

S-100 Часть 1
Язык концептуальной схемы

Contents

1-1 Цель	3
1-2 Соответствие	3
1-3 Нормативные ссылки	3
1-4 UML Профиль S-100	4
1-4.1 Введение	4
1-4.2 Общие правила использования UML	4
1-4.3 Классы	5
1-4.4 Атрибуты	5
1-4.5 Базовые типы данных	6
1-4.5.1 Общие положения	6
1-4.5.2 Типы примитивов	7
1-4.5.3 Комплексные типы	9
1-4.6 Предопределенные производные типы	13
1-4.7 Перечисляемые типы	14
1-4.8 Типы списков кодов	15
1-4.9 Взаимосвязи и ассоциации	17
1-4.9.1 Взаимосвязи	17
1-4.9.2 Ассоциация, композиция и агрегация	18
1-4.10 Стереотипы	22
1-4.10.1 Использование стандартных стереотипов UML для классов/классификаторов	22
1-4.11 Опционные, условные и обязательные – атрибуты и ассоциации	23
1-4.12 Присвоение имен и поименованные пробелы	23
1-4.13 Примечания	25
1-4.14 Пакеты	25
1-4.15 Документирование моделей в S-100	26

1-1 Цель

Эта часть определяет язык концептуальной схемы и базовые типы данных для использования сообществом ИЮ. Она определяет комбинацию статической структурной диаграммы Унифицированного языка моделирования (UML) и набора типовых определений базовых данных как язык концептуальной схемы для спецификаций географической информации. (UML - стандартизованный язык моделирования общего назначения в области разработки программного обеспечения. Он включает набор техник графической нотации для создания абстрактных моделей конкретных систем. UML объединяет передовую практику, основанную на таких концепциях моделирования данных, как диаграммы связей субъектов, рабочий процесс, моделирование объектов и моделирование компонентов).

Кроме того, эта часть является руководством по использованию UML для создания стандартизованных моделей географической информации и сервисов, которые являются основой достижения целей функциональной совместимости. Поскольку это все связано с UML, раздел специальных терминов и определений для UML дается в Приложении А (Термины и Определения).

1-2 Соответствие

Любая концептуальная схема, написанная для спецификации, которая претендует на соответствие этой части S-100, должна строиться по правилам, изложенным в пункте 5. Этот профиль использует класс соответствия 2 ISO 19106:2004.

1-3 Нормативные ссылки

Для использования этой части S-100 требуются следующие справочные документы. К документам с указанной датой выпуска применяется только упомянутое издание. Для недатированных ссылок применяется последнее издание документа (включая поправки).

ISO 19103:2005(E), *Geographic information — Conceptual schema language*

ISO 8601:2004(E), *Data elements and interchange formats — Information interchange — Representation of dates and times*

ISO 19136: *Geographic Information – Geography Markup Language*

ISO 25964-1: *Information and documentation — Thesauri and interoperability with other vocabularies — Part 1: Thesauri for information retrieval.*

ISO 25964-2: *Information and documentation — Thesauri and interoperability with other vocabularies — Part 2: Interoperability with other vocabularies*

OGC 10-129r1: *Geographic Information – Geography Markup Language (GML) – Extended schemas and encoding rules*

OMG Unified Modelling Language (OMG UML), Superstructure, V2.1.2

RFC 3986, *Uniform Resource Identifier (URI): Generic Syntax*. T. Berners-Lee, R. Fielding, L. Masinter. Internet Standard 66, IETF. URL: <http://www.ietf.org/rfc/rfc3986.txt> or <http://www.rfc-editor.org/info/std66>

RFC 2141, *URN Syntax*. R. Moats. IETF RFC 2141, May 1997. URL: <http://www.rfceditor.org/info/rfc2141>

RFC 8089, *The "file" URI Scheme*, February 2017. URL: <https://www.ietf.org/rfc/rfc8089.txt>

SKOS: *SKOS – Simple Knowledge Organization System – Reference*. W3C Recommendation, 2009. <http://www.w3.org/TR/2009/REC-skos-reference-20090818/>

1-4 UML профиль S-100

1-4.1 Введение

Этот раздел определяет правила и руководство по использованию UML для географической информации.

Подразделы данного раздела представлены следующим образом:

- 1) Общие правила использования UML
- 2) Классы
- 3) Атрибуты
- 4) Базовые типы данных
- 5) Предопределенные производные типы
- 6) Перечисляемые типы
- 7) Типы списков кодов
- 8) Взаимосвязи и ассоциации
- 9) Стереотипы
- 10) Дополнительные, условные и обязательные атрибуты и ассоциации
- 11) Присвоение имен и пространства имен
- 12) Примечания
- 13) Пакеты
- 14) Документирование моделей в S-100

1-4.2 Общие правила использования UML

UML (Unified Modelling Language) должен использоваться в соответствии с UML 2. Нормативные модели должны использовать диаграммы классов и диаграммы пакетов. Другие типы диаграмм UML могут использоваться информативно. Все

нормативные модели должны содержать полные определения атрибутов, ассоциаций и соответствующих типов данных.

1-4.3 Классы

Класс – это описание набора объектов, которые имеют одинаковые атрибуты, функции, методы, связи, поведение и ограничения. Класс представляет собой концепцию, являющуюся предметом моделирования. В зависимости от типа модели, концепция может основываться на объектах реального мира (для концептуальной модели) или на реализации нескольких платформо независимых концепций (для специфицированных моделей), или нескольких концепциях на базе специфических платформ (для применяемых моделей (implementation models)).

Классификатор является генерализацией классов, который включает другие классоподобные элементы, такие как типы данных, активные объекты и компоненты. Класс UML имеет имя, набор атрибутов, набор функций и ограничений. В S-100 функции не используются. Класс может участвовать в ассоциациях.

Класс в соответствии с S-100 рассматривается как спецификация, а не реализация.

Использование множества производных должно быть минимизировано, поскольку это ведет к увеличению сложности модели.

Абстрактный класс обозначается именем, написанным курсивом

1-4.4 Атрибуты

UML обозначение для атрибутов имеет форму:

```
optvisibilityopt имя: optpackage ::opt opttypeopt opt[multiplicity]opt opt= initial valueopt  
opt{propertystring}opt
```

Атрибут должен быть уникальным в рамках контекста класса и его супер-типа, или еще быть производным атрибутом, переопределенным из супер-типа.

Видимость (visibility) атрибутов показана условными знаками в таблице ниже. Защищенная и частная видимости в стандартных спецификациях обычно не используются. Должны использоваться соответствующие условные знаки. Такие же условные знаки видимости должны использоваться для ассоциаций.

Таблица 1-1 — Видимость атрибутов

Условный знак	Значение
+	Публичная видимость
#	Защищенная видимость
-	Частная видимость
/	Производный атрибут

Все атрибуты должны быть классифицированы по типам и тип должен существовать, как конструированный/определенный тип. Тип всегда должен быть определен, типа по умолчанию быть не должно.

Если явной множественности (multiplicity) не дано, считается, что она равна 1 (*т. е. обязательная*).

Атрибут может определять значение по умолчанию, которое используется в случаях, когда создается объект такого типа. Значения по умолчанию определяются явными значениями по умолчанию в UML определении атрибута.

Могут использоваться следующие свойства:

- readOnly (только для чтения) – значение атрибута не может быть изменено и должно быть инициализировано.
- ordered (упорядоченный) – применяется для множества (более одного) атрибутов, с имеющим значение порядком элементов.

