Диаграммы прецедентов

Диаграммы прецедентов представляют собой один из пяти типов диаграмм, применяемых в UML для моделирования динамических аспектов системы (остальные четыре типа - это диаграммы деятельности, состояний, последовательностей и кооперации). Диаграммы прецедентов играют основную роль в моделировании поведения системы, подсистемы или класса. Каждая такая диаграмма показывает множество прецедентов, актеров и отношения между ними.

Диаграммы прецедентов применяются для моделировании вида системы с точки зрения прецедентов (или вариантов использования). Чаще всего это предполагает моделирование контекста системы, подсистемы или класса либо моделирование требований, предъявляемых к поведению указанных элементов.

Диаграммы прецедентов имеют большое значение для визуализации, специфицирования и документирования поведения элемента. Они облегчают понимание систем, подсистем или классов, представляя взгляд извне на то, как данные элементы могут быть использованы в соответствующем контексте. Кроме того, такие диаграммы важны для тестирования исполняемых систем в процессе прямого проектирования и для понимания их внутреннего устройства при обратном проектировании.

Введение

Предположим, что вы получили по почте загадочное устройство, с одной стороны которого находятся несколько кнопок и маленький жидкокристаллический дисплей. Помимо этого в нем нет ничего примечательного, и у вас нет даже намека на то, как его использовать. Можно, конечно, использовать метод проб и ошибок: произвольно нажимать на кнопки и смотреть, что получится, но таким путем на изучение устройства вам придется потратить огромное количество времени.

Программные системы часто выглядят столь же непривычно. Вам как пользователю могут передать некоторое приложение и посоветовать поработать с ним. Если это приложение следует принятым соглашениям используемой вами операционной системы, то, быть может, после нескольких попыток вы добьетесь каких-то полезных результатов, но понять таким образом более сложные и тонкие аспекты поведения программы вам никогда не удастся. Если вы разработчик, то вам могут передать на сопровождение унаследованное приложение или набор компонентов и сказать, что их надо применить в работе. Вряд ли вы сумеете постичь, как нужно с ними обращаться, пока не сформируете концептуальную модель их работы.

В языке UML диаграммы прецедентов как раз и позволяют визуализировать поведение системы, подсистемы или класса, чтобы пользователи могли понять как их использовать, а разработчики - реализовать соответствующий элемент. На рис. 17.1 приводится диаграмма, описывающая использование устройства, которое упоминалось в начале главы, - обычно его называют сотовым телефоном.

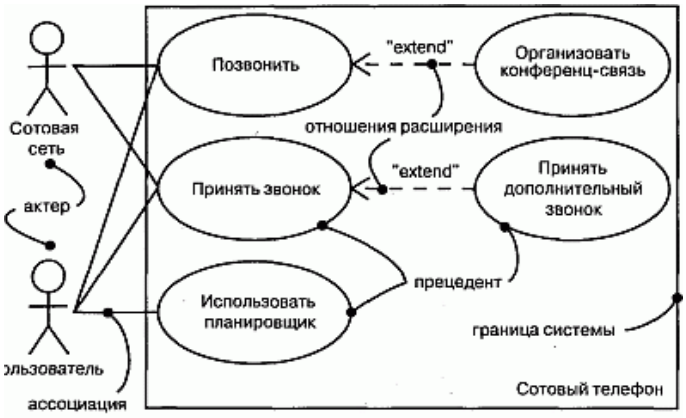

Рис. 17.1 Диаграмма прецедентов

Термины и понятия

Диаграммой прецедентов, или использования (Use case diagram), называется диаграмма, на которой показана совокупность прецедентов и актеров, а также отношения между ними.

Общие свойства

Диаграмма прецедентов обладает стандартными свойствами, присущими любой диаграмме, - именем и графическим содержанием, которое представляет собой одну из проекций модели. Диаграмма прецедентов отличается от прочих своим конкретным содержанием.

Содержание

Диаграммы прецедентов обычно включают в себя:

- прецеденты;
- актеры;
- отношения зависимости, обобщения и ассоциации.

Как и все остальные диаграммы, они могут содержать примечания и ограничения.

Иногда в диаграммы прецедентов помещают пакеты, применяемые для группирования элементов модели в более крупные блоки, а в ряде случаев и экземпляры прецедентов, особенно если надо визуализировать конкретную исполняемую систему.

