Глава 24

Уточнение прецедентов

Основные положения

- Для поддержки действий по проектированию и написанию кода необходимо детализировать разработанные при выявлении требований прецеденты.
- Прецеденты лучше всего использовать в тех случаях, когда система имеет много функций и должна поддерживать различные типы пользователей.
- Если у системы пользователей мало или они вовсе отсутствуют, а интерфейсы минимальны, т.е. основную массу требований составляют нефункциональные требования и ограничения проектирования, то применять прецеденты не эффективно.

В частях 1 и 2 мы начали рассматривать прецеденты как средство выражения требований к системе. В последнее время данный метод приобрел популярность.

Использование прецедентов имеет ряд преимуществ по сравнению с традиционным подходом, когда определяются отдельные декларативные программные требования.

- Прецеденты относительно легко писать.
- Прецеденты пишутся на языке пользователя.
- Прецеденты предлагают связные нити поведения (или сценарии), которые понятны как пользователю, так и разработчику.
- Благодаря описанию "нитей поведения" и содержащимся в языке UML специализированным элементам и нотациям моделирования, прецеденты обеспечивают дополнительные возможности связывать деятельность по разработке требований с проектированием и реализацией. (Мы будем более подробно обсуждать эту тему в главе 30.)
- Графическое представление прецедентов в UML и их поддержка различными инструментальными средствами моделирования обеспечивают визуализацию связей между прецедентами, что может содействовать пониманию сложной программной системы.
- Сценарий, описанный с помощью прецедента, может практически без изменений использоваться в качестве сценария тестирования во время проверки правильности.

Вопросы, на которые нужно ответить

Когда следует использовать методологию прецедентов

Для фиксации основной массы требований к системе следует использовать прецеденты в том случае, если для приложения выполнено одно или оба из приведенных ниже условий.

- Система является функционально ориентированной; существует несколько различных типов пользователей и функционального поведения. Так как прецеденты отдельно описывают поведение системы *для каждого типа пользователя*, их применение наиболее эффективно, когда существует много типов пользователей системы и система должна предоставлять различные наборы функций каждому из них.
- Команда реализует систему, используя UML и объектно-ориентированные (ОО) методы. Определенные ОО-понятия (такие, как унаследованное поведение акторов и прецедентов, абстрактные акторы) хорошо приспособлены к методу прецедентов и создают дополнительные удобства для аналитика или разработчика моделей. UML-нотация прецедентов также поддерживает визуальное моделирование системы и обеспечивает парадигму моделирования, которая поддерживает представление требуемого поведения системы (прецедент), а также того, как это поведение реализуется в программе (посредством реализаций прецедентов).

Когда прецеденты не являются наилучшим вариантом

Однако прецеденты подходят не для всех типов систем и требований. Особенно это касается рассматриваемых ниже разновидностей систем. Работая над такой системой, необходимо дополнить модель прецедентов или даже вовсе от нее отказаться.

Системы, где пользователей мало или нет вообще, а интерфейсы минимальны. Существует много классов функционально наполненных систем, которые имеют мало внешних интерфейсов и пользователей. Примерами могут служить системы, проектируемые для проведения научных вычислений или имитаций, встроенные системы, системы управления процессами, система проверки наличия вирусов, работающая без вмешательства оператора, а также различные служебные программы, такие как компиляторы и программы управления памятью. Хотя и здесь можно применять прецеденты и, возможно, они будут полезны в качестве дополнения к традиционному подходу, в данном случае существуют более простые способы выразить большую часть требований.

Системы, в которых доминируют нефункциональные требования и ограничения проектирования. Как уже отмечалось ранее, прецеденты не предназначены для представления нефункциональных требований — атрибутов системы и ее среды, специальных требований и ограничений проектирования. Для включения требований данного типа в каждом прецеденте есть раздел “специальные требования”. Это работает удовлетворительно, если подобные требования применяются к одному или нескольким прецедентам. Но в общем случае не все такие требования удается связать с конкретным прецедентом.