ПРИМЕРЫ + center: Point = (0,0) {readOnly}
+ origin: Point [0..1] // множественность 0..1 означает, что это опция
+ controlPoints : Point [2..*] {ordered}

1-4.5 Базовые типы данных

1-4.5.1 Общие положения

Базовые типы данных делятся на две категории:

- 1) Типы примитивов: Фундаментальные типы представления значений, например, символьная строка (CharacterString), Integer, Boolean, Date, Time, etc.
- 2) Типы комплексов: Комбинация типов, например, комбинация типов измерения и единиц измерения.

Содержание базовых типов данных описано в последующих подразделах.

Форматы данных S-100 могут представлять значения с использованием соответствующих встроенных или стандартных типов. Например, формат ISO 8211

(Часть 10a) представляет значения всех тематических атрибутов фичеров в строках вместо использования знакового целого по ISO 8211, без знакового целого, или знакового представления с плавающей точкой для тематических атрибутов типа S-100 целый (Integer) или действительный (Real).

1-4.5.2 Типы примитивов

Следующие типы примитивов используются в диаграммах UML S-100.

Таблица 1-2 — Типы данных

Название	Описание
Integer	Целое число со знаком, представление целого числа зависит от инкапсуляции и назначения данных. ПРИМЕР 29, -65547
PositiveInteger	Целое число без знака больше 0.
NonNegativeInteger	Целое число без знака большее или равное 0.
Real	Действительное (с плавающей запятой) число со знаком, состоящее из мантиссы и экспоненты, представление действительного числа зависит от инкапсуляции и назначения данных. ПРИМЕР 23.501, -1.234E-4, -23.0
Boolean	Значение, представляющее бинарную логику. Значение может быть или истинным (true) или ложным (false).
CharacterString	CharacterString - последовательность знаков произвольной длины, включающая диакритические и специальные знаки из какого-либо принятого набора знаков.
Date	Значение года, месяца и дня по Григорианскому календарю. Закодированная дата представляет собой строку, соответствующую формату календарной даты по ISO 8601. ПРИМЕР 19980918 (YYYYMMDD) В форматах XML вместо базового представления по ISO 8601 следует использовать стандартный тип XML-схемы (который не является стандартным типом для XML). ПРИМЕР : 1998-09-18
Time	В 24-часовой системе часов время определяется в часах, минутах и секундах. Кодировка времени должна представляться в полном базовом формате, определенном в ISO 8601. Полный базовый формат означает, что используются часы, минуты и секунды. Базовый формат означает, что разделительные знаки опускаются. Время желательно выражать как Всемирное Время UTC. ПРИМЕР 183059Z Время может выражаться как местное время с определенным смещением относительно UTC. ПРИМЕР 183059+0100 Время может выражаться как местное время без смещения относительно UTC. ПРИМЕР 183059 Полное представление времени 15 часов 27 минут и 46 секунд в Женеве (зимой на час раньше UTC) и в Нью-Йорке (зимой на пять

	<p>часов позже UTC), вместе с указанием разницы во времени местного времени и UTC, даны в качестве примеров. Geneva: 152746+0100 New York:152746-0500</p> <p>Часы работы служб, которые доступны в течение всего года в районе, где летнее время влияет на смещение относительно UTC, могут быть выражены как местное время без определенного смещения.</p> <p>Открыто: 074500 Закрыто: 161500</p> <p>В форматах XML вместо базового представления ISO 8601 следует использовать стандартный тип XML-схемы (который не является стандартным типом для XML).</p> <p>ПРИМЕРЫ: 18:30:59Z;18:30:59+01:00;18:30:59</p>
<p>DateTime</p>	<p>DateTime – это комбинация даты и времени. Кодирование DateTime выполняется в соответствии с ISO 8601 (см. выше). ПРИМЕР: 19850412T101530</p> <p>В форматах XML вместо базового представления ISO 8601 следует использовать стандартный тип XML-схемы (который не является стандартным типом для XML). ПРИМЕРЫ: 1985-04-12T10:15:30; 1985-04-12T10:15:30+01:00; 1985-04-12T10:15:30Z</p>
<p>S100_TruncatedDate</p>	<p>S100_TruncatedDate позволяет указывать дату или частичную дату. По крайней мере один из следующих компонентов должен содержать опущенные элементы, замененные эквивалентным числом переносов, определяемым форматом.</p> <p>Компоненты: YYYY Целое число года между 0000 и 9999 MM Целое число месяца между 01 – 12 (включительно) DD Целое число дня между 01 и 28, 29, 30, или 31 (включительно), в соответствии со значениями года и месяца, если они указаны.</p> <p>Этот тип может быть использован для кодирования повторяющихся моментов (см. Часть 3, пункт 3-8).</p> <p>ПРИМЕР 1 (ISO 8211, HDF5): YYYYMMDD с неуказанным компонентом(ами), замененным переносами, так что длина поля всегда составляет 8 символов: - - - -1217 представляет 17 декабря любого года</p> <p>ПРИМЕР 2 (XML): Следует использовать соответствующий тип схемы XML: - - 12-17 представляет 17 декабря любого года (соответствует типу XML gMonthDay)</p> <p>Часть 10b содержит дополнительные сведения о кодировании в наборах данных GML.</p>

1-4.5.3 Комплексные типы

1-4.5.3.1 UnlimitedInteger (неограниченное целое число)

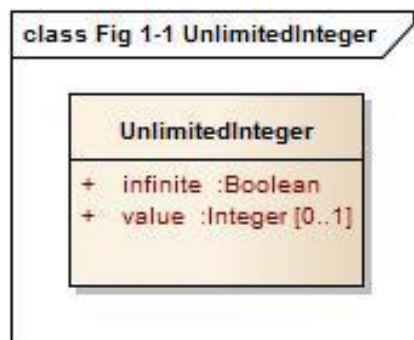


Рисунок 1-1 - UnlimitedInteger

Целое число со знаком, значение которого не ограничено.

1-4.5.3.2 Matrix (массив, матрица)

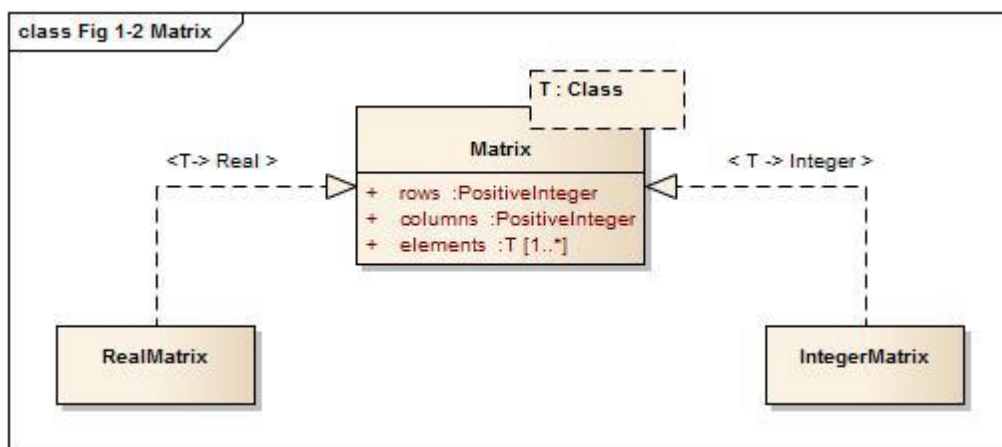


Рисунок 1-2 – Матрица

Массив действительных или целых чисел.

