**Диаграмма коммуникации (Communication diagram)**

Диаграмма коммуникации определена в UML 2. Эта же диаграмма в UML 1 называлась диаграммой кооперации. Она выражает ту же семантику, что и диаграмма последовательности – описывает поведение проектируемой системы в форме последовательности обмена сообщениями взаимодействующих экземпляров классификаторов. Различие между диаграммами коммуникации и последовательности состоит в следующем:

1) диаграммы используют различные графические средства;

2) в диаграмме коммуникации внимание читателя сосредотачивается на структуре связей между конкретными экземплярами;

3) в диаграмме последовательности внимание читателя сосредотачивается на временных аспектах взаимодействий между конкретными экземплярами классификаторов.

В диаграмме последовательности используется один тип сущностей – экземпляры взаимодействующих классов (объекты и роли классов) и один тип отношений – связи.

Для обозначения взаимодействующих объектов применяется стандартная нотация – прямоугольник с подчёркнутым именем объекта – экземпляра класса. Синтаксическая форма имени имеет вид: ИмяОбъекта : ИмяКласса

Синтаксис записи имени не является жёстким: можно не указывать имя классификатора даже имя объекта может отсутствовать.

Примеры записи имён объектов в диаграммах UML:

Круг: Фигура – Полное имя

: Фигура – Анонимный объект – роль класса

Круг – Имя объекта с подразумеваемым именем класса

Круг: – Одно имя объекта с неизвестным именем класса

Сокращённые имена объектов применяются в тех случаях, когда какая-то часть имени является несущественной. Например, может быть важным только факт наличия объекта конкретного типа, а имя объекта может быть любым. Возможно рассмотрение множества объектов одного заданного класса, при этом,  естественно, указывать имя известного класса необязательно. Иногда бывает существенным только наличие какого-либо объекта, а класс его совершенно не важен. В этом случае используются объекты без состояний – роли классов.

Взаимное положение элементов на диаграмме коммуникации не имеет значения - важны только связи, вдоль которых передаются сообщения.

Связь между объектами – это:

1) экземпляр ассоциаций между классами этих объектов; таким образом, объекты только тогда могут иметь связь, когда их классы имеют ассоциацию; всегда предполагается, что каждый класс имеет ассоциацию сам с собою, поэтому и каждый объект имеет связь с собою;

2) канал (путь) для передачи сообщений от одного объекта к другому.

Связь может иметь характеристику видимости в форме стереотипа над дальним концом связи. Предусмотрены следующие стандартные стереотипы видимости:

«global» – объект–поставщик сообщения находится в глобальной области определения;

«local» – объект–поставщик сообщения находится в локальной области определения объекта-клиента;

«parameter» – объект–поставщик сообщения является параметром операции объекта-клиента;

«self» – один и тот же объект является и поставщиком сообщения, и клиентом.

Сообщение – это данные и управление, передаваемые от одного объекта (отправителя) к другому (получателю), вызывающие выполнение получателем некоторого действия. Отправка сообщения также квалифицируется как действие, получение сообщения трактуется как событие.

В UML 1 следующие действия, связаны с порождением информации и отправкой сообщений:

1) вызов метода (call);

2) создание объекта (create);

3) уничтожение объекта (destroy);

4) возврат значения (return);

5) посылка сигнала (send).

Действие записывается в виде текста над стрелкой, символизирующей сообщение, или рядом с ней. Если действие имеет параметры (вызов метода, создание объекта, посылка сигнала), то аргументы, соответствующие параметрам по числу и типам, записываются справа от имени действия в круглых скобках.

Синтаксис вызова метода имеют различия в UML 1 и в UML 2. Если действием является вызов метода, возвращающего значения, то в UML 1 слева от имени метода записывается список переменных для возвращаемых значений и знак присваивания :=. Таким образом, та часть нотации сообщений, которая относится к выполняемому действию по вызову метода, в UML 1 имеет следующий синтаксис.

Переменные := ИмяСообщения(Аргументы)

В UML 2 используется несколько иной синтаксис:

Атрибут = ИмяСообщения(Аргументы):Переменные

где Аргументы – это данные, передаваемые получателю сообщения (входные параметры); Переменные – данные, возвращаемые отправителю сообщения (выходные параметры); Атрибут – один из возвращаемых результатов деятельности получателя сообщения.

Сообщение может символизироваться и процедурным обращением к методу, как с аргументами, так и без аргументов:

ИмяСообщения(Аргументы):Переменные

или

ИмяСообщения(Аргументы) или ИмяСообщения

Процедурное обращение без аргументов, в частности, используется как форма передачи сигнала.

В UML имеются так называемые стандартные сообщения. Они представляются в виде стереотипов. Например, сообщение о необходимости создания объекта имеет вид «create».

Получатель сообщения вместе с данными получает и управление – для того, чтобы иметь возможность выполнить действия, инициируемые полученным сообщением.

В UML различается несколько типов передачи управления с помощью сообщения. Чтобы отличить тип передачи сообщения, в UML применяется специальная графическая нотация, а именно, различаются виды стрелок, которыми обозначаются сообщения. На диаграммах коммуникации и последовательности сообщения обозначаются различным образом, но принципы изображения одинаковы.

