Crystal Collection для профессиональных разработчиков программного обеспечения
Научный руководитель проекта – Алистер Коберн

Современные методы описания функциональных требований к системам

Алистер Коберн

Writing effective use cases
(The Crystal Collection for software professionals)
Alistair Cockburn

Современные методы описания функциональных требований к системам
Алистер Коберн

Переводчик Е. Борисова
Научный редактор А. Вендров
Корректор И. Красненкова
Верстка И. Нагорновой

Pearson Education Corporate Sales Division
One Lake Street
Upper Saddle River, NY 07458
(800) 382-3419
corpsales@pearsontechgroup

ISBN 0-201-70225-8

Изд. № : OAI (03)
ЛР № : 070612 30.09.97 г.
ISBN 5-85582-152-8

Подписано в печать 04.01.2002. Формат 70 x 100/16
Гарнитура Литературная Печать офсетная
Печ. л. 18 Тираж 3200 Заказ № 5800
Цена договорная

Издательство "Лори". Москва 123 557, Б. Тишинский пер., д. 40, корп. 2
Телефон для оптовых покупателей: (095)256-02-83
Размещение рекламы: (95)259-01-62

WWW.LORY–PRESS.RU

Отпечатано в полном соответствии с качеством
предоставленных диапозитивов в ППП "Типография "Наука"
121099 Москва, Шубинский пер., 6

Современные методы описания функциональных требований к системам

Алистер Коберн

Издательство "Лори"

Содержание

Предисловие

Все больше разработчиков прибегают к помощи вариантов использования для описания бизнес-процессов или поведения программных систем и требований к ним. Составление описания использования системы может оказаться сложной задачей. Проблема в том, что создание вариантов использования можно сравнить с написанием прозаического эссе, где необходимы четкие формулировки, которые свойственны письменной прозе. Трудно сказать, на что похож хороший вариант использования. Однако еще труднее научиться писать хорошие варианты использования.

Здесь содержатся правила, которые я использую при написании вариантов использования: как может думать индивидуум, чего он может твердо придерживаться, чтобы создать лучший вариант использования и лучший набор вариантов использования.

Я включил примеры хороших и плохих вариантов использования, привел разные стили написания. Вариант использования не должен быть выдающимся, чтобы быть полезным. Даже посредственные варианты использования полезны, причем больше, чем многие конкурирующие с ними технические требования. Итак, напишите что-нибудь внятное, и вы окажете услугу вашей организации.

Круг читателей

Эта книга предназначена преимущественно для профессионалов, которые обучаются самостоятельно. В книге содержится подготовительная информация для усвоения нового материала: понятия, примеры, памятки и упражнения (некоторые с ответами).

Преподавателям следует найти подходящие объяснения и примеры для своих групп. Разработчики курсов должны суметь построить материал на основе этой книги, составляя задания по своему усмотрению. (Однако, поскольку я включил ответы ко многим упражнениям, им придется построить экзаменационный материал самим.)

Структура книги

Книга построена как общее введение в написание вариантов использования, за которым следует подробное описание составных частей варианта использования, часто задаваемые вопросы, памятки и заключительные замечания.

Введение дает начальное представление о ключевых понятиях. Обсуждается, как выглядит вариант использования, когда он описывается, какие вариации допустимы. Варианты использования выглядят по-разному в зависимости от того, когда, где, с кем и почему вы их пишете.

Часть 1, *Составные части варианта использования*, содержит главы, посвященные каждому основному понятию, которое необходимо усвоить, и части шаблона, которые следует применять. Сюда входят главы Вариант использования как соглашение о поведении, Область действия, Участники и действующие лица, Три поименованных уровня цели, Предусловия, триггеры и гарантии, Сценарии и шаги, Расширения, Изменения в технологии и данных, Связывание вариантов использования и Форматы вариантов использования.

Часть 2, *Часто обсуждаемые темы,* освещает следующее: Когда считать работу завершенной, Как работать с большим количеством вариантов использования, CRUD и параметризованные варианты использования, Моделирование бизнес-процессов, Пропущенные требования, Роль вариантов использования в общем процессе, Краткие описания вариантов использования и экстремальное программирование (XP, EXtreme Programming) и Распространенные ошибки.

Часть 3, *Памятки для занятых,* содержит основные правила для тех, кто закончил читать книгу, или уже знает этот материал и хочет возвратиться к ключевым идеям. Главы называются: Памятки для каждого варианта использования, Памятки для набора вариантов использования, Памятки для работы над вариантами использования.

Далее следуют четыре приложения: Варианты использования в UML, Ответы к (некоторым) упражнениям, Глоссарий и Список использованных материалов.

Преемственность идей

В конце 60-х годов Ивар Якобсон, работая над телефонными системами в компании Ericsson, изобрел то, что позднее получило название вариантов использования (use cases). В конце 80-х он представил их специалистам по объектно-ориентированному программированию. Они получили признание, заполнив брешь в процессе формирования требований. В начале 90-х я прошел курс у Якобсона. Ни он, ни его команда не пользовались моими терминами цель (goal) и неудача (goal failure), однако на самом деле они применяли эти понятия. В нескольких случаях мы с ним не нашли серьезных противоречий между его и моей моделью. Я расширил его модель, чтобы осовременить ее.

Я создал концептуальную модель Действующие лица и цели (Actors and Goals) в 1994 г., когда писал руководство по вариантам использования для IBM Consulting Group. Модель объясняла многие непонятные моменты в вариантах использования и одновременно служила руководством по их структурированию и написанию. Модель Действующие лица и цели неформально распространялась с 1995 г. Она находилась на *http://members.aol.com/acocburn*, позднее на *www.usecase.org*, а в 1997 г. в журнале *Journal of Object-Oriented Programming* была опубликована моя статья о структурировании вариантов использования с целями.

В период с 1994 по 1999 гг. эти идеи не претерпели изменений, хотя в теории оставались еще "белые пятна". В конце концов я понял, почему многие с таким трудом усваивали столь простые идеи (не говоря уж о том, что я обнаружил много ошибок в своих первых опытах). Кроме того, выявились недостатки в модели Действующие лица и цели. Все это привело к трактовке, изложенной в этой книге, и к новой представленной здесь идее — модели Участники и интересы (Stakeholders and Interests).

Унифицированный язык моделирования UML (Unified Modeling Language) не оказал заметного влияния на эти идеи, обратное также верно. Гуннар Овергаард, бывший коллега Якобсона, написал большую часть материала о вариантах использования на основе UML и поддержал наследие Якобсона. Однако группа стандартизации UML подвержена серьезному влиянию графических средств, что привело к выхолащиванию текстовой природы вариантов использования в стандарте UML. Гуннар Овергаард и Ивар Якобсон обсудили мои идеи и уверили меня, что большая часть того, что я говорю о варианте использования, умещается внутри одного из эллипсов UML и, следовательно, никогда не подвергнется воздействию стандарта UML и не повлияет на него. Это значит, что идеи этой книги совместимы со стандартом варианта использования в UML версии 1.3. С другой стороны, если вы прочтете только стандарт UML, в котором не обсуждается содержание и способ написания варианта использования, вы не поймете, что такое вариант использования и как его применять, и получите ложное представление о вариантах использования как графической, так и текстовой конструкции. Цель этой книги — показать, как писать эффективные варианты использования. Стандарт немного говорит по этому поводу, поэтому я вынес свои замечания по UML в приложение А.

Используемые примеры

Примеры, приведенные в этой книге, в большинстве своем взяты из реальных проектов и могут показаться несовершенными. Однако они удовлетворяли потребностям команды разработчиков, которая их писала. Эти несовершенства находятся в допустимых пределах. В издательстве Addison-Wesley меня убедили убрать шероховатости, чтобы выделить корректный вид на фоне реального и адекватного представления. Надеюсь, для вас будет полезно посмотреть на примеры и распознать подлинные описания реальных проектов. Вы можете применить некоторые из моих правил к этим примерам, чтобы улучшить их. Так как совершенствовать кем-то написанное можно бесконечно, я принимаю возражения и любую критику.

Варианты использования в серии Crystal Collection

Это одна из книг из серии Crystal Collection для профессионалов в области программного обеспечения. Она посвящена простым и доступным методам разработки программных продуктов. Некоторые книги описывают какой-либо метод, другие — единственную роль в проектировании или вопросы взаимодействия в группе разработчиков.

В основу Crystal Collection положены два принципа:

- Разработка программного обеспечения — это кооперативная игра, построенная на изобретательстве и общении. Она совершенствует навыки разработчиков и повышает эффективность работы группы в целом.
- Разным проектам присущи разные потребности. Программные продукты отличаются характеристиками и разрабатываются различными по численности группами, состоящими из специалистов с неодинаковыми системами оценок и приоритетов. Невозможно указать лучший способ производства программного обеспечения.