1-4.5.3.3 S100_Multiplicity (Множественность S100)

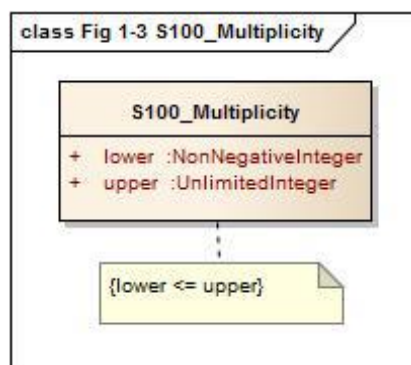


Рисунок 1-3 – S100_Multiplicity

Определяет диапазон множественности от наименьшего до наибольшего значения. Наибольшее значение может быть неограниченным.

1-4.5.3.4 S100_NumericRange (Числовой диапазон S100)

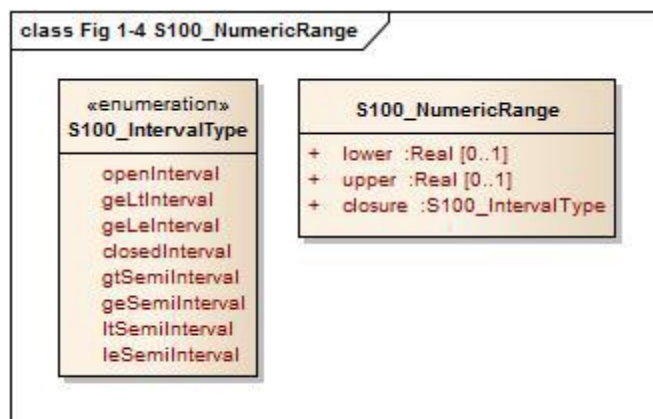


Рисунок 1-4 – S100_NumericRange

Определяет допустимые числовые интервалы, их нижние и верхние границы и тип завершения интервала.

ПРИМЕЧАНИЕ: Атрибут **lower (нижний)** должен использоваться для всех завершений кроме **ltSemiInterval** или **leSemiInterval**. Атрибут **upper (верхний)** должен использоваться для всех завершений кроме **gtSemiInterval** или **geSemiInterval**.

ПРИМЕЧАНИЕ: Интервал с одиночным значением должен кодироваться с **upper = lower** и устанавливать **closure** для **closedInterval**.

Закрытие интервала определяется оператором S100_IntervalType (Тип интервала S100). Знаки имеют следующие значения:

Таблица 1-3 — Типы интервалов

Имя	Описание	Обозначение	Определение (где $a \leq b$)
openInterval	Открытый интервал	$(lower, upper)$	$lower < x < upper$
geLtInterval	Полуоткрытый интервал справа	$[lower, upper)$	$lower \leq x < upper$
gtLeInterval	Полуоткрытый интервал слева	$(lower, upper]$	$lower < x \leq upper$
closedInterval	Закрытый интервал	$[lower, upper]$	$lower \leq x \leq upper$
gtSemiInterval	Полуоткрытый луч слева	$(lower, \infty)$	$lower < x$
geSemiInterval	Закрытый луч слева	$[lower, \infty)$	$lower \leq x$
ltSemiInterval	Полуоткрытый луч справа	$(-\infty, upper)$	$x < upper$
leSemiInterval	Закрытый луч справа	$(-\infty, upper]$	$x \leq upper$

ПРИМЕЧАНИЕ: Интервалы, имеющие круглые скобки (или), как у первого в таблице интервала (a,b), или в примерах (- 1,3) и (2,4) называются **open intervals** и конечные точки не включаются в набор значений. Интервалы, имеющие квадратные скобки [или], как у закрытого интервала [a,b] или в примерах [-1,3] и [2,4] называются **closed intervals**, а конечные точки включаются в набор значений. Интервалы, имеющие как квадратные так и круглые скобки [и) или (и], как это указано в

таблице (a,b] и [a,b), или в примерах [-1,3) и (2,4] называются **half-closed intervals** или **half-open intervals**.

ПРИМЕЧАНИЕ: Интервалы, имеющие одну из $\pm\infty$ как конечную точку, называются лучами или полу-линиями.

ПРИМЕР: Интервал "(10,42)" показывает набор всех действительных чисел между значениями 10 и 42, но не включает в себя значения 10 или 42, то есть первое и последнее число интервала соответственно. Интервал "[10,42]" включает все числа в интервале между 10 и 42 и значения 10 и 42 в том числе.

1-4.5.3.5 S100_UnitOfMeasure (Единицы измерения S100)

Единицы измерения являются хорошим компаратором для величин. В S-100 единица измерения включает в себе название и, дополнительно, определение и символ.

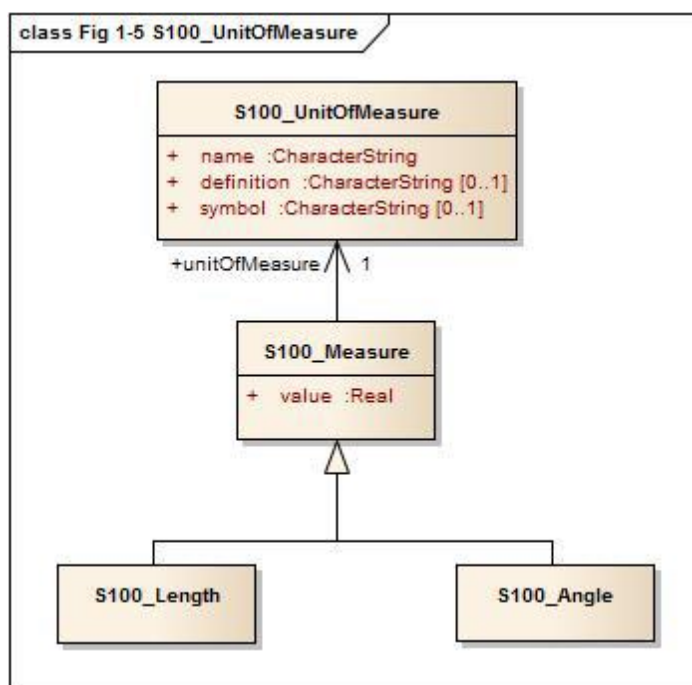


Рисунок 1-5 – S100_UnitOfMeasure

1-4.5.3.6 S100_Measure (Измерение S100)

Измерение – это измеренное значение. Измерение является оценкой величины каких-либо характеристик сущностей реального мира, таких как длина или вес, в

принятых единицах измерения. Измерение состоит из фактического значения (величины) и единицы измерения.

1-4.5.3.7 S100_Length (Длина S100)

Измерение расстояния как интеграла, например, длины кривой или периметра полигона как длины его границы.

1-4.5.3.8 S100_Angle (Угол S100)

Величина поворота, необходимого для совмещения двух линий или плоскостей, измеряется в радианах или градусах.

1-4.5.3.9 S100_IndeterminateDate (Неопределенная дата S100)

Неопределенный момент времени – это момент времени, заданный временной единицей, связанной с конкретной датой, и в определенном временном формате. Допустимыми временными связями являются «до» и «после», определяющими, что событие произошло до или после указанного компонента дата-время.

