Лекция 3 часть3

Элементы графической нотации диаграммы кооперации

Назначение диаграммы кооперации. Объекты, их имена и графическое изображение. Активные и пассивные объекты, их графическое изображение. Мультиобъекты и составные объекты. Графическое изображение связей, посылаемых и принимаемых сообщений между объектами. Формат и синтаксис записи сообщений. Стереотипы сообщений. Рекомендации по построению диаграмм кооперации.

Ключевые слова: диаграмма, кооперация, объект, сообщение, вариант использования, связь, представление, UML, очередь,экземпляр ассоциации, поток, диаграммы классов, операции, полное имя, имя роли, класс, имя класса, запись, пассивный,деятельность, активный объект, active object, мультиобъект, отношение, множества, анонимный, письмо, составной объект,элемент языка, место, кратность, агрегация, композиция, реакция, запрос, список, выражение, потоки управления, терм,синтаксис, значение, истина, итерация, имя события, псевдокод, язык программирования

Диаграмма кооперации предназначена для описания поведения системы на уровне отдельных *объектов*, которые обмениваются между собой *сообщениями*, чтобы достичь нужной цели или реализовать некоторый *вариант использования*. С точки зрения аналитика или архитектора системы в проекте важно представить структурные *связи* отдельных *объектов* между собой. Такое *представление* структуры модели как совокупности взаимодействующих *объектов* и обеспечивает *диаграмма кооперации*.

Кооперация

Кооперация (collaboration) — спецификация множества *объектов* отдельных классов, совместно взаимодействующих с целью реализации отдельных вариантов использования в общем контексте моделируемой системы.

Понятие *кооперации* – одно из фундаментальных в языке *UML*. Цель самой *кооперации* состоит в том, чтобы специфицировать особенности реализации отдельных вариантов использования или наиболее значимых операций в системе. *Кооперация*определяет структуру поведения системы в терминах взаимодействия участников этой *кооперации*.

На диаграмме *кооперации* размещаются *объекты*, представляющие собой экземпляры классов, *связи* между ними, которые в свою *очередь* являются *экземплярами ассоциаций* и *сообщения*. *Связи* дополняются стрелками *сообщений*, при этом показываются только те *объекты*, которые участвуют в реализации моделируемой *кооперации*. Далее, как и на диаграмме классов, показываются структурные отношения между *объектами* в виде различных соединительных линий. *Связи* могут дополняться именами ролей, которые играют *объекты* в данной взаимосвязи. И, наконец, изображаются динамические взаимосвязи — потоки *сообщений* в форме стрелок с указанием направления рядом с соединительными линиями между *объектами*, при этом задаются имена*сообщений* и их порядковые номера в общей последовательности *сообщений*.

Одна и та же совокупность *объектов* может участвовать в реализации различных *кооперацией*. В зависимости от рассматриваемой *кооперации*, могут изменяться как *связи* между отдельными *объектами*, так и *поток сообщений* между ними. Именно это отличает диаграмму *кооперации* от *диаграммы классов*, на которой должны быть указаны все без исключения классы, их атрибуты и *операции*, а также все ассоциации и другие структурные отношения между элементами модели.

Объекты и их графическое изображение

Объект (object) — сущность с хорошо определенными границами и индивидуальностью, которая инкапсулирует состояние и поведение.

В контексте языка *UML* любой *объект* является экземпляром класса, описанного в модели и представленного на диаграмме классов. *Объект* создается на этапе реализации модели или выполнения программы. Он имеет собственное имя и конкретные значения атрибутов. Следует рассмотреть особенности семантики и графической нотации *объектов*, из которых строятся диаграммы.

Для диаграмм *кооперации полное имя объекта* в целом представляет собой строку текста, разделенную двоеточием и записанную в формате:

<собственное имя объекта >'/'<Имя роли класса>:<Имя класса >.

Имя роли класса указывается в том случае, когда соответствующий *класс* отсутствует в модели или разработчику необходимо акцентировать внимание на особенности его использования в рассматриваемом контексте моделирования взаимодействия. *Имя класса* – это имя одного из классов, представленного на диаграмме классов. Важно отметить, что вся *запись* имени *объекта*подчеркивается, что является визуальным признаком *объектов* на различных диаграммах языка *UML*.