* — Вложенный поток управления. Данный тип передачи сообщения подразумевает, что отправитель может отправить следующее сообщение только после того, как завершится выполнение всех действий, инициированных данным сообщением. Обычно применяется при вызове методов.
* — Только UML 1. Простой поток управления. Данный тип передачи подразумевает, что управление передается от отправителя сообщения получателю (возможно, безвозвратно). Обычно применяется при моделировании повеления на уровне действующих лиц и вариантов использования.
* В UML 1  в UML 2 — Асинхронный поток управления. Данный тип передачи подразумевает, что сообщение асинхронно передается от отправителя получателю, при этом у отправителя сохраняется свой поток управления, независящий от потока управления получателя. Обычно применяется при отправке сигналов.

 — Возврат управления. Данный тип передачи подразумевает возврат управления после выполнения всех действий, инициированных передачей сообщения с вложенным потоком управления. При этом могут быть указаны возвращаемые значения. Данный тип передачи сообщения можно не отображать на диаграмме, поскольку он подразумевается по умолчанию при вызове методов.

Не определяется — Допускается использование при моделировании других, не определяемых в UML, типов передачи управления, например, передача управления по истечении времени.

Для отображения упорядоченности сообщений во времени применяется иерархическая десятичная нумерация. Последовательно передаваемые сообщения нумеруются последовательно: сообщение с большим номером может быть отправлено только после того как отправлены все сообщения с предыдущими номерами. Одно сообщение может порождать одно или несколько вторичных сообщений. Это наблюдается, когда имеет место коллективное выполнения действия, предполагаемого исходным сообщением. Объект, получивший исходное сообщение, может «запросить помощи» посредством вторичных сообщений у одного другого объекта (формируется одно вторичное сообщение) или у нескольких других объектов (формируется несколько вторичных сообщений). Вторичные сообщения, естественно, могут порождать третичные и так далее. Так образуются сообщения, называемые вложенными. Уровень вложенности не ограничен. При этом действует правило: следующее сообщение конкретного уровня передаётся только после передачи всех вложенных сообщений предыдущего сообщения этого уровня.

Вложенные сообщения нумеруются составным номером. Например, 2.4.1 – это первое сообщение третьего уровня, вложенное в четвёртое сообщение второго уровня, вложенное во второе сообщение первого уровня.

Различие между синхронным и асинхронным типом передачи сообщений состоит в следующем:

При простой синхронной передаче существует один поток управления –управление передаётся от объекта к объекту; действия объектов синхронизируются – выполняются строго последовательно, передатчик дожидается готовности приёмника к получению сообщения, после чего передаёт ему и данные, и управление.

Процедурный синхронный тип передачи сообщения является частным случаем простой синхронной передачи, он отличается те, что предполагает обязательный возврат управления источнику сообщения. Эта особенность обусловливает существование внешнего уровня последовательности сообщений, – аналогичного последовательности операторов главной программы. Номер сообщения этого уровня состоит из одного числа.

При асинхронной передаче действия передатчика и приёмника не согласовываются. Передатчик, не интересуясь состоянием приёмника, сбрасывает сообщение в канал, как в почтовый ящик, сохраняя собственный поток управления. Приёмник, пребывая в собственном потоке управления, по мере готовности извлекает сообщение из канала и начинает на него реагировать. Таким образом, при асинхронной передаче имеют место, по крайней мере, два потока управления.

Пример диаграммы коммуникации. В этой диаграмме представлены не конкретизированные объекты, а роли классов. На внешнем уровне диаграмма содержит единственное сообщение – с номером 1. Это сообщение получает объект класса СерверПечати, который вовлекает в реализацию операции печати либо объект класса Принтер, если тот свободен, либо объект класса Очередь, если первый объект занят. Поскольку сообщения от сервера печати, направлены на реализацию операции, инициализируемой сообщением № 1, то сообщения от него находятся на втором уровне и имеют составной номер из двух чисел, первое из которых 1. Сообщения от сервера печати альтернативны – посылается только одно сообщение в зависимости от условия, выражающего состояние принтера. По этой причине оба сообщения имеют один и тот же номер, за которым в квадратных скобках размещается условие передачи сообщения.

:Компьютер

1.1:[принтер занят] сохранить (файл)

:Очередь

1: печать (файл)

:Принтер

:СерверПечати

1.1:[принтер свободен] печать (файл)

**Состав обеспечивающих подсистем**

Подсистемы, обеспечивающие реализацию функциональных подсистем, называются обеспечивающими. ГОСТ 34.602-89 определяет требования к следующим обеспечивающим подсистемам:

1) математическое обеспечение;

2) информационное обеспечение;

3) лингвистическое обеспечение;

4) программное обеспечение;

5) техническое обеспечение;

6) метрологическое обеспечение;

7) организационное обеспечение;

8) методическое обеспечение.

Указанный список является общим – он предназначен для выборочного использования в зависимости от вида проектируемой автоматизированной системы. В частности, в информационных системах, как правило, не определяются требования к метрологическому и методическому обеспечениям, хотя это и не является правилом. В некоторых случаях список обеспечений ГОСТ может дополняться, например, требованиями к технологическому обеспечению.