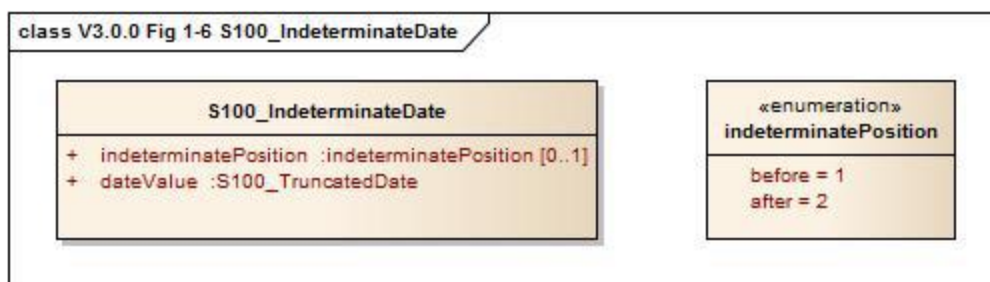


Рисунок 1-6 – S100_IndeterminateDate

Пример (информативный): Донесение мореплавателя, датированное неопределенным моментом времени до 1950 года, указывается атрибутом *reportDate* со значением под-атрибутов как указано в таблице:

Под-атрибут	Значение	Примечание
indeterminatePosition	1 (до)	В неопределенное время до 1 января 1950 года
value	1950----__	

1-4.5.3.10 S100_TM_Instant (мгновенное значение времени)

S100_TM_Instant представляет собой концепцию момента времени по ISO19108. Момент времени — это 0-мерный геометрический примитив, представляющий положение во времени [ISO 19108:2002].

Имя роли	Имя	Описание	Мнж.	Тип данных	Примечания
Класс	S100_TM_Instant	Момент времени. Несколько точек могут быть указаны как усеченные даты, используемые для представления повторяющихся моментов	-		Один из атрибутов <i>date</i> , <i>time</i> или <i>dateTime</i> должен быть заполнен
Атрибут	date	Дата или сокращенная дата (см. таблицу 1-2)	0..1	S100_TruncatedDate	
Атрибут	time	Время (см. таблицу 1-2)	0..1	Time	
Атрибут	dateTime	DateTime (см. таблицу 1-2)	0..1	DateTime	

1-4.5.3.11 S100_TM_Period (период времени)

S100_TM_Period представляет собой концепцию периода времени по ISO19108. Период времени — это одномерный геометрический примитив, представляющий протяженность во времени. [ISO19108:2002].

Имя роли	Имя	Описание	Мнж.	Тип данных	Примечания
Класс	S100_TM_Period	Протяженность во времени	-		Одиночное значение интервала кодируется с $begin = end$, а $closure = closedInterval$ (или пропущен)
Атрибут	closure (закрытый)	S100_IntervalType. По умолчанию <i>closedInterval</i>	0..1	S100_IntervalType	
Атрибут	Begin (начало)	Начало периода	0..1	S100_TM_Instant	
Атрибут	End (конец)	Конец периода	0..1	S100_TM_Instant	

1-4.6 Предопределенные производные типы

Производные типы происходят из базовых типов или других производных типов путем ограничения диапазона допустимых значений. Следующие производные типы определяются в S-100. Спецификации на производство могут определять дополнительные производные типы.

Таблица 1-4 — Предопределенные производные типы

Name	Description	Derived From
URI	Унифицированный ресурсный идентификатор, определенный в RFC 3986. Кодирование знаков URI должно соответствовать правилам, определенным в RFC 3986. ПРИМЕР: http://registry.iho.int	CharacterString
URL	Унифицированный ресурсный локатор (URL) – это URI, который указывает на расположение ресурса путем описания его первичного механизма доступа (RFC 3986). ПРИМЕР: http://registry.iho.int	URI
URN	Постоянный, независимый от расположения идентификатор ресурса, соответствующий синтаксису и семантике, описанной в RFC 2141. ПРИМЕР: urn:s101:1:0:0:AnchorageArea	URI

Атрибуты, содержащие ссылки на файлы поддержки, должны иметь типы атрибутов URI и соответствовать синтаксису RFC 8089 для создания ссылок на файлы.

ПРИМЕР: Минимальное представление локального файла без поля полномочий и абсолютного пути, начинающегося с косого слеша "/".

* "file:/path/to/file"

ПРИМЕЧАНИЕ: В контексте наборов обмена наборы данных могут ссылаться на файлы поддержки со значением атрибута, таким как file:/CABLES01.TXT, который можно интерпретировать как

<ExchangeSetRoot>/SUPPORT_FILES/CABLES01.TXT

И как

/root/installation/folder/some/thing/else/support/files/folder/CABLES01.TXT

в инсталляции конечного пользователя.

1-4.7 Перечисляемые типы

Перечисляемый тип означает, что предлагается список валидных идентификаторов мнемонических слов. Атрибуты перечисляемого типа могут принимать только значения, указанные в списке (перечне).

ПРИМЕР



Рисунок 1-7 — Перечень

Перечни моделируются как классы, которые обозначаются как <<enumeration>>. Перечисляемый класс может содержать только простые атрибуты, которые представляют собой перечисляемые значения. Другая информация в пределах перечисляемого класса остается пустой. Перечень является определяемым пользователем типом данных, чьи примеры формируют список допустимых значений. Обычно объявляются имя перечня и список допустимых значений. Расширение типа перечисления будет подразумевать модификацию схемы.

1-4.8 Типы списков кодов (Codelist)

Типы списков кодов могут использоваться для открытых перечислений, чья принадлежность неизвестна на уровне спецификации на производство, для повторного использования фрагментов информационной модели или для более эффективного управления каталогом. Конкретно они могут использоваться:

- a) для перечней чья принадлежность не полностью известна на уровне прикладной схемы;
- b) для списков, определяемых или контролируемых внешними властями;
- c) для списков общих для нескольких доменов S-100;
- d) если набор допустимых значений требует расширения без значительного пересмотра спецификации данных;
- e) для длинных списков потенциальных значений, которые загромождали бы или увеличивали каталоги фичеров.

Например, ISO 19115 (Metadata) определяет несколько списков кодов, потому что необходимо определить перечисляемые типы, чье существование определено доменом и обстоятельствами (например, дистрибутивный носитель).

Декларация типа списка кодов должна быть одной из следующих 3-х типов:

- 1) **Открытый перечень**, которое является списком допустимых комбинаций ключевых значений (это отображения кодовых значений) с указанием допустимых сообществ пользователей, чтобы показать допустимые значения в заданном формате.
- 2) **Закрытый словарь**, который является словарем комбинаций ключевых значений в заданном формате, идентифицируемом Унифицированным Идентификатором Ресурса (URI) и местонахождение которого может указываться применением стандартных современных технологий для ориентирующих ресурсов. Дополнительные значения не обеспечиваются.
- 3) **Открытый словарь**, который является словарем комбинаций ключевых значений в заданном формате, идентифицируемом Унифицированным Идентификатором Ресурса (URI), как это определено выше, с дополнительным

условием того, что дополнительные значения, соответствующие заданному формату, могут обеспечиваться.

Списки кодов моделируются как стереотипные классы <<S100_Codelist>>. Списки кодов первого типа должны перечислять известные литеры как атрибуты. Во втором и третьем типах атрибуты не перечисляются, а определяется словарь с помощью URI. Классификатор списка кодов должен иметь целевые (указанные) значения, которые определяют его отображение, расширение и предполагаемое кодирование. Рисунок 1-8 показывает 3 примера списка кодов:

1) Список кодов нулей высот (датумов) (**VerticalDatum**) является примером списка кодов, моделируемого как расширяемый открытый перечень (индицируется тегом *codelistType="open enumeration"*), который может дополняться (расширяться) значениями в форме "other: ...", индицируемые тегом *encoding="other: [something]"*.