Типичные примеры применения

Диаграммы прецедентов, или использования, применяют для моделирования статического вида системы с точки зрения прецедентов. Этот вид охватывает главным образом поведение системы, то есть видимые извне сервисы, предоставляемые системой в контексте ее окружения.

При моделировании статического вида системы с точки зрения прецедентов диаграммы использования обычно применяются двумя способами:

- для *моделирования контекста системы*. Моделирование контекста подразумевает, что мы обводим систему воображаемой линией и выявляем актеры, которые находятся за этой линией и взаимодействуют с системой. Диаграммы прецедентов нужны на этом этапе для идентификации актеров и семантики их ролей;
- для *моделирования требований* к системе. Моделирование требований к системе предполагает указание на то, что система должна делать (с точки зрения внешнего наблюдателя), независимо от того, как она должна это делать. Диаграммы прецедентов нужны здесь для специфицирования желаемого поведения системы. Они позволяют рассматривать всю систему как "черный ящик": вы видите все, что находится вне нее, наблюдаете за ее реакцией на события, но ничего не знаете о ее внутреннем устройстве.

Типичные приемы моделирования

Контекст системы

Любая система содержит внутри себя какие-либо сущности, в то время как другие сущности остаются за ее пределами. Например, в системе проверки кредитных карточек имеются счета, транзакции и механизмы проверки подлинности. В то же время обладатели кредитных карточек и торговые предприятия находятся вне системы. Сущности внутри системы отвечают за реализацию поведения, которого ожидают сущности, находящиеся снаружи. Сущности, находящиеся вне системы и взаимодействующие с ней, составляют ее контекст. Таким образом, контекстом называется окружение системы.

UML позволяет моделировать контекст с помощью диаграмм прецедентов, в которых внимание акцентируется на окружающих систему актерах. Важно правильно определить актеры, поскольку это позволяет описать класс сущностей, взаимодействующих с системой. Еще важнее определить, что не является актером, так как при этом ограничивается окружение системы: в нем остаются только те элементы, которые участвуют в ее работе.

Моделирование контекста системы состоит из следующих шагов:

1. Идентифицируйте окружающие систему актеры. Для этого нужно найти группы, которым участие системы требуется для выполнения их задач; группы, которые необходимы для осуществления системой своих функций; группы, взаимодействующие с внешними программными и аппаратными средствами, а также группы, выполняющие вспомогательные функции администрирования и поддержки.
2. Организуйте похожих актеров с помощью отношений обобщения/специа лизации.
3. Введите стереотипы для каждого актера, если это облегчает понимание.
4. Поместите актеров на диаграмму прецедентов и определите способы их свя зи с прецедентами системы.

Например, на рис. 17.2 показан контекст системы, работающей с кредитными карточками, где основное внимание уделяется окружающим ее актерам. В первую очередь это Клиенты двух типов (Индивидуальный клиент и Корпоративный клиент), соответствующие ролям, которые играют люди при взаимодействии с системой. В этом контексте показаны и актеры, представляющие другие организации, такие как Торговые предприятия (с ними покупатели совершают карточные транзакции, приобретая вещи или услуги) и Субсидирующие финансовые институты (выполняющие роль клиринговой палаты для карточных счетов). В реальном мире последние два актера, скорее всего, сами будут программными системами.

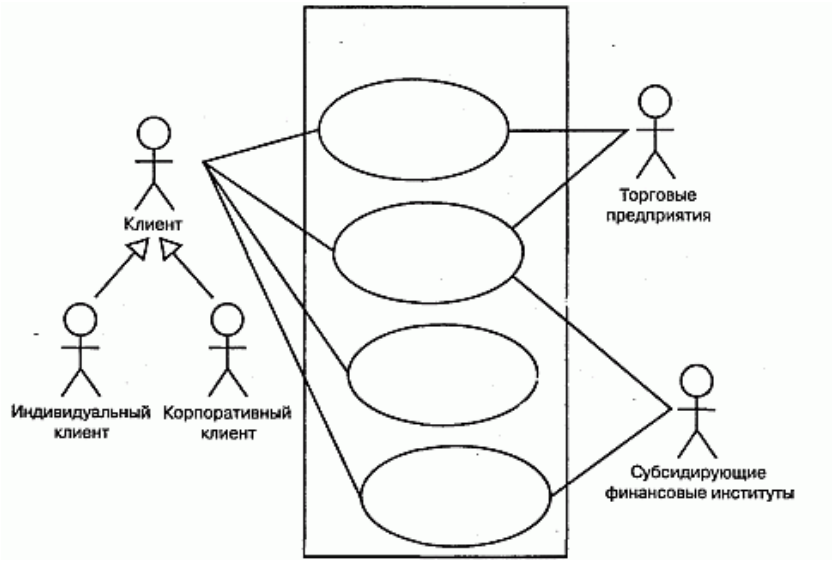

Рис. 17.2 Моделирование контекста системы

Тот же метод позволяет моделировать и контекст подсистемы. Вообще, элемент, который на одном уровне абстракции выглядит как система, часто становится подсистемой на другом, более высоком уровне абстракции. Моделирование контекста подсистемы может пригодиться при построении системы из нескольких взаимосвязанных частей.

Требования к системе

Требование (Requirement) - это особенность проекта, свойство или поведение системы. Приступая к сбору требований, вы как бы описываете условия контракта, заключаемого между системой и сущностями вне нее, в котором декларируется, что система должна делать. При этом, как правило, вас заботит не то, как именно система будет выполнять поставленные задачи, а только то, что она будет делать. Хорошо спроектированная система должна полностью выполнять все требования, причем делать это предсказуемо и надежно. Ее создание начинается с соглашения о том, каково ее назначение, хотя в ходе разработки понимание требований будет постепенно изменяться. Аналогично при работе с готовой системой понимание того, как она себя ведет, имеет принципиальное значение для ее правильного использования.

Требования можно выразить по-разному, от неструктурированного текста до выражений на формальном языке (или, например, с помощью примечаний). Большая часть функциональных требований к системе - или даже все они - может быть выражена в виде прецедентов использования, в чем помогают диаграммы прецедентов UML.

Моделирование требований к системе производится следующим образом:

1. Установите контекст системы, идентифицировав окружающие ее актеры.
2. Для каждого актера рассмотрите поведение, которого он ожидает или требу ет от системы.
3. Поименуйте эти общие варианты поведения как прецеденты.
4. Выделите общее поведение в новые прецеденты, которые будут использо ваться другими; выделите вариации поведения в новые прецеденты, расши ряющие основные потоки событий.
5. Смоделируйте эти прецеденты, актеры и отношения между ними на диа грамме прецедентов.
6. Дополните прецеденты примечаниями, описывающими нефункциональные требования; некоторые из таких примечаний можно присоединить к систе ме в целом.

Рис. 17.3 расширяет предыдущую диаграмму. Хотя отношения между актерами и прецедентами на ней опущены, но присутствуют дополнительные прецеденты, которые описывают важные, пусть и невидимые для обычного пользователя, элементы поведения системы. Эта диаграмма удобна, поскольку дает возможность пользователям, экспертам и разработчикам совместно визуализировать, специфицировать, конструировать и документировать свои решения относительно функциональных требований к системе. Например, прецедент Обнаружение фальшивых карточек описывает поведение, важное как для Торговых предприятий, так и для Субсидирующих финансовых институтов. Другой прецедент - Отчет о состоянии счета - также описывает поведение, требуемое от системы различными организациями в своих контекстах.

Рис. 17.3 Моделирование требований к системе

Требование, моделируемое прецедентом Управление сбоями в сети, немного отличается от остальных, поскольку представляет вспомогательное поведение системы, необходимое, чтобы гарантировать надежное и непрерывное функционирование.

Такая же методика применима и при моделировании требований к подсистеме.

Прямое и обратное проектирование

Большинство других диаграмм UML, включая диаграммы классов, компонентов и состояний, удобно для прямого и обратного проектирования, поскольку у каждой из них имеется аналог в исполняемой системе. С диаграммами прецедентов дело обстоит несколько иначе, поскольку они скорее отражают, чем определяют реализацию системы, подсистемы или класса. Прецеденты описывают то, как ведет себя элемент, а не то, как реализуется соответствующее поведение, и поэтому не могут быть непосредственно подвергнуты прямому и обратному проектированию.

Прямое проектирование подразумевает преобразование модели в исполняемый код на каком-либо языке программирования. Прямое проектирование диаграмм прецедентов приводит к получению тестов для соответствующего элемента. Каждый прецедент в диаграмме определяет поток событий (и его варианты), а эти потоки специфицируют ожидаемое поведение элементов - иначе говоря, именно то, что подлежит тестированию. Хорошо структурированный прецедент содержит также пред- и постусловия, с помощью которых можно определить начальное состояние теста и критерии успешности. Для каждого сценария в диаграмме вы можете разработать тест, а затем запускать его при появлении каждой новой версии элемента, подтверждая тем самым, что он работает корректно, и, стало быть, другие элементы могут на него полагаться.

Прямое проектирование диаграммы прецедентов состоит из следующих шагов:

1. Идентифицируйте основной и альтернативный потоки событий для всех прецедентов, представленных на диаграмме.
2. В зависимости от предполагаемой глубины тестирования сгенерируйте тестовый скрипт для каждого потока, используя предусловия в качестве начального состояния, а постусловия - в качестве критерия успешности.
3. При необходимости сгенерируйте тестовое окружение, в котором представ лены все актеры, взаимодействующие с прецедентом. Актеры, которые пере дают элементу информацию или на которых элемент воздействует, можно имитировать или же подставить вместо них реальные эквиваленты.
4. С помощью инструментальных средств выполняйте эти тесты всякий раз после завершения разработки новой версии элемента, к которому применяется диаграмма.

Обратным проектированием называется процесс преобразования кода, написанного на каком-либо языке программирования, в модель. Автоматическое обратное проектирование диаграмм прецедентов на данном этапе развития отрасли практически невыполнимо из-за потери информации при переходе от спецификации поведения элемента к его реализации. Тем не менее можно изучить существующую систему и разобраться в ее поведении самостоятельно, а затем представить результаты анализа в виде диаграммы прецедентов. Фактически это придется делать всякий раз, когда вы столкнетесь с недокументированным программным обеспечением. Диаграммы прецедентов UML в данном случае просто предоставляют в ваше распоряжение стандартный и выразительный язык для описания того, что вы обнаружите.

Обратное проектирование диаграмм прецедентов производится так:

1. Идентифицируйте все взаимодействующие с системой актеры.
2. Изучите способы, посредством которых актеры взаимодействуют с системой, изменяют состояние системы или ее окружения, а также реагируют на события.
3. Осуществите трассировку потока событий в исполняемой системе относительно каждого актера. Начинать при этом нужно всегда с главных потоков и только потом рассматривать альтернативные.
4. Сгруппируйте родственные потоки, объявив соответствующий прецедент. Подумайте о моделировании вариаций с помощью отношений расширения и о моделировании общих потоков с помощью отношений включения.
5. Изобразите актеры и прецеденты на диаграмме прецедентов и установите отношения между ними.

Советы

Создавая диаграммы прецедентов в UML, помните, что каждая из них является всего лишь графическим представлением статического вида системы с точки зрения прецедентов. Это означает, что ни одна диаграмма прецедентов, взятая в отдельности, не может, да и не должна охватывать весь этот вид целиком. В совокупности диаграммы прецедентов дают полное представление о виде системы с точки зрения прецедентов, а каждая из них в отдельности - только об одном из его аспектов.

Хорошо структурированная диаграмма прецедентов обладает следующими свойствами:

- акцентирует внимание только на одном аспекте статического вида системы с точки зрения прецедентов;
- содержит только такие прецеденты и актеров, которые важны для понимания этого аспекта;
- содержит только такие детали, которые соответствуют данному уровню абстракции; следует показывать только те дополнения (например, точки расширения), которые необходимы для понимания системы;
- не настолько лаконична, чтобы ввести читателя в заблуждение относительно важной семантики.

При изображении диаграммы прецедентов руководствуйтесь следующими принципами:

- дайте ей имя, соответствующее назначению;
- расположите элементы так, чтобы свести к минимуму число пересечений;
- пространственно организуйте элементы так, чтобы семантически близкие сущности физически располагались рядом;
- используйте примечания и цвет, чтобы привлечь внимание читателя к важным особенностям диаграммы;
- старайтесь не показывать слишком много видов отношений. В общем случае, если есть много сложных отношений включения и расширения, выделите их в отдельную диаграмму.