В основополагающей книге серии Crystal Collection, Software Development as a Cooperative Game, детально разрабатываются идеи о проектировании программного обеспечения как о кооперативной игре и о методологии как о гармонизации культуры. В книге рассматриваются аспекты методологии, технологии и деятельности, рабочих продуктов и стандартов. Сущность дискуссии в необходимом для вариантов использования объеме изложена в разделе 1.2 данной книги.

Настоящая книга — это техническое руководство по основным элементам написания вариантов использования. Этот метод подходит почти для любого случая, однако шаблоны и стандарты следует выбирать в соответствии с потребностями конкретного проекта.

Благодарности

Я благодарен многим людям. Спасибо тем, кто просматривал эту книгу в черновиках и указывал на темы, которые были не совсем понятны. Особая благодарность Расселу Уолтерсу за его содействие и организацию обратной связи. Выражаю благодарность компаниям FirePond и Fireman's Fund Insurance Company за жизненные примеры вариантов использования. Спасибо Питу Мак-Брину, впервые испытавшему модель Участники и интересы. Он добавил к ней практический подход и внес предложения по улучшению. Благодарю Silicon Valley Patterns Group за внимательное прочтение первых вариантов книги и квалифицированные комментарии к различным документам и идеям. Спасибо Майку Джоунсу из Beans & Brew в Fort Union, придумавшему пиктограмму болта для варианта использования подсистемы.

Особая благодарность Сьюзен Лилли за тщательное прочтение и корректуру. Она внесла огромный вклад в совершенствование окончательной версии.

Благодарю других рецензентов, представивших подробные комментарии и одобривших работу: Пола Рэмни, Энди Полза, Мартина Фаулера, Карла Ваклавека, Алана Уильямса, Брайана Хендерсон-Селлерса, Ларри Константайна и Рассела Гоулда. Редакторы в Addison-Wesley проделали хорошую работу, выправив мои нескладные предложения и многочисленные опечатки.

Благодарю моих учащихся за помощь в совершенствовании идей этой книги.

Еще раз спасибо моей семье, Динне, Кэмерону, Шону и Кирану, а также сотрудникам из Beans & Brew в Fort Union, обеспечившим теплую атмосферу.

Дополнительную информацию о вариантах использования см. на web-сайтах, которые я поддерживаю: *members.aol.com/acockburn* и *www.usecases.org*. Во избежание возможных недоразумений скажу, что моя фамилия произносится Ко-о-берн, с долгим о.

Глава 1

Введение

Как выглядят варианты использования?

Почему каждой команде разработчиков необходим свой стиль описания вариантов использования?

Какое место занимают варианты использования в работе по формированию требований?

Как подготовиться к описанию вариантов использования?

Прежде чем вникать в подробности вариантов использования как таковых, следует ответить на эти вопросы.

1.1. Что такое варианты использования

Вариант использования фиксирует соглашение между участниками системы о ее поведении. Вариант использования описывает поведение системы при ее ответах на запрос одного из участников, называемого *основным действующим лицом*, в различных условиях. Основное действующее лицо инициирует взаимодействие с системой, чтобы добиться некоторой цели. Система отвечает, соблюдая интересы всех участников. Различные модели поведения, или сценарии, развертываются в зависимости от определенных запросов и условий, при которых делались эти запросы. Вариант использования собирает вместе эти сценарии.

Варианты использования представлены большей частью в текстовой форме, хотя возможны блок-схемы, циклограммы, сети Петри или языки программирования. При нормальных обстоятельствах они служат средством связи между лицами, часто не имеющими специальной подготовки. Поэтому простой текст обычно является наилучшим выбором.

Вариант использования как форма описания стимулирует обсуждение проектируемой системы в группе разработчиков. Одна команда разработчиков может с помощью варианта использования документировать действительные требования,

другая — окончательный проект. Все вышеперечисленное может применяться как для большой системы масштаба компании, так и для небольшой системы, например для фрагмента прикладной программы. Одинаковые основные правила создания вариантов использования применимы в любой ситуации, даже если документация имеет различные уровни детализации и технические подробности.

Когда варианты использования документируют бизнес-процессы организации, *рассматриваемая система* (SuD, system under discussion) и является этой организацией. Участники — это акционеры компании, заказчики, поставщики, органы государственного управления. Основные действующие лица — это заказчики компании и, возможно, их поставщики.

Если варианты использования описывают требования к поведению части программного обеспечения, SuD — это компьютерная программа. Участники — это пользователи программы, компания, владеющая ею, органы государственного управления и другие компьютерные программы. Основное действующее лицо — это сидящий за компьютером пользователь или другая компьютерная система.

Хорошо написанный вариант использования легко читается. Он состоит из предложений, написанных в единой грамматической форме (простых шагов действий), в результате которых действующее лицо достигает цели или передает информацию другому действующему лицу. Обучение чтению варианта использования занимает несколько минут.

Труднее научиться писать хорошие варианты использования. Для этого придется освоить три понятия, которые применяются к каждому предложению варианта использования и варианту использования в целом. Необходимо всегда иметь в виду эти понятия, что может оказаться непростой задачей. С трудностями вы столкнетесь, как только начнете писать первый вариант использования. Вот эти три понятия:

- *Область действия (Scope):* какова на самом деле рассматриваемая система?
- *Основное действующее лицо (Primary actor):* у кого есть цель?
- *Уровень (Level):* какой уровень имеет эта цель?

Далее следуют примеры вариантов использования. Части варианта использования описаны в следующей главе. А сейчас запомните определения:

- *Действующее лицо (Actor):* кто-то (или что-то), обладающий поведением.
- *Участник (Stakeholder):* кто-то (или что-то), проявляющий интерес к поведению рассматриваемой системы (SuD).
- *Основное действующее лицо (Primary actor):* участник (некто или нечто), инициирующий взаимодействие с SuD для достижения некоторой цели.
- *Вариант использования (Use case):* соглашение относительно поведения SuD.
- *Область действия (Scope):* идентифицирует рассматриваемую систему.
- *Предусловия и гарантии (Preconditions and guarantees):* то, что должно быть истинным до и после реализации варианта использования.

- *Основной сценарий (Main success scenario):* вариант, в котором не возникает никаких ошибок.
- *Расширения (Extensions):* различные отклонения от основного сценария.
- Номера расширений соответствуют номерам шагов основного сценария, в которых обнаруживаются данные отклонения (например, шаги 4a и 4b указывают на две отличные от основного сценария ситуации, которые могут возникнуть на шаге 4).
- Когда один вариант использования ссылается на другой, последний подчеркивается.

В приведенных ниже примерах первый вариант использования описывает лицо, собирающееся купить некоторые ценные бумаги через Сеть. Чтобы обозначить, что данную цель можно достигнуть за один сеанс, я помечаю вариант использования как находящийся на *уровне цели пользователя* и снабжаю его ярлыком в виде символа *уровня моря*, 🌊. Второй вариант использования описывает лицо, пытающееся получить компенсацию за автомобильную аварию. Эта цель требует более одного сеанса. Чтобы показать это, я помечаю вариант использования как находящийся на *обобщенном уровне* и снабжаю его ярлыком в виде другого символа — *над уровнем моря*, 🪁. Эти знаки объясняются в главе 5 и представлены на второй странице обложки.

Первый вариант использования описывает взаимодействие индивидуума с программой (PAF), выполняющейся на рабочей станции, подключенной к Интернету. Знак "черного ящика", 📦, указывает, что рассматриваемая система является компьютерной. Второй вариант использования описывает взаимодействие индивидуума с компанией, которую я обозначаю символом здания, 🏠. Использовать именно эти символы не обязательно, в то время как некоторые метки области действия и уровня в любом случае необходимы.

Ниже следуют варианты использования 1 и 2.

Вариант использования 1

📦 Покупка ценных бумаг через Интернет 🌊

Основное действующее лицо: покупатель

Область действия: персональные консультанты / финансовый пакет (PAF)

Уровень: цель пользователя

Участники и интересы:

Покупатель — хочет купить ценные бумаги, причем они должны автоматически попасть в портфель PAF.

Биржевое агентство — хочет получить полную информацию о покупке.

Предусловие: программа PAF у пользователя уже открыта.

Минимальные гарантии: в наличии достаточно регистрационной информации, чтобы PAF могла обнаружить несоответствие и запросить у пользователя дополнительные данные.

Гарантия успеха: удаленный web-сайт подтвердил покупку; регистрационные файлы и портфель пользователя обновлены.