2) Список кодов производителей ЭНК (**ENCProducerCodes**) является примером списка кодов, моделируемого внешним словарем, который может использовать только значения, содержащиеся в этом словаре (индицируется тегом *codelistType="closed dictionary"*). Словарь определяется тегом *URI=http://www.iho.int/producers/enc/ver1_5*.

3) Список кодов агентств-производителей (**Agency**) является примером списка кодов, моделируемого внешним словарем, который может использовать дополнительные значения (показывается тегом *codelistType="open dictionary"*). Словарь определяется тегом *URI=http://www.iho.int/agency/ver1_5*. Список может быть расширен значениями в форме "other: ...", показываемыми тегом *encoding="other: [something]"*.

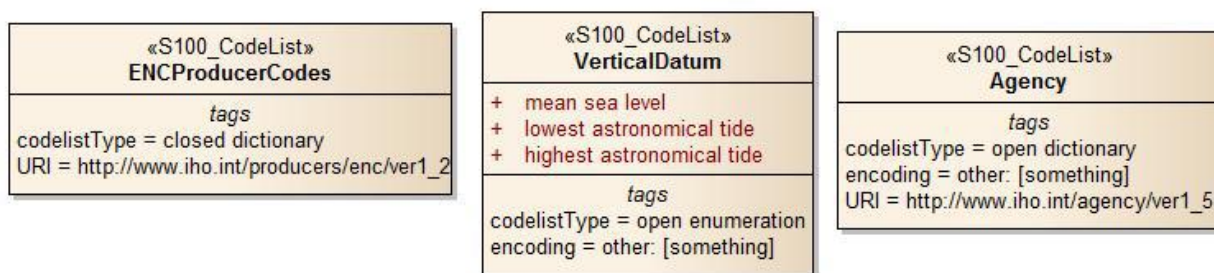


Рисунок 1-8 – Списки кодов

Допускаются реализации (и специальное кодирование), позволяющие избежать подсказок кодирования. Различные реализации могут использовать различные схемы кодирования (и таблицы перевода в другие схемы кодирования). Например, подготовка каталога фичеров в соответствии с ISO 8211 может трансформировать словарь в фрагмент XML, который вливается в (или *Xinclude*) XML каталог фичеров (обычно требуется дополнительная процедура для этого). Это дает возможность

при кодировании XML/GML использовать словарь, позволяющий кодирование по другой схеме.

1-4.9 Взаимосвязи и ассоциации

1-4.9.1 Взаимосвязи






	Ассоциация	Семантическая связь между двумя реализациями
	Генерализация	Связь между элементом и под-элементами, которые могут его заменять
	Зависимость	Использование одного элемента другим
	Агрегация	Взаимосвязь частей
	Композиция	Сильная агрегация, младшие элементы удаляются, если удаляются старшие элементы

Рисунок 1-9 — Различные виды взаимоотношений

Взаимосвязи в UML – это конкретные семантические связи между элементами модели. Виды взаимоотношений включают ассоциацию, генерализацию, агрегацию/композицию, мета отношения, поток и несколько типов зависимого группирования. В ISO 19103 дается ясное различие между общим термином «взаимоотношение» и более специфическим термином «ассоциация». Оба применяются в понятии связи класса с классом, но ассоциация используется для описания ассоциаций реализаций с реализациями. “Генерализация” и “зависимость” являются взаимоотношениями класса с классом. “Агрегация” и другие связи объектов с объектами более строго называются “ассоциациями.” Всегда лучше использовать более ограничительный термин в любом случае, поэтому по возможности чаще используется термин “ассоциация.”

В S-100, генерализация, зависимость и уточнение используются в соответствии со стандартной системой обозначения UML. В дальнейшем будет подробнее описано использование ассоциации, агрегации и композиции.

1-4.9.2 Ассоциация, композиция и агрегация

Ассоциация в UML является семантическим взаимоотношением между двумя или более классификаторами (например, класс, интерфейс, тип ...), которые включают связи между их реализациями.

Ассоциация используется для описания взаимосвязей между двумя или более классами. В дополнение к обычной ассоциации UML определяет два специальных типа ассоциации, называемые агрегацией и композицией. Эти три типа имеют разную семантику. Обычная ассоциация должна использоваться для представления общего взаимоотношения между двумя классами. Агрегативная и композиционная ассоциации должны использоваться для создания взаимоотношения частей и целого (part-whole) между двумя классами.

Бинарная ассоциация имеет имя и два ассоциированных конца. Ассоциированный конец имеет ролевое имя, оператор кратности и дополнительный агрегативный символ. Ассоциированный конец всегда должен быть связан с классом.

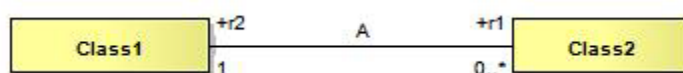


Рисунок 1-10 — Ассоциация

На рисунке 1-10 показана ассоциация, именуемая "А" с ее двумя соответствующими ассоциированными концами. Ролевое имя используется для идентификации конца ассоциации.

Ролевое имя r1 определяет ассоциированный конец, который связан с классом, названным class2.

Кратность ассоциированного конца может быть ровно-один (exactly-one) (1), ноль-или-один (zero-or-one) (0..1), один-или-больше (one-or-more) (1..*), ноль-или-больше (zero-or-more) (0..*) или интервал (interval) (n..m).

С точки зрения класса, ролевое имя противоположного ассоциированного конца определяет роль целевого класса. Мы говорим, что class2 имеет ассоциацию с class1, который определяется ролью r2 и который кратен точно одному. И наоборот, мы можем сказать, что class1 ассоциирован с class2, определяется ролевым именем r1 и имеет кратность ноль-или-один. На показанной выше модели мы говорим, что объекты class1 ссылаются на объекты ноль-или-больше class2 и что объекты class2 ссылаются на ровно один объект class1.

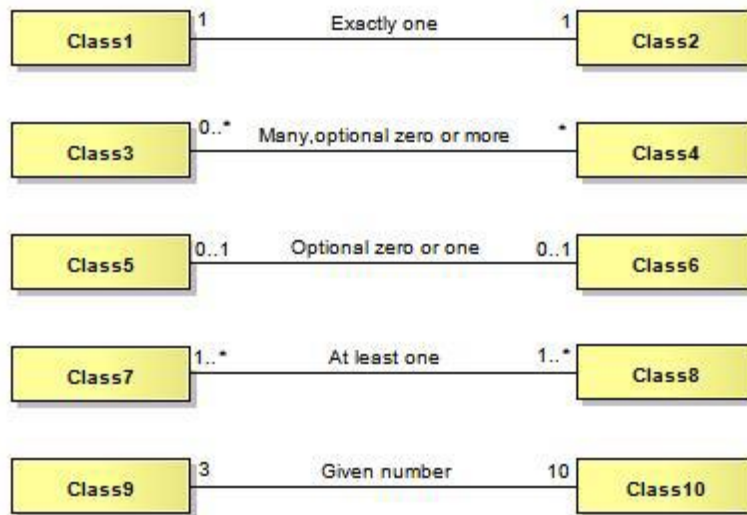


Рисунок 1-11 — Спецификация множественности

Количество реализаций, участвующих на одном конце ассоциации (или атрибута) показаны на рисунке 1-11.

