежание путаницы в терминологии ОРО и LOTOS в последующих подразделах курсивом выделены специфические для LOTOS термины.

4.1.1 Основные моделирующие понятия

4.1.1.1 *Объект*

Реализация определения процесса LOTOS, на которую можно однозначно ссылаться.

4.1.1.2 *Среда (объекта)*

Часть модели, которая не является частью объекта. В LOTOS среда объекта в данный момент времени в спецификации задается средой спецификации и другими *выражениями поведения*, которые сочленены с данным объектом в данное время.

Примечание — Среда спецификации пуста, если спецификация не параметризована.

4.1.1.3 *Действие*

В LOTOS действия моделируются либо как *внутренние*, либо как *наблюдаемые события*. Все события в LOTOS являются элементарными. Внутреннее действие может быть явно выражено символом *внутреннего события* i или совершением события, с которым связаны *скрытые* от среды *ворота*.

Взаимодействие представляется в LOTOS *синхронизацией* между двумя или несколькими *выра-*

жениями поведения, связанными с объектами в общей точке взаимодействия (*воротах*). Взаимодействия могут быть следующих видов:

- чистая *синхронизация* в общих *воротах* без представления: передачи значений между объектами не происходит;
- ! и ! для чистой *синхронизации:* нет обмена значениями между объектами;
- ! и ? для передачи значения при условии, что событие ? содержит событие !: другой способ рассмотрения — событие ! выбирает значение из значений для события?;
- ? и ? для установления значения: результатом является согласование значения из пересечения множеств значения. Если это пересечение является пустым множеством, то нет *синхронизации* и, следовательно, взаимодействие не происходит.

Если требуется неэлементарное дробление действий, то можно использовать уточнение событий. Это позволит моделировать нереализуемые и перекрывающиеся действия. Следует заметить, что уточнение событий является нетривиальной задачей, особенно когда должна обеспечиваться поведенческая совместимость.

В LOTOS нет конструкций для выражения взаимосвязей причина — результат, однако иногда они могут быть выражены неформально.

4.1.1.4 *Интерфейс*

Это — абстракция поведения объекта, которая состоит из подмножества наблюдаемых действий данного объекта. Так как всем наблюдаемым действиям объекта в LOTOS требуются *ворота*, синхронизированные со средой, то обычно подмножество наблюдаемых действий получается делением *ворот*, заданных в *определении процесса*, связанного с объектом. Для получения рассматриваемого интерфейса не обязательно доходить до *скрытых ворот*. Альтернативно можно использовать *синхронизацию* на подмножестве *ворот*, связанных с объектом. В этом случае действия, происходящие через те *ворота* в *определении процесса*, которые не входят в синхронизированное множество, могут рассматриваться как внутренние действия объекта до тех пор, пока рассматривается образующая интерфейс синхронизация со средой в этих *воротах*.

Следует учитывать, что данное определение требует, чтобы интерфейсы объекта использовали разные имена *ворот*, так как невозможно различать интерфейсы, использующие одни и те же *ворота*.

4.1.1.5 *Деятельность*

Деятельность является однонаправленным ациклическим графом действий, в котором каждый узел представляет состояние системы, а каждая дуга — действие. Для того чтобы действие могло осуществиться, должны быть выполнены предусловия состояния системы.

4.1.1.6 *Поведение (объекта)*

Поведение объекта определяется в LOTOS *выражением поведения,* связанным с *определением процесса,* которое образует шаблон объекта. *Выражение поведения* может состоять из последовательности как видимых внешне событий, так и *внутренних событий*. Фактическое поведение объекта, которое может быть записано при трассировке, зависит от *выражения поведения,* связанного с объектом, и от того, как он конфигурирован со средой. Фактическое поведение объекта показывает зависимость от *выражения поведения* объекта и от того, как оно синхронизировано со средой. Объект может демонстрировать недетерминированное поведение.

4.1.1.7 *Состояние (объекта)*

Условие объекта, определяющее множество всех последовательностей действий, в которых может участвовать объект. Этим условием управляет *выражение поведения*, определенное в шаблоне объекта, из которого был создан объект, и, возможно, текущими значениями любых существующих локальных переменных.

4.1.1.8 *Коммуникация*

Перенос информации (через передаваемые значения) между двумя или несколькими взаимодействующими объектами. Невозможно выразить непосредственно взаимоотношения причина — результат. Следует отметить, что *синхронизация* сама может быть построена как коммуникация.

4.1.1.9 *Положение в пространстве*

LOTOS абстрагируется от понятия положения в пространстве. Можно отождествлять пространство со структурой модели спецификации. Положение события (структурное положение относительно модели спецификации) задается в LOTOS *воротами* для взаимодействий. От понятия положения в пространстве, в котором может произойти *внутреннее событие,* LOTOS полностью абстрагирован. Эта абстракция достигается неявным использованием в LOTOS конструкции «*скрыть . . . в*», которая делает *ворота*, используемые внутри процесса, невидимыми для среды процесса, или явным использованием символа *внутреннего события* i.

Некоторое положение в пространстве может использовать несколько точек взаимодействия. В LOTOS это возможно с помощью единственных *ворот* с разными *обозначениями действий*.

Положение объекта задается объединением положений связанных с ним *ворот*, т. е. объединением всех положений действий, в которых объект может участвовать.

4.1.1.10 *Положение во времени*

LOTOS абстрагируется от понятия времени, рассматривая только временно́й порядок; таким образом, нет *абсолютного положения в реальном метрическом времени*. Однако положение во времени возможно, если расширить LOTOS за счет включения вопросов времени.

4.1.1.11 *Точка взаимодействия*

Ворота с (возможно пустым) списком ассоциированных значений.

Примечание — В спецификации изменения положения могут отражаться в ассоциированных значениях.

4.1.2 Специфицирующие понятия

4.1.2.1 *Композиция*

- **Композиция объектов** — составной объект, описанный путем применения одного или нескольких операторов комбинации LOTOS. В их число входят

оператор чередования (| | |),

операторы параллельной композиции (| | и | [*список-ворот*]|),

оператор разрешения (>>),

оператор запрета ([>),

оператор выбора ([]).

- **Композиция поведений** — композиция *выражений поведения*, связанных с объектами-компонентами при создании составного объекта с помощью композиции. Операторы для композиции поведений те же самые, что и для композиции объектов.

4.1.2.2 *Составной объект*

Объект, описанный с использованием одного или нескольких *операторов чередования, параллельной композиции, разрешения, запрета* и *выбора*.

4.1.2.3 *Декомпозиция*

- **Декомпозиция объектов** — выражение данного объекта как составного. Может быть несколько способов *декомпозиции объекта*.

- **Декомпозиция поведений** — выражение данного поведения как составного. Может быть несколько способов декомпозиции поведения.

Примечание — Также можно считать, что представление декомпозиции поведения обеспечивается *операциями* ACT ONE и уравнениями, связанными с *сортом*. Эти *операции* и *уравнения* дают все возможные комбинации поведения. Так, например, последовательная композиция может быть создана последовательным применением *операций*. Каждое применение операции в последовательности должно удовлетворять обязательным для появления уравнениям. Хотя остается спорным, является ли это композицией поведений, так как *операции* и уравнения уже существуют и определяют все возможные поведения.

4.1.2.4 *Поведенческая совместимость*

В LOTOS были разработаны специальные теории для проверки поведенческой совместимости. Нет специфических для языка LOTOS синтаксических особенностей для построения и гарантии поведенческой совместимости в общем случае. Однако в стандарте LOTOS разработано представление *соответствия*, которое обеспечивает основу для рассмотрения поведенческой совместимости.

Для определения того, являются ли два объекта поведенчески совместимыми, должно быть введено представление *соответствия*. *Соответствие* нацелено на оценку функционирования реализации относительно спецификации; здесь термин реализация может рассматриваться как менее абстрактное описание спецификации.

Если **P** и **Q** — два процесса LOTOS, то утверждение «**Q** соответствует **P**» (записывается «**Q** conf **P**») означает, что **Q** является допустимой реализацией **P**. Это, в свою очередь, означает, что если **P** может пройти по некоторой трассе σ и ведет себя при этом как некоторый процесс **P'** и **Q** может пройти по трассе σ и ведет себя как **Q'**,то для **P'** и **Q'** должны выполняться следующие условия: когда **Q'** может отказаться от осуществления любого события из данного множества наблюдаемых действий **A**, тогда **P'** также может отказаться от осуществления любого события из **A**.

Таким образом, **Q** conf **P** только в том случае, если **Q**, помещенный в среду, допускающую только трассы для **P**, не может привести к взаимоблокировке, когда **P** не может этого сделать. Другой способ определения: **Q** имеет взаимоблокировки **P** в среде, трассы которой ограничены трассами **P**.

Объект может быть сделан поведенчески совместимым с другим объектом после некоторой модификации его поведения, которая может включать в себя *расширение* (добавление нового поведения) или *сужение* (ограничение) поведения объекта. Этот процесс модификации объекта называется уточнением (см. 4.1.2.5).

4.1.2.5 *Уточнение*

Уточнение является процессом, с помощью которого объект может быть модифицирован путем расширения или сужения его поведения (либо комбинацией этих способов) таким образом, чтобы соответствовать другому объекту. Если **P** и **Q** — процессы LOTOS, то «**Q** *расширяет* **P**» (записывается «**Q extends P**») означает, что **Q** имеет не меньше трасс, чем **P**, но в среде, трассы которой ограничиваются трассами **P, Q** имеет те же самые взаимоблокировки. «**Q** *суживает* **P**» (записывается «**Q reduce P**») означает, что **Q** имеет не больше трасс, чем **P**, но в среде, трассы которой ограничиваются трассами **Q, P** имеет те же самые взаимоблокировки.

4.1.2.6 *Трасса*

Трасса поведения объекта из его начального состояния в некоторое другое является записью конечной последовательности взаимодействий (*наблюдаемых событий*) между объектом и его средой.

4.1.2.7 *Тип <X>*

Типы, которые могут быть явно записаны в LOTOS для объектов и интерфейсов, являются типами шаблонов. В LOTOS нет явных конструкций, позволяющих моделировать типы действий как таковые. Спецификация LOTOS состоит из *выражения поведения*, которое само состоит из *обозначений действий* (шаблонов действий). Эти шаблоны действий либо проявляются как часть поведения системы, и в этом случае их появление может неформально рассматриваться как реализация шаблона действия, либо не проявляются, и в этом случае шаблон действия остается нереализованным. Сами шаблоны действий могут задаваться либо символом *внутреннего события* i либо представлениями событий в воротах, которые могут иметь или не иметь конечной последовательности значений и (или) деклараций переменных.

LOTOS не предоставляет возможностей для непосредственной характеристики действий, но ограниченная форма характеристики действия встроена в *синхронизацию* LOTOS. А именно, можно считать, что синхронизированные *обозначения действий* (шаблоны действий) должны соответствовать одному и тому же типу действия для того, чтобы действие могло произойти. Однако LOTOS не классифицирует характеризующие признаки этих произвольных *обозначений действий* и, таким образом, нет возможности ввести формальный тип для любого заданного действия. Может встретиться случай, когда неформально представлению события, участвующему во взаимодействии, приданы роли причины и результата, но в общем случае это не так. См. 4.1.1.8.

Символ *внутреннего события* может использоваться для представления типа действия, когда общая характеристика этой совокупности действий состоит в том, что они не имеют характеристик.

Следует заметить, что если ограничиться единственным возможным в LOTOS предикатом для объектов (и интерфейсов), которые соответствуют типу шаблона, то понятия типа и шаблона, определенные в ГОСТ Р ИСО/МЭК 10746-2, суживаются до методов моделирования в LOTOS. В LOTOS нет разницы между типом в широком смысле слова и типом шаблона в более узком смысле реализации шаблона.

4.1.2.8 *Класс <X>*

Представление класса есть зависимость от характеризующего тип предиката, которому должны удовлетворять члены класса. Объекты, интерфейсы и действия могут удовлетворять многим произвольным характеризующим тип предикатам. Тип, который может быть в нем записан, является типом шаблона. Класс объектов, интерфейсов или действий, связанных с этим типом, является классом шаблонов.

Примечание — Следует заметить, что если ограничиться единственной возможной в LOTOS классификацией объектов, интерфейсов и действий, которые соответствуют типу шаблона, то понятия класса и класса шаблонов, определенные в ГОСТ Р ИСО/МЭК 10746-2, суживаются до методов моделирования в LOTOS. В LOTOS нет разницы между классом в широком смысле слова и классом шаблонов в более узком смысле как множества реализаций данного типа шаблонов.