Основной сценарий:

1. Покупатель выбирает покупку ценных бумаг через Сеть.
2. PAF получает от пользователя адрес нужного сайта (E*Trade, Schwab и т. д.).
3. PAF подключается к сайту, сохраняя контроль над процессом.
4. Покупатель выбирает и покупает ценные бумаги на данном сайте.
5. PAF перехватывает ответы web-сайта и обновляет портфель покупателя.
6. PAF показывает покупателю новое состояние портфеля.

Расширения:

2a. Покупатель запросил web-сайт, не поддерживаемый PAF.
  2a1. Система получает от покупателя новое предложение с возможностью отменить вариант использования.

3a. Отказ любого рода в Сети во время установки:
  3a1. Система сообщает покупателю о неудаче, дает совет и возвращается на предыдущий шаг.
  3a2. Покупатель либо отказывается продолжать этот вариант использования либо делает новую попытку.

4a. Во время транзакции покупки компьютер выходит из строя или выключается:
  4a1. (Что здесь нужно делать?)

4b. web-сайт не подтверждает покупку, а задерживает ее:
  4b1. PAF регистрирует задержку, устанавливает таймер, чтобы запросить покупателя о выходе.

5a. web-сайт не возвращает необходимую информацию о покупке:
  5a1. PAF регистрирует отсутствие информации о том, совершилось ли в портфеле покупателя обновление для зависшей покупки.

Вариант использования 2

Получить страховую компенсацию за автомобильную аварию

Основное действующее лицо: истец

Область действия: страховая компания (MyInsCo)

Уровень: обобщенный

Участники и интересы:

Истец — хочет получить максимально возможную сумму.

Компания MyInsCo — хочет заплатить как можно меньше.

Министерство страхования — хочет убедиться, что соблюдены все нормы и правила.

Предусловие: отсутствует.

Минимальные гарантии: MyInsCo регистрирует заявление и все действия.

Гарантия успеха: истец и MyInsCo приходят к соглашению о сумме страхового возмещения. Истец получает оговоренную сумму.

Триггер: истец представляет заявление на рассмотрение.

Основной сценарий:

1. Истец представляет на рассмотрение заявление с обоснованием.
2. Страховая компания проверяет законность страхового полиса истца.
3. Страховая компания назначает агента для расследования страхового случая.
4. Страховая компания проверяет, укладываются ли все детали в нормативы полиса.
5. Страховая компания выплачивает истцу страховое возмещение и закрывает дело.

Расширения:

1a. Предоставленные данные не полны:

1a1. Страховая компания запрашивает недостающую информацию.

1a2. Истец предоставляет недостающую информацию.

2a. Полис истца недействителен:

2a1. Страховая компания отклоняет заявление, извещает истца, документирует все действия и прекращает дело.

3a. На этот момент нет свободных агентов.

3a1. (Что в этом случае делает страховая компания?)

4a. Обстоятельства аварии противоречат основным правилам полиса:

4a1. Страховая компания отклоняет заявление, извещает истца, документирует все действия и прекращает дело.

4b. Обстоятельства аварии противоречат второстепенным правилам полиса:

4b1. Страховая компания начинает переговоры с истцом относительно суммы платежа.

Большинство вариантов использования взяты из реальных проектов, и я старался не подправлять их (кроме добавления ярлыков области действия и уровня, если их не было). Мне хочется, чтобы вы увидели практические примеры, а не созданные для классной комнаты. У людей редко хватает времени, чтобы написать варианты использования по форме, в полном объеме, да еще отредактировать их. Обычно их делают лишь "достаточными", а именно это и необходимо. Я показываю жизненные примеры, потому что создать совершенный вариант использования удается редко. Даже я не часто могу написать идеальный вариант использования.

Вариант использования 3 написал Торфин Аас из Центрального банка Норвегии для своего коллеги, представителя пользователя и для себя самого. Он демонстрирует, как можно изменить форму, сохранив ценность. Автор внес в документ дополни-

тельный бизнес-контекст, иллюстрируя работу приложения в течение рабочего дня. Это практично, так как избавляет от необходимости писать отдельный документ, описывающий бизнес-процесс. Это никого не запутало и увеличило информативность для заинтересованных лиц.

Вариант использования 3

Зарегистрировать привезенную коробку

П — приемщик

Р — регистратор

Основное действующее лицо: П

Область действия: программа регистрации ночных поступлений

Уровень: цель пользователя

Основной сценарий:

1. П получает и открывает коробку (идентификатор (ID) коробки, сумки с ID), полученную от транспортной компании (ТК).
2. П проверяет, совпадает ли ID коробки с зарегистрированными ID ТК.
3. П, возможно, подписывает бумагу посыльного.
4. П регистрирует поступление коробки в системе, которая хранит:
   ID П
   дату, время
   ID коробки
   название транспортной компании
   <Фамилию посыльного?>
   количество сумок (с ID сумок?)
   <оценку?>
5. П извлекает сумки из коробки, кладет в тележку, доставляет Р.

Расширения:

2a. ID коробки не соответствует ID транспортной компании.

4a. Срабатывает пожарная сигнализация и прерывает регистрацию.

4b. Выходит из строя компьютер.
   Оставьте деньги на столе и подождите, когда компьютер заработает.

Вариации:

4'. С персональным ID или без него.

4''. С оценкой или без оценки.

5'. Поставляет сумки в коробке.

1.2. У каждого свой вариант использования

Варианты использования — это вид документации, который можно использовать в работе в различных ситуациях, например, если требуется:

- Описать рабочий процесс в бизнесе.
- Сконцентрировать усилия на обсуждении принципиальных требований к разрабатываемой системе, а не на подробном их описании.
- Описать функциональные требования к системе.
- Документировать проект системы.
- Написать документ в небольшой компактной группе или в большой распределенной группе.

В каждой ситуации требуется свой стиль. Ниже приведены основные формы описания, диктуемые заданной *целью*.

- Компактная группа, собирающая требования, или более крупная группа, обсуждающая требования к разрабатываемой системе, пишет бессистемно, в противоположность вариантам использования, созданным по полной программе, которые принадлежат более многочисленной, географически распре деленной команде. *Бессистемная* (casual) форма в целях экономии времени упрощает шаблон варианта *использования* (подробнее см. ниже). Варианты использования 1-3 написаны *по полной форме* (fully dressed) с использованием полного шаблона варианта использования и схемы нумерации шагов. Вариант использования 4 — это пример бессистемной формы.
- Участникам бизнес-процессов варианты использования нужны, чтобы описать свои *деловые* операции, тогда как группа разработки аппаратного или программного обеспечения пишет системные варианты использования для своих требований. Команда разработчиков может писать и другие системные варианты использования, чтобы документировать проект или разбить требования на небольшие подсистемы.
- В зависимости от уровня, разработчик описывает многосеансовую, или *сложную* цель; односеансовую, или *пользовательскую* цель; часть цели пользователя, или *подфункцию*. Указывать, какой из этих уровней описывается, настолько важно, что мои студенты используют два способа их обозначения: с помощью высоты над уровнем моря (выше, на уровне, ниже) и цвета (белый, голубой, индиго).
- Лицо, формулирующее требования для проектируемой системы, компьютерной либо для бизнеса, пишет варианты использования типа "*черный ящик*". При этом внутренняя структура системы не обсуждается. Разработчики бизнес-процессов описывают варианты использования типа "*прозрачный ящик*", показывающие, как в компании или организации протекают внутренние процессы. Группа технической разработки может сделать то же самое, чтобы документировать эксплуатационный контекст системы, которую ей предстоит разработать, а также создать варианты использования типа "прозрачный ящик" для описания функционирования только что разработанной системы.

Замечательно, что форму описания варианта использования можно применять в столь различных ситуациях, но это и сбивает с толку. Несколько человек не придут к согласию по какому-нибудь вопросу описания просто потому, что их варианты использования имеют разные цели. И вы, вероятно, встретите со временем несколько комбинаций этих характеристик.

Обсуждая варианты использования, мы будем стараться найти общий язык, допуская в то же время многообразие в этом вопросе. Лучшее, что я могу сделать, это указать на проблему, а примеры будут говорить самим за себя.

Вы можете протестировать себя на предмет составления вариантов использования в этой главе. Варианты использования 1, 3 и 5 были написаны с целью создания требований к системе, поэтому они полностью соответствуют форме, имеют тип "черный ящик" для системы и уровень цели пользователя. Вариант использования 4 того же типа, но написан произвольно. Вариант использования 2 с контекстной установкой, предназначенный для документирования бизнес-процесса, написан по полной форме, имеет тип "черный ящик" и сложный уровень.

