**Лекция 8. Диаграммы прецедентов (диаграмма вариантов использования)**

Если изобразить процесс разработки ПО в виде "*черного ящика*", на выходе которого мы получаем программный продукт, то на *вход* этого "черного ящика" будет подаваться **набор требований к программному продукту** (рис. 1).

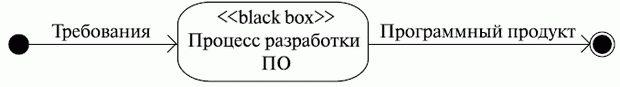


Рис. 1.

Техническое задание - основной документ, без составления которого не начинался в советские времена ни один проект. Документ это был большой, многостраничный, с четкой структурой, определяемой ГОСТами (государственными отраслевыми стандартами). Но время шло, и постепенно техническое задание уступило место набору, состоящему из документов двух видов: диаграммы прецедентов и нефункциональные требования.

**Диаграммы прецедентов** составляют ***модель прецедентов*** (вариантов использования, use-cases). ***Прецедент*** - это функциональность системы, позволяющая пользователю получить некий значимый для него, ощутимый и измеримый результат. Каждый прецедент соответствует отдельному сервису, предоставляемому моделируемой системой в ответ на запрос пользователя, т.е. определяет способ использования этой системы. По этой причине use cases, или прецеденты, часто в русской терминологии фигурируют как ***варианты использования***. Варианты использования чаще всего применяются для спецификации внешних требований к проектируемой системе или для спецификации функционального поведения уже существующей системы. Кроме этого, варианты использования неявно описывают типичные способы взаимодействия пользователя с системой, позволяющие корректно работать с предоставляемыми системой сервисами.

***Нефункциональные требования*** - это описание таких свойств системы, как особенности среды и реализации, производительность, расширяемость, надежность и т. д. Часто нефункциональные требования не привязаны к конкретному варианту использования и потому выносятся в отдельный список дополнительных требований к системе (рис. 2).

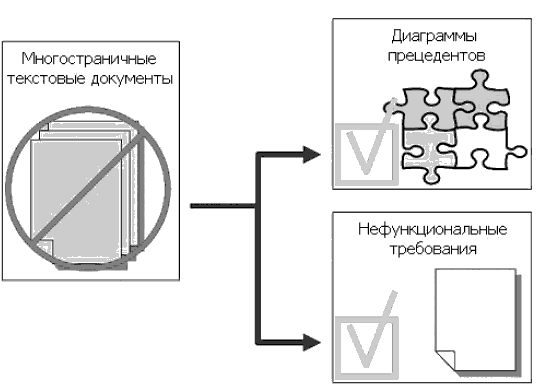


Рис. 2.

Идентифицировать прецеденты и действующие лица - обязанность системного аналитика. И делает он это для того, чтобы:

* четко разграничить систему и ее окружение;
* определить, какие действующие лица и как именно взаимодействуют с системой, какой функционал (варианты использования) ожидается от системы;
* определить и описать в словаре предметной области (глоссарии) общие понятия, которые необходимы для детального описания функционала системы (прецедентов).

Подобный вид деятельности обычно выполняется в такой последовательности:

1. Определение действующих лиц.
2. Определение прецедентов.
3. Составление описания каждого прецедента.
4. Описание модели прецедентов в целом (этот этап включает в себя создание словаря предметной области).

Вначале требования оформляются в виде обычного текстового документа, который создается или самим пользователем, или пользователем и разработчиком вместе. Далее требования оформляют в виде таблицы. В левую колонку помещают прецеденты, а в правую - действующих лиц, участвующих в прецеденте.

Рассмотрим **пример**. Секретарь размещает на сервере меню обеденных блюд на неделю. Сотрудники должны иметь возможность ознакомиться с меню и сделать заказ, выбрав блюда на каждый день следующей недели. Офис-менеджер должен иметь возможность сформировать счет и оплатить его.

Таблица с описанием требований может быть, например, такой:

|  |  |
| --- | --- |
| **Прецедент** | **Действующее лицо** |
| разместить меню | секретарь |
| ознакомиться с меню | сотрудник, секретарь, офис-менеджер |
| сделать заказ | сотрудник, секретарь, офис-менеджер |
| сформировать счет | офис-менеджер |
| оплатить счет | офис-менеджер |

Диаграмма прецедентов, построенная на основе этой таблицы, может быть, например, такой (рис. 3).

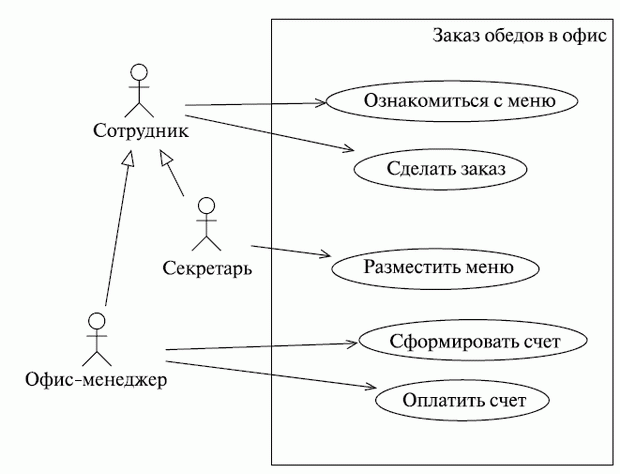
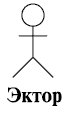


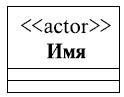
Рис. 3.

**Диаграммы прецедентов и их нотация**

Первое, что бросается в глаза, - большой прямоугольник, внутри которого размещаются **эллипсы**, обозначающие прецеденты. В верхней части прямоугольника указано название моделируемой системы. Этот элемент диаграммы показывает границу между тем, что вы как аналитик показали в виде прецедентов (внутри этих рамок), и тем, что вы изобразили как действующие лица (вне их). Чаще всего таким прямоугольником показывают ***границы самой моделируемой системы***. То есть внутри границы находятся прецеденты - тот функционал, который реализует система (и в этом смысле прецеденты могут рассматриваться как представления подсистем и классов модели), а снаружи - ***действующие лица***: пользователи и другие внешние сущности, взаимодействующие с моделируемой системой.

Кроме рамок системы или ее контекста на диаграмме мы видим еще два вида связанных с ней сущностей - это *действующие лица* (экторы, actors) и *прецеденты*. ***Эктор***- это набор ролей, которые исполняет пользователь в ходе взаимодействия с некоторой сущностью (системой, подсистемой, классом). Эктор может быть человеком, другой системой, подсистемой или классом, которые представляют нечто за пределами рассматриваемой сущности. Экторы "общаются" с системой путем обмена сообщениями. Четко выделив экторов, вы тем самым ясно определяете границу между тем, что внутри системы, и тем, что снаружи, - рамки системы.

На диаграммах UML экторы изображаются в виде стилизованных человечков:  

Несмотря на "человеческий" вид этого обозначения, не следует забывать, что экторы - это не обязательно люди. Эктором может быть внешняя система, подсистема, класс и т. д. Кстати, человечек ("stick-person") - это не единственное обозначение эктора, используемое в UML. На диаграммах прецедентов обычно применяется именно "человекоподобная" форма эктора, но на других диаграммах, и особенно *в случаях, когда эктор имеет атрибуты*, которые важно показать, используется изображение эктора как класса со стереотипом <<actor>>: 

С системой экторы общаются через сообщения, в терминах модели прецедентов - через прецеденты. Один и тот же эктор может быть связан с несколькими прецедентами, и наоборот, один прецедент может быть связан с несколькими разными экторами. Ассоциации между эктором и прецедентом всегда бинарные - т. е. представляют отношения типа "один к одному", использование кратности недопустимо. Это не противоречит сказанному выше: действительно, один эктор может быть связан с несколькими прецедентами, но только с помощью отдельных ассоциаций - по одной на каждый прецедент. Мы видели это в нашем примере. Кстати, там мы видели ассоциации, изображенные не просто в виде линий, а стрелками.

Думаем, смысл этого обозначения вполне понятен: это *направленная ассоциация* и стрелка (как и на других диаграммах) всегда направлена в сторону той сущности, от которой что-то требуют, чьим сервисом пользуются и т. д.

И еще - экторы не могут быть связаны друг с другом. Единственное допустимое отношение между экторами - ***генерализация* (*наследование*)**. Опять-таки, в нашем примере с заказом обедов в офис, вы могли увидеть именно такой вид отношений между экторами. Это не значит, что в реальной жизни офис-менеджер и секретарь (да и вообще любые два сотрудника) не могут общаться: просто при создании модели прецедентов такое общение не попадает в область наших интересов, считается несущественным.

***Прецедент***- это описание набора последовательных событий (включая возможные варианты), выполняемых системой, которые приводят к наблюдаемому эктором результату. Прецеденты описывают сервисы, предоставляемые системой экторам, с которыми она взаимодействует. Причем прецедент никогда не объясняет, "как" работает сервис, а только описывает, "что" делается.

Изображаются прецеденты в виде эллипса, внутрь контура которого помещается имя (описание) прецедента. Имя прецедента описывает взаимодействие эктора с системой, говорит о том, какими сообщениями они обмениваются между собой. В примере с заказом обедов мы видели несколько прецедентов и могли видеть, что имя прецедента - это, скорее, название сценария, воспроизводящегося в ходе взаимодействия эктора с системой. Причем это всегда описание *с точки зрения эктора*, описание услуг, предоставляемых системой пользователю.

Приведем пример простейшей диаграммы, иллюстрирующей сказанное нами об обозначениях прецедента (рис. 4).

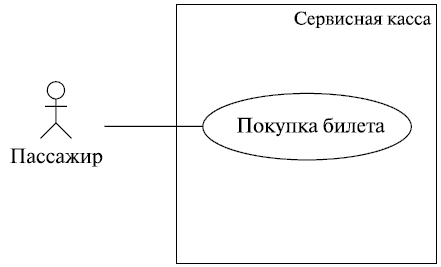


Рис. 4.

В этом примере пассажир может купить в сервисной кассе билет на некоторый вид транспорта. Покупка билета - это название сценария, по которому эктор (пассажир) может взаимодействовать с системой (кассой). Заметьте, это *не описание* сценария, а именно название - оно говорит нам, *что* делает эктор в процессе взаимодействия, но не говорит, как именно.

Прецеденты определяют *непересекающиеся* сценарии поведения. Выполнение одного прецедента не может быть прервано в результате работы другого прецедента. Другими словами, выполнение одного прецедента не может быть прервано в результате событий или действий, вызванных выполнением другого прецедента.

**Сценарий** - это конкретная последовательность действий, иллюстрирующая поведение. Сценарий - это повествовательный рассказ о совершаемых эктором действиях, история, эпизод, происходящий в данных временных рамках и данном контексте взаимодействия. Сценарии (в различных формах представления) широко применяются в процессе разработки программного обеспечения. При разработке пользовательского интерфейса сценарии описывают взаимодействие между пользователем (или категорией пользователей, например, администраторами системы, конечными пользователями) и системой.

Такой сценарий состоит из последовательного описания комбинаций отдельных действий и задач (например, нажатий клавиш, щелчков по элементам управления, ввода данных в соответствующие поля и т. д.). Сценарии также иногда можно увидеть на диаграмме прецедентов. Иногда их изображают в виде "*листа бумаги*", на котором написано имя файла, - прямоугольника с загнутым нижним левым уголком. В этом случае указанный файл содержит в себе описание данного сценария. А иногда сценарий записывается в комментарий. Комментарии (ноутсы, notes) изображаются прямоугольниками с загнутым верхним правым углом и соединяются с элементом, который они поясняют, пунктирной линией ([рис.](https://www.intuit.ru/studies/courses/1007/229/lecture/5962?page=2#image.6.7) 5).



Рис. 5.

Вот **пример** простого (неформализованного) текстового описания сценария.

*Пользователь вводит логин, пароль, адрес электронной почты и код подтверждения и нажимает кнопку "Далее". Система запрашивает ввод проверочного кода. Пользователь вводит код и нажимает кнопку "Далее". Система проверяет соответствие кода изображенному на картинке*.

А вот тот же сценарий в табличном представлении:

|  |  |
| --- | --- |
| **Действия пользователя** | **Реакция системы** |
| Ввод логина, пароля, адреса электронной почты и нажатие кнопки "Далее" | Запрос ввода проверочного кода |
| Ввод проверочного кода и нажатие кнопки "Далее" | Проверка кода на соответствие изображенному на картинке |

*Как связаны понятия сценария и прецедента?* Прецеденты рождаются из требований к системе. Но говорят они о том, что делает система. Как система это делает, говорят сценарии. Таким образом, прецедент можно специфицировать путем описания потока действий или событий в текстовой форме - в виде, понятном для не занятого в непосредственной разработке системы читателя.

Таким образом, *сценарии специфицируют прецеденты*. Поскольку сценарии - это, по сути, рассказы, они являются весьма эффективным средством извлечения информации из бесед с заказчиком и предоставляют превосходное, понятное непрофессионалу описание создаваемого приложения.

Сценарии, да и вообще диаграммы прецедентов (дополненные сценариями) являются отличным *средством общения между разработчиками и заказчиком*, причем, в силу простоты нотации, - средством, понятным обеим сторонам. Взаимосвязь между требованиями, прецедентами и сценариями можно изобразить такой "псевдодиаграммой" ([рис.](https://www.intuit.ru/studies/courses/1007/229/lecture/5962?page=2#image.6.8) 6).

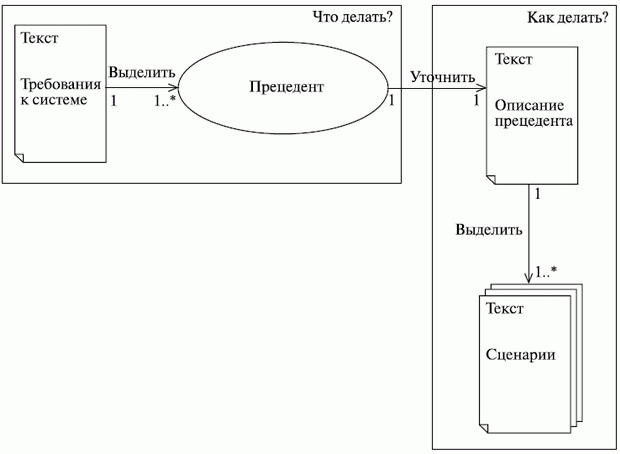


Рис. 6.

Точно так же, как мы обычно поступаем с классами, после того как мы выделили и описали каждый прецедент, мы должны просмотреть их все на предмет наличия одинаковых действий - поискать, а не выполняются ли (используются) некоторые действия совместно несколькими вариантами использования. Этот совместно используемый фрагмент лучше описать в отдельном прецеденте. Таким образом, мы уменьшим избыточность модели за счет применения обобщения прецедентов.

Экземпляры обобщенных прецедентов (потомков) сохраняют поведение, присущее обобщающему прецеденту (предку). Другими словами, наличие (использование) в варианте использования X обобщенного варианта использования Y говорит нам о том, что экземпляр прецедента X *включает в себя* поведение прецедента Y.

Обобщения применяются, чтобы упростить понимание модели вариантов использования за счет многократного задействования "заготовок" для создания прецедентов, необходимых заказчику. Такие "полные" прецеденты называются *конкретными прецедентами*. "Заготовки" прецедентов, созданные лишь для многократного использования в других прецедентах, называют ***абстрактными прецедентами***. Абстрактный прецедент (как и абстрактный класс) не существует сам по себе, но экземпляр конкретного прецедента демонстрирует поведение, описываемое абстрактными прецедентами, которые он (повторно) использует. Прецедент, который экторы наблюдают при взаимодействии с системой ("полный" прецедент, как мы называли его ранее), часто называют еще " *реальным* " прецедентом.

Изображается обобщение, линией с "незакрашенной" треугольной стрелкой на конце. **Обобщение** - это отношение между предком и потомком, и **стрелка всегда указывает на предка** (рис. 7).

Экторы-потомки наследуют от предка базовые характеристики и дополняют их своей спецификой. Точно так же прецедент-потомок наследует поведение и семантику прецедента-родителя и дополняет его поведение.

Следующий вид отношений между прецедентами - ***включение***. Отношение включения означает, что ***в некоторой точке базового прецедента содержится поведение другого прецедента***. Включаемый прецедент не существует сам по себе, а является всего лишь частью объемлющего прецедента. Таким образом, базовый прецедент как бы заимствует поведение включаемых, раскладываясь на более простые прецеденты. Например, когда мы покупаем в магазине некоторую вещь, в момент считывания кассиром штрих-кода обновляется состояние базы данных товаров, имеющихся в наличии, - количество наличных единиц купленного товара уменьшается. То же самое действие выполняется и в том случае, если купленный товар оказался бракованным, непригодным к использованию или попросту нам не понравился: состояние упомянутой базы данных вновь обновляется - но теперь уже в сторону увеличения количества наличных единиц определенного товара. Т. е. оба этих действия - и покупка, и возврат - содержат (включают в себя) такое действие, как обновление содержимого БД.

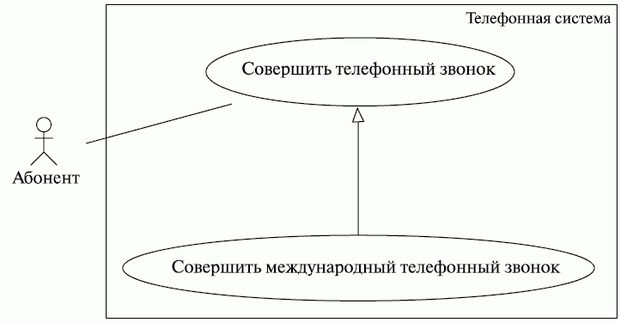


Рис. 7.

Изображается включение как зависимость (пунктирная линия со стрелкой) со стереотипом **<<include>>**. При этом стрелка направлена, естественно, **в сторону включаемого прецедента** (стрелка всегда направлена в сторону того элемента, от которого что-то требуется, чьими сервисами пользуются)(рис. 8).

Как хорошо видно из этого примера, использование включения позволяет избежать многократного описания одного и того же набора действий - общее поведение можно просто описать в виде прецедента, включаемого в базовые.

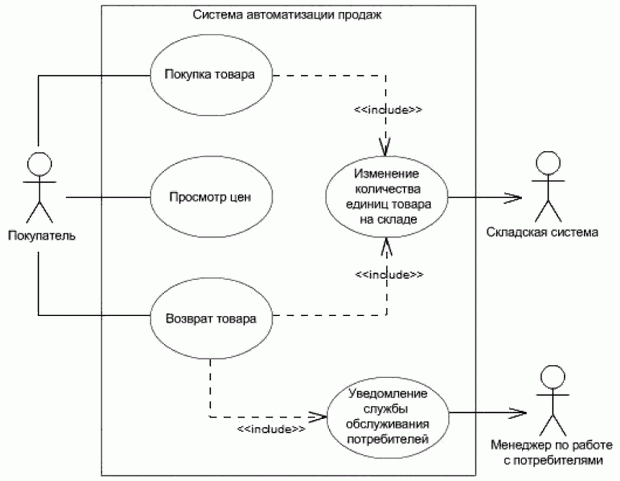


Рис. 8.

**Отношение расширения.** Чтобы уяснить себе смысл расширения, представим себе, что мы говорим об оплате некоторого купленного нами товара. Мы можем оплатить товар наличными, если сумма не превышает $ 100. Или оплатить кредитной картой, если сумма находится в пределах от $ 100 до $ 1000. Если же сумма превышает $ 1000, нам придется брать кредит. Таким образом, мы расширили понимание операции оплаты купленного товара и на случаи, когда используются другие средства оплаты, нежели наличные. Но сами эти случаи возникают только при строго определенных условиях: когда цена товара попадает в определенные рамки.

Расширение дополняет прецедент другими прецедентами, "срабатывающими" при некоторых условиях, - просто добавляет в исходный прецедент последовательность действий, содержащуюся в другом прецеденте. Отношение расширения прецедента А к прецеденту В означает, что экземпляр прецедента В может включать в себя (при определенных условиях, которые могут быть описаны в расширении) поведение, описанное в прецеденте А. Пример показан на следующей диаграмме (рис. 9):

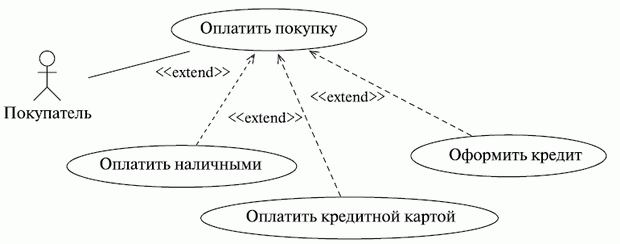


Рис. 9.

Однако в приведенном примере не видно, при каких именно условиях человек использует каждый конкретный способ оплаты. В то же время, при моделировании с использованием расширения можно указать как условия осуществления расширенного поведения, так и место - ***точку расширения*** прецедента, в которой подключаются действия из расширяющих прецедентов. Вспомните оператор условного перехода - когда исходный прецедент (а именно, последовательность действий, содержащаяся в нем) приходит в точку расширения, происходит оценка условий расширения. Если условия выполняются, прецедент включает в себя последовательность действий из расширяющего прецедента.

Точка расширения описывается в дополнительном разделе прецедента, отделенном от его названия горизонтальной линией - точно так же, как в отдельных разделах перечисляются атрибуты класса и его операции. Ниже показан пример описания точки расширения (рис. 10).

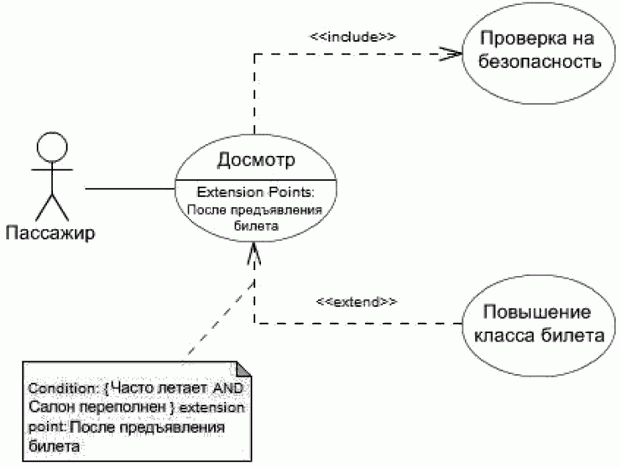


Рис. 10.

В этом примере регистрация пассажиров авиарейса включает в себя контроль службы безопасности, а при условии (указанном в примечании после служебного слова "Condition:"), что человек часто летает и салон переполнен (обратите внимание на оператор AND, говорящий об одновременности выполнения условий), класс билета может быть повышен, например, с "эконом" до "бизнес-класса". Причем такой апгрейд может произойти только после того, как билет предъявлен на стойку регистрации - это и есть точка расширения. Она описана (ее имя указано) в дополнительном разделе прецедента после служебной фразы "Extension points:". Прецедент может иметь сколь угодно много точек расширения.

Сопоставить конкретный расширяющий прецедент с определенной точкой расширения можно, прочитав условия расширения, указанные в комментариях, - само условие записывается после служебного слова "Condition:" в фигурных скобках, за которыми идет служебная фраза "Extension point:", и после нее указывается имя точки расширения. Посмотрите еще раз на наш пример с регистрацией пассажиров в аэропорту и убедитесь сами, что все это очень просто!

Некоторое недоумение может вызвать то, что стрелка направлена всегда в сторону расширяемого прецедента. Но и это легко объяснить с точки зрения нашего тезиса, что "стрелка всегда указывает на того, от которого что-то требуют": ведь для того, чтобы прецедент был расширен, нужно, чтобы он попал в точку расширения и проверилась истинность условий - только тогда действия, содержащиеся в расширяющем прецеденте, смогут быть добавлены в последовательность действий исходного прецедента. Так что все правильно - от расширяемого прецедента требуется точка расширения и проверка условий, потому и стрелка направлена к нему.

Подытоживая все вышесказанное, можно сказать, что ***расширение позволяет моделировать необязательное поведение системы*** (был бы класс билета повышен, если бы пассажир не налетал нужного количества миль, а салон был бы почти пуст?). Сам факт расширения зависит от выполнения условий - расширения ведь может и не произойти! Это просто отдельные последовательности действий, выполняемые лишь при определенных обстоятельствах и включаемые в определенных точках сценария (обычно в результате явного взаимодействия с эктором).

Организация прецедентов с помощью выделения общего поведения (включение) и различных вариантов поведения (расширение) - важная составляющая часть процесса разработки простого, сбалансированного и понятного набора прецедентов. Можно сказать даже, что использование включения и расширения - признак хорошего стиля в моделировании прецедентов.

**Моделирование при помощи диаграмм прецедентов**

Модель прецедентов, по сути, является концептуальной моделью системы. В ней в общих чертах описывается только поведение (функциональность) системы, а о деталях реализации речь не идет - на данном этапе реализация не важна, гораздо важнее собрать требования к системе и оформить их в наглядном виде, понятном и разработчикам, и заказчику.

Подводя итоги, мы можем сформулировать три причины и три способа использования прецедентов в ходе работы над системой.

* *Прецеденты дают возможность аналитикам, пользователям и разработчикам говорить на одном языке*: используя прецеденты, аналитики (эксперты в предметной области) могут на основе пожеланий заказчика описать поведение системы с точки зрения пользователя с такой степенью детализации, что разработчики смогут без труда сконструировать "внутренности" системы. В то же время, нотация диаграмм прецедентов настолько проста, что даже неподготовленный пользователь (заказчик) способен понять их смысл и помочь в их уточнении - ведь картинки (а тем более комиксы, каковыми, по сути, являются диаграммы UML) воспринимаются намного легче, чем текст!
* *Прецеденты позволяют разработчикам понять назначение элемента*: система, подсистема или даже класс могут быть сложными образованиями, состоящими из большого числа составных частей и имеющими большое число атрибутов и операций. Моделирование прецедентов позволяет лучше представить себе поведение системы, понять, какие элементы модели играют какие роли в реализации этого поведения, в какие кооперации входят, и какой именно прецедент (функционал системы) реализуют.
* *Прецеденты являются основой для тестирования элемента в течение всей разработки*: модель прецедентов описывает желаемое поведение системы (ее функционал) с точки зрения пользователя. Так что, постоянно сопоставляя предоставляемый элементом (фактический) функционал с имеющимися прецедентами, можно надежно контролировать корректность реализации элемента. Вот вам и надежный источник регрессионных тестов. Кроме этого, появление нового прецедента зачастую заставляет пересмотреть реализацию элемента, дабы убедиться, что она обладает достаточной гибкостью, изменяемостью и масштабируемостью.

**Пример диаграммы прецедентов**

Пример демонстрирует включение, расширение и наследование прецедентов. Обратите внимание на стрелки, которые направлены к экторам, изображающим шлюзы. Все правильно - ведь система пользуется их услугами при отправке сообщений, в то время как маркетолог, наоборот, пользуется услугами системы, и потому стрелки направлены от него ([рис. 1](https://www.intuit.ru/studies/courses/1007/229/lecture/5962?page=4#image.6.15)1).

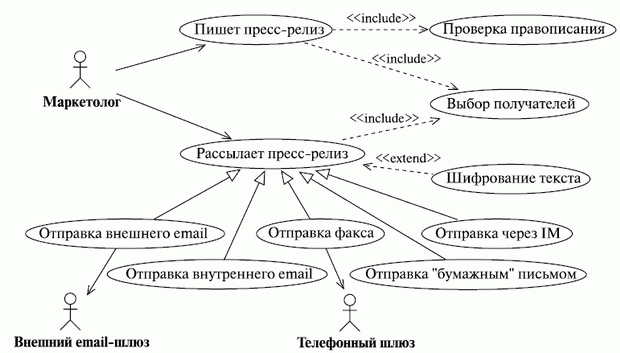


Рис. 11.