Агрегативная ассоциация определяет взаимоотношение между двумя классами, в котором один из классов играет роль контейнера, а другой играет роль содержимого контейнера. На рисунке 1-12 показан пример агрегации. Агрегативный символ в виде ромба на ассоциативном конце возле объекта class1 показывает, что class1 является агрегацией, состоящей из class3. Это означает, что class3 является частью class1. На рисунке модели показано, что объекты **class1** будут содержать один-или-больше объектов **class3**. Агрегативная ассоциация будет использоваться тогда, когда компоненты контейнера (те, что представляют части объекта – контейнера) могут существовать без объекта – контейнера. Агрегация является символической краткой формой ассоциацией частей, но не имеет явной семантики. Это позволяет включать одни и те же объекты в несколько агрегаций одновременно. Если требуется более строгая агрегативная семантика, должна использоваться композиция, которая описана далее. Также возможно присвоение ролевого имени и кратности на ромбовидном конце.

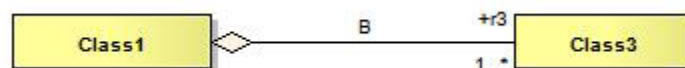


Рисунок 1-12 — Агрегация

Композиционная ассоциация является сильной (строгой) агрегацией. В композиционной ассоциации, если объект-контейнер удаляется, то все его контейнерные объекты тоже удаляются. Композиционная ассоциация должна использоваться тогда, когда объекты, представляющие части объекта-контейнера, не могут существовать без него. (Например, створ). Рисунок 1-13 показывает композиционную ассоциацию, в которой ромбовидный символ композиционной ассоциации имеет сплошную закрашку. Здесь объекты **class1** состоят из один-или-

больше объектов **class4** и объекты **class4** не могут существовать если не существует объект **class1**. Требуемая множественность владящего класса тоже один. Контейнерные объекты или части не могут использоваться несколькими владельцами.

Можно также указывать ролевое имя на ромбовидном конце, но кратность всегда будет не более одного. Композиция должна использоваться для определения семантического эффекта содержимого. Композиция должна использоваться с осторожностью, в частности нужно учитывать различные требования различных перспектив применения прежде, чем использовать ее. Применение композиционной конструкции должно рассматриваться в контексте модели, где контекст означает домен применения, в пределах которого применение должно работать. Это делается для того, чтобы предотвратить проблемы, когда различные приложения имеют разные требования к композиции.

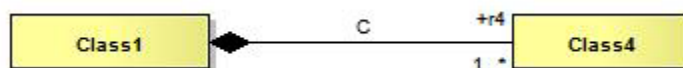


Рисунок 1-13 — Композиция (сильная агрегация)

Все ассоциации должны иметь кардинальность, определенную для обоих концов ассоциации. Как минимум одно ролевое имя должно быть определено. Если только одно ролевое имя определено, другое по умолчанию должно быть **inv_rolename**.

Все ассоциативные концы (роли), представляющие направление взаимосвязи, должны именоваться или иначе сама ассоциация должна иметь имя. Имя ассоциативного конца (ролевое имя) должно быть уникальным в контексте класса и его супертипов. Должно быть указано направление ассоциации. Если направление не указано, это будет означать двухстороннюю ассоциацию (two-way).

Если создается односторонняя ассоциация, направление этой ассоциации может быть указано стрелкой в конце линии. Направление ассоциации должно указываться только если ассоциация имеет имя.

Каждая ассоциация UML имеет навигационные атрибуты, указывающие какой участник ассоциации имеет прямой доступ к обратной роли ассоциации. Логика по умолчанию для немаркированной ассоциации говорит о том, что она двухсторонняя. Ассоциации, у которых не указана навигация, являются двухсторонними, то есть оба участника имеют равный доступ к обратной роли (opposite role). Двухсторонняя навигация не является общепринятой или обязательной во множестве приложений клиент-сервер. Контрпримером этого могут быть сервисы извещений, в которых сервер часто провоцирует связь по подписке. Использование двухсторонних связей, предоставляющих

необоснованный пакет услуг, должен быть минимизирован. Односторонние связи должны использоваться, когда предоставляется все что нужно.

Если ассоциация имеет навигацию в определенном направлении, модель должна обеспечить «ролевое имя» приемлемое для роли целевого объекта относительно исходного объекта. Таким образом, в двухсторонней ассоциации будет создано два ролевых имени. Ролевое имя по умолчанию “the<target class name>”, в котором целевой класс указывается из исходного класса (это имя по умолчанию, используемое во многих инструментах UML). Ассоциативные имена имеют второстепенную важность и фактически служат целям документирования. Тем не менее, иногда они могут использоваться для генерирования ассоциативно управляющих объектов в условиях, поддерживающих ассоциации в качестве концепции гражданина первого класса (first-class citizen concept).

Множественность относится к количеству связей определенного типа, которые объект может иметь. Если ассоциативный конец не имеет навигации, наложение кратных ограничений на него может потребовать реализации трекинга использования ассоциации другими объектами (или сбора данных по множественности через очередь). Если это важно для модели, ассоциация должна быть с двухсторонней навигацией, чтобы более надежно применить ограничение. Иными словами, односторонняя связь подразумевает определенное беззаботное (“don’t care”) отношение к не навигационному концу.

N-ые взаимоотношения для $N > 2$ следует избегать, где это возможно, для того чтобы уменьшить комплексность (сложность). Кратности ассоциаций указываются как спецификации кратности UML. Ассоциация с ролевым именем может рассматриваться как подобная определяющим атрибутам для двух участвующих классов с дополнительным ограничением того, что обновления и удаления постоянно обрабатываются для обоих участников. Для односторонних ассоциаций это будет эквивалентно определению атрибута. Рекомендуется в S-100 использовать ассоциативное обозначение для всех случаев кроме тех, которые включают атрибуты базовых типов данных.

1-4.10 Стереотипы

1-4.10.1 Использование стандартных стереотипов UML для классов/классификаторов

В S-100 используются следующие стереотипы:

- a) <<Interface>> определение набора операций, поддерживаемых объектами имеющими этот интерфейс.
- b) <<Type>> стереотипный класс, используемый для спецификации домена объектов. Тип может иметь атрибуты и ассоциации.
- c) <<Enumeration>> Тип данных, реализации которых формируют список поименованных фиксированных значений. Декларируются как имя перечня, так и его фиксированные значения. Enumeration (перечень) означает короткий список хорошо понятных потенциальных значений в рамках класса. Классическим примером могут служить булевы значения, которые имеют только 2 (или 3) потенциальных значения TRUE, FALSE (и NULL). Большинство перечислений будут кодироваться как последовательный набор целочисленных значений, если не указано другое. Фактическое кодирование обычно осуществляется компиляторами программного языка. В S-100 Списки кодирования взяты из стандарта ISO 19100 и классифицируются как перечни.
- d) <<MetaClass>> Класс чьи реализации являются классами. Мета классы обычно используются в конструкциях мета моделей. Значением мета класса является класс объекта, чье назначение содержать метаданные о другом классе. Например, "FeatureType" и "AttributeType" являются мета классами для "Feature" и "Attribute".
- e) <<DataType>> Дескриптор набора значений, не имеющих идентичности (независимое существование и возможность побочных эффектов). Типы данных включают примитивные предопределенные типы и определяемые пользователем типы. Поэтому DataType является классом с малым количеством или с отсутствием операций, чья первичная цель держать абстрактное состояние другого класса для передачи, хранения, кодирования или постоянного хранения.
- f) <<Codelist>> Тип данных, формирующий список поименованных реализаций, некоторые или все из которых могут быть неизвестны. Имя **Codelist** объявляется в прикладной схеме. Перечисленные в списке данные могут быть описаны либо (i) списком кодов с соответствующим символьным дополнением с шаблоном, позволяющим вносить дополнительные значения, соответствующие определенному формату, или (ii) указателем на ресурс, состоящий из мэппингов списка кодов/символов. Этот ресурс называется словарем. Целевые значения, указанные в декларации **Codelist**, показывают, какая форма используется и местонахождение ресурса (обычно как URI). Списки кодов должны использоваться только тогда, когда перечень или не используем или неэффективен (например, если полный список значений не известен авторам спецификации или список допустимых значений слишком длинный, непостоянный, контролируемый другими и/или используется множеством разных доменов).