Самое большое различие между форматами вариантов использования заключается в их соответствии полной форме. Рассмотрим совершенно разные ситуации:

- Команда работает над программным обеспечением для большого, критически важного для организации проекта. Разработчики решили, что дополнительная процедура стоит затрат, поэтому (a) шаблон варианта использования должен быть длиннее и подробнее, (b) члены пишущей команды должны писать в одном стиле, чтобы сократить количество двусмысленных и непонятных мест, (c) варианты использования следует тщательно проверять на предмет пропусков и двусмысленностей. Учитывая низкую устойчивость к ошибкам, они также решили сократить допустимые отклонения при написании варианта использования.

- Команда из 3-5 человек разрабатывает систему, самая серьезная неисправность которой заключается в потере комфорта. С ней легко справиться, позвонив по телефону специалисту. Разработчики посчитали соблюдение формальностей напрасной тратой времеии, сил и денег. Поэтому они выбрали: (a) более простой шаблон, (b) более свободный стиль изложения, (c) меньшее количество менее строгих проверок. Обнаружение ошибок и упущений в тексте варианта использования перекладывается на другие механизмы проекта, возможно, на обсуждение среди коллег и пользователей. Разработчикам позволено допускать больше ошибок в письменном общении между собой, а потому менее формально подойти к созданию вариантов использования при большем разнообразии неформальных отношений между людьми.

И то, и другое допустимо. Такой выбор должен предоставляться для каждого проекта. Это наиболее важный урок, который я как методист усвоил за последние пять лет. Конечно, мы годами говорили, что не все можно мерить одной меркой. Но как конкретизировать это, осталось тайной.

Не нужно гнаться за точностью и строгостью, если в них нет большой необходимости — на это уйдет много времени и сил. Как написал Джим Сэвайер, участвуя в дискуссии по электронной почте, "...пока шаблоны не станут такими формальными,

что вы потеряетесь в рекурсивном спуске, который, как червоточины, пронизывает пространство проекта. Если это начнется, я велю разобрать эту сомнительную конструкцию до основания и стану рассказывать истории и писать на салфетках".

Я пришел к выводу, что недостаточно иметь только один шаблон. Должно быть по меньшей мере два: в свободном стиле для менее формальных проектов и строгой формы для более педантичного стиля проектирования. В любом проекте можно адаптировать одну из этих форм к соответствующей ситуации. Следующие два варианта использования описывают одно и то же, но в разных стилях.

Вариант использования 4

Совершить покупку (бессистемная версия)

Инициатор запроса делает запрос и направляет его своему Утверждающему лицу. Утверждающее лицо проверяет, есть ли деньги в бюджете, проверяет цены на товары, завершает оформление запроса для подачи и посылает его Покупателю. Покупатель просматривает содержимое памяти, отыскивая лучшего поставщика. Уполномоченное лицо подтверждает подлинность подписи Утверждающего лица. Покупатель формирует запрос на заказ, инициирует денежный перевод по почте Поставщику. Поставщик доставляет товары Приемщику, получает подтверждение доставки (вне сферы проектируемой системы). Приемщик регистрирует доставку, отсылает товары Инициатору запроса. Инициатор запроса отмечает, что заказ доставлен.

В любой момент, предшествующий получению товаров, Инициатор запроса может изменить или отменить запрос. При отмене запрос выводится из любой активной обработки (удаляется из системы?). Снижение цены оставляет его неизменным. При повышении цены он снова отсылается Утверждающему лицу.

Вариант использования 5

Совершить покупку (полная версия)

Основное действующее лицо: инициатор запроса

Цель в контексте: инициатор запроса покупает что-нибудь с помощью этой системы, получает заказ. Не включена оплата заказа.

Область действия: бизнес — общий механизм совершения покупки (электронный и обычный), как его представляют в компании.

Уровень: обобщенный

Участники и интересы:

Инициатор запроса: хочет получить то, что заказал, без особых хлопот.

Компания: позволяя сделать необходимые покупки, хочет проконтролировать расходы.

Поставщик: хочет, чтобы ему заплатили за все доставленные товары.

Предусловие: отсутствует.

Минимальные гарантии: каждый отосланный заказ одобрен законным уполномоченным лицом. Заказ отслеживается, чтобы компании присылали счета только за действительно доставленные товары.

Гарантии успеха: инициатор запроса получает товары, правильный бюджет готов к дебетованию.

Триггер: инициатор запроса решает что-то купить.

Основной сценарий:

1. *Инициатор запроса:* инициирует запрос.
2. *Утверждающее лицо:* проверяет наличие денег в бюджете, цены товаров, завершает оформление запроса для подачи.
3. *Покупатель:* просматривает содержимое памяти, ищет лучшего поставщика товаров.
4. *Уполномоченное лицо:* удостоверяет подпись утверждающего лица.
5. *Покупатель:* завершает формирование запроса для заказа, инициирует денежный перевод поставщику.
6. *Поставщик:* доставляет товары приемщику, получает квитанцию о доставке (вне сферы проектируемой системы).
7. *Приемщик:* регистрирует доставку, отправляет товары инициатору запроса.
8. *Инициатор запроса:* отмечает, что заказ доставлен.

Расширения:

1a. Инициатор запроса не знает поставщика или цены — пропустить эти части бланка и продолжить.

1b. В любой момент до приема товаров инициатор запроса может изменить или отменить запрос:

При отмене запроса прекращается его активная обработка (удалить из системы?).

Снижение цены не затрагивает обработку запроса.

При повышении цены запрос снова отправляется утверждающему лицу.

2a. Утверждающее лицо не знает поставщика или цену — оставить пропуск и дать покупателю его заполнить или забрать.

2b. Утверждающее лицо не является руководителем покупателя — не страшно, если утверждающее лицо подписывает запрос.

2c. Утверждающее лицо отклоняет запрос — вернуть инициатору запроса для изменения или аннулирования.

3a. Покупатель находит товары в памяти — отослать их, уменьшить запрос на эту величину и продолжить.

3b. Покупатель заполняет пропущенные поля поставщика и цены — запрос повторно посылается утверждающему лицу.

4a. Уполномоченное лицо отказывает утверждающему лицу — возвратить запрос покупателю и прекратить его активную обработку. (Что это значит?)

5a. Запрос включает несколько поставщиков — покупатель осуществляет несколько денежных переводов.

5b. Покупатель объединяет несколько запросов — тот же процесс, но следует пометить, к какому запросу относится каждый денежный перевод.

6a. Поставщик не доставляет заказ вовремя — система сигнализирует о том, что заказ не доставлен.

7a. Частичная доставка — приемщик отмечает частичную доставку в денежном переводе и продолжает.

7b. Частичная доставка по переводу на несколько запросов: — приемщик записывает количество полученных товаров для каждого запроса и продолжает.

8a. Товары не те или не того качества — инициатор запроса отказывается от доставленных товаров (Что это значит?).

8b. Инициатор запроса ушел из компании — покупатель решает с руководителем инициатора запроса, назначить ли другого инициатора или возвратить товары и отменить запрос.

Список изменений в технологии и данных: отсутствуют.

Приоритет: различный

Реализации: несколько

Время реакции: различное

Частота использования: 3 раза в день

Канал для Основного действующего лица: браузер Интернета, почтовая система или равноценный механизм

Второстепенные действующие лица: поставщик

Каналы для второстепенных действующих лиц: факс, телефон, автомобиль

Открытые вопросы:

Когда отмененный запрос удаляется из системы?

Чье разрешение нужно, чтобы отменить запрос?

Кто может изменить содержание запроса?

В каком объеме должна поддерживаться история изменений для запросов?

Что происходит, когда инициатор запроса отказывается от доставленных товаров?

В чем отличие требования от заказа?

Как при размещении заказов происходит обращение к внутренней памяти и ее использование?

Нельзя просто сказать, что мы пишем варианты использования для этого проекта. Любая рекомендация или определение процесса, типа "Напишите варианты использования", будет недостаточной. Вариант использования, достоверный для одного проекта, не применим к другому. Кроме того, следует указать, применяются ли варианты использования по полной или бессистемной форме, какие части шаблона и форматы обязательны и каковы пределы допустимого отклонения разработчиков от формы.

Всестороннее обсуждение возможных отклонений и вариаций в проектах содержится в книге *Software Development as a Cooperative Game*, упоминавшейся в предисловии. Чтобы научиться писать варианты использования, не требуется подробное обсуждение. Что действительно необходимо, так это различать метод написания, качество варианта использования и стандарты проекта.