Глобальные нефункциональные требования вообще не удается удовлетворительно представить с помощью прецедентов. Это относится к требованиям соответствия зако-

нодательству и инструкциям, операционным средам и стандартам разработки программ. (Например, в Rational одна спецификация используется исключительно для задания требований глобализации программных продуктов. Эти требования практически полностью состоят из ограничений, которыми следует руководствоваться при проектировании программного обеспечения, чтобы сделать возможным и относительно дешевым его перевод на другие языки. Прецеденты нужны только для описания отдельных вариантов предполагаемого использования, таких как "Человек, говорящий по-французски, использует немецкий вариант ОС".)

Проблема избыточности

Многие прецеденты весьма похожи, но в то же время и достаточно различны, чтобы требовать отдельного описания. Это приводит к значительной избыточности описания, что увеличивает объем документации требований. Кроме того, возникают проблемы обслуживания, если необходимо изменить общее поведение, выраженное во многих прецедентах. В этом случае для уменьшения избыточности можно использовать такие отношения прецедентов, как генерализация, отношения включения и наследования (Буч (Booch), 1990).

Однако использование этих отношений само по себе создает дополнительные сложности и может оказаться неэффективным, если поведение можно описать другими способами. Действительно, некоторые примеры достаточно сложного поведения проще описать на естественном языке (например, "если система находится в состоянии готовности и каждый из двух офицеров нажимает на кнопку запуска и удерживает ее в этом положении более 1 секунды, произойдет запуск ракеты"). Конечно, можно попытаться и в этих случаях применить прецеденты, но задача состоит в том, чтобы выбрать наиболее подходящий метод для существующих обстоятельств, который обеспечит простое представление и возможность понимания, а не в том, чтобы использовать некий метод потому, что вы думаете, что так надо. В большинстве проектов для создания оптимального подхода можно использовать прецеденты совместно с традиционными методами.

Совершенствование спецификаций прецедентов

В данной главе мы продолжим изучение прецедентов, которые начали рассматривать в главах 2 и 13, и используем их для уточнения спецификации системы. Это удобно, так как можно вернуться к прецедентам, разработанным на более ранних этапах, и описать их более детально. В зависимости от достигнутого в процессе их выявления уровня конкретизации, разработанные ранее прецеденты могут быть уже достаточно подробными для проведения проектирования и реализации. Однако более вероятно (и мы рекомендуем поступать именно так), что на стадии выявления требований был сохранен достаточно высокий уровень абстракции, чтобы не запутаться в деталях. Может оказаться, что определены еще не все необходимые прецеденты, исключительные условия, условия состояния и другие специальные условия, которые менее интересны для пользователя, но могут заметно повлиять на проектирование системы. Теперь пришло время обеспечить этот дополнительный уровень конкретизации.

Замечание. В задачу данной книги не входит изложение полного курса по использованию прецедентов. Если вы заинтересованы в более полном овладении данной методологией и сопутствующими технологиями, рекомендуем обратиться к двум хорошим книгам по данной теме: Шнайдер и Уинтерс (Schneider, Winters, 1998), а также Джейкобсон (Jacobson, 1999). Тем не менее, мы рассмотрим некоторые основополагающие принципы методологии прецедентов.

Для того чтобы конкретизировать прецеденты, нам нужно использовать более строгий подход, чтобы лучше понять некоторые нюансы. Рассмотрим еще раз определение прецедентов, обращая основное внимание на то, что говорится о них в UML.

> *Прецедент – это описание множества действий (включая варианты), которые система выполняет для того, чтобы доставить полезный осязаемый результат определенному актору* (Буч (Booch), 1999).

Вот это да! Выглядит так, как будто это определение составлялось на собрании адвокатов![1] Как мы уже отмечали ранее, методология прецедентов выделяет два элемента, которые должны присутствовать во всех реализациях прецедентов.