4.1.2.9 *Подтип/супертип*

Так как типы, которые могут быть написаны в LOTOS для объектов, интерфейсов и, отчасти, действий, являются типами шаблонов, то отношение подтипа в LOTOS является отношением, которое может существовать между типами шаблонов. Однако в LOTOS существует непрямой способ записи отношения подтипа непосредственно. Если требуется отношение подтипа, то можно исполь-

зовать *расширение* для получения этого отношения на основе возможности подстановки, но этот способ в LOTOS не предназначен для таких целей.

4.1.2.10 *Подкласс/суперкласс*

Так как типы, которые могут быть написаны в LOTOS для объектов, интерфейсов и, отчасти, действий, являются типами шаблонов, то отношение подкласса существует между двумя классами, когда отношение подтипа существует между двумя соответствующими типами шаблонов.

4.1.2.11 *Шаблон <X>*

Шаблон объектов — *определение процесса* с некоторым средством, которое позволяет однозначно идентифицировать реализацию процесса. Если не задан список параметров, то будет невозможна идентификация объекта для более чем одной реализации из этого шаблона объектов.

Что касается комбинации шаблонов объектов в LOTOS, то не существует операторов комбинации, за исключением ограниченного вида области действия, использующей термин LOTOS «where».

Шаблон интерфейсов — любое поведение, полученное из *определения процесса* при рассмотрении только взаимодействий на подмножестве *ворот*, связанных с *определением процесса*. Это подмножество ворот получается путем *сокрытия ворот*, не нужных для рассматриваемых взаимодействий.

Что касается комбинации шаблонов интерфейсов в LOTOS, то не существует операторов комбинации, за исключением ограниченного вида области действия, использующей термин LOTOS «where».

Шаблон действий — *обозначение действия*, которое может быть символом *внутреннего события*, идентификатором ворот или идентификатором ворот с конечной последовательностью значений и (или) деклараций переменных.

Примечание — Здесь определение *обозначение действия* является придуманным, так как LOTOS фактически не обеспечивает понятие шаблона действия. В LOTOS возможное поведение специфицируется заданием скомбинированных в некотором виде *обозначений действий*. Аналогия между шаблоном и *обозначением действия* — лучшее, что может быть сделано в LOTOS. Однако ГОСТ Р ИСО/МЭК 10746-2 требует, чтобы шаблоны действий были сгруппированы по характеристикам действий. Этого нет в LOTOS, так как представления событий (*обозначения действий*) существуют изолированно, их невозможно собрать вместе и применить шаблон для их характеристики.

Композиция шаблонов действий может быть связана с *синхронизацией* с передачей или генерацией значения. В этом случае два (или несколько) шаблонов действий согласуют общий шаблон действия для осуществления *синхронизации*, т. е. шаблон действия с общими характеристиками всех участвующих в *синхронизации* (композиции) шаблонов действий.

4.1.2.12 *Сигнатура интерфейса*

Сигнатура интерфейса как набор шаблонов действий, связанных с взаимодействиями интерфейса, представляется в LOTOS как набор *обозначений действий*. Члены этого набора являются теми *обозначениями действий*, которым для осуществления требуется *синхронизация* со средой.

4.1.2.13 *Реализация (шаблона <X>)*

Реализация шаблона объекта — результат процесса, который использует шаблон объекта для создания нового объекта в начальном состоянии. Этот процесс включает в себя *актуализацию формального списка ворот* и *формальных параметров определения процесса* подстановкой из заданного списка ворот и списка фактических параметров. Характеристиками создаваемого объекта управляют шаблон объекта и параметры, использованные при его реализации.

Реализация шаблона интерфейса — результат процесса, которым из шаблона интерфейса создается интерфейс. Создаваемый интерфейс может в дальнейшем использоваться объектом, с которым он связан, для взаимодействия со средой. Характеристики создаваемого интерфейса определяются шаблоном интерфейса и параметрами, использованными при его реализации.

Реализация шаблона действия — осуществление действия в LOTOS. Оно может включать в себя перезапись выражений ACT ONE.

4.1.2.14 *Роль*

Имя, связанное с *определением процесса* в шаблоне для составного объекта (т. е. композиция LOTOS *выражений поведения*). Роль не может быть использована в качестве параметра. Однако каждой роли в композиции можно присвоить значения данных для их различения или адресации.

4.1.2.15 *Создание (<X>)*

Создание объекта — реализация шаблона объекта как часть поведения существующего объекта.

Создание интерфейса — так как объекты и интерфейсы моделируются в LOTOS одинаково

(через *определения процессов*), создание объектов соответствует созданию интерфейсов. Таким образом, определение создания интерфейса дано определением создания объекта.

4.1.2.16 *Введение (объекта)*

Реализация поведения, связанного со спецификацией на LOTOS.

4.1.2.17 *Удаление (<X>)*

Удаление объекта — прекращение процесса *реализации*. Оно может быть достигнуто использованием *запрещающего оператора* LOTOS, остановкой (*стоп*) *выражения поведения*, которая не допускает передачи управления, или успешным завершением (*выход*) *выражения поведения*, когда передача управления возможна через *разрешающий оператор*.

Удаление интерфейса — процесс, которым будущее поведение объекта ограничивается в том, что этому поведению не требуется удаляемый интерфейс.

4.1.2.18 *Экземпляр типа*

Экземпляр типа шаблона объекта — экземпляр данного шаблона объекта представляется в LOTOS реализацией этого шаблона объекта или приемлемой подстановкой для реализации этого шаблона объекта. Приемлемая подстановка должна охватывать характеристики, идентифицирующие тип. Таким образом, приемлемой подстановкой может быть другой шаблон, поведенчески совместимый с первым. Он может быть получен путем *расширения*, как описано в 4.1.2.4. Использование этого отношения гарантирует, что включены все характеристики рассматриваемого типа. Однако возможен случай, когда могут быть найдены различные формы соответствия типу, которым требуются не все характеристики, связанные с данным шаблоном, а только некоторое их подмножество.

Экземпляр типа шаблона интерфейса — так как шаблон интерфейса представляется точно так же, как и шаблон объекта (через *определение процесса* в LOTOS), то сказанное выше применимо (с заменой термина «объект» на «интерфейс») к реализации шаблона интерфейса.

Экземпляр типа шаблона действия — экземпляр шаблона действия (*обозначение действия*) представляется в LOTOS другим *обозначением действия*, предлагающим эквивалентное событие.

4.1.2.19 *Тип шаблона (<X>)*

Предикат, выражающий, что <*X*> является экземпляром данного шаблона, где <*X*> может быть объектом, интерфейсом или действием.

4.1.2.20 *Класс шаблонов (<X>)*

Класс шаблона <*X*> — это множество всех <*X*>, которые являются экземплярами этого шаблона <*X*>, где <*X*> может быть объектом, интерфейсом или действием.

Примечание — Применение класса шаблона действия ограничено в LOTOS, так как LOTOS не обеспечивает явно шаблоны действий, реализации и типы шаблонов действий.

4.1.2.21 *Производный класс/базовый класс*

Если шаблон класса является нарастающей модификацией шаблона другого класса, то первый класс является производным от второго, второй — базовым классом первого.

Шаблоны LOTOS могут нарастающе модифицироваться путем *расширения*, *обогащения* и изменения *типов данных* или поведения. Проблема возникает с модификацией поведения, в частности:

- подтипы: в систему может быть внесена неопределенность, когда начальное состояние наследуемого шаблона и его модификации одинаковы и, таким образом, отношение подтипа не может быть гарантировано;
- необходимость перенаправления автоссылки: любая ссылка на производный шаблон из родительского должна быть перенаправлена на производный шаблон, что не всегда возможно.

Удовлетворительного решения этих проблем в стандартном LOTOS нет.

4.1.2.22 *Инвариант*

В LOTOS могут быть записаны только инварианты, являющиеся *определениями процессов*. Нет способа присоединить к *определению процесса* инвариант, который сам не является *определением процесса*.

4.1.2.23 *Предусловие*

Предикат, который обязательно должен быть истинным для того, чтобы действие могло осуществиться; он может быть непосредственно выражен в LOTOS с использованием одного или нескольких:

- последовательностей действий;
- *предохранителей* и *предикатов выбора*.

4.1.2.24 *Постусловие*

В LOTOS осуществление действия зависит от состояния системы после осуществления действия. Таким образом, LOTOS не предоставляет средства для непосредственного выражения постусловий.

4.2 Архитектурная семантика в ACT ONE

В настоящем разделе приведена формализация основных моделирующих и специфицирующих понятий, установленных в ГОСТ Р ИСО/МЭК 10746-2. Хотя конструкция ACT ONE не является ЯФС, она используется в стандартизованных ЯФС LOTOS и SDL и предоставляет альтернативную возможность формализации указанных выше понятий. Поэтому формализация на ACT ONE понятий, установленных в ГОСТ Р ИСО/МЭК 10746-2, приводится в отдельном разделе.

4.2.1 Основные моделирующие понятия

4.2.1.1 *Объект*

Экземпляр *сорта*, который может быть однозначно указан. Следует отметить, что объекты, моделируемые таким способом, должны быть специфицированы таким образом, чтобы они имели некоторую форму существования. Это может быть достигнуто использованием алгебраического стиля спецификации. Примеры такого стиля включают в себя рекурсию в *определениях процессов*, когда объект является элементом *списка значений параметров* связанного с этим *определением процесса*. Альтернативно для моделирования объектов с формой существования могут использоваться разделы **let . . . in.** В обоих стилях требуются *предохранители* и *предикаты выбора* для гарантии того, что реализации определений *сорта* являются уникальными.

4.2.1.2 *Среда (объекта)*

В ACT ONE среда объекта не обеспечивается. Это понятие может рассматриваться только через алгебру обработки и выражения ACT ONE. В результате среда объекта может рассматриваться как вся алгебра обработки, отличная от выражений ACT ONE, представляющих рассматриваемый объект, и от *операций* на этом объекте. Таким образом, среда используется как причина осуществления *операций* на объекте. Такое понятие среды не требует, чтобы *операции* на объекте вызывались другими объектами. Следствием этого является такое понятие, как взаимодействие, т. е. здесь взаимодействие происходит не между объектами, а между объектом и некоторым внешним агентом — алгеброй обработки.

4.2.1.3 *Действие*

Осуществление *операции*. Следует заметить, что это справедливо только в общем случае, так как нет существенного различия между взаимодействием и внутренним действием с позиций только ACT ONE. А именно, возможные действия моделируются *операциями* в сигнатуре *сорта* ACT ONE, и они могут происходить или не происходить в зависимости от выражений ACT ONE, существующих в алгебре обработки. Таким образом, внутренние действия не обеспечиваются явным образом в ACT ONE. Однако возможны случаи, когда внутреннее действие может моделироваться определенными локально в алгебре обработки *сортами*. Альтернативно все *операции*, объявленные в алгебре обработки, могут трактоваться как взаимодействия. *Операции*, используемые для удовлетворения таких операций, т. е. *уравнения*, связанные с рассматриваемыми *операциями*, могут трактоваться как внутренние действия. Например, если процесс вызывает *операцию pop2*, которая удаляет два элемента из очереди, а она в связанных с нею *уравнениях* дважды использует *операцию pop*, то *pop2* может трактоваться как взаимодействие, а *pop* — как внутреннее действие. Однако при такой трактовке внутренних действий проблема заключается в том, что нет понятия спонтанного перехода как такового, например как для экземпляра с символом *внутреннего события* i в алгебре обработки.

Следует отметить, что в таком виде взаимодействия не требуется, чтобы два или более объектов взаимодействовали в смысле алгебры обработки, т. е. через *синхронизацию* в общих *воротах*. Здесь взаимодействие может быть интерпретировано как нечто, что вызвано косвенно средой, а не обязательно объектом, т. е. не другим экземпляром *сорта*, моделирующего объект. Таким образом, возможен случай, когда появление предложения события, которое не содержит выражений ACT ONE, вызывает осуществление взаимодействия, например появление предложения события, приводящего к *реализации определения процесса, список значений параметров* которого содержит *операцию* (взаимодействие) на объекте (или объектах).

4.2.1.4 *Интерфейс*

Операции и *уравнения*, связанные с объектом.