Метод — это последовательность размышлений или действий, которые люди используют при построении вариантов использования. Эта книга в значительной степени касается метода: как думать, как строить предложения, в какой последовательности работать. Методы хороши тем, что в основном не зависят от размера проекта. Овладевший методом специалист способен применять его как к малым, так и к большим проектам.

Качество говорит о том, насколько точно написанные варианты использования соответствуют их цели. В этой книге я описываю лучший способ, который я знаю, для каждой части варианта использования, для варианта использования в целом и для разных целей. В конечном счете, однако, способ, которым вы оцениваете качество ваших вариантов использования, зависит от вашей цели, допустимых пределов и степени следования форме.

В стандартах записано, о чем договорились разработчики, создавая варианты использования. В этой книге я рассматриваю альтернативные приемлемые стандарты, показывая различные шаблоны и разные стили предложений и заголовков. Я предлагаю несколько конкретных рекомендаций, но, в конце концов, дело организации или разработчиков проекта устанавливать или адаптировать стандарты и решать, насколько строго следовать им.

В большей части этой книги я рассматриваю насущный вопрос: разработку точных требований. Далее консультант Стив Эдолф описывает применение вариантов использования скорее для выявления требований, чем для их документирования.

▪ Стив Эдолф:

"Выявление" требований на новой территории

Варианты использования обычно предлагают способ сбора и моделирования известных функциональных требований. Считается, что повествователь- ная форма легче для понимания, чем традиционные длинные списки требований. Тогда действительно понятно, что должна делать система.

Но что, если никто не знает, что должна делать система? Автоматизация процесса обычно изменяет сам процесс. В полиграфической промышленности недавно произошла одна из самых больших перемен со времен изобретения офсетной печати: появилась технология "непосредственно на печатную форму/непосредственно на пресс". Раньше установка печатного пресса была трудоемким, многошаговым процессом. Новая технология сделала промышленную полиграфию таким же простым делом, как распечатка документа, подготовленного в текстовом процессоре.

Как бы вы в качестве аналитика, ответственного за управление технологическим процессом, собирали требования для системы на основе такой совершенно новой технологии, как полиграфический метод "непосредственно на печатную форму"? Для начала вы могли бы описать использование существующей системы и определить действующих лиц и службы новой системы. Однако в результате вы получили бы лишь существующую систему. Никто еще не делал новую работу, так что все специалисты в этой области изучают новую систему одновременно с вами. Вы параллельно разрабатываете новый процесс и новое программное обеспечение. Остается только пожелать вам удачи. Как вы действуете в этой ситуации? Берете существующую модель и спрашиваете себя, что же меняется? Ответом вполне может быть: "Все".

Когда вы пишете варианты использования, чтобы документировать требования, у кого-то уже сложилось представление о системе. Вы просто выражаете это представление так, чтобы каждому было понятно. Однако при выявлении требований вы сами создаете представление о системе.

Применяйте варианты использования в качестве инструмента "мозгового штурма". Спросите себя, какие шаги в этих вариантах использования при новой технологии теряют смысл. Создайте новое повествование о том, как действующие лица достигают своих целей. Цели пока те же самые, но некоторые второстепенные действующие лица ушли или изменились.

Используйте метод "погрузиться и всплыть". Создайте широкую модель высокого уровня, описывающую ваше представление о том, как могла бы работать новая система. Стремитесь к простоте, поскольку это неисследованная территория. Придумайте, как мог бы выглядеть основной сценарий. Проиграйте его со специалистами по старой системе.

Затем погрузитесь в подробности одного варианта использования. Рассмотрите альтернативы. Учтите, что людям проще понять рассказ, и отложите формирование требований. Прочитайте шаг в варианте использования и задайтесь вопросом, что происходит, когда клиент предпочитает твердую, а не цифровую копию корректуры. Это проще, чем пытаться построить полную мысленную модель работы системы.

Наконец, вернитесь на поверхность. Что изменилось, после того как вы погрузились в детали? Настройте модель, затем повторите погружение для другого варианта использования.

Мой опыт показал, что применение вариантов использования для выявления требований приводит к созданию функциональных требований более высокого качества. Они лучше организованы и более полны.

1.3. Требования и варианты использования

Если вы пишете варианты использования как требования, имейте в виду:

- *Это действительно требования.* Не следует превращать их в другую форму требований к поведению системы. Написанные надлежащим образом, они точно описывают, что должна делать система.

- *Это не все требования.* Они не детализируют внешние интерфейсы, форматы данных, бизнес-правила и сложные формулы. Они устанавливают только часть (возможно, треть) всех требований, которые вам необходимо собрать, очень важную, но лишь часть.

Каждая организация занимается сбором и систематизацией требований в соответствии со своими нуждами. Существуют даже стандарты описания требований. В каждом из них варианты использования занимают лишь часть общего множества документированных требований.

Следующая схема требований представляется мне весьма полезной. Я взял ее из шаблона, который Сьюзен Робертсон и консорциум Atlantic Systems Guild опубликовали на своем web-сайте и в книге *Managing Requirements* (1999). Их шаблон впечатляет своей полнотой, так что я урезал его до приведенной ниже схемы, которую использую как руководство. Однако она слишком длинна для большинства проектов, с которыми я сталкиваюсь, поэтому я стараюсь сокращать ее и далее. Каков бы ни был ее размер, она задает много интересных вопросов, которые в другом случае не возникли бы, например: "Как влияет человеческий потенциал на сбои системы? Какие политические соображения управляют каждым требованием?"

В задачи этой книги не входит полная стандартизация ваших требований, но я знаю, что многие никогда не видели схему требований. Я объясню, что это такое. Ее главная цель — показать место вариантов использования в общей схеме требований и подчеркнуть, что варианты использования не содержат всех требований, а только описывают часть поведения, требуемую функцию.

Приемлемая схема требований

Раздел 1. Цель и область действия
- 1a. Что представляют собой общая область действия и цель?
- 1b. Участники (Кого это интересует?)
- 1c. Что входит в область действия и что нет?

Раздел 2. Используемые термины/Глоссарий

Раздел 3. Варианты использования
- 3a. Основные действующие лица и их общие цели
- 3b. Варианты использования для бизнес-процессов
- 3c. Системные варианты использования

Раздел 4. Используемая технология
- 4a. Какие технологические требования предъявляются к данной системе?
- 4b. С какими системами будет взаимодействовать данная, каковы требования?

Раздел 5. Другие требования
- 5a. Процесс разработки

Q1. Кто участвует в проекте?
Q2. Какие оценки проекта будут отражены (простой, ранний, быстрый или гибкий)?
Q3. Какая обратная связь или прозрачность проекта нужна пользователям и организаторам?
Q4. Что мы можем купить, что должны построить, с кем конкурируем?
Q5. Какие еще существуют технологические требования (тестирование, установка и т.д.)?
Q6. От чего зависит развитие проекта?

5b. Бизнес-правила
5c. Производительность
5d. Эксплуатация, безопасность, документация
5e. Использование (простота использования)
5f. Сопровождение и мобильность
5g. Нерешенные или отложенные вопросы

Раздел 6. Людские резервы, правовые, политические, организационные вопросы

Q1. Как влияют людские резервы на функционирование системы?
Q2. Какие существуют правовые и политические требования?
Q3. Какие последствия для людей будет иметь создание этой системы?
Q4. Каковы требования к обучению?
Q5. Какое влияние оказывает система на окружающее сообщество?

Вариантам использования посвящен только Раздел 3 требований. Это не вообще все требования, а только требования к поведению. Бизнес-правила, глоссарий, производительность, технологические требования и многое другое не попадает в категорию поведения. Для этих аспектов нужны собственные разделы (см. рис. 1.1).

Варианты использования как структура связей проекта

Варианты использования связаны со многими другими деталями требований. Они помогают связать информацию в различных частях требований, включая сведения о профиле пользователя, бизнес-правила и требования к форматам данных.

Вне документа о требованиях варианты использования помогают систематизировать информацию о планировании проекта, такую как даты выпуска, команды разработчиков, приоритеты и состояние разработки. Кроме того, они служат для отслеживания определенных результатов деятельности команды разработчиков, в частности интерфейса пользователя и системных тестов.