Прецедент

1. *Прецедент.* В UML прецедент изображается в виде овала. Несмотря на то что прецедент является текстовым описанием, данная пиктограмма служит вспомогательным средством, помогающим визуально моделировать систему и отображать взаимодействия между прецедентами и другими элементами модели.


Актор

2. *Акторы.* Актор — это некто или нечто, взаимодействующее с нашей системой. Существует три типа акторов: пользователи ("техник Билл"), приборы ("контроллер двигателя руки робота") и другие системы ("контроллер ЦБУ системы HOLIS"). Акторы не являются частью описываемой системы, а находятся за ее пределами.

Рассмотрим некоторые другие ключевые фразы UML-определения: "Прецедент — это описание множества действий (включая варианты), которые выполняет система , чтобы доставить полезный осязаемый результат определенному актору".

- *Варианты.* Прецедент описывает основной поток событий (или нить), а также варианты (или альтернативные потоки).
- *Множество действий.* Описывает выполняемую функцию или алгоритмическую процедуру, которая приводит к результату. Когда актор инициирует прецедент, предлагая системе некий ввод, происходит вызов этого множества. Отдельное действие атомарно, т.е. оно либо выполняется целиком, либо не выполняется вовсе. Требование атомарности строго обязательно при выборе уровня детализации прецедента. Следует изучить предлагаемый прецедент и, если некое действие не является атомарным, обеспечить дальнейшую детализацию.

[1] В действительности это было собрание специалистов по методологии. Айвар Джейкобсон (Ivar Jacobson) как-то рассказал анекдот: В чем разница между специалистом по методологии и террористом? С террористом можно договориться.

- *Система выполняет.* Это означает, что система обеспечивает описанные в прецеденте функциональные возможности. Это то, что делает система в зависимости от полученного ввода.
- *Полезный осязаемый результат.* Важно отметить, что результат прецедента должен быть полезен пользователю, т.е. "жилец нажимает кнопку выключателя" не является правильным прецедентом; (система ничего не делает для пользователя). Но "жилец нажимает кнопку выключателя, и система включает свет" является осмысленным прецедентом.
- *Определенный актор.* Это человек или прибор (жилец по имени Линда или сигнал от кнопки нештатной ситуации), инициирующий данное действие (переключение света или активацию систему безопасности).

Эволюция прецедентов

На ранних итерациях в части 3, "Определение системы", было выявлено большинство наиболее важных прецедентов. Однако описывались только немногие из них — те, которые считались архитектурно значимыми или особенно хорошо описывали поведение системы. Как правило, описания этих прецедентов выполнены в виде дополнения к документу-концепции, в котором описывается, как предполагается использовать содержащиеся в документе функции.

В процессе уточнения завершается разработка всех прецедентов, необходимых для определения системы. Показателем того, что прецедентов "достаточно", является то, что полный набор прецедентов описывает все возможные способы использования системы на уровне конкретизациипригодном для проектирования, реализации и тестирования.

Следует отметить, что детализация прецедента *не является* декомпозицией системы. Другими словами, мы не начинаем с прецедента высокого уровня, чтобы затем разбивать его на все большее число прецедентов. Вместо этого мы стараемся более точно описать взаимодействия акторов с системой. Таким образом, разработка прецедентов более похожа на уточнение последовательностей действий, а не на иерархическое разбиение их на более мелкие действия. В модели часто есть прецеденты, которые настолько просты, что не нуждаются в детальном описании потока событий; достаточно краткого описания. Критерием для принятия такого решения является то, что пользователи понимают, что означает прецедент, а для разработчиков и тестологов данный уровень детализации удобен.

Какие действия включить в прецедент

Часто трудно решить, является ли множество взаимодействий пользователя с системой (или диалог) одним прецедентом или несколькими. Рассмотрим использование машины, принимающей банки и бутылки. Клиент загружает в нее консервные банки и бутылки, нажимает кнопку и получает квитанцию, по которой может затем получить деньги.