4.2.1.5 *Деятельность*

Последовательность применения *операций* на данном *сорте*. Эти *операции* должны удовлетворять *уравнениям*, связанным с *сортом*. Каждая *операция* в последовательности встретившихся *операций*, т. е. каждая *операция* в активности должна иметь предусловия, которые удовлетворяют

постусловиям ранее встретившихся *операций*. Предусловия и постусловия для *операций* определены в 4.2.2.23 и 4.2.2.24, соответственно.

4.2.1.6 *Поведение (объекта)*

Поведение объекта, моделируемого в ACT ONE, зависит от *операций,* связанных с шаблоном объекта, и от текущего состояния привязанных к нему объектов. А именно, значение состояния привязанного объекта может быть использовано для ограничения возможных *операций,* например путем исключения некоторых *операций, уравнения* которых не выполняются для этого состояния.

ACT ONE не обеспечивает явно такие ограничения на появление *операций,* как последовательность, недетерминизм, конкуренция и ограничения реального времени. ACT ONE предоставляет *операции* и *уравнения,* которые сами по себе обозначают все возможные ограничения, т. е. все возможное поведение с ассоциированными с ним ограничениями.

В ACT ONE нет характеристик специально для моделирования внутренних действий. Конкретнее, нет понятия спонтанного перехода, которое может встретиться в алгебре обработки с символом *внутреннего события* i, например.

Следует отметить, что реально нет понятия поведения, фактически осуществляющегося только в ACT ONE. Осуществляющееся поведение ACT ONE обычно связывается с выражениями ACT ONE, которые вычисляются в алгебре обработки.

4.2.1.7 *Состояние (объекта)*

Текущее значение того экземпляра *сорта*, моделирующего объект, к которому это значение привязано. *Сорт,* моделирующий объект, должен содержать идентификатор, используемый для отличия друг от друга разных экземпляров *сорта*. Значение этого идентификатора не является частью состояния, т. е. это значение должно оставаться неизменным в *операциях* и *уравнениях,* связанных с *сортом.*

4.2.1.8 *Коммуникация*

Это понятие не поддерживается в ACT ONE. Возможен случай, когда коммуникация абстрактного вида может быть смоделирована при использовании алгебраического стиля спецификации. Однако и это не будет отражать текста ГОСТ Р ИСО/МЭК 10746-2, т. е. это не будет переносом информации от одного объекта к другому. Скорее, среда (алгебра обработки) используется для коммуникации с объектами.

4.2.1.9 *Положение в пространстве*

Понятие положения в пространстве не поддерживается явным образом в ACT ONE. Если требуется, то это понятие может быть построено в модели спецификации, например через *сорт*, моделирующий положение в пространстве, который используется в тех *операциях,* где нужно установить положение в пространстве.

4.2.1.10 *Положение во времени*

Понятие положения в пространстве не поддерживается явным образом в ACT ONE. Однако, если понятие времени относится к текущему состоянию данного объекта, т. е. к изменениям состояния, которые уже произошли и которые могут произойти, то положение во времени, при котором данное действие может произойти, можно определить, с некоторыми расширениями, текущим состоянием данного объекта.

Положение во времени, при котором данное действие может произойти, может быть построено в спецификации, например через *сорт*, моделирующий положение во времени, который используется в тех *операциях,* где нужно установить положение во времени.

4.2.1.11 *Точка взаимодействия*

Это понятие не поддерживается явным образом в ACT ONE. Однако возможен случай, когда это понятие может быть построено в спецификации, например через сорт, используемый в операциях, которые входят в множество интерфейсов в одном и том же месте. А именно, всем *операциям* в множестве интерфейсов в одном и том же месте требуется входной параметр (*сорт*), указывающий положение, в котором эти *операции* существуют. При необходимости можно смоделировать несколько точек взаимодействия так, чтобы они существовали в одном и том же месте. Этого можно достичь через *операцию* создания *сорта* «положение», которая требует на входе несколько *сортов* «точка взаимодействия». *Операции и уравнения*, ассоциированные с этими сортами, должны позволять идентифицировать разные точки взаимодействия и положения.

4.2.2 Специфицирующие понятия

4.2.2.1 *Композиция*

Композиция объектов — в общем случае в ACT ONE нельзя скомбинировать два произвольных объекта и получить осмысленный результат, т. е. составной объект со своим собственным поведе-

нием и пр. Наиболее похожей формой композиции в ACT ONE является *операция* «конструктор», имеющая два и более объектов в качестве входных параметров; это означает, что объекты могут быть однозначно идентифицированы.

Рассмотрим следующую *операцию* «конструктор» ACT ONE:

$$makeCO: Id, Ob1, Ob2 \rightarrow CO$$

Здесь составной объект создается из двух других объектов, имеющих свои собственные *операции* и *уравнения*, т. е. свое собственное поведение. Хотя объект *co = makeCO (id1, ob1, ob2)* является составным объектом из объектов *ob1* и *ob2*, сам объект *co* не имеет такого поведения. А именно, поведение, ассоциированное с *ob1* и *ob2*, не применимо к экземплярам *CO*.

Для решения этой проблемы составной объект может иметь дополнительные *операции* и *уравнения*. Тогда вид этих операций и уравнений и их взаимосвязь с объектами-компонентами определят вид композиции. Например, если *операции* и *уравнения* составного объекта просто позволяют получить доступ к изолированным компонентам-объектам, то получается вид делегирования с композицией объектов-компонентов, представленных в агрегации. Если *операции* и *уравнения* составного объекта заданы таким образом, что изменяют поведение составляющих объектов, то получается вид композиции, более похожий на описанный в ГОСТ Р ИСО/МЭК 10746-2. Однако следует заметить, что данное в ГОСТ Р ИСО/МЭК 10746-2 понятие композиции не требует дальней спецификации, т. е. для создания новых объектов и поведений нужна композиция существующих объектов и поведений, а не спецификация дополнительного поведения, позволяющего создавать осмысленную композицию объектов-компонентов.

Композиция поведений — так как ACT ONE не предоставляет специальных операторов композиции, понятие композиции поведений явно не обеспечивается. Надо отметить, что в ACT ONE существует форма композиции, и это — *обогащение*. Однако в общем случае оно не может быть классифицировано как композиция, так как не обеспечивает явную композицию поведений (и объектов). *Обогащение* само по себе, т. е. без дополнительной спецификации, не обеспечивает свойств композиции. При применении *обогащения* все *типы данных* существуют независимо. Понятие композиции может применяться только в том случае, когда специфицированы *операции* и *уравнения*, которые позволяют использовать *сорта*, доступные через *обогащение*. Однако так нельзя адекватно отразить текст ГОСТ Р ИСО/МЭК 10746-2, т. е. это не тот случай, когда два поведения просто образуют комбинацию. Для комбинации поведения требуется дополнительная спецификация. *Обогащение* позволяет скомбинировать все существующие *операции* и *уравнения сортов*. Таким образом, повторная спецификация необходима для комбинации поведения, не включенного *операциями* и *уравнениями сортов* через *обогащение*.

Некоторая форма композиции поведений может быть получена путем *актуализации параметризованных типов данных*. *Актуализация типа* данных должна удовлетворять любым *формальным сортам*, операциям и *уравнениям актуализируемого типа данных*. Это ближе, чем может быть получено с ACT ONE, к охвату понятия композиции поведений, очерченного в ГОСТ Р ИСО/МЭК 10746-2. Спорным остается вопрос, представляется ли таким образом композиция поведений, так как поведение *параметризованного типа данных* не может существовать до того, как будет *актуализировано*, т. е. экземпляр этого сорта не может появиться в алгебре обработки и применяемых *операциях*.

4.2.2.2 Составной объект

Результат композиции объектов.

4.2.2.3 *Декомпозиция*

Декомпозиция объектов — в ACT ONE объекты могут быть разложены при условии, что в связанной с объектом сигнатуре существуют операции, допускающие декомпозицию. Например, следующий тип данных позволяет разложить составной объект на компоненты:

```
type Z is X, Y, IdType
  sorts Z
  opns makeZ: Id, X, Y — > Z
    getX: Z — > X
    getY: Z — > Y
  eqns forall x:X, y:Y, id:ID
    ofsortX
      getX (makeZ (id, x, y)) = x;
    of sortY
```

getY (makeZ (id, x, y)) = y;
endtype (* Z *)

Таким образом, данный *z:Z = makeZ (id1, x, y)*, где *x* и *y* — экземпляры *сортов*, моделирующих объекты, а *id1* — уникальный идентификатор, можно разложить на *x* и *y*, т. е. на его компоненты, операциями *getX (Z)* и *getY (Z)*, соответственно.

Примечание — Эта интерпретация основана на идее о том, что можно разделить составной объект на части-компоненты (объекты). Однако ГОСТ Р ИСО/МЭК 10746-2 только требует, чтобы декомпозиция определяла данный объект как комбинацию двух или нескольких объектов, т. е. композицию. В ACT ONE составные объекты всегда специфицированы как комбинация объектов-компонентов. Следовательно, различие между композицией и декомпозицией, определенное ГОСТ Р ИСО/МЭК 10746-2, несколько размывается при представлении в ACT ONE.

Декомпозиция поведений — понятие декомпозиции поведений зависит от спецификации композиции поведений. Это понятие не обеспечивается в ACT ONE явным образом (см. 4.2.2.1). А именно, поведение представляется *операциями* и *уравнениями*, действующими на *сорт*. Это не тот случай, когда два произвольных поведения *сортов* могут быть скомбинированы и дадут новое поведение.

Примечание — Так же можно считать, что понятие декомпозиции поведений обеспечивается *операциями* и *уравнениями* ACT ONE, связанными с *сортом*. А именно, эти *операции* и *уравнения* обеспечивают все возможные комбинации поведений. Так например, последовательная композиция может быть создана последовательным применением *операций*. Каждое применение *операции* в последовательности должно удовлетворять необходимым для появления *уравнениям*. Остается спорным, является ли это композицией поведений, так как *операции* и *уравнения* уже существуют и определены для всех возможных поведений.

4.2.2.4 *Поведенческая совместимость*

В LOTOS поведенческая эквивалентность *типов данных* основана на эквивалентности имен *сортов* и, возможно, на значении, с которым эти сорта связаны. В результате, в общем случае, объект не может быть заменен в некоторой среде другим объектом, если объекты получены из разных шаблонов объектов, т. е. являются экземплярами разных *сортов*. Однако иногда может оказаться возможным заменить один объект другим, полученным из другого шаблона объектов. Для этого требуется, чтобы среда предлагала *операции*, применимые к обоим *сортам*, а результаты этих *операций* были одинаковы. Например, в некоторой среде могут оказаться поведенчески совместимыми *сорта*, представляющие стек целых чисел и очередь целых чисел, если среда предлагает только *операцию «верх»*, а стек и очередь имеют одинаковое количество помещенных в них целых чисел. А именно, в обоих случаях результатом будет целое число. Если среда предлагает *операцию «вытолкнуть»* или *«втолкнуть»*, то между этими объектами не будет поведенческой совместимости, так как *операции* вернут *сорта* «стек» или «очередь». Так как в общем случае среда объекта может вызвать любую *операцию* из сигнатуры, то эта форма поведенческой совместимости является ограниченной.

4.2.2.5 *Уточнение*

Хотя понятие уточнения обеспечивается алгеброй обработки LOTOS явным образом, например через отношения тестирования *соответствия* и эквивалентности, в ACT ONE для уточнения сделано очень мало. Интуитивно ясно, что уточнение в ACT ONE может иметь разные формы, например через расширение сигнатуры данного *сорта*, т. е. обеспечение дополнительных *операций*. Эта форма уточнения будет создавать поведенческую совместимость, т. е. существующие *операции* и *уравнения* остаются неизменными. Возможны и другие формы уточнения, например изменение *уравнений*, связанных с *операциями* над *сортом*. Не похоже, что обеспечение поведенческой совместимости будет тривиальным в этом случае.