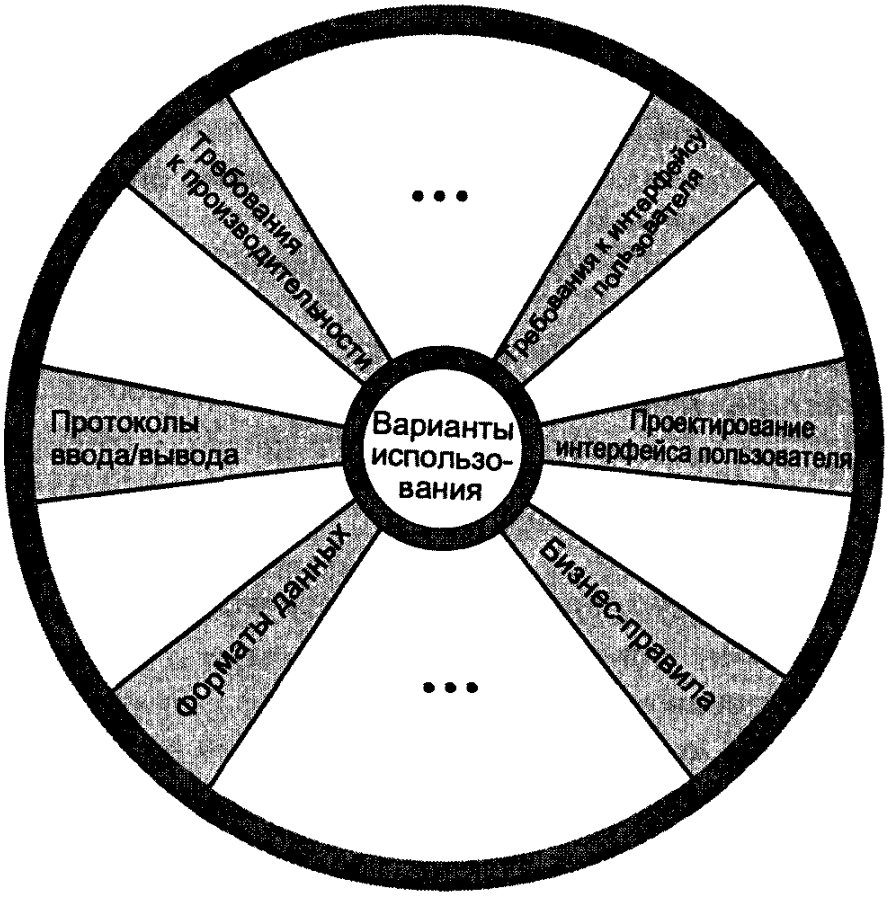


Рис. 1.1. *Модель требований в виде колеса*

Хотя эти требования и не входят в варианты использования, все они с ними связаны. Варианты использования играют роль втулки колеса (см. рис. 1.1), а другие виды информации выглядят как спицы, направленные в разные стороны. Именно по этой причине варианты использования представляются людям центральным узлом требований или даже центральным узлом процесса разработки системы.

1.4. Когда варианты использования повышают ценность проекта

Варианты использования популярны прежде всего потому, что связно рассказывают о поведении системы при ее использовании. Пользователи системы могут увидеть, что это будет за система. Они получают возможность на ранней стадии влиять на точную настройку. В этом состоит значимость вариантов использования для проекта, но это лишь один пример.

Во-первых, варианты использования являются особенно ценными, когда именуются целями пользователя, которые будет реализовывать система, и собираются в список. Этот список объявляет, что будет делать система, раскрывая ее область применения и цели. Он становится средством связи между различными участниками проекта.

Список целей просматривается представителями пользователя, должностными лицами, специалистами-разработчиками и руководителями проекта, оценивающими стоимость и сложность системы, разработка которой начинается с этого списка. Они

обсуждают, какие функции реализовать в первую очередь и как сформировать команды. Список — это каркас, на который нанизываются сложность, стоимость, сроки и состояние проекта. Он накапливает разнообразную информацию в течение всего периода разработки.

Во-вторых, варианты использования важны, когда их авторы выявляют методом "мозгового штурма" все факты, которые могли бы вызвать сбой в успешном сценарии, перечисляют их и начинают документировать реакцию системы на них. В этот момент разработчики, возможно, обнаруживают что-то удивительное, что-то, о чем они или те, кто предоставил им требования, и не думали.

Когда мне надоедает писать варианты использования, я переключаюсь на условия отказа. Здесь я регулярно открываю нового участника, систему, цель или бизнес-правило, пока документирую обработку отказа. Когда мы определяем, что делать при возникновении одного из этих условий, я часто представляю себе экспертов по бизнесу, сбившихся в кучу или говорящих по телефону, чтобы решить, что в этих обстоятельствах должна делать система.

Без дискретизации шагов вариантов использования и обсуждения возможных отказов в режиме "мозгового штурма" многие условия ошибок остаются необнаруженными до тех пор, пока какой-нибудь программист не наткнется на них, кодируя фрагмент программы. При этом уже слишком поздно открывать новые функции и бизнес-правила. Эксперты по бизнесу, как правило, уже отключились от проекта, и время поджимает, так что программисты кодируют то, что им придет в голову в этот момент, вместо того чтобы выяснить, какое поведение требуется.

Те, кто пишет варианты использования длиной в один абзац, экономят время и наслаждаются преимуществами вариантов использования. Те же, кто упорно занимается обработкой отказов, тоже берегут время, обнаруживая тонкие моменты в поведении системы на ранней стадии проектирования.

1.5. Дозируйте свою энергию

Берегите свои силы или по крайней мере управляйте ими. Если вы стремитесь описать все детали за один раз, вы не успеете раскрыть все разделы за отведенное время. Если вы записываете только схему, чтобы начать и затем описать сущность каждого варианта использования, вы можете:

- Дать участникам шанс внести необходимые коррективы и понять приоритеты на ранней стадии.
- Позволить распределить работу между несколькими группами, увеличивая параллелизм и производительность.

Часто говорят: "Дайте мне обзор с расстояния 15 км", "Дайте мне только схему" или "Детали добавим позднее". Это означает: "Пишите пока не слишком подробно; детали добавим позже".

Точность определяется тем, как много вы сумели сказать. Когда вы утверждаете: "Заказчик захочет взять напрокат видео", вы не тратите слишком много слов, но действительно сообщаете важную информацию читателям. Когда вы показываете

список всех целей, которые будет поддерживать проектируемая система, вы предоставляете участникам огромное количество информации, содержащееся в небольшом количестве слов.

Точность — это не то же самое, что правильность. Если кто-то говорит вам: "Число Пи равно 4,141592", он выдает высокую точность, однако далек от правильности. Если вам говорят: "Пи равно 3", точность здесь невелика (маловато цифр), но это правильно в определенных пределах. Та же идея справедлива для вариантов использования.

Вы можете добавить массу деталей к каждому варианту использования, повысив его точность. Однако если первоначальная формулировка целей содержит ошибки, усилия на детализацию их описания будут потрачены впустую. Лучше сделать корректным список целей, а не тратить время и силы на детальное описание вариантов использования.

Усилия, потраченные на описание вариантов использования, можно разделить на четыре стадии точности:

1. *Действующие лица и цели.* Перечислите действующих лиц и каждую из их целей, которые будет обеспечивать система. Проанализируйте правильность и полноту этого списка. Расставьте приоритеты и распределите цели между командами разработчиков и версиями. Теперь у вас есть функциональные требования к первому уровню точности.

2. *Краткое изложение варианта использования или основной сценарий.* Для вариантов использования вы выбрали следование основному сценарию в общих чертах. Рассмотрите его в форме набросков, чтобы быть уверенным, что система действительно удовлетворяет интересам участников, о которых вы заботитесь. Это второй уровень точности функциональных требований. Этот материал довольно легко набрать, в отличие от двух следующих уровней.

3. *Условия отказа.* Завершите основной сценарий и продумайте, какие отказы могут произойти. Занесите их в список, прежде чем решать, как система должна их обрабатывать. Описание процессов обработки отказов более трудоемкая работа, чем перечисление отказов. Те, кто начинает описывать обработку отказов немедленно, часто выдыхаются, не закончив списка условий отказов.

4. *Обработка отказа.* Напишите, как система должна реагировать на каждый отказ. Это часто непростой, утомительный и непредсказуемый процесс. Непредсказуемый потому, что довольно часто вопрос о малопонятном бизнес-правиле всплывает во время работы над описанием, или при обработке отказа неожиданно обнаружится новое действующее лицо или новая цель, достижение которой должна обеспечить система.

Большая часть проектов требует немного времени и усилий. Поэтому управление уровнем точности, к которому вы стремитесь, должно быть приоритетом проекта. Я настоятельно рекомендую работать в описанном выше порядке.

1.6. Разминка с помощью повествовательного стиля

Описание использования системы в *повествовательном* (narrative) стиле — это пример варианта использования в действии при определенных обстоятельствах. Это единичный, очень специфический пример того, как действующее лицо использует систему. Это даже не вариант использования, и в большинстве проектов он не доживает до официального документа "Функциональные требования". Однако это очень полезный метод, который стоит того, чтобы вы о нем прочитали.