Является ли загрузка сдаваемых предметов одним прецедентом, а получение квитанции — другим? Или это все один прецедент? Имеют место два действия, но в отдельности они не приносят пользы клиенту. Именно полный диалог, со всеми загрузками и получением квитанции, имеет смысл для клиента. Следовательно, именно полный диалог — от загрузки первого сдаваемого предмета до нажатия кнопки и получения квитанции — является полным вариантом использования, т.е. прецедентом.

Кроме того, хотелось бы хранить эти два действия вместе, чтобы иметь возможность одновременно их просматривать, модифицировать, вместе проверять, изменять, когда это необходимо, писать характеризующую их пользовательскую документацию и обращаться с ними, как с единым целым. Это становится особенно важным в более крупных системах.

Рабочий пример. Строение простого прецедента

Рассмотрим шаг за шагом процедуру определения прецедента. Будем использовать простой пример из системы HOLIS: жилец включает освещение в комнате, используя автоматическую систему управления освещением HOLIS.

Определение акторов

Первым делом необходимо точно установить, кто взаимодействует с прецедентом. Во многих системах, спроектированных для пользователей, следует сначала выявить людей, которые будут использовать систему. В нашем случае жилец взаимодействует с системой, чтобы управлять освещением в комнате. Таким образом, выявлен единственный актор, пользователь (Жилец), нажимающий кнопку пульта.

> **Предостережение.** При определении акторов полезно вести "именной список", чтобы видеть, какие из них уже определены, и случайно не создать некий актор повторно под другим именем.

Дать название прецеденту

Каждый прецедент должен иметь имя, показывающее, что достигается при его взаимодействии с актором(ами). Имя может состоять из нескольких слов, проясняющих его смысл. Различные прецеденты не должны иметь одно и то же имя.

К заданию имени следует подходить ответственно. Оно должно быть уникальным и в то же время таким, чтобы его можно было легко отличить от имен других прецедентов данного проекта. В английском языке имена прецедентов часто начинаются с глагола, обозначающего действие, которое отражает предназначение данного прецедента. Назовем наш прецедент "Управление освещением".

Можно создавать структуру имен формальным методом, чтобы сгруппировать аналогичные прецеденты в аналогично именованные группы. Можно также ввести порядковый номер или другой уникальный идентификатор в имя прецедента, чтобы было удобнее работать со списком прецедентов. Например, разработчик может указать имя данного прецедента как "031 Управление освещением". Однако опыт свидетельствует, что простого именования прецедентов и, возможно, применения неких вспомогательных программ, позволяющих нам их находить, сортировать и анализировать, обычно вполне достаточно.

Составление краткого описания

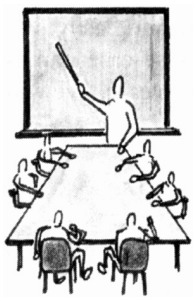
Краткое описание прецедента должно отражать его роль и цель. При его написании следует рассмотреть, какие акторы участвуют в нем, а также обратиться к глоссарию. Если необходимо, можно определить новые понятия.

Это описание должно служить своего рода неформальным обзором функциональных возможностей. Их полное описание содержится в следующем разделе прецедента "Поток событий". Описание прецедента предназначено для получения "общего впечатления" и ничего более. В нашем случае можно описать прецедент следующим образом.

> **Описание прецедента "Управление освещением"**
>
> Данный прецедент определяет способ включения и выключения света, а также изменения его яркости, в зависимости от того, как долго пользователь удерживает кнопку пульта в нажатом состоянии.

Определение потока событий

Сердцевиной прецедента является поток событий. Как правило, это текстовое описание операций актора и различных ответов системы. Поток событий описывает, что, как предполагается, будет делать система в зависимости от поведения актора. Между прочим, не обязательно описывать поток в текстовой форме. Для этого можно использовать диаграммы взаимодействия UML, а также другие формальные методы, обсуждаемые в главе 28. Все они вполне пригодны для документации прецедентов. Главное — обеспечить *понимание*, и не существует единственного подхода на все случаи жизни. Однако, как правило, вполне подходит описание на естественном языке.