4.2.2.6 *Трасса*

Так как в ACT ONE взаимодействия не представлены явным образом, то понятие трассы ограничено, т. е. нельзя гарантировать, что она не содержит внутренних действий. Если взаимодействия рассматривать как *операции*, которые происходят в алгебре обработки, а внутренние действия — как *операции*, используемые для вычисления уравнений, связанных с этими *операциями*, то в некоторых ограниченных пределах можно моделировать трассу. В таком случае она соответствует последовательности применения *операций*, связанных с экземпляром моделирующего объект *сорта*. Нужно отметить, что если *уравнения*, связанные с моделирующими взаимодействия *операциями*, были переписаны, то запись взаимодействий объекта, т. е. трасса, вероятно, является некорректной. Например, применение к очереди *операции «втолкнуть»* с последующей *«вытолкнуть»* будет, веро-

ятно, переписано как *«очередь»*, а не как выражение *pop (push (x, q))*. Таким образом, понятие трассы в ACT ONE ограничено.

4.2.2.7 *Тип <X>*

Объекты, интерфейсы и действия, специфицированные в ACT ONE, могут удовлетворять многим различным произвольным характеризующим предикатам. Типы, которые могут быть записаны явно, являются типами шаблонов.

4.2.2.8 *Класс <X>*

Понятие класса зависит от характеризующего тип предиката, которому должны удовлетворять члены класса. Объекты, интерфейсы и действия могут удовлетворять многим произвольным характеризующим тип предикатам. Тип, который может быть в нем записан, является типом шаблона. Класс объектов, интерфейсов или действий, связанных с этим типом, является классом шаблонов.

Примечание — Следует заметить, что если ограничиться единственной возможной в LOTOS классификацией объектов, интерфейсов и действий, что они соответствуют типу шаблона, то понятия класса и класса шаблонов, определенные в ГОСТ Р ИСО/МЭК 10746-2, суживаются до методов моделирования в LOTOS. В LOTOS нет разницы между классом в широком смысле слова и классом шаблонов в более узком смысле как множества реализаций данного типа шаблонов.

4.2.2.9 *Подтип/супертип*

Понятия подтипа и супертипа, вообще говоря, не поддерживаются в ACT ONE, так как при проверке типа LOTOS использует эквивалентность имен. Например, два типа объектов являются одинаковыми только тогда, когда они представляются одним и тем же *сортом*. Таким образом, в общем случае один *сорт* не может быть подставлен вместо другого. Однако возможен случай, когда существует ограниченная форма отношений подтипа между двумя разными *сортами*, если характеризующий тип предикат основан на некотором аспекте сорта, отличном от имени, например такая-то *операция* допустима над этим *сортом* и возвращает такой-то результат. Это — ограниченная форма создания типа, и вряд ли она существует в большинстве случаев.

4.2.2.10 *Подкласс/суперкласс*

Понятия подкласса и суперкласса поддерживаются в ACT ONE только в очень ограниченных пределах, т. е. когда характеризующий тип предикат основан на некотором аспекте *сорта*. В результате эти понятия не полностью поддерживаются в ACT ONE. Если между двумя *сортами* существует отношение подтип/супертип, то между экземплярами *сортов* в алгебре обработки существует отношение подкласс/суперкласс.

4.2.2.11 *Шаблон <X>*

Шаблон объектов — определение *сорта* (с соответствующими *операциями* и *уравнениями*), моделирующего объект.

Шаблон интерфейсов — набор *операций* и *уравнений*, связанных с определением *сорта*, моделирующего объект. Понятия шаблонов интерфейсов и объектов можно рассматривать как идентичные, так как *операции* и *уравнения* всегда должны действовать на определении *сорта*. Это справедливо и в том случае, когда декларация экземпляра сорта в алгебре обработки имеет неявно связанные с ней *операции* и *уравнения*, что устанавливается в части ACT ONE спецификации.

Шаблон действий — *операции* с соответствующими *уравнениями*.

4.2.2.12 *Сигнатура интерфейса*

Операции, которые применяются к переменной, объявленной как экземпляр *сорта*, моделирующего объект.

4.2.2.13 *Реализация (шаблон <X>)*

Реализация шаблона объекта — для реализации шаблона объекта требуется инициализация моделирующего объект *сорта* допустимым начальным состоянием. Эта инициализация *сорта* должна гарантировать, что экземпляр *сорта* может быть однозначно указан.

Реализация шаблона интерфейса — реализация шаблона интерфейса происходит при реализации шаблона объекта. В таком случае объект имеет единственный интерфейс, заданный *операциями* и *уравнениями*, действующими на *сорт*, из которого объект был реализован.

Реализация шаблона действия — появление *операции* ACT ONE в алгебре обработки. Эта *операция* должна удовлетворять *уравнениям*, связанным с *операцией*.

4.2.2.14 *Роль*

Понятие роли лучше всего может быть промоделировано в ACT ONE *сортом*. Это происходит потому, что роль представляет собой идентификатор поведения. Именно через декларацию *сорта* становятся доступными *операции* и *уравнения*, применяемые к этому сорту.

4.2.2.15 *Создание (<X>)*

Создание объекта или интерфейса — так как объекты и интерфейсы имеют форму существования, когда ACT ONE используется совместно с алгеброй обработки, то сам ACT ONE не может использоваться для моделирования создания. ACT ONE совместно с алгеброй обработки может использоваться для моделирования в ограниченных пределах создания объектов и интерфейсов при условии следования определенному стилю спецификации. Например, как *операция*, связанная с моделирующим объект *сортом*, т. е. *операция* над уже существующим объектом, которая приводит к созданию нового объекта. Должна быть возможность уникальной ссылки на вновь созданный объект. Этот новый объект будет использоваться в алгебре обработки, так что он имеет некоторую форму существования (подробнее см. 4.2.1.1).

4.2.2.16 *Введение (объекта)*

Введение объекта в ACT ONE может быть достигнуто несколькими способами при использовании совместно с алгеброй обработки. Например, с помощью предложения события, появление *обозначения действия* которого приводит к созданию нового экземпляра сорта, моделирующего объект. Эти новые экземпляры должны находиться в допустимом начальном состоянии, быть однозначно указываемыми и иметь некоторую форму существования в алгебре обработки. Альтернативно объекты могут быть введены через разделы **let . . . in.** Эти объекты тоже должны иметь допустимое начальное состояние, использоваться в алгебре обработки так, что они имеют форму существования, и должна быть возможность их однозначной идентификации.

4.2.2.17 *Удаление (<X>)*

Удаление объекта или интерфейса — удаление объекта или интерфейса в ACT ONE может быть достигнуто при совместном использовании с алгеброй обработки путем перезаписи соответствующего появления *уравнений*, связанных с *операциями* над *сортами*, моделирующими объекты. Например, для моделирования удаления может быть использована *операция*, которая перемещает элемент из множества, т. е. объект с конкретным идентификатором перемещается (удаляется) из множества реализованных объектов, существующего как часть *списка значений параметров* рекурсивного *определения процесса*.

4.2.2.18 Экземпляр типа

Экземпляр типа шаблона объекта, интерфейса или действия — экземпляр типа зависит от определяющего его характеризующего предиката. Если предикат типа является типом шаблоном для объектов и интерфейсов, то экземпляр типа объекта или интерфейса соответствует появлению в алгебре обработки *сорта*, моделирующего рассматриваемые объекты и интерфейсы. Аналогично, если предикат типа является типом шаблона для действия, то экземпляр типа действия задается появлением в алгебре обработки *операции*, моделирующей тип действия.

4.2.2.19 *Тип шаблона (<X>)*

Тип шаблона объекта или интерфейса — предикат на реализациях *сорта*, используемого для моделирования объекта. Все реализации (появления) шаблона (*сорта*) в алгебре обработки имеют *операции* и *уравнения*, ассоциированные с этим *сортом*. Так как шаблоны для объектов и интерфейсов моделируются одним и тем же образом, т. е. определением *сорта* с соответствующими *операциями* и *уравнениями*, то тип шаблона объекта и тип шаблона интерфейсов для этого объекта в ACT ONE являются синонимами: они оба соответствуют появлению *сорта* в алгебре обработки.

Тип шаблона действия — предикат на появлениях *операции* в алгебре обработки. А именно, все реализации (появления) шаблона (*операции*) в алгебре обработки должны соответствовать требованиям шаблона, т. е. они должны иметь те же самые входные данные и давать тот же самый результат, что и рассматриваемое определение *операции*, а вычислением *операции* управляют *уравнения*, ассоциированные с этой *операцией*.

4.2.2.20 *Класс шаблонов (<X>)*

Класс шаблонов объектов — множество реализаций данного *сорта*, моделирующего объект, в алгебре обработки.

Класс шаблонов интерфейсов — множество реализаций данного *сорта*, моделирующего интерфейс объекта, в алгебре обработки.

Класс шаблонов действий — множество реализаций данной *операции* в алгебре обработки.

4.2.2.21 *Производный класс/базовый класс*

В ACT ONE производные и базовые классы не поддерживаются. Это происходит потому, что в ACT ONE классы обычно задаются только через классы шаблонов, т. е. (для объектов) множеством

реализаций данного *сорта*, моделирующего объект, в алгебре обработки. В данном случае *сорта* не могут модифицироваться нарастающим образом. Конкретно, *сорта* и присвоенные им метки, т. е. имена *сортов*, не позволяют ссылаться на другой *сорт*, т. е. всегда существует автоссылка. Таким образом, *операции* и *уравнения*, связанные с данным *сортом*, применимы только к нему, но не к другим *сортам*.

Понятие *актуализации* параметризованных классов, хотя интуитивно и имеет характеристики отношения производный/базовый класс, не влияет на представление этого отношения. Дело в том, что в данном случае экземпляры параметризованного класса не могут появляться в алгебре обработки, т. е. они должны быть *актуализированы* так, чтобы класс мог существовать.

4.2.2.22 *Инвариант*

Это понятие неявно подразумевается в ACT ONE, т. е. объекты всегда должны удовлетворять *операциям* и *уравнениям*, которые к ним применяются.

4.2.2.23 *Предусловие*

В ACT ONE все *операции* должны удовлетворять всем *уравнениям* (и любым ассоциированным *предохранителям*), которые применяются к ним до их появления.

4.2.2.24 *Постусловие*

Это понятие неявно подразумевается в ACT ONE, т. е. появление данной *операции* (действия) требует, чтобы соответствующие *уравнения* были определены (справедливы).

4.3 Архитектурная семантика в SDL

SDL является стандартизированным ЯФС. Методические материалы приведены в приложении к Рекомендации МСЭ-Т Z.100. Существует ряд руководств по SDL и его коммерческих реализаций, поддерживающих различные аспекты SDL: от работы с графикой до анализа и генерации программного кода на основе SDL.

В SDL *система* моделируется как набор (множество) запоминающих конечных автоматов, взаимодействующих с помощью сообщений, называемых *сигналами*. Понятие данных в SDL основано на ACT ONE. Конечные автоматы являются запоминающими в том смысле, что могут быть определены локальные переменные для хранения части их истории. Сигналы передаются асинхронно, представляя таким образом потерю связи между компонентами в распределенной системе.

В настоящем разделе представлен один из способов, которым основные моделирующие понятия могут быть выражены на SDL. Это представление не является единственно возможным. Он показывает, что большинство фундаментальных понятий может быть выражено на SDL. Альтернативный подход к моделированию многих понятий основан на использовании ACT ONE. Подробнее он описан в 4.2.

Использована версия SDL-92, определенная в Рекомендации МСЭ-Т Z.100. Эта версия по сравнению с SDL-88 имеет несколько расширений. Среди них важнейшими являются:

- объектно-ориентированные конструкции;
- возможность *каналов* без задержки;
- недетерминизм;
- возможность включения понятия альтернативных данных и вызовов *удаленных процедур*.

Наиболее важной характеристикой альтернативных данных является возможность комбинированного использования ACT ONE и ACH.1 в SDL. Семантика комбинации SDL-92 с ACH.1 определена в Рекомендации МСЭ-Т Z.105.

Во избежание путаницы термины SDL при необходимости выделены курсивом.

4.3.1 Основные моделирующие понятия

4.3.1.1 *Объект*

В SDL объектами являются экземпляры *типа систем*, *типа блоков*, *типа процессов*, *типа услуг*, *таймера*, *канала* и *сигнального пути*. Эти экземпляры характеризуются состоянием и поведением.

Каждый экземпляр инкапсулирован, т. е. любые изменения в его состоянии могут произойти только в результате внутренних действий или взаимодействия с его средой.

Ссылки для каждого вида объектов должны быть обеспечены явно.

4.3.1.2 *Среда (объекта)*

Среда объекта зависит от его вида. См. таблицу 1.