Начиная новый проект, вы можете не иметь опыта в написании вариантов использования или, возможно, вы не продумали детали функционирования системы. Чтобы освоиться с материалом, составьте краткий отчет об одном дне жизни одного из действующих лиц.

Придумайте конкретное действующее лицо и кратко опишите, почему он хочет именно это или какие условия заставляют его действовать именно так. Как и при создании всех вариантов использования, много писать не надо. Удивительно, сколько информации можно передать с помощью всего нескольких слов. Напишите, как все работает в этом отдельном случае от начала до конца ситуации.

Важна краткость, читатель должен сразу ухватить суть. Подробности и причины или эмоциональное содержание имеют значение. Каждый читатель — от лнца, утверждающего требования, до разработчика программного обеспечения, проектировщика тестов и создателя учебных материалов — должен понять, как следует оптимизировать систему, чтобы повысить ее ценность для пользователя.

Такое описание не требует много усилий и места и вводит читателя в сам вариант использования легко и незаметно. Вот пример.

Описание использования: быстрое получение наличных денег

Мэри с двумя дочками по пути на работу заезжает в центр медицинской помощи, подъезжает к банкомату, пропускает карточку через считывающее устройство, вводит свой PIN-код, выбирает режим FAST CASH (быстрое получение наличных денег) и вводит сумму, $35. Банкомат выдает банкноту в $20 и три по $5, а также квитанцию, показывающую остаток на ее счете после дебетования $35. После каждой транзакции FAST CASH банкомат восстанавливает экран, так что Мэри может не беспокоиться о том, что следующий пользователь получит доступ к ее счету. Мэри нравится FAST CASH, поскольку он не задает лишних вопросов, затягивающих общение. Она пользуется этим банкоматом, так как он выдает банкноты в $5, которые она использует для платы за медицинскую помощь, к тому же для получения наличности ей не приходится выходить из машины.

Описания использования нужны, чтобы понять, как работать с системой. Писать их полезно и в качестве разминки для проработки деталей перед созданием варианта использования. Иногда группа разработчиков публикует описания в начале всех вариантов использования или только в начале конкретных вариантов использования, которые они иллюстрируют. Одна группа собирала пользователей, аналитиков и

специалистов по созданию требований и анимировала описание использования, чтобы легче было представить систему и выстроить общее видение ее использования.

Описание использования — это не требования, скорее это ступень для создания более продуманных и обобщенных описаний требований. Описание использования служит отправной точкой для создания варианта использования. Сам по себе вариант использования — это сухой остаток описания использования, формула с обобщенными именами лиц, действующих в описании использования.

1.7. Упражнения

Файл требований

1.1. Какие разделы файла требований зависят от вариантов использования, какие нет? Обсудите это с коллегой и подумайте, почему ваши ответы различны.

1.2. Сделайте набросок других приемлемых требований, который можно распространить по корпоративной сети интранет. Уделите внимание структуре вашего подкаталога и соглашениям о календарных штампах (зачем вам понадобятся такие соглашения?).

Описание использования

1.3. Создайте два описания использования для банкомата, которым вы пользуетесь. Чем и почему они отличаются от примера описания, приведенного выше? Насколько важны эти различия для разработчиков, которые будут создавать систему?

1.4. Создайте описание использования для лица, идущего в новый фирменный салон видеопроката, чтобы взять напрокат оригинал фильма *The Parent Trap*.

1.5. Создайте описание использования для вашего текущего проекта. Попросите коллегу написать аналогичный документ. Сравните и обсудите записи. Почему они различаются, находятся ли эти различия в пределах допустимого или они серьезны?

ЧАСТЬ 1

Составные части варианта использования

Глава 2

Вариант использования как соглашение о поведении

Разрабатываемая система — это механизм для выполнения соглашения между участниками. Варианты использования обеспечивают поведенческую часть этого соглашения. Каждое предложение в варианте использования описывает действие, защищающее некоторый интерес какого-либо участника. Предложение может описывать взаимодействие двух действующих лиц или внутренние действия системы, которые она должна выполнять для защиты интересов участников. Сначала рассмотрим вариант использования как способ фиксации взаимодействия участников, имеющих определенные цели. Отталкиваясь от этого положения, мы можем расширить обсуждение, изучая вариант использования как соглашение между участниками. Я называю первую часть концептуальной моделью Действующие лица и цели, а вторую — концептуальной моделью Участники и интересы.

2.1. Взаимодействия действующих лиц, имеющих цели

Действующие лица имеют цели

Представьте себе служащего, ответственного за прием запросов на обслуживание по телефону (на рис. 2.1 служащий представлен основным действующим лицом). Когда поступает звонок, цель служащего —открыть компьютерный журнал и инициировать запрос.

Обязанность системы — зарегистрировать и инициировать запрос на обслуживание в нашем примере. (В действительности она отвечает за защиту интересов всех участников, причем служащий, основное действующее лицо, только один из них. Сосредоточимся на обязанности системы обеспечить обслуживание для основного действующего лица.)

Чтобы выполнить свою обязанность, система формулирует подцели. Некоторые подцели она способна реализовывать внутри себя, но для других ей необходима помощь *вспомогательного действующего лица*. Этим вспомогательным действующим лицом может быть подсистема печати или другая организация, например партнерская компания или правительственная организация.

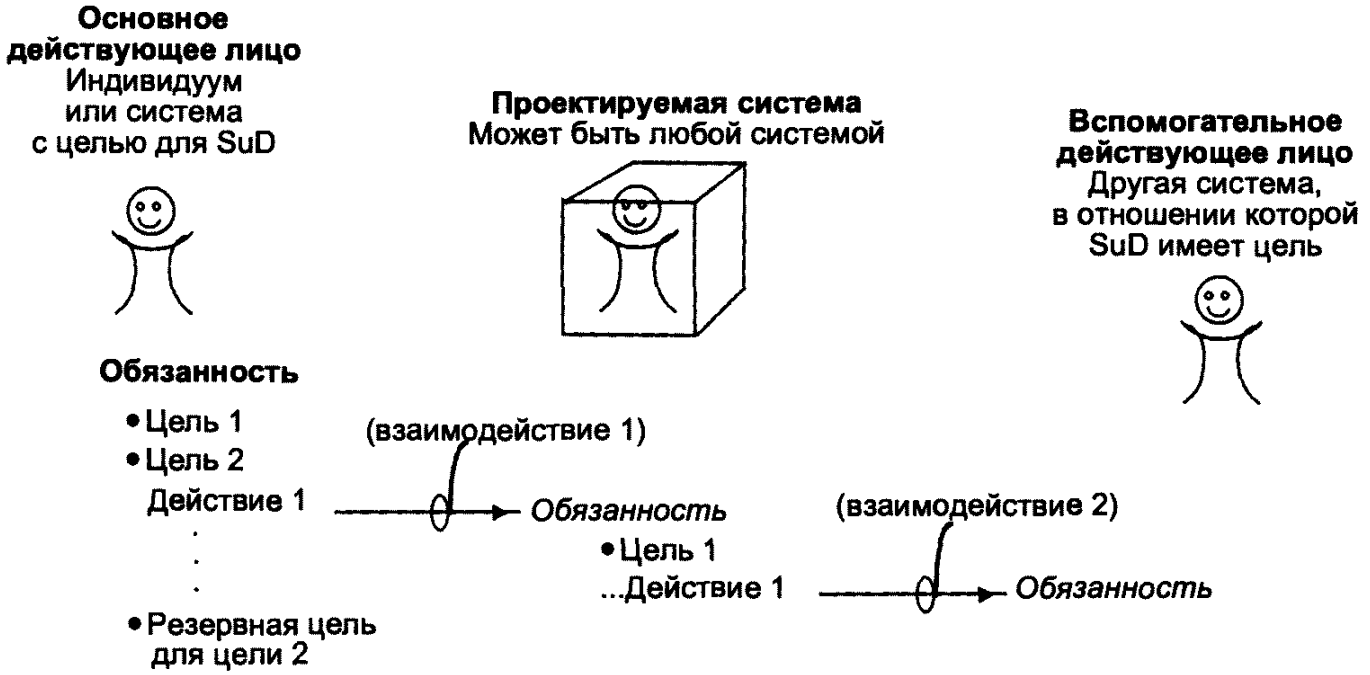


Рис. 2.1. *Действующее лицо, имеющее цель, инициирует выполнение обязанностей другого действующего лица*

Вспомогательное действующее лицо обычно выполняет свое обещание и добивается подцели для разрабатываемой системы (SuD). SuD взаимодействует с внешними действующими лицами. Она достигает своих подцелей через некоторую последовательность действий, пока не выполнит своей обязанности, т.е. не окажет обещанную услугу.

Предоставление обещанной услуги — высшая цель, достигаемая через подцели. Подцели можно, в свою очередь, разбить на подподцели и т.д. до бесконечности. Подпод(...под)целей может быть бесконечное множество, если мы разбиваем действия действующих лиц на мелкие составляющие. В стихотворении "Похвала поэзии" Джонатан Свифт говорит о блохе, на которой натуралисты нашли блох еще меньшего размера, которые, в свою очередь, имеют еще более мелких блох и т.д. Предела этому нет.

Вероятно, наиболее трудная часть создания хороших вариантов использования — это управление блохами на блохах, т.е. подподцелями. Подробнее об этом см. в главе 5, в главе 7 (правило 6) и в главе 20 (памятка 6).

Концептуальная модель Действующие лица и цели удобна, поскольку применима и к бизнесу, и к компьютерным системам. Действующими лицами может быть человек, организация или компьютер. Мы можем описать смешанные системы, включающие людей, компании и компьютеры. Мы можем описать программную систему, управляе-

мую другой компьютерной системой, которая обращается к одушевленному вспомогательному действующему лицу, или организацию, которая обращается к компьютерной системе либо индивидууму. Это полезная обобщенная модель.

Целей можно и не достигнуть

Что, по-вашему, будет делать служащий, разговаривающий с клиентом по телефону, если компьютер отказывает во время записи запроса? Если система не может предоставить обещанную услугу, служащий должен придумать резервную цель, в этом случае, вероятно, с помощью карандаша и бумаги. Сохраняя за собой основную обязанность по работе, служащий должен иметь план на случай отказа системы.

Точно так же система может столкнуться с отказом на пути к одной из своей подцелей. Возможно, основное действующее лицо дало неверные данные, может быть, это внутренняя неисправность или вспомогательное действующее лицо не предоставило обещанную услугу. Как же должна вести себя система? Это действительно интересная часть требований к поведению SuD.

Иногда система может устранить неисправность и вернуться к нормальному поведению. В других случаях она должна просто отказаться от этой цели. Если вы идете к своему банкомату и пытаетесь извлечь больше денег, чем есть у вас на счету, ваша цель выудить наличные просто не будет достигнута. Ничего у вас не выйдет и при отключении банкомата от сети. Но если вы всего лишь ошиблись, набирая код, система предоставит вам вторую попытку ввести код правильно.

Такая фокусировка на неудачах в достижении целей и реакциях на неудачи объясняет, почему варианты использования являются хорошим описанием поведения системы и отличными функциональными требованиями в целом. Специалисты по функциональной декомпозиции и декомпозиции потоков данных отметили это как наиболее значительное преимущество, которое дали им варианты использования.

Взаимодействие имеет составную структуру

Простейшее взаимодействие — это отправление сообщения: "Привет, Джин, — говорю я, проходя по холлу. Это и есть простое взаимодействие. В процедурном программировании простому взаимодействию соответствует вызов функции, например print (значение). В объектно-ориентированном программировании один объект посылает сообщение другому, скажем, **objectA->print(значение)**.

Последовательность сообщений, или *сценарий* — это составное взаимодействие. Предположим, я иду к автомату с газированной водой, опускаю в него долларовую банкноту за 80-центовый напиток и получаю сообщение, что нужно опустить точную сумму. Вот мое взаимодействие с автоматом:

1. **Я опускаю долларовую банкноту.**
2. **Нажимаю кнопку "cola".**
3. **Автомат требует точную сумму мелочью.**
4. **Я нажимаю на "Возврат монет".**
5. **Автомат возвращает доллар монетами.**
6. **Забираю монеты и ухожу.**

Можно сжать последовательность, как если бы это был единственный шаг ("Я попытался купить колу в автомате, но он потребовал точную сумму"), и развернуть этот сжатый шаг в более длинную последовательность:

1. **Я пошел в банк компании и взял некоторую сумму.**
2. **Попытался купить колу в автомате, но тот потребовал точную сумму.**
3. **Прошел до кафетерия и купил воду там.**

Таким образом, при необходимости взаимодействие можно свернуть или, напротив, разбить на шаги, как позволяют цели. Каждый шаг сценария фиксирует цель, и поэтому каждый шаг можно развернуть как отдельный вариант использования. Так что взаимодействия напоминают блох, у которых есть блохи, да и цели тоже.

Удобно то, что поведение системы можно представить на весьма высоком уровне обобщения, когда цели и взаимодействия находятся в свернутом виде. Постепенно развертывая их, можно описать поведение системы сколь угодно точно. Я часто называю набор вариантов использования постоянно разворачивающейся историей. Наша задача — писать эту историю так, чтобы читатель легко в ней ориентировался.

Я использую слово последовательность в широком смысле. Во многих случаях взаимодействия не обязательно должны происходить в каком-то определенном порядке. Чтобы купить воду за 80 центов, я мог бы опустить восемь монет по 10 центов, три монеты по 25 центов и монету в 5 центов либо... (можете продолжить список). Не имеет значения, какую монету опускать первой.

Строго говоря, последовательность — не совсем подходящее слово. Правильный математический термин — *"частичное упорядочение"*. Однако слово последовательность точнее по смыслу и гораздо легче воспринимается теми, кто пишет варианты использования. Если кто-нибудь спросит вас о сообщениях, которые могут поступать одновременно, попросите кратко написать об этом. Посмотрите, с чем они к вам придут. Мой опыт показывает, что люди создают удивительно ясные описания после весьма непродолжительного обучения. Поэтому я продолжаю употреблять слово последовательность. Пример сложных последовательностей вы найдете в варианте использования 22.

С формальным языком для вариантов использования дело обстоит не так просто. Большинство разработчиков языков либо заставляют пишущего варианты использования перечислять все возможные порядки, либо изобретают сложные нотации для произвольного порядка событий. Мы создаем варианты использования для людей, а не для компьютеров, поэтому просто пишем: "Покупатель опускает 80 центов пятицентовыми, десятицентовыми или двадцатипятицентовыми монетами в любом порядке".

Последовательности хороши для описания взаимодействий в прошлом, поскольку прошлое полностью определено. Чтобы описать взаимодействия в будущем, необходимы наборы возможных последовательностей, по одной для каждого условия, которое может когда-либо возникнуть. Если я вам рассказываю о том, как вчера просил повышения, я говорю:

> "Вчера я имел серьезный разговор с боссом. Я сказал... Она сказала... Я сказал...", и т.д.

Но о будущем я расскажу так:

"Меня действительно беспокоит предстоящий разговор с боссом".

"Почему?"

"Я собираюсь просить повышения".

"Как?"

"Ну, сначала я скажу... Затем, если она скажет..., я отвечу...".

"Но если она скажет..., я попробую...", и т.д.

Точно так же, объясняя кому-либо, как купить содовой воды, мы скажем:

"Прежде всего, запаситесь монетами".

"Если у вас есть точная сумма, опустите монеты в автомат и нажмите кнопку "cola".

"Если нет, опустите деньги и посмотрите, сможет ли автомат дать сдачи. Если да..."

Чтобы описать взаимодействие в будущем, придется иметь дело с различными условиями, создавая наборы последовательностей. Для каждой последовательности мы указываем условие, саму последовательность и результат.

Можно спрятать набор последовательностей в одном предложении: "Сначала пойдите купить воды в автомате". Или: "Затем попросите у босса повышение". Как и с последовательностями, эти предложения можно вместить в краткое описание высокого уровня обобщения или развернуть их в подробное описание в соответствии с нашими потребностями.

Теперь вы знаете, что вариант использования содержит набор возможных сценариев для достижения цели. Для завершенности добавим:

- Все взаимодействия имеют отношение к одной и той же цели одного и того же основного действующего лица.
- Вариант использования начинается с инициирующего события и продолжается, пока цель не достигнута или ее достиженне не прервано и система не освобождается от своих обязательств по отношению к данному взаимодействию.

Вариант использования как сборник сценариев

У основного действующего лица есть цель; система должна помогать действующему лицу достичь этой цели. Некоторые сценарии приводят к достижению заданной цели; другие заканчиваются, если цель оказывается недостижимой. Каждый сценарий содержит последовательность шагов, показывающих, как раскрываются действия и взаимодействия. Вариант использования собирает все эти сценарии вместе, показывая все пути, которые могут привести или не привести к цели.

Рассмотрим "полосатые брюки" (рис. 2.2). Ремень на брюках указывает цель, которая объединяет все сценарии. Из двух половинок брюк одна предназначена для

сценариев, которые приводят к успеху, а другая — для сценариев, которые кончаются неудачей.