1-4.11 Опциональные, условные и обязательные – атрибуты и ассоциации

В UML все атрибуты по умолчанию обязательные. Возможность показать множественность для ролевых имен атрибутов и ассоциаций предоставляет путь для описания дополнительных и условных атрибутов.

По умолчанию значит обязательный, который поэтому не требует уточнения. Там где указана множественность 0..1 или 0..*, означает, что этот атрибут может быть указан или может быть пропущен. Условный атрибут должен быть показан как дополнительный атрибут с ограниченным значением в OCL. Условие должно быть выражено как OCL ограничение в зависимости от объявленного класса. Это означает, что нулевое значение должно быть представлено в указанной модели, например, как элемент, резервирующий место, или как нулевое значение. Дополнительный или условный атрибут никогда не должен иметь значения, определенного как значение по умолчанию.

Атрибут может быть определен как условный, означающий что он является дополнительным в зависимости от значений других атрибутов. Зависимости могут быть от существования-зависимости других (дополнительных) атрибутов или от значения других атрибутов. Условный атрибут показывается как дополнительный с прикрепленным условным выражением. Условие должно быть записано в примечании, напрямую ассоциированном с атрибутом, или с классом и именем атрибута в первой строке. Условный атрибут никогда не должен иметь значения, определенного как значение по умолчанию.

Если не указано, кратность по умолчанию для ассоциаций равна 0..*, а кратность по умолчанию для атрибутов равна 1.

1-4.12 Присвоение имен и поименованные пробелы

Все классы должны иметь уникальные имена. Все классы должны быть определены в рамках пакета. Имена классов должны начинаться с заглавной буквы. Класс не должен иметь имени, основанном на внешнем использовании, поскольку это может ограничить повторное использование. Имя класса не должно содержать пробелов. Отдельные слова в имени класса должны быть связаны друг с другом (конкатенированы). Каждое под-слово (часть слова) в имени должно начинаться с заглавной буквы, например, "XnnnYmmm".

Чтобы обеспечить уникальность имен в глобальном плане, все имена классов должны быть указаны с двухзначным префиксом. Двухзначные префиксы допускают использовать после себя символ пробела _, например, GM_Object.

Модель геометрии использует двухзначные префиксы (GM и TP). Для других областей должны быть указаны другие префиксы.

Имя ассоциации должно быть уникальным в контексте класса и ее супертипов или еще они должны быть производными.

Имена атрибутов должны начинаться с маленькой буквы.

Пример: `firstName`, `lastName`.

Точные технические имена должны использоваться в атрибутах и операциях, чтобы избежать путаницы.

Пример: `alphaCodeIdentifier`, `dateOfLastChange`

Для описания элемента должны широко использоваться поля документирования.

Не повторяйте имен классов внутри имен атрибутов. Старайтесь по возможности использовать короткие имена.

Пример: класс `S-100_WorkingGroup`, атрибут `workingGroupName`.

Правила присвоения имен используются по множеству причин, главным образом для улучшения читаемости, согласованности и защиты от чувствительных к заглавным и прописным буквам программ.

Имена элементов UML должны:

1) Использовать точные и понятные технические имена классов и атрибутов.

Пример: `index`, а не `i`

2) Для ролей атрибутов и ассоциаций заглавными должны быть только первая буква каждого слова после первого слова, составляющего имя. Заглавной должна быть первая буква первого слова каждого имени класса, пакета, спецификации типа и ассоциации.

Пример: `computePartialDerivatives` (не `computepartialderivatives` или `COMPUTEPARTIALDERIVATIVES`)

Пример: `CoordinateTransformation` (не `coordinateTransformation`)

3) Быть короткими насколько это возможно. Использовать стандартные аббревиатуры, если они понятны, и пропускать предлоги и глаголы, если они не влияют значительно на смысл имени.

- numSegment вместо numberOfSegments
- Equals вместо isEqual
- value() вместо getValue()
- initObject вместо initializeObject
- length() вместо computeLength()

Элемент наименования UML **package::package::className** позволяет один и тот же **className** определять в разных пакетах. Однако многие инструменты UML в настоящее время этого не позволяют.

Поэтому была одобрена ограниченная конвенция присвоения имен:

- 1) Хотя модель чувствительна к регистру (case sensitive), все имена классов должны быть уникальны с учетом нечувствительности к регистру.
- 2) Имя класса должно быть уникальным в рамках всей модели (так чтобы не создавать проблем в разных инструментах UML).
- 3) Имена пакетов должны быть уникальны в рамках всей модели (по той же причине).
- 4) Необходимо избегать многократность классов для одной и той же концепции.

1-4.13 Примечания

Окна примечаний используются для комментариев по модели в общих и специфических элементах (классы или ассоциации) модели.



Рисунок 1-14 — Пример примечания

1-4.14 Пакеты

Пакет UML – это контейнер, используемый для декларирования групп под-пакетов, классов и их ассоциаций. Структура пакета в UML позволяет поддерживать иерархическую структуру под-пакетов, деклараций классов и ассоциаций. Пакет должен использоваться для представления схемы.