Если указано собственное имя *объекта*, то оно должно начинаться со строчной буквы. В то же время имя *объекта*, *имя роли* с символом " / " или *имя класса* могут отсутствовать. Однако двоеточие всегда должно стоять перед именем класса, а косая черта – перед именем роли.

Таким образом, на диаграммах *кооперации* могут встретиться следующие варианты возможных записей полного имени *объекта*:

- **o : C** – *объект* с собственным именем o, экземпляр класса C.
- **: C** – анонимный *объект*, экземпляр класса C.
- **o:**(или просто o) — *объект* -сирота с собственным именем o.
- **o / R : C** — *объект* с собственным именем o, экземпляр класса C, играющий роль R.
- **/ R : C** — анонимный *объект*, экземпляр класса C, играющий роль R.
- **o / R** — *объект* -сирота с собственным именем o, играющий роль R.
- **/ R** — анонимный *объект* и одновременно *объект* -сирота, играющий роль R.

Примеры изображения *объектов* на диаграммах *кооперации* приводятся на рис. 3-3.1.

Рис. 3-3.1. Примеры графических изображений объектов на диаграммах кооперации уровня примеров

Если собственное имя *объекта* отсутствует, то такой *объект* принято называть **анонимным**. Однако в этом случае обязательно ставится двоеточие перед именем соответствующего класса (рис. 3-3.1, в). Отсутствовать может и *имя класса* – такой *объект* называется **сиротой**. Для него записывается только собственное имя *объекта*, двоеточие не ставится, *имя класса* не указываются (рис. 3-3.1, г). Если для *объектов* указываются атрибуты, то в большинстве случаев они принимают конкретные значения (рис. 3-3.1, б). Для отдельных *объектов* (рис. 3-3.1, д, е) могут быть дополнительно указаны роли, которые они играют в *кооперации*.

В контексте языка *UML* все *объекты* делятся на две категории: пассивные и активные. *Пассивный объект* оперирует только данными и не может инициировать *деятельность* по управлению другими *объектами*. Однако пассивные *объекты* могут посылать сигналы в процессе выполнения запросов, которые они обрабатывают. На диаграмме *кооперации* пассивные *объекты* изображаются обычным образом без дополнительных стереотипов.

Активный объект (*active object*) имеет собственный процесс управления и может инициировать деятельность по управлению другими *объектами* .

Активный объект на диаграмме *кооперации* обозначается прямоугольником с утолщенными границами (рис. 3-3.2). Каждый *активный объект* является владельцем определенного процесса управления. В данном фрагменте диаграммы *кооперации активный объект* **а : Клиент** является инициатором открытия счета, который представлен анонимным *объектом* **: Счет**.

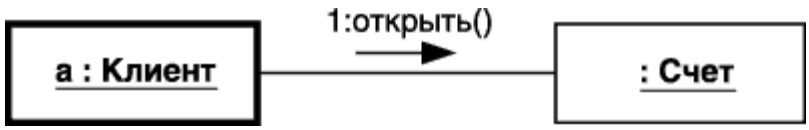

Рис. 3-3.2. Графическое изображение активного объекта (слева) на диаграмме кооперации

Мультиобъект (multiobject) представляет собой множество анонимных *объектов*, которые могут быть образованы на основе одного класса.

На диаграмме *кооперации мультиобъект* используется для того, чтобы показать *операции* и сигналы, которые адресованы всему множеству
анонимных *объектов*. *Мультиобъект* изображается двумя прямоугольниками, один из которых выступает из-за верхней правой вершины другого (рис. 3-3.2, а). При этом стрелка взаимосвязи относится ко всему множеству *объектов*, которые обозначает данный *мультиобъект*. На диаграмме *кооперации* может быть явно указано *отношение* агрегации (композиции) между *мультиобъектом* и отдельным *объектом* из его *множества* (рис. 3-3.3, б).

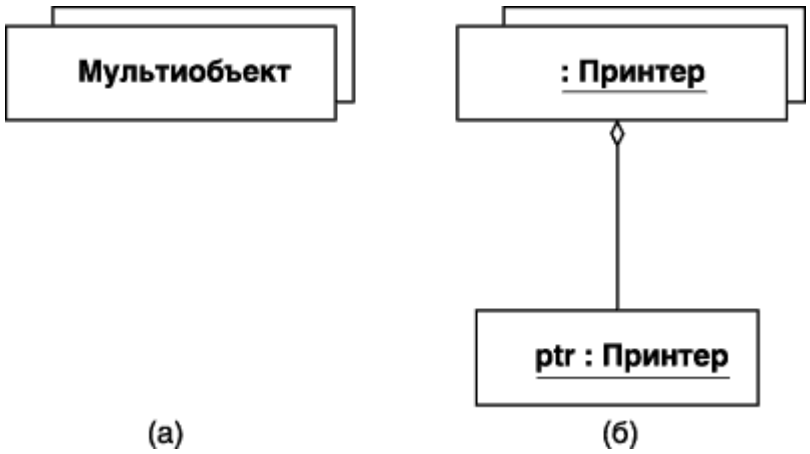

Рис. 3-3.3. Графическое изображение мультиобъектов на диаграмме кооперации

В следующем примере рассматривается ситуация с отправкой почтового *сообщения* клиенту из редактора электронной почты (рис. 3-3.4). *Анонимный активный объект* класса РедакторEmail вначале посылает *сообщение* анонимному *мультиобъекту* класса Клиент.
Это *сообщение* инициирует выбор единственного *объекта* класса Клиент, удовлетворяющего дополнительным условиям. После этого выбранному *объекту* посылается *сообщение* о необходимости отправить электронное *письмо*.

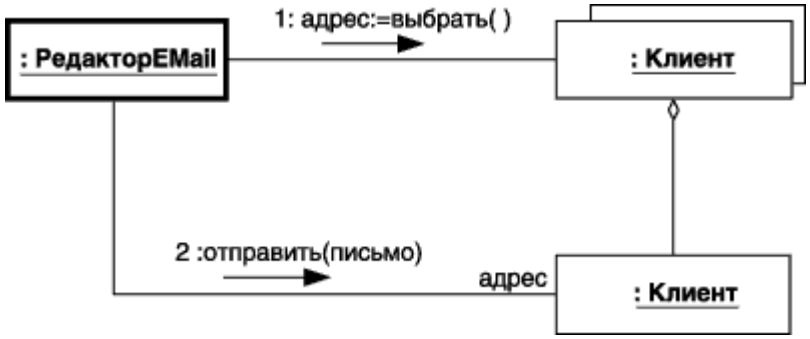

Рис. 3-3.4. Фрагмент диаграммы кооперации для выбора адреса клиента для отправки электронного письма

Составной объект (composite object) или объект-композит предназначен для представления *объекта*, имеющего собственную структуру и внутренние потоки (нити) управления.

Составной объект является экземпляром класса-композита, который связан отношением композиции со своими частями. Аналогичные отношения связывают между собой и соответствующие *объекты*. На диаграммах *кооперации* такой *составной объект* изображается как обычный *объект*, состоящий из двух секций: верхней и нижней. В верхней секции записывается

имя *составного объекта*, а в нижней – его объекты-части вместо списка атрибутов (рис. 3-3.5). При этом допускается иметь в качестве частей другие *составные объекты*.

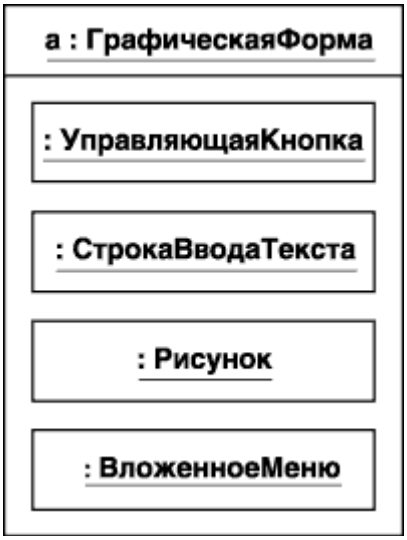

Рис. 3-3.5. Графическое изображение составного объекта на диаграмме кооперации

При изображении диаграммы *кооперации* отношения между *объектами* описываются с помощью *связей*, которые являются экземплярами соответствующих ассоциаций.

Связи на диаграмме кооперации

Связь (link) — любое семантическое отношение между некоторой совокупностью *объектов* .

Связь как *элемент языка UML* является экземпляром или примером произвольной ассоциации и может иметь *место* между двумя и более *объектами*. Бинарная *связь* на диаграмме *кооперации* изображается отрезком сплошной линии, соединяющей два прямоугольника *объектов* (рис. 3-3.4). На концах этой линии дополнительно могут быть явно указаны имена ролей соответствующей ассоциации.

Связи не имеют собственных имен, поскольку идентичны как экземпляры некоторой ассоциации. Другими словами, все *связи* на диаграмме *кооперации* могут быть только анонимными и при необходимости записываются без двоеточия перед именем ассоциации. Однако чаще всего имена *связей* на диаграммах *кооперации* не указываются. Для *связей* не указывается также и*кратность* концевых точек. Однако другие обозначения специальных случаев отношений, такие как *агрегация* и *композиция*, могут присутствовать на отдельных концах *связей*. Например, символ *связи* типа агрегации между *мультиобъектом* класса Клиент и отдельным анонимным *объектом* класса Клиент (рис. 3-3.4).

Примеры *связей* с различными стереотипами изображены на рис. 3-3.6. Здесь представлена обобщенная схема компании с именем с, которая состоит из департаментов (*анонимный мультиобъект* класса Департамент). В последние входят сотрудники (*анонимный мультиобъект* класса Сотрудник). Рефлексивная *связь* указывает на то, что руководитель департамента является одновременно и его сотрудником.

Рис. 3-3.6. Графическое изображение связей с различными стереотипами

Как было отмечено выше, особенности моделирования взаимодействия в контексте языка *UML* заключаются в том, чтобы специфицировать коммуникацию между множеством взаимодействующих *объектов*. Каждое взаимодействие описывается совокупностью *сообщений*, которыми участвующие в нем *объекты* обмениваются между собой.

Сообщения и их графическое изображение

Сообщение (message) — спецификация передачи информации от одного элемента модели к другому с ожиданием выполнения определенных действий со стороны принимающего элемента.

При этом первый *объект* предполагает, что после получения *сообщения* вторым *объектом* последует выполнение некоторого действия. На диаграмме *кооперации сообщение* является причиной или стимулом начала выполнения операций, отправки сигналов, создания и уничтожения отдельных *объектов*. *Связь* обеспечивает канал для направленной передачи *сообщений* между *объектами* от объекта-источника к объекту-получателю.

В этом смысле *сообщение* представляет собой законченный фрагмент информации, который отправляется одним *объектом* другому. При этом прием *сообщения* может инициировать выполнение определенных действий, направленных на решение отдельной задачи тем *объектом*, которому это *сообщение* отправлено. *Сообщения* не только передают информацию, но и требуют или предполагают от принимающего *объекта* выполнения ожидаемых действий. *Сообщения* могут инициировать выполнение операций *объектом* соответствующего класса, а параметры этих операций передаются вместе с *сообщением*.

В таком контексте каждое *сообщение* имеет направление от *объекта*, который инициирует и отправляет *сообщение*, к *объекту*, который его получает. Иногда отправителя *сообщения* называют клиентом, а получателя – сервером. При этом *сообщение* от клиента имеет форму запроса некоторого сервиса, а *реакция* сервера на *запрос* после получения *сообщения* может быть связана с выполнением определенных действий или передачи клиенту необходимой информации тоже в форме *сообщения*.

Сообщения в языке *UML* также специфицируют роли, которые играют *объекты* — отправитель и получатель *сообщения*. *Сообщения* на диаграмме *кооперации* изображаются дополнительными стрелками рядом с соответствующей *связью* или ролью ассоциации. Направление стрелки указывает на получателя *сообщения*. Внешний вид стрелки *сообщения* имеет определенный смысл. На диаграммах *кооперации* может использоваться один из трех типов стрелок для обозначения *сообщений* (рис. 3-3.7).

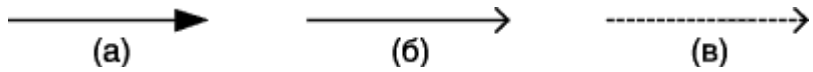

Рис. 3-3.3-3. Графическое изображение различных типов сообщений на диаграмме кооперации

- Сплошная линия с треугольной стрелкой (рис. 3-3.7, а) обозначает вызов процедуры (операции) или передачу потока управления. *Сообщения* этого типа могут быть использованы параллельно *активными объектами*, когда один из них передает *сообщение* этого типа и ожидает, пока не закончится некоторая последовательность действий, выполняемая вторым *объектом*. Обычно все такие *сообщения* синхронны, т.е. инициируются по завершении деятельности или при выполнении определенного условия.
- Сплошная линия с V-образной стрелкой (рис. 3-3.7, б) обозначает асинхронное *сообщение* в простом потоке управления. В этом случае клиент передает асинхронное *сообщение* и продолжает выполнять свою деятельность, не ожидая ответа от сервера.
- Пунктирная линия с V-образной стрелкой (рис. 3-3.7, в) обозначает возврат из вызова процедуры. Стрелки этого типа зачастую отсутствуют на диаграммах *кооперации*, поскольку неявно предполагается их существование после окончания процесса выполнения операции или деятельности.

Каждое *сообщение* может быть помечено строкой текста, которая имеет следующий формат:

<Предшествующие сообщения> <Выражение последовательности> <Возвращаемое значение := имя сообщения> <(Список аргументов)>

Предшествующие *сообщения* — это разделенные запятыми номера *сообщений*, записанные перед наклонной чертой: <Номер *сообщения* ','>* <'/'>. Если *список* номеров *сообщений* пуст, то вся *запись*, включая наклонную черту, опускается. Если номера *сообщений* указываются, то они должны соответствовать номерам других *сообщений* на этой же диаграмме *кооперации*. Смысл указания предшествующих *сообщений* заключается в том, что данное *сообщение* не может быть передано, пока не будут переданы своим адресатам все *сообщения*, номера которых записаны в данном списке.

Выражение последовательности — это разделенный точками *список* отдельных термов последовательностей, после которого записывается двоеточие: <Терм последовательности'.'…> ':'

Каждый из термов представляет отдельный уровень процедурной вложенности в форме законченной итерации. Наиболее верхний уровень соответствует самому левому терму последовательности. Если все *потоки управления* параллельные, то вложенность отсутствует. Каждый *терм* последовательности имеет следующий *синтаксис*: [Целое число | Имя] [Рекуррентность].

- Целое число указывает на порядковый номер *сообщения* в процедурной последовательности верхнего уровня. *Сообщения*, номера которых отличаются на единицу, следуют подряд один за другим.
- Имя в форме буквы алфавита используется для спецификации параллельных потоков (нитей) управления. *Сообщения*, которые отличаются только именем, являются параллельными на

этом уровне вложенности. На одном уровне вложенности все нити управления эквивалентны в смысле приоритета передачи *сообщений*.

- Рекуррентность используется для указания итеративного или условного характера выполнения передачи *сообщений*. Семантика рекуррентности представляет ноль или больше *сообщений*, которые должны быть выполнены в зависимости от записанного условия. Возможны два варианта записи рекуррентности:
- '*'['Предложение-итерация']' для записи итеративного выполнения соответствующего выражения. Итерация представляет последовательность *сообщений* одного уровня вложенности. Предложение-итерация может быть опущено, если количество итераций никак не специфицируется. Наиболее часто предложение-итерация записывается на псевдокоде или языке программирования. В языке UML формат записи этого предложения строгим образом не определен.
- '['Предложение-условие']' для записи ветвления. Эта форма записи специфицирует условие для данного *сообщения*, передача которого по данной ветви возможна только при его истинности.

В общем случае предложение-условие - обычное булевское *выражение* и предназначено для синхронизации отдельных нитей потока управления. Записывается в квадратных скобках и может быть опущено, если оно отсутствует у данного *сообщения*. Наличие этого условия обеспечивает передачу *сообщения* только в том случае, если это условие принимает *значение* "*истина*". Предложение-условие может быть записано на обычном тексте, псевдокоде или некотором языке программирования.

Предложение-условие записывается так же, как и *итерация*, но без звездочки. Это можно понимать как некоторую одношаговую итерацию. В общем случае предполагается, что специфицированная *итерация* выполняется последовательно. Если необходимо отметить возможность параллельного выполнения итерации, то для этой цели в языке *UML* используется символ " *|| ".*Итерация* не распространяется на вложенные уровни данного потока или нити. Каждый уровень должен иметь собственное *представление* для итеративного повторения процедурной последовательности.

Имя *сообщения*, записанное в сигнатуре после возвращаемого значения, означает *имя события*, которое инициируется объектом-получателем *сообщения* после его приема. Наиболее часто таким событием является вызов *операции* у объекта-получателя. Это может быть реализовано различными способами, один из которых – явное указание в качестве имени *сообщения* вызываемой *операции*. Тогда соответствующая операция должна быть определена в том классе, которому принадлежит объект-получатель.

Список аргументов представляет собой разделенные запятыми и заключенные в круглые скобки действительные параметры той *операции*, вызов которой инициируется данным *сообщением*. *Список* аргументов может быть пустым, однако скобки все равно записываются. Для записи аргументов также может быть использован некоторый *псевдокод* или *язык программирования*.

В языке *UML* предусмотрены стандартные действия, выполняемые в ответ на получение соответствующего *сообщения*. Они могут быть явно указаны на диаграмме *кооперации* в форме стереотипа перед именем *сообщения*, к которому они относятся, или выше его. В этом случае они записываются в угловых кавычках.

В языке *UML* определены следующие стереотипы *сообщений*:

- <<call>> (вызвать) – *сообщение*, требующее вызова операции или процедуры объекта-получателя. Если *сообщение* с этим стереотипом рефлексивное, то оно инициирует локальный вызов операции у пославшего это *сообщение объекта*.
- <<return>> (возвратить) – *сообщение*, возвращающее значение выполненной операции или процедуры вызвавшему ее *объекту*. Значение результата может инициировать ветвление потока управления.
- <<create>> (создать) – *сообщение*, требующее создания другого *объекта* для выполнения определенных действий. Созданный *объект* может стать активным (ему передается поток управления), а может остаться пассивным.
- <<destroy>> (уничтожить) – *сообщение* с явным требованием уничтожить соответствующий *объект*. Посылается в том случае, когда необходимо прекратить нежелательные действия со стороны существующего в системе *объекта*, либо когда *объект* больше не нужен и должен освободить задействованные им системные ресурсы.
- <<send>> (послать) – обозначает посылку другому *объекту* сигнала, который асинхронно инициируется одним *объектом* и принимается (перехватывается) другим. Отличие сигнала от *сообщения* заключается в том, что сигнал должен быть явно описан в том классе, *объект* которого инициирует его передачу.

Рекомендации по построению диаграмм кооперации

Построение диаграммы *кооперации* можно начинать сразу после построения *диаграммы классов*. При разработке диаграмм *кооперации* вначале изображаются *объекты* и *связи* между ними. Далее на диаграмму *кооперации* необходимо нанести все *сообщения*, указав их порядок и другие семантические особенности. *Диаграмма кооперации* может содержать только те *объекты* и *связи*, которые уже определены на построенной ранее диаграмме классов. В противном случае, если возникает необходимость включения в диаграмму *кооперации объектов*, которые создаются на основе отсутствующих классов, то *диаграмма* классов должна быть модифицирована посредством включения в нее явного описания этих классов.

Процесс построения диаграммы *кооперации* должен быть согласован с процессами построения *диаграммы классов* и диаграммы последовательности. В первом случае, как уже отмечалось, необходимо следить за использованием только тех *объектов*, для которых определены порождающие их классы. Во втором случае необходимо согласовывать последовательности передаваемых *сообщений*. Речь идет о том, что не допускается различный порядок следования *сообщений* для моделирования одного и того же взаимодействия на диаграмме *кооперации* и диаграмме последовательности.

Необходимо помнить, что *диаграмма кооперации*, с одной стороны, обеспечивает концептуально согласованный переход от статической модели *диаграммы классов* к динамическим моделям поведения, представляемым диаграммами последовательности, состояний и деятельности. С другой стороны, *диаграмма* этого типа предопределяет особенности реализации модели на диаграммах компонентов и развертывания, которые будут рассмотрены в последующих лекциях.