Таблица 1 — Среда объекта

Вид объекта	Ограничения среды
Система	- входящие *сигналы* каналов
Блок	- глобальные типы данных - входящие *сигналы* каналов - вызовы *экспортированных процедур* - импортированные переменные
Процесс	- глобальные типы данных - входящие *сигналы* явных и неявных *сигнальных путей* - вызовы *экспортированных процедур* - видимые/импортированные переменные - ограничения времени для *входящих* действий
Услуга	- глобальные переменные */таймеры/* типы данных, совместно используемые буфера сигналов (принадлежащие объемлющему экземпляру *процесса*) - входящие *сигналы* явных и неявных *сигнальных путей* - вызовы *экспортированных процедур* - видимые/импортированные переменные - ограничения времени для *входящих* действий
Таймер	- вызовы *стоп*, *установить* и *переустановить* - *процессами* владельца
Канал	- входящие *сигналы* от соединенных *блоков*
Сигнальный путь	- входящие *сигналы* от соединенных экземпляров *процессов/услуг*

4.3.1.3 *Действие*

В SDL действие — это единичное *предложение действия, ввести* или *сохранить*, вся транзакция или завершенное выполнение *процедуры*. Возможными единичными *предложениями действия* являются:

- *задача, импорт, экспорт, вид;*
- *вывести;*
- *создать;*
- *установить, переустановить, активизировать;*
- *вызов процедуры;*
- *стоп/вернуть;*
- *следующее-состояние.*

Передача *сигналов каналами* или *сигнальными путями* также является действием, как и генерация *сигнала таймера*. Взаимодействиями являются *ввод/вывод сигнала, вызов* и *возврат удаленной процедуры* и использование разделяемых переменных (глобальных переменных процесса для *услуг, обнаруженных/видимых* и *экспортированных/импортированных* переменных *процесса*). Последовательность действий отправки, передачи и, возможно, приема *сигнала (вывести—ввести)* может рассматриваться как одно взаимодействие.

4.3.1.4 *Интерфейс*

В зависимости от вида объекта в SDL имеются различные способы описания интерфейсов, как показано в таблице 2.

В случае объекта *блок* должны быть инкапсулированы в один или несколько процессов набор *экспортированных/импортированных удаленных процедур* и набор *сигналов*, отправляемых/получаемых процессами этого *блока*. Тогда эти *процессы* работают как интерфейсы объекта *блок*. Этими описаниями интерфейсов определены потенциальные взаимодействия объекта, а каждый интерфейс описывает подмножество потенциальных взаимодействий объекта.

Таблица 2 — Интерфейсы объекта

Вид объекта	Интерфейс характеризуется:
Система	- воротами системы с их списками *сигналов* - входными/выходными *каналами* с их списками *сигналов*
Блок	- *воротами блока* с их списками *сигналов* - входными/выходными *каналами* с их списками *сигналов* - входными/выходными *путями сигналов* с их списками *сигналов* - набором (*экспортированных/импортированных*) *удаленных процедур* - набором (*экспортированных/импортированных*) удаленных переменных
Процесс	- *воротами процесса* с их списками *сигналов* - набором всех допустимых *входных/выходных сигналов* - набором всех *экспортированных/импортированных процедур* - набором *(обнаруженных/видимых)* разделяемых переменных
Услуга	- воротами услуги с их списками *сигналов* - набором всех допустимых *входных/выходных сигналов* - набором всех *экспортированных/импортированных процедур*
Таймер	- идентификацией *таймера*
Канал	- наборами всех *сигналов*, передаваемых в каждом направлении
Сигнальный путь	- наборами всех сигналов, передаваемых в каждом направлении

4.3.1.5 *Деятельность*

В общем случае деятельность не может быть обозначена явно, так как она может охватывать несколько объектов.

Одним из частных случаев деятельности является выполнение локальной или *удаленной процедуры* с действием *вызов* в начале деятельности и потенциальными действиями возврата в конце.

4.3.1.6 *Поведение (объекта)*

Поведение *процесса/услуги* является множеством всех переходов этого (ой) *процесса/услуги*. Действия *ввести* дают ограничения на обстоятельства, при которых могут происходить переходы. Дополнительные ограничения могут быть введены с помощью конструкций *обеспечить* и *непрерывный сигнал*. Объект может демонстрировать недетерминированное поведение.

Поведение *блока* собирается из поведений содержащихся в нем *процессов*. Поведение системы собирается из поведений содержащихся в ней *блоков*.

Поведением *канала* или *сигнального пути* является перенос *сигналов* (немедленный или с задержкой).

В SDL поведение *таймера* является предопределенным в смысле работы в качестве будильника.

4.3.1.7 *Состояние (объекта)*

Множество всех последовательностей действий, в которых может участвовать объект, в данный момент времени для *процесса* или *услуги* определяется текущим состоянием SDL в это время, значениями локальных переменных и содержимым входного порта.

Состоянием *блока* или *системы* является общее состояние всех содержащихся в нем (ней) *процессов* и *блоков* плюс всех содержащихся *каналов* и *сигнальных путей*.

Состояние *канала* явно или неявно задается *блоком*. Это состояние зависит от того, имеет ли *канал* свойство *задержки*.

Состояние *сигнального пути* всегда задано неявно.

Состоянием *таймера* — активен или не активен. Состояние активного *таймера* определяется временем задержки до отправки *сигнала* тайм-аута.

4.3.1.8 *Коммуникация*

Перенос информации между двумя или несколькими объектами осуществляется явными или неявными (в случае удаленных *процедур* или *экспортированных/импортированных* переменных) *каналами* или *сигнальными путями*. Информацию переносят *сигналы*.

4.3.1.9 *Положение в пространстве*

Действия осуществляются в экземплярах *процессов* и *услуг*. Передача действий осуществляется в *каналах* и *сигнальных путях*.

4.3.1.10 *Положение во времени*

Каждое действие характеризуется датами начала и окончания действия. Действия могут быть мгновенными. Длительность действия не может быть обозначена явно. Для действий может быть составлено расписание: происходить в конкретное время или после заданной задержки.

Примечание

а) Глобальное время доступно в SDL через *now*. При этом ничего не говорится о единицах измерения.

б) Для двух последовательных действий *a1* и *a2* справедливо отношение *now (a1)* <= now (a2).

в) Явным образом обращаться ко времени можно только с помощью действий установки таймера, а *now* применяется в *разрешающих условиях* и *непрерывных сигналах*. Следует избегать составления расписания действий для фиксированных моментов времени.

4.3.1.11 *Точка взаимодействия*

Точками взаимодействия являются ворота экземпляров *блоков*, *процессов и услуг* и оконечные точки (возможно, неявные) каналов и *сигнальных путей*. Совместно используемые переменные так же являются точками взаимодействия. Они имеют положение. Объект может иметь несколько точек взаимодействия.

4.3.2 Специфицирующие понятия

4.3.2.1 *Композиция*

Композиция объектов:

а) объект *система* может быть конкурентной композицией объектов *блок*, которые могут быть соединены *каналами;*

б) объект *блок* может быть конкурентной композицией объектов *блок* (которые могут быть соединены *каналами*) или конкурентной композицией объектов *процесс* (которые могут быть соединены сигнальными путями);

в) объект *процесс* может быть чередующейся композицией объектов *услуга*;

г) *канал* может быть композицией *блоков*, соединенных *каналами.*

Композиция поведений:

а) поведение *системы* является конкурентной композицией поведений ее *блоков*;

б) поведение *блока* является конкурентной композицией поведений его *подблоков* или *процессов*;

г) поведение *процесса* является чередующейся композицией поведений его услуг или последовательной композицией действий графа *процесса*;

д) поведение *услуги* является последовательной композицией взаимодействий графа *услуги*;

е) поведение *канала* может быть композицией поведений составляющих его *блоков* и соединяющих их *каналов.*

4.3.2.2 *Составной объект*

Согласно 4.3.2.1 следующие объекты могут быть представлены как композиции:

- *система,*
- *блок,*
- *процесс,*
- *канал.*

4.3.2.3 *Декомпозиция*

Декомпозиция объектов — спецификация данного объекта как композиции.

Декомпозиция поведений — спецификация данного поведения как композиции.

4.3.2.4 *Поведенческая совместимость*

В SDL нет общих способов явного описания поведенческой совместимости, однако семантические основы языка допускают определение поведенческой совместимости и ее проверки в терминах переходных систем.

Экземпляр переходного класса может рассматриваться как ограниченно поведенчески совместимый с экземпляром соответствующего базового класса, а экземпляр *переопределенного* типа может рассматриваться как ограниченно поведенчески совместимый с экземпляром соответствующего *виртуального* типа. Для требований ограниченной поведенческой совместимости может быть использован раздел *at least.*

4.3.2.5 *Уточнение*

В SDL имеется два способа уточнения спецификации объекта:

- построение подструктур (на уровне *блока* или *системы*);
- использование объектно-ориентированных свойств (наследования, виртуальных и родовых параметров).

4.3.2.6 *Трасса*

В SDL нет общих способов явной спецификации трасс. Трассы могут быть получены в результате интерпретации спецификации в соответствии с динамической семантикой SDL.

Примечание — Диаграмма записи последовательностей (ДЗП) предоставляют синтаксис и семантику, подходящие для представления трасс спецификаций SDL. Имеется тесная взаимосвязь между синтаксисом и семантикой ДЗП, и синтаксисом и семантикой SDL. ДЗП определены и стандартизированы в рекомендации МСЭ-Т Z.120.

4.3.2.7 *Тип <X>*

В SDL нет общих явных предикатов.

4.3.2.8 *Класс <X>*

Это понятие в общем случае поддерживается только для типов шаблонов.

4.3.2.9 *Подтип/супертип*

В общем случае это понятие не поддерживается.

4.3.2.10 *Подкласс/суперкласс*

Это понятие в общем случае поддерживается только для типов шаблонов.

4.3.2.11 *Шаблон <X>*

Шаблон объектов — шаблоны объектов являются определениями типов для подходящих видов объектов (*система, блок, процесс, услуга*). Для *таймеров*, *каналов* и *сигнальных путей* шаблонами объектов являются соответствующие декларации.

Шаблон интерфейсов — в зависимости от вида интерфейса его шаблон может быть задан декларацией (*канала, сигнального пути*) неявно или определением типа *процесса* (см. 4.3.1.4) явно.

Шаблон действий — шаблон действий специфицируется определением *графа процесса, процедуры* или *услуги*. Шаблонами элементарных действий являются *ввести, вывести, сохранить, установить, переустановить, создать, задача, стоп, возврат, следующее-состояние, вызов, импорт, экспорт, вид.*

Шаблоны могут быть специфицированы с (формальными или формального контекста) параметрами. Параметры могут иметь ограничения. Шаблоны могут быть скомбинированы (т. е. определение типа может содержать определения других типов).

4.3.2.12 *Сигнатура интерфейса*

Набор *типов сигналов* и *типов удаленных процедур*, допустимых для интерфейса.

4.3.2.13 *Реализация (шаблона <X>)*

В SDL имеется два способа реализации шаблонов:

- неявная реализация (*системы, блока, каналов, сигнальных путей, процессов, услуг*) осуществляется декларацией объекта;
- явная реализация осуществляется с помощью действия *создать* (только для *процессов*).

Реализации всегда являются результатами действий по реализации шаблонов. Параметры формального контекста должны быть актуализированы до того, как может быть получена реализация (специализацией типа или декларацией *процесса*).

4.3.2.14 *Роль*

Нет общих средств для спецификации ролей.

Роли могут быть описаны как параметры формального контекста.

Примечание — Для дальнейшего уточнения ролей могут использоваться разделы *Atleast.*

4.3.2.15 *Создание (<X>)*

В SDL имеется два способа создания (см. 4.3.2.13):

- неявное создание;
- явное создание — интерпретация действия *создать.*

4.3.2.16 *Введение (объекта)*

Неявная реализация (см. 4.3.2.13) может рассматриваться как введение.

4.3.2.17 *Удаление (<X>)*

Могут быть удалены только объекты *процесс. Процесс* может удалить только самого себя, что осуществляется путем интерпретации действия *стоп.* Если действие стоп интерпретирует *услуга*, то это приводит к удалению данной *услуги*, удалению всех других *услуг*, относящихся к тому же самому процессу, и самого этого *процесса.*

Примечания

1 Удаление одного процесса другим может моделироваться с использованием *вывода* специального *сигнала*, который при приеме его получателем приводит к интерпретации получателем действия *стоп.*

2 Удаление всех *процессов блока* может рассматриваться как удаление самого этого *блока.*

4.3.2.18 *Экземпляр типа*

Объект является экземпляром *типа системы, блока, процесса* или *услуги* X, если имеется явная или неявная реализация или подстановка этого X. Подстановка является реализацией типа шаблона, который, в свою очередь, есть специализация типа в SDL.

4.3.2.19 *Тип шаблона (<X>)*

Тот факт, что **<X>** является реализацией шаблона **<X>**, может быть выражен для *процессов, услуг, блоков* и *систем* с помощью обозначения того, что объект является реализацией в SDL определения типа.

4.3.2.20 *Класс шаблонов (<X>)*

Нет общей явной нотации для характеристики класса шаблонов **<X>**, однако класс шаблонов является множеством всех экземпляров *процессов, блоков, услуг* или *систем* из определений *типов*, соответственно, *процессов, блоков, услуг* или *систем*.

4.3.2.21 *Производный класс/базовый класс*

Определение типа может быть получено из определения другого типа с *помощью специализации*, которая может включать в себя:

- связывание параметров контекста и добавление новых параметров контекста;
- наследование определений;
- переопределение виртуальных компонентов;
- добавление новых определений.

Ограничения могут применяться с помощью конструкций *atleast* и *finalised*.

Примечание — Кратное наследование не поддерживается.

4.3.2.22 *Инвариант*

В SDL нет нотации для инвариантов.

4.3.2.23 *Предусловие*

Для спецификации предусловий передачи могут использоваться *разрешающие условия, непрерывные сигналы* и *сигналы*.

4.3.2.24 *Постусловие*

В SDL нет нотации для постусловий.

4.4 Архитектурная семантика в Z

Нотация Z является спецификацией нотации, основанной на строгой теории множеств и исчислении предикатов первого порядка. Z пока еще не является стандартным ЯФС, но соответствующий стандарт разрабатывается ИСО ПК22 РГ19. Последней версией стандарта является Z-Base.

Стандарт де-факто для Z существует. Он стабилен, близок к стандарту Z-Base, и существуют средства, обеспечивающие проверку синтаксиса и семантики. До тех пор, пока документы ИСО не получат широкого распространения, для архитектурной семантики БМ-ОРО используется стандарт де-факто.

Во избежание путаницы в терминологии ОРО и Z в последующих подразделах курсивом выделены специфические для Z термины.

4.4.1 Основные моделирующие понятия

4.4.1.1 *Объект*

Объект может быть описан в Z совокупностью фрагментов спецификации. Эти фрагменты должны содержать совокупность *схем операций* (представляющих интерфейсы объекта), включая соответствующие *схемы состояний*. Эти схемы состояний, в свою очередь, могут включать в себя предикаты, которые используются для представления (фрагментов) инвариантов объекта. Фрагменты спецификации должны иметь некоторые средства, позволяющие однозначно ссылаться на эти фрагменты. Это может быть достигнуто с помощью идентификатора в *схеме (ах) состояний* объекта, который остается постоянным при всех определенных над объектом операциях. Наконец, для данного объекта должно существовать допустимое начальное состояние. Это может быть достигнуто с помощью схемы инициализации, которая задает допустимые связывания переменных, объявленных в *схеме состояний*, и содержит предикаты, гарантирующие единственность объекта в спецификации.

Примечание — При спецификации объектов в Z следует проявлять осторожность, так как язык не предоставляет собственных средств инкапсуляции (особенно для описываемых объектов), как указано в примечании к 4.4.1.3.

4.4.1.2 *Среда (объекта)*

В спецификации Z среда объекта описывается в терминах ввода/вывода. Ввод в объект посту-

пает из среды. Вывод объекта идет в среду. Среда объекта может быть специфицирована непосредственно или оставлена неспецифицированной. Если она не специфицирована, то появление *схем операций*, производящих вывод или требующих ввода, может произойти со средой, производящей ввод или получающей вывод, соответственно. Если же среда объекта специфицирована, то этим подразумевается, что для каждой связанной с объектом *схемы операций* существует другая *схема операций* (возможно, связанная с другим объектом), которая требует вводов и выводов того же самого типа, что и рассматриваемый объект. Эти две *схемы операций* сопрягаются друг с другом и ввод/вывод рассматриваемой операции переименовывается в вывод/ввод операции, представляющей среду.

Среда объекта может быть задана переменными, на которые ссылается объект, имеющий глобальную область действия, например так происходит в *аксиоматических описаниях*.

4.4.1.3 *Действие*

Действие моделируется в Z осуществлением операции, специфицированной в *схеме операций*. Результатом является осуществление изменений (или нулевых изменений) в состояниях объектов, с которыми ассоциировано действие. Действие может привести к недетерминированному результату.

Так как в Z нет явной нотации для инкапсуляции, то обычно в Z не определено, является действие наблюдаемым или внутренним, следовательно, различия между взаимодействиями и внутренними действиями не определено явно. В настоящем стандарте используется соглашение, что *схема операций*, представляющая действие, которое имеет ввод, вывод или глобальные переменные, специфицирует взаимодействие со средой. Среда может быть задана или нет (см. 4.4.1.2). Действия, требующие ввода от неспецифицированной среды и не дающие выводов, могут рассматриваться как вызываемые внешне ненаблюдаемые действия. Действия, дающие вывод в неспецифицированную среду, могут рассматриваться как вызываемые внутренние (спонтанные) наблюдаемые действия. Действия, требующие ввода от неспецифицированной среды и дающие вывод в эту среду, могут рассматриваться как вызываемые внешне наблюдаемые действия.

Если среда объекта специфицирована, то это подразумевает, что для каждой *схемы операции*, требующей вводов и выводов, которые ассоциированы с интерфейсом объекта, т. е. для наблюдаемого действия, существует другая *схема операции* (возможно, ассоциированная с другим объектом), которая требует вводов и выводов того же типа, что и для рассматриваемого объекта. При соотнесении этих двух *схем операций* вводы/выводы рассматриваемой операции переименовываются в вывод/вводы операции, представляющей среду.

Появление операций, ссылающихся на глобальные для спецификации переменные, можно рассматривать как взаимодействия.

Все операции в Z являются элементарными: *схемы операции* в Z либо осуществляются полностью, либо не осуществляются совсем. Следовательно, в Z невозможны действия, не являющиеся элементарными.

Объект, взаимодействующий сам с собой, неформально может быть смоделирован композицией *схем операций* Z. Например, операция OpA с выводом $a!:A$ может быть скомпонована с операцией OpB с вводом $b?:A$ и дополнена предикатом *конъюнкции*, устанавливающим, что $a! = b?$.

В Z понятие взаимоотношения причины и результата не является строгим. Однако, если операция для своего осуществления требует ввода, то это можно рассматривать как то, что среда является причиной осуществления операции, т. е. среда действует как производитель, а *схема операции* — как потребитель. Аналогично, если *схема операции* создает вывод, то можно считать, что среда действует как потребитель, а операция — как производитель. Если данная *схема операции* требует ввода и дает вывод или не имеет ни ввода, ни вывода, то для конкретного действия нельзя установить отношение причины и результата.

Примечание — Следует отметить, что приведенное синтаксическое соглашение для различия внутренних и наблюдаемых действий является ограниченным, так как нет семантического различия между операциями, которые должны интерпретироваться как спонтанные или внутренние, и операциями, которые требуют участия среды; различие может быть установлено только в комментариях на естественном языке, которые должны сопровождать спецификации на Z. Как следствие этого, приведенное выше определение трактует потерянную очередь как подтип очереди. Ясно, что введенная дополнительная (потерянная) операция над потерянной очередью будет вести себя недетерминировано.

4.4.1.4 *Интерфейс*

Абстракция поведения объекта, полученная путем идентификации операций, ассоциированных с тем объектом, который образует основу для интерфейса. Во всех остальных *схемах операций* все вводы и выводы скрыты, и появление определенных в них *операций* рассматривается как внутренние

действия, т. е. они не требуют участия объектов среды. Получившийся в результате текст Z, представляющий данный объект, является шаблоном интерфейса. Любой экземпляр шаблона интерфейса является интерфейсом.

4.4.1.5 *Деятельность*

Понятие деятельности как однонаправленного ациклического графа действий непосредственно не существует в языке Z. Однако понятие деятельности может быть смоделировано путем некоторого расширения, а именно с помощью записи, что если действие *x* предшествует действию *y* в некоторой деятельности, то *постусловие* действия *x* должно удовлетворять *предусловию* для действия *y*.

4.4.1.6 *Поведение (объекта)*

Поведение объекта в данном состоянии является множеством всех возможных деятельностей, которые могут быть осуществлены из этого состояния. Фактическая последовательность действий, которые могут произойти, может зависеть от влияния среды объекта и ограничений, выраженных в *предусловиях*.

4.4.1.7 *Состояние (объекта)*

Значения используемых для вычисления *предусловий* переменных состояния, объявленных в ассоциированной (ых) с шаблоном объекта *схема (ы) состояния*.

4.4.1.8 *Коммуникация*

Коммуникация может быть смоделирована в Z через ввод и вывод операций. Обычно вводы/выводы *операционных схем* рассматриваются как коммуникации со средой объекта. Так как коммуникация осуществляется между объектами, то для моделирования коммуникации должна быть специфицирована среда объекта (см. 4.4.1.2). Тогда коммуникация получается с помощью первоначальной нормализации *операционных схем*, ассоциированных со взаимодействующими объектами и последующего их соединения, при котором вывод одной *схемы операции* переименовывается во ввод другой. Такое моделирование коммуникации требует, чтобы вводы и выводы соответствующих *схем операций* были одного типа.

Альтернативно коммуникации могут быть представлены появлением *схем операций*, ссылающихся на глобальные для спецификации переменные; при этом значения глобальных переменных после появления операции будут переданы всем другим ссылающимся на эти переменные *схемам операций*.

4.4.1.9 *Положение в пространстве*

Понятие пространства в Z не рассматривается. Положение в пространстве, в котором происходит действие, может быть задано в Z в терминах модели спецификации, а не в терминах реальной моделируемой системы. Таким образом, положение в пространстве может быть введено как тип Z. Если это сделано, то могут быть специфицированы отношения, связывающие *схемы операций* с положениями в пространстве. Тогда можно будет говорить о положениях в пространстве, в которых происходят действия.

4.4.1.10 *Положение во времени*

Понятие времени в Z не рассматривается. Положение во времени, в котором происходит действие, может быть задано в Z в терминах модели спецификации, а не в терминах реальной моделируемой системы. Таким образом, положение во времени может быть введено как тип Z, который может быть связан с данными действиями, например, некоторым отношением. Если это сделать, то можно будет количественно определять моменты времени, когда могут произойти действия.

4.4.1.11 *Точка взаимодействия*

Понятие точки взаимодействия зависит от взаимодействия и положений в пространстве и времени (см. 4.4.1.3, 4.4.1.9 и 4.4.1.10).

4.4.2 Специфицирующие понятия

4.4.2.1 *Композиция*

Композиция объектов — композиция объектов не обеспечивается явным образом в языке Z, так как в нем, кроме всего прочего, отсутствует инкапсуляция. Однако некоторые характеристики композиции можно смоделировать, включая в схему и переопределяя операции с помощью продвижения.

Композиция поведений — так как поведение в самом вырожденном случае может рассматриваться как действие, а действие в Z является осуществлением операции, определенной *схемой операции*, то композиция действий соответствует в Z комбинации *схем операций*. Схемы операций могут быть скомбинированы несколькими способами как:

- *схема вычисления,*

- *схема композиции* (;),
- *перекрытие* (⊕).

В общем случае из композиции могут быть получены характеристики результирующего поведения, которые не могут быть получены из отдельных комбинируемых поведений. Кроме того, могут быть подавлены нежелательные характеристики комбинируемых поведений.

4.4.2.2 *Составной объект*

См. выше интерпретацию композиции.

4.4.2.3 *Декомпозиция*

Декомпозиция объектов — см. выше интерпретацию композиции объектов.

Декомпозиция поведений — см. выше интерпретацию композиции поведений.

4.4.2.4 *Поведенческая совместимость*

Основана на понятии заменяемости в данной среде. Расширение является одним из возможных способов ее достижения. Расширение базового шаблона может иметь дополнительные компоненты в ассоциированной *схеме состояний*, более сильные инвариант состояний и начальные условия и больше *схем операций*. *Схемы операций*, ассоциированные с расширением типа шаблона, могут иметь более широкие *предусловия* и более сильные *постусловия*, чем соответствующие *схемы операций* в базовом типе шаблона.

4.4.2.5 *Уточнение*

Процесс преобразования одной спецификации в более подробную. Так как Z имеет дело с абстракциями систем, где данные и операции над этими данными используются для представления рассматриваемых систем, то могут быть идентифицированы две основные формы уточнения:

- *уточнение операции;*
- *уточнение данных.*

Для того чтобы уточнить спецификацию, уточнение должно гарантировать поведенческую совместимость между спецификацией и уточнением. С учетом этого существуют определенные условия для гарантии того, что уточнение спецификации Z создает допустимую более подробную спецификацию. Этими условиями являются *сохранность* и *жизнеспособность*. Условие сохранности уточнения спецификации состоит в том, что любые обстоятельства, достижимые для спецификации, должны быть достижимы и для уточнения. Условие жизнеспособности уточнения спецификации состоит в том, что при любых обстоятельствах, достижимых для спецификации, поведение уточнения должно допускаться спецификацией.

Условия сохранности и жизнеспособности должны применяться к уточнениям как операций, так и данных.

4.4.2.6 *Трасса*

Моделирование трассы Z ограничено по двум причинам. Во-первых, нет прямого способа записи действий объекта, во-вторых, нет семантического различия между внутренними и наблюдаемыми действиями, что уже указано в примечании к 4.4.1.3.

4.4.2.7 *Тип <X>*

Объекты, интерфейсы и действия могут иметь много различных ОРО типов. Типы ОРО соответствуют множествам в Z, а характеризующий предикат определяется принадлежностью к множеству.

4.4.2.8 *Класс <X>*

Множество всех *<X>*, для которых предикат принадлежности к множеству, т. е. тип ОРО, является истинным.

4.4.2.9 *Подтип/супертип*

Подтипы и супертипы ОРО соответствуют подмножествам и супермножествам в Z.

4.4.2.10 *Подкласс/суперкласс*

Подклассы и суперклассы ОРО соответствуют отношениям подмножества и супермножества в Z.

4.4.2.11 *Шаблон <X>*

Шаблон объектов — фрагменты спецификации, которые представляют общие характеристики возможных состояний объектов, имеющих однозначную (неизменную) идентификацию, по которой на них можно сослаться, и соответствующее множество *схем операций*, действующих на это состояние. Если шаблон объекта является родовым, то точная форма шаблона будет дана только тогда, когда будут даны параметры типа.

Шаблон интерфейсов — множество *схем операций*, полученных из представляющего шаблон объекта текста Z способом, описанным при интерпретации интерфейса (см. 4.4.1.4). Если шаблон объекта является родовым, то точная форма шаблона интерфейса будет дана только тогда, когда

будут даны параметры типа. Шаблоны интерфейсов могут быть скомбинированы операциями Z для комбинации схем.

Шаблон действий — *схема операции*. Шаблоны действий могут быть скомбинированы операциями Z для комбинации схем. Если шаблон действия является родовым, то точная форма шаблона действия будет дана только тогда, когда будут даны параметры типа.

4.4.2.12 *Сигнатура интерфейса*

Набор шаблонов действий, связанных с взаимодействиями интерфейса.

Примечание — Следует заметить, что ГОСТ Р ИСО/МЭК 10746-2 трактует сигнатуру интерфейса как набор шаблонов действий, ассоциированных с взаимодействиями интерфейса. Если принять, что шаблон действий является общими характеристиками совокупности действий, похоже, что это определение некорректно. А именно, шаблон действий, вероятно, включает в себя семантическую информацию и синтаксис. Однако общая интерпретация сигнатуры интерфейса имеет дело, главным образом, с синтаксисом. Если рассматривать синтаксическое понятие сигнатуры интерфейса, то оно соответствует множеству нормализованных шаблонов действий, ассоциированных с взаимодействиями интерфейса, со всеми не относящимися к вводу/выводу переменными, объявленными в сигнатуре *схемы операции* и присутствующими в предикатной части *схемы операции*.

4.4.2.13 *Реализация (шаблона <X>)*

Реализация шаблона объекта — спецификация начальных значений переменных, указанных в шаблоне объекта. В Z часто это делается явно путем инициализации схемы или множества схем. После инициализации значения этих переменных должны соответствовать допустимому состоянию, т. е. состоянию, удовлетворяющему всем *инвариантам,* которые могут присутствовать.

Примечание — Эти *инварианты* могут ссылаться на другие объекты.

Реализация шаблона интерфейса — ограничение фрагментов шаблона интерфейса спецификации Z путем задания значений родовых параметров и удовлетворения соответствующих предикатов для переменных, указанных в шаблоне интерфейса.

Реализация шаблона действия — осуществление операции, специфицированной в *схеме операций*.

4.4.2.14 *Роль*

Роль может быть представлена именем, которое идентифицирует индивидуальный объект, например через идентификатор, находящийся в ассоциированной с объектом *схеме состояний*. Это имя может использоваться в *схеме кадрирования* для продвижения операций индивидуального объекта до уровня (спецификации) системы в целом.

Примечание — Данное описание не адекватно отражает назначение ГОСТ Р ИСО/МЭК 10746-2.

4.4.2.15 *Создание (<X>)*

Создание объекта — задается предоставлением допустимого начального состояния для текста Z, ассоциированного с шаблоном объекта. Конкретно, путем подстановки начальных значений переменным, заданным в *схеме состояний* объекта. В Z это часто делается явно через схему инициализации. В таком случае действие создания представляется осуществлением операции, заданной в схеме инициализации.

Создание интерфейса — в Z внутренне связано с созданием объектов. А именно, когда инициализируется текст, связанный с шаблоном объекта, инициализируются и любые шаблоны интерфейсов, которые могут присутствовать.

При создании необходимо гарантировать уникальность в пределах спецификации создаваемого объекта. Этого можно достичь с помощью *схемы кадрирования* с соответствующим предикатом, гарантирующим уникальность всех создаваемых объектов. Эта *схема кадрирования* может использоваться для продвижения инициализации объекта до влияния на спецификацию в целом.

Примечания

1 Подразумевается, что так как схема инициализации дана как часть спецификации, то объект создается. Однако в Z нет понятия спецификации, фактически применяющей эту схему инициализации, так как сам Z не является выполняемым языком. Таким образом, фактически это является введением объекта, т. е. объект реализуется методами, не охваченными моделью. Понятие реализации в спецификации Z частично является созданием (в части задания схемы инициализации), а частично — введением (в части не охватываемого моделью применения схемы инициализации).

2 Обычно, после применения схемы инициализации следует проверка обязательств для гарантии того, что объект находится в допустимом начальном состоянии.

4.4.2.16 *Введение (объекта)*

См. создание (объекта), 4.4.2.15.

4.4.2.17 *Удаление (<X>)*

Абстрактное представление удаления можно получить, если используется *схема кадрирования*. Удаление можно смоделировать как осуществление операции по удалению состояния и идентификации объекта из системы в целом.

Возможен случай, когда может быть смоделировано удаление на основе безактивности объекта. Например, объект, связанное с которым будущее поведение не может быть осуществлено, так как при этом будут нарушены *инварианты*, может рассматриваться в некотором смысле как удаленный. Эта форма удаления не охватывается корректно определением, данным в ГОСТ Р ИСО/МЭК 10746-2, так как оно не является разрушением объекта как таковым.

4.4.2.18 *Экземпляр типа*

Экземпляр типа ОРО представляется в Z элементом множества, члены которого удовлетворяют предикату, т. е. типа ОРО. Для более специфического понятия типа, такого как тип шаблона, экземпляр типа объекта или интерфейса соответствует инициализации текста на Z (или его расширения), представляющего рассматриваемый объект или интерфейс, такой, что для этого объекта или интерфейса существует допустимое начальное состояние. Здесь схемой инициализации должны удовлетворяться характеризующий предикат, заданный текстом спецификации, и ассоциированные предикаты для возможных допустимых связываний (*инвариантов*).

Так как *схема операции* задает характеристики типа действия, то экземпляр типа действия задается появлением этой или другой *схемы операции*, являющейся расширением данной, т. е. расширение включает в себя характеризующие свойства типа действия.

4.4.2.19 *Тип шаблона (<X>)*

В Z тип шаблона объекта или интерфейса является предикатом, который схема инициализации сохраняет для объекта в допустимом состоянии и ассоциированных с ним интерфейсов. Таким образом, все переменные состояния должны получить значения, а все необходимые предикаты (*инварианты*) должны быть удовлетворены. Часто для проверки типа шаблона объекта или интерфейса требуется проведение доказательства.

Тип шаблона действия соответствует предикату, устанавливающему, когда может появиться шаблон действия, заданный *схемой операции*, т. е. реализация данной *схемы операции*. Таким образом, все реализации должны удовлетворять предикатам для допустимых значений переменных, как установлено в предикатах *схемы операции*.

4.4.2.20 *Класс шаблонов (<X>)*

Класс шаблонов <*X*> — это множество всех <*X*>, которые являются экземплярами этого шаблона <*X*>, где <*X*> может быть объектом, интерфейсом или действием.

4.4.2.21 *Производный класс/базовый класс*

В Z для двух шаблонов А и В, где А — нарастающая модификация В и экземпляры А и В находятся в отношении производный/базовый класс, нарастающие модификации В для создания А могут включать в себя:

- добавление или удаление параметров состояния,
- добавление, удаление или изменение операций, или
- усиление или ослабление *инвариантов*.

4.4.2.22 *Инвариант*

Предикат, постоянная справедливость которого требуется спецификацией. Z позволяет записывать *инварианты* непосредственно в схемах и *аксиоматических описаниях*. Часто *инварианты* устанавливают ограничения на возможные значения, которые могут принимать переменные в схемах и *аксиоматических описаниях*. Если это так, то *инвариант* часто используется для ограничения возможного поведения, заданного в спецификации.

4.4.2.23 *Предусловие*

Такое условие для состояния системы до осуществления операции, определенной *схемой операции*, и на ее ввод, что существуют возможное состояние после осуществления операции и вывод, удовлетворяющие *постусловиям*. Z позволяет записывать *предусловия* непосредственно.

4.4.2.24 *Постусловие*

Предикат, описывающий множество состояний, в которых может оказаться система после осуществления операции, определенной *схемой операции*. Z позволяет записывать *постусловия* непосредственно.

4.5 Архитектурная семантика в ESTELLE

ESTELLE является стандартизированным (ИСО/МЭК 9074) языком формальных спецификаций (ЯФС). Методические материалы приведены в указанном стандарте. Существуют различные программные средства, поддерживающие моделирование и распределенную реализацию спецификаций ESTELLE.

ESTELLE основан на расширенном конечном автомате. Система моделируется иерархически структурированным множеством экземпляров модулей, взаимодействующих асинхронным образом с помощью обмена сообщениями через каналы. Синтаксис языка и определения типов данных и переменных основаны на языке Паскаль.

В спецификации на ESTELLE видимый внешне интерфейс модуля определен в *заголовке модуля*, тогда как *тело модуля* описывает его внутреннюю структуру и поведение. *Экземпляры модуля* определяют *точки взаимодействия*, через которые они могут отправлять и получать сообщения. *Две точки взаимодействия* могут быть соединены, если они были определены для противоположных ролей одного и того же *определения канала*. Определение канала содержит две *роли* для концов *канала*. Для каждой *роли* определены сообщения (в ESTELLE они называются *взаимодействиями*), которые могут быть отправлены. *Определение взаимодействия* состоит из имени и набора параметров. Для каждой *точки взаимодействия* устанавливается очередь, в которой хранятся входящие сообщения. *Экземпляр модуля* может иметь общую очередь, которая совместно используется несколькими или всеми *точками взаимодействия*.

Структура спецификации ESTELLE динамическая, т. е. *экземпляры модулей* динамически могут быть реализованы и удалены, а точки взаимодействия — соединены и отсоединены. Экземпляры модулей спецификации ESTELLE расположены в строгом иерархическом порядке. Каждый экземпляр модуля может реализовывать и удалять дочерние *экземпляры модулей* или соединять и отсоединять свои *точки взаимодействия*. Единственный способ для *экземпляра модуля* достичь экземпляра одного с ним уровня (или другие *экземпляры модулей*, не являющиеся его дочерними) — через обмен *взаимодействиями в канале*.

Первоначально ESTELLE разрабатывался для спецификации коммуникационных услуг и протоколов. ESTELLE поддерживает инкапсуляцию, но он не содержит объектно-ориентированных характеристик наследования и создания подтипов. Несмотря на это ESTELLE позволяет выразить большинство понятий ОРО. Спецификации ESTELLE просты для чтения, и так как ESTELLE является конструктивным методом, то он хорошо подходит для реализации.

В настоящем разделе объясняется, как основные моделирующие понятия ОРО могут быть выражены на ESTELLE.

Во избежание путаницы в терминологии ОРО и ESTELLE в последующих подразделах курсивом выделены специфические для ESTELLE термины. Следует обратить внимание на то, что в ESTELLE понятие *взаимодействия* означает обмен сообщениями между *экземплярами модулей*.

4.5.1 Основные моделирующие понятия

4.5.1.1 *Объект*

Моделируется в ESTELLE *экземпляром модуля*.

4.5.1.2 *Среда (объекта)*

Часть спецификации, которая не является частью *экземпляра модуля* (в частности, *родительский экземпляр* и другие экземпляры), которые через *каналы* связаны с *точками взаимодействия* экземпляра модуля.

4.5.1.3 *Действие*

Действие в ESTELLE представляется выполнением *раздела-WHEN*, предложения действия, всей *транзакции* или *процедуры*. Возможными предложениями действия являются:

- *output*,
- *init*,
- *connect*,
- *attach*,
- *release*,
- *disconnect*,
- *detach*,
- *присваивание*.

Имеется несколько типов **взаимодействий.** Выполнение предложения output является, как и выполнение *раздела-WHEN*, взаимодействием. Кроме того, последовательность действий, состоящая из *ввода взаимодействия* через *точку взаимодействия* и последующего его потребления при выпол-

нении *раздела-WHEN,* может рассматриваться как одно взаимодействие. Другим типом взаимодействия является обновление *экспортированных переменных.* Все другие действия являются **внутренними**.

Примечание — Так как *каналы* ESTELLE имеют бесконечные очереди, то среда всегда готова к участию во *взаимодействиях.*

4.5.1.4 *Интерфейс*

В ESTELLE имеется два вида интерфейсов:

- интерфейс первого вида образуется множеством всех *взаимодействий*, определенных для назначенной *роли* (в соответствующем *определении канала*) в *точке внешнего взаимодействия* объекта;
- интерфейс второго вида образуется множеством всех *экспортированных переменных* объекта.

В первом случае множество взаимодействий (образующих интерфейс) содержит набор предложений *output* и (или) *разделов-WHEN,* которые объект выполняет в *точке взаимодействия.* Во втором случае множество взаимодействий состоит из всех предложений, которые читают или записывают *экспортированные переменные.* Через интерфейс этого вида возможны взаимодействия только между *экземпляром модуля* и его *родительским экземпляром.*

4.5.1.5 *Деятельность*

В общем случае деятельность не может быть обозначена явно, так как она может охватывать несколько объектов. Деятельность является некоторой последовательностью взаимосвязанных действий, например *переходов*, процедур или *функций*, если отдельное предложение рассматривать как действие.

4.5.1.6 *Поведение (объекта)*

Определяется множеством всех переходов этого объекта. Ограничения на обстоятельства, при которых могут происходить заданные действия, определяются в *разделах переход* (например, в *разделе-FROM* или *разделе-PROVIDED*). Объект может демонстрировать недетерминированное поведение.

4.5.1.7 *Состояние (объекта)*

Содержит следующие компоненты:

- *состояние контроля экземпляра модуля,*
- содержимого очередей *точек взаимодействия,*
- значений *внутренних* и *экспортированных переменных* и формальных параметров *экземпляра модуля*,
- состояний существующих дочерних *экземпляров*, их структур соединений и *экспортированных переменных.*

Вместе эти компоненты определяют множество всех последовательностей действий (*переходов*), в которых может принять участие *экземпляр модуля.*

4.5.1.8 *Коммуникация*

Информация между объектами переносится двумя путями:

- через *вывод* и последующее получение *взаимодействия* (т. е. через последовательность действий, образующую взаимодействие),
- через чтение и обновление *экспортированных переменных.*

4.5.1.9 *Положение в пространстве*

Действия происходят в *экземплярах модулей* или в их *точках взаимодействия.* Таким образом, положение действия в пространстве соответствует положению в пространстве соответствующего *экземпляра модуля.*

4.5.1.10 *Положение во времени*

В ESTELLE время представляется только в *разделах-DELAY,* возможно, с соответствующими *переходами.* Относительно времени можно лишь принять, что оно равномерно возрастает по мере выполнения процесса.

4.5.1.11 *Точка взаимодействия*

Представляется либо *точкой взаимодействия,* либо множеством всех *экспортированных переменных* объекта.

4.5.2 Специфицирующие понятия

Выражение на ESTELLE специфицирующих понятий затруднено, так как в нем ограничена поддержка объектно-ориентированных понятий.

4.5.2.1 *Композиция*

Композиция объектов — *экземпляр модуля* может быть составлен из набора дочерних *экземпляров модулей.* На верхнем уровне, спецификация может быть составлена из набора *экземпляров систем.*

Композиция поведений — когда *экземпляр модуля* является композицией дочерних *экземпляров*, их поведение составлено:

- параллельно, с перекрытием, если родительский модуль имеет атрибут *activity* или *systemactivity*, или
- параллельно, синхронно, если родительский модуль имеет атрибут *process* или *systemprocess*.

4.5.2.2 *Составной объект*

Экземпляр модуля, который описывается множеством дочерних *экземпляров модулей*.

4.5.2.3 *Декомпозиция*

Декомпозиция объектов — спецификация данного объекта как композиции.

Декомпозиция поведений — спецификация данного поведения как композиции.

4.5.2.4 *Поведенческая совместимость*

В ESTELLE нет прямого способа выразить поведенческую совместимость. Однако семантическая основа языка в терминах переходных систем допускает определение поведенческой совместимости и ее верификации.

4.5.2.5 *Уточнение*

Объект может быть уточнен путем построения подструктуры кооперирующихся дочерних *экземпляров*.

4.5.2.6 *Трасса*

Может быть получена из динамической интерпретации (выполнения или моделирования) спецификации ESTELLE.

4.5.2.7 *Тип <X>*

В ESTELLE нет способа для явной формулировки предикатов.

4.5.2.8 *Класс <X>*

Не поддерживается.

4.5.2.9 *Подтип/супертип*

Не поддерживается.

4.5.2.10 *Подкласс/суперкласс*

Не поддерживается.

4.5.2.11 *Шаблон <X>*

Шаблон объекта представлен *определением тела модуля* вместе с соответствующим *определением заголовка модуля*. Шаблонами действий являются:

- *output;*
- *init;*
- *release;*
- *connect;*
- *disconnect;*
- *attach;*
- *detach;*
- *присваивание;*
- *раздел-WHEN.*

Так как существует два вида интерфейсов, то шаблон интерфейса задается:

- соответствующим *определением канала* (для множества *взаимодействий в точке взаимодействия*). В этом случае соответствующее поведение интерфейса может быть задано через дочерний *экземпляр модуля*, который реализуется и присоединяется к *точке взаимодействия* при создании интерфейса. Таким образом, шаблон интерфейса содержит *заголовок модуля* и *определение* тела этого модуля;
- в *определении заголовка модуля* объекта (для *экспортированных переменных*).

4.5.2.12 *Сигнатура интерфейса*

Представляется двумя способами:

- определениями *взаимодействий*, которые содержатся в интерфейсе (для множества *взаимодействий* в *точке взаимодействия*);
- действиями присваивания *экспортированным переменным*.

4.5.2.13 *Реализация (шаблона <X>)*

Реализацией объекта является *экземпляр модуля* соответствующего *определения модуля*. Реализацией интерфейса является конкретная *точка взаимодействия* (возможно, с присоединенным к ней конкретным дочерним *экземпляром модуля*, представляющим поведение интерфейса) или множество *экспортированных переменных* конкретного *экземпляра модуля*.

4.5.2.14 *Роль*

Роль ассоциирована с каждой декларацией *точки взаимодействия*. Для того чтобы *точки взаимодействия* были соединены, они должны иметь противоположные *роли* в одном и том же *определении канала*. ESTELLE поддерживает спецификацию разных *тел модуля* для одного и того же *заголовка модуля*. Это сделано для обеспечения выбора поведения при реализации объекта. Интерфейсы объекта не зависят от выбранного *тела модуля*.

4.5.2.15 *Создание (<X>)*

Объекты создаются действием init. Интерфейсы создаются неявно при создании объекта. Динамического создания интерфейсов нет. Однако создание интерфейса как *точки взаимодействия* может быть смоделировано выбором *точки взаимодействия* (из массива *точек взаимодействия*) с соответствующим *каналом* и *ролью*. Если для представления поведения интерфейса специфицирован дочерний *модуль*, то этот *модуль* реализуется и присоединяется к *точке взаимодействия*.

4.5.2.16 *Удаление (<X>)*

Объект удаляется явно действием *release*. Интерфейсы удаляются неявно вместе с объектом. Если моделируется динамическое создание интерфейсов так, как указано выше (см. 4.5.2.15),то удаление интерфейса соответствует *освобождению* присоединенного *дочернего экземпляра модуля* (если он есть) и превращению *точки взаимодействия* в точку, не представляющую интерфейс.

4.5.2.17 *Введение (объекта)*

Нет способов для введения объектов. В общем случае методы, не охваченные моделью, могут быть задействованы с помощью *примитивов*, *внешних процедур* и *функций*.

4.5.2.18 *Экземпляр типа*

Так как в ESTELLE нет подтипов и подклассов, то экземпляры шаблона получаются реализацией этого шаблона.

4.5.2.19 *Тип шаблона (<X>)*

Для объекта — предикат, который может утверждать, что объект является экземпляром соответствующего *определения модуля*. Интерфейс (множество *взаимодействий* в *точке взаимодействия*) является экземпляром соответствующего *определения канала*. Интерфейс, представленный множеством всех *экспортированных переменных*, является экземпляром соответствующего *определения заголовка модуля*. Действие является экземпляром соответствующего шаблона действия.

Примечание — В ESTELLE может быть установлен лишь тот факт, что объект, интерфейс или действие являются реализацией данного шаблона. Так как понятие подкласса фактически не поддерживается, то таким образом могут быть идентифицированы только реализации шаблона, но не экземпляры.

4.5.2.20 *Класс шаблона (<X>)*

Классом шаблона объектов является множество всех *экземпляров* одного модуля. Классом шаблона интерфейсов является множество всех точек взаимодействия, определенных с использованием одного *определения канала* и *роли*.

4.5.2.21 *Производный класс/базовый класс*

Не поддерживается.

4.5.2.22 *Инвариант*

В ESTELLE нет способов для явной формулировки инвариантов.

4.5.2.23 *Предусловие*

Предусловия осуществления *перехода* устанавливаются в *разделах переходов*. Предусловия действия в *блоке перехода* задаются предусловиями *перехода* и действиями, содержащимися в этом *блоке* и предшествующими рассматриваемому действию.

4.5.2.24 *Постусловие*

Постусловие перехода определяется *разделом-TO* перехода и действиями в *блоке перехода*.

УДК 681.324:006.354 ОКС 35.080 П85 ОКСТУ 4002

Ключевые слова: взаимосвязь открытых систем, управление данными, открытая распределенная обработка, архитектура, базовая модель

Редактор *В.П. Огурцов*
Технический редактор *В.Н. Прусакова*
Корректор *М.С. Кабашова*
Компьютерная верстка *И.А. Налейкиной*

Изд. лиц. № 02354 от 14.07.2000. Сдано в набор 12.02.2004. Подписано в печать 23.03.2004. Усл. печ. л. 4,18. Уч.-изд.л. 4,10.
Тираж 260 экз. С 1222. Зак. 326.

ИПК Издательство стандартов, 107076 Москва, Колодезный пер., 14.
e-mail: info@standards.ru
Набрано в Издательстве на ПЭВМ
Отпечатано в филиале ИПК Издательство стандартов — тип. «Московский печатник», 105062 Москва, Лялин пер., 6.
Плр № 080102

gost.e-ecolog.ru