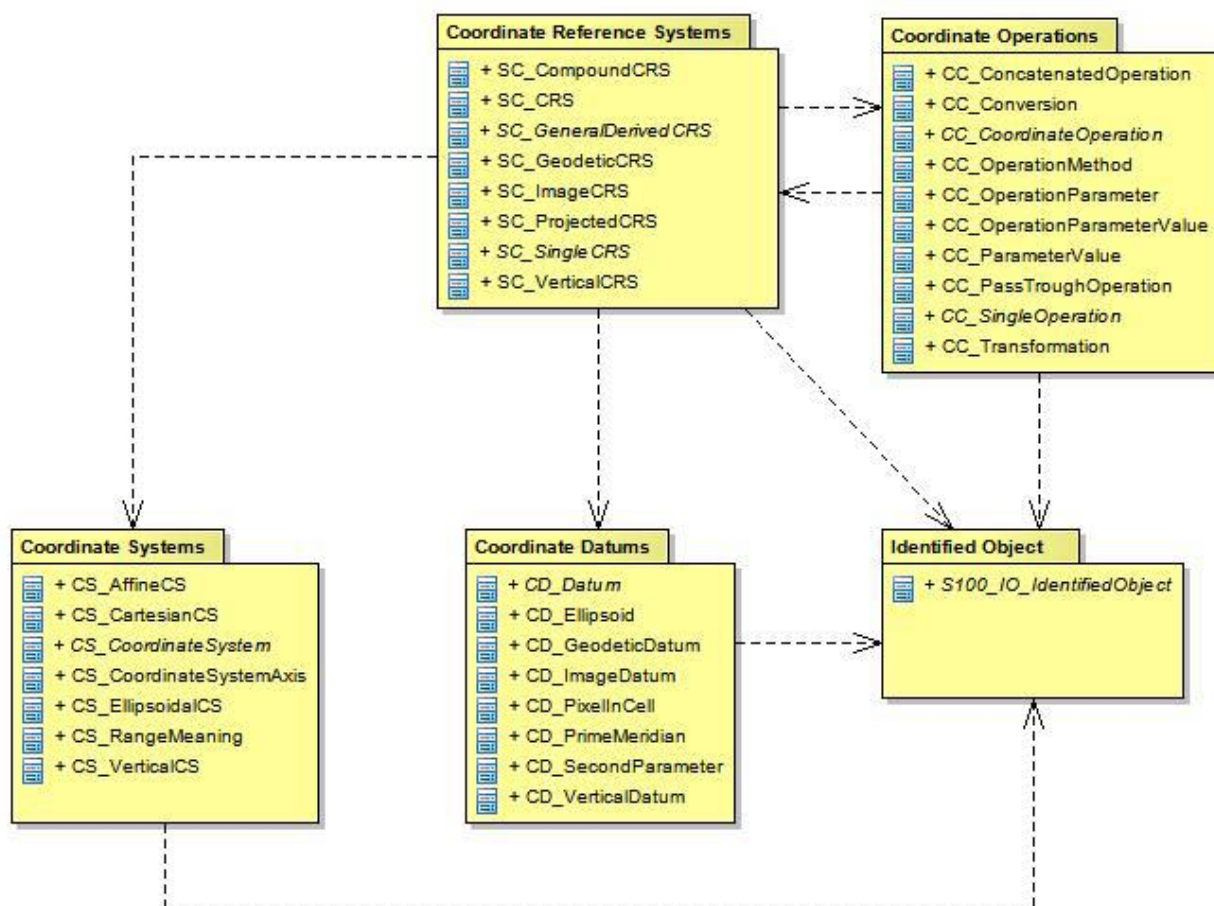


Рисунок 1-15 — Пример структуры пакета

Пакеты, классы и атрибуты в модели схемы могут быть идентифицированы квалифицированным именем. Форма квалифицированного имени следующая: **name1** : :**name2** : :**name3**, где **name1** – это имя крайнего пакета, **name2** – имя, которое появляется внутри пространства имени **name1**, и **name3** – имя, которое появляется внутри пространства имени **name2**. Стандартный символ UML “: :” должен использоваться как разделитель имен. Ограничений по глубине этой иерархии пространства имени нет.

ПРИМЕР: В схеме Spatial имеется подпакет с именем Geometry, определяемый классом с именем GM_Object. Этот класс имеет ассоциацию с ролевым именем SRS (Spatial Reference System). Полное квалифицированное имя этой ассоциации будет: Spatial.Geometry : :GM_Object.SRS.

1-4.15 Документирование моделей в S-100

В дополнение к диаграммам необходимо документировать семантику модели. Значения атрибутов, ассоциаций, операций и связей требуют объяснений. Это

делается посредством контекстных таблиц. Контекстная таблица определяется для каждого класса и имеет следующие колонки:

- Ролевое имя (Role Name)
- Имя (Name)
- Описание (Description)
- Множественность (Multiplicity)
- Тип данных (Data Type)
- Комментарии (Remarks)

Колонка **Ролевое имя** уточняет свойства класса, указанного в этой колонке. Возможны следующие значения:

- Class – Указание на класс
- Attribute – Атрибут класса
- Association – Ассоциация с другим классом
- Enumeration – Перечисляемый тип данных (перечень)
- Literal – Буквенное значение перечисляемого типа данных

Колонка **Имя** содержит имя характеристики. Для ассоциаций это ролевое имя, используемое для данного класса. В колонке Описание дается семантика характеристики.

Колонка **Множественность** содержит число появлений характеристики в классе. Здесь также указывается, какие характеристики являются обязательными, а какие дополнительными.

Колонка **Тип данных** содержит имя типа данных характеристики.

В колонке **Комментарии** дается дополнительная информация. Это могут быть ограничения или условия. Для документирования перечисляемых типов колонки **Множественность** и **Тип данных** не используются.

Пример ниже иллюстрирует использование контекстных таблиц:

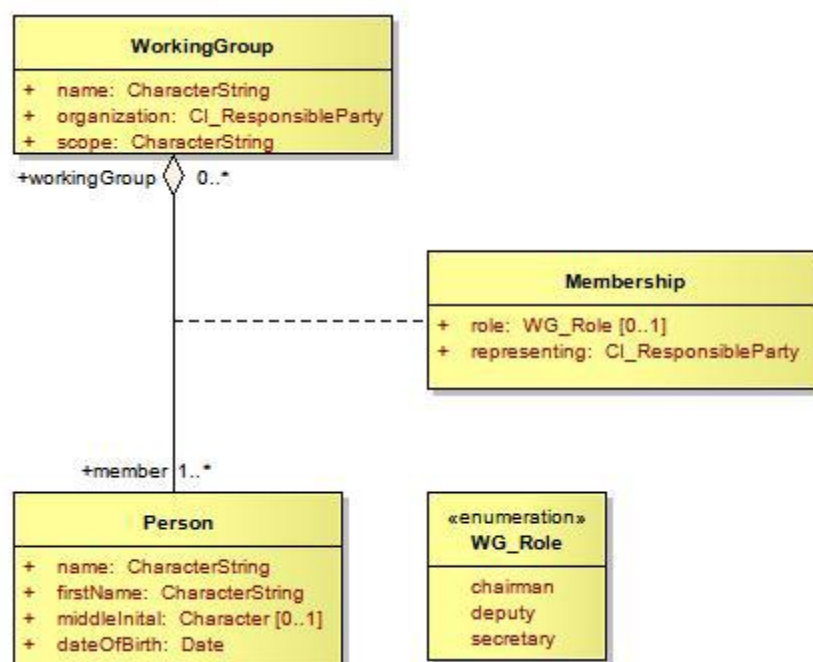


Рисунок 1-16 — Использование контекстных таблиц

Имя роли	Имя	Описание	Множественность	Тип данных	Примечание
Класс	WorkingGroup	Группа экспертов выполняющих полезную работу	-	-	
Атрибут	name	Название рабочей группы	1	CharacterString	
Атрибут	organization	Ответственная за рабочую группу организация	1	CI_ResponsibleParty	
Атрибут	scope	Причина поездок такого количества людей	1	CharacterString	
Ассоциация	member	Личность, назначенная для работы в группе	1..*	Person	

Имя роли	Имя	Описание	Множественность	Тип данных	Примечание
Класс	Person	Наличие человека	-	-	
Атрибут	name	Имя человека	1	CharacterString	
Атрибут	firstName	Первое имя человека	1	CharacterString	
Атрибут	middleInitial	Второе имя человека	0..1	Character	
Атрибут	dateOfBirth	Дата рождения человека	1	Date	
Ассоциация	workingGroup	Рабочая группа, в которой работает человек	0..*	WorkingGroup	

Имя роли	Имя	Описание	Множественность	Тип данных	Примечание
Класс	Membership	Класс, описывающий членство человека в рабочей группе	-	-	
Атрибут	role	Роль человека в рабочей группе	0..1	WG_Role	Простой человек роли не имеет

Атрибут	representing	Ответственная за рабочую группу организация	1	CI_ResponsibleParty	
---------	--------------	---	---	---------------------	--

Имя роли	Имя	Описание	Примечание
Перечень	WG_Role	Роли, которые могут иметь члены рабочей группы	
Буквенный	chairman	Председатель	
Буквенный	deputy	Заместитель председателя	
Буквенный	secretary	Секретарь	