Поток событий описывает достижение цели прецедента и предназначен для рассмотрения следующими заинтересованными лицами.

- Клиентами, которые одобряют результат и "благословляют" функции
- Пользователями, которые будут работать с системой
- Разработчиками прецедентов, которые заинтересованы в точном описании желаемого поведения системы
- Рецензентами, которые составляют непредвзятое мнение о системе
- Проектировщиками, которые анализируют прецеденты в поисках классов, объектов и т.п.
- Тестологами, которым нужно создать тестовые примеры
- Менеджером проекта, которому необходимо понимать весь проект в целом
- Составителем технической документации, которому нужно документировать функции системы так, чтобы было понятно пользователю
- Людьми, из отдела маркетинга и продаж, которым необходимо понимать функции продукта и объяснять его достоинства остальным

Вы, возможно, думаете, что *практически не бывает ситуаций, когда можно описать простой поток событий, который работает во всех случаях; всегда нужен способ для описания некоторых альтернативных потоков.* Ничего страшного. Определение потока событий

прецедента допускает альтернативные потоки. Но сначала создадим основной поток для нашего примера.

Основной поток для прецедента "Управление освещением". Отметим, что данный поток событий нигде не указывает, *как* система выполняет то или иное действие. Он указывает только,*что* происходит.

> Основной поток начинается, когда Жилец нажимает любую кнопку пульта. Если Жилец отпускает кнопку в течение периода времени, отсчитываемого таймером, система"переключает" состояние освещения.
>
> - Если освещение было включено, оно полностью выключается.
> - Если свет был выключен, освещение включается с тем же уровнем яркости, который был перед последним выключением.
>
> Конец основного потока.

Альтернативный поток событий. Во многих случаях прецедент может иметь различные потоки, в зависимости от возникающих условий. Иногда эти потоки связаны с выявленными в процессе обработки ошибочными условиями, в других случаях они могут описывать дополнительные способы обработки конкретных условий. Например, прецедент, печатающий квитанцию для операции с кредитной карточкой, может обнаружить, что у принтера закончилась бумага. Этот специальный случай будет описан в прецеденте как альтернативный поток событий. При записи альтернативных потоков не забывайте документировать условия, приводящие к ним! На количество альтернативных потоков не накладываются ограничения, поэтому будьте внимательны и документируйте все альтернативные потоки, в том числе все возможные условия возникновения ошибок.

В нашем примере альтернативный поток событий возникает, когда Жилец удерживает кнопку пульта в нажатом состоянии дольше одной секунды. Итак, нам нужно добавить в прецедент альтернативный поток.

> **Альтернативный поток событий: постепенное изменение яркости**
>
> Если Жилец удерживает кнопку пульта в нажатом состоянии дольше одной секунды, система инициирует действия по изменению яркости для данной кнопки пульта.
>
> Пока Жилец продолжает удерживать кнопку, происходит следующее.
>
> 1. Яркость контролируемого источника постепенно повышается до максимального значения со скоростью 10 процентов в секунду.
> 2. Когда достигнуто максимальное значение, яркость контролируемого источника постепенно понижается до минимального уровня со скоростью 10 процентов в секунду.
> 3. Когда достигнуто минимальное значение, процесс продолжается с шага 1.
>
> Когда Жилец отпускает кнопку, происходит следующее.
>
> 4. Система прекращает изменять яркость освещения.

Выявление пред- и постусловий

Иногда необходимо выявить предусловия, которые влияют на поведение системы, описанное прецедентом, а также описать постусловия, такие как состояние системы и перманентные данные, полученные после завершения прецедента. Но использовать пред- и постусловия нужно только тогда, когда необходимо прояснить поведение, выраженное прецедентом.

Важно проводить различие между событиями, которые запускают потоки прецедента, и предусловиями, которые должны быть выполнены до того, как можно будет инициировать прецедент. Например, предусловием для прецедента "Управление освещением" является то, что домовладелец (Жилец) должен обеспечить наличие определенного набора осветительных приборов, способных изменять яркость. Еще одним предусловием является то, что выбранная кнопка пульта должна быть запрограммирована для управления этим набором. (Предполагается, что другие прецеденты описывают, как осуществляются эти предусловия.) Итак, нам нужно сформулировать предусловия.

Предусловия для прецедента "Управление освещением"

- Для выбранной кнопки пульта должен быть предусмотрен режим "Изменение яркости".
- Выбранная кнопка пульта должна быть предварительно запрограммирована для управления неким набором осветительных приборов.

Часто необходимо выявлять и включать в документацию постусловия. Они позволяют точно указывать состояние, которое должно быть истинным по окончании прецедента, даже если использовались альтернативные пути.

Чтобы яркость была такой, как нужно, когда Жилец включает свет в следующий раз, система должна "помнить" уровень яркости, который был установлен для выбранной кнопки пульта после действий по изменению яркости. Следовательно, это постусловие, которое мы должны записать для данного прецедента.

Постусловия для прецедента "Управление освещением"

- После окончания данного прецедента запоминается текущий уровень яркости для выбранной кнопки пульта.

Теперь соберем все это вместе. В табл. 24.1 содержится все, что получится после того, как мы объединим все важные "кусочки" нашего прецедента. (Хотя для прецедента можно определить еще много других элементов, они не так важны для нас на данном этапе.) Этот прецедент документирован в повествовательном стиле, его можно найти среди артефактов системы HOLIS в приложении А.

Таблица 24.1. Определение прецедента

Элемент	Значение
Название прецедента	*Управление освещением*
Краткое описание	Данный прецедент определяет способ включения и выключения света, а также изменения его яркости в зависимости от того, как долго пользователь удерживает кнопку пульта в нажатом состоянии

Окончание табл. 24.1

Элемент	Значение
Поток событий	Основной поток событий начинается, когда Жилец нажимает кнопку пульта (“Управление включением”). Если Жилец отпускает кнопку в течение периода времени, отсчитываемого таймером, система “переключает” состояние освещения. Это означает следующее. ■ Если свет был включен, он полностью выключается ■ Если свет был выключен, он включается с тем же уровнем яркости, который был перед последним выключением
Альтернативный поток событий	Если Жилец удерживает кнопку пульта в нажатом состоянии более одной секунды, система инициирует действия по изменению яркости для указанной кнопки. Пока Жилец продолжает удерживать кнопку в нажатом сотоянии, производятся следующие действия. 1. Яркость контролируемого источника постепенно повышается до максимального значения со скоростью 10 процентов в секунду 2. Когда достигнуто максимальное значение, яркость контролируемого источника постепенно снижается до минимального уровня со скоростью 10 процентов в секунду 3. Когда достигнуто минимальное значение, процесс продолжается с шага 1 Когда Жилец отпускает кнопку пульта происходит следующее. 4. Система прекращает изменять яркость источника
Предусловия	■ Выбранная кнопка пульта должна иметь режим, допускающий изменение яркости ■ Выбранная кнопка должна быть предварительно запрограммирована для управления неким набором осветительных приборов
Постусловия	По окончании прецедента установленная яркость запоминается системой
Специальные требования	Минимальный уровень освещения не должен быть равен 0. Он должен быть не очень ярким, соответствующим уровню ночного освещения

Далее…

После того как все прецеденты выявлены и описаны приблизительно на таком уровне детализации, процесс уточнения той части системы, которую мы решили описывать с помощью прецедентов, заканчивается. В следующей главе мы рассмотрим создание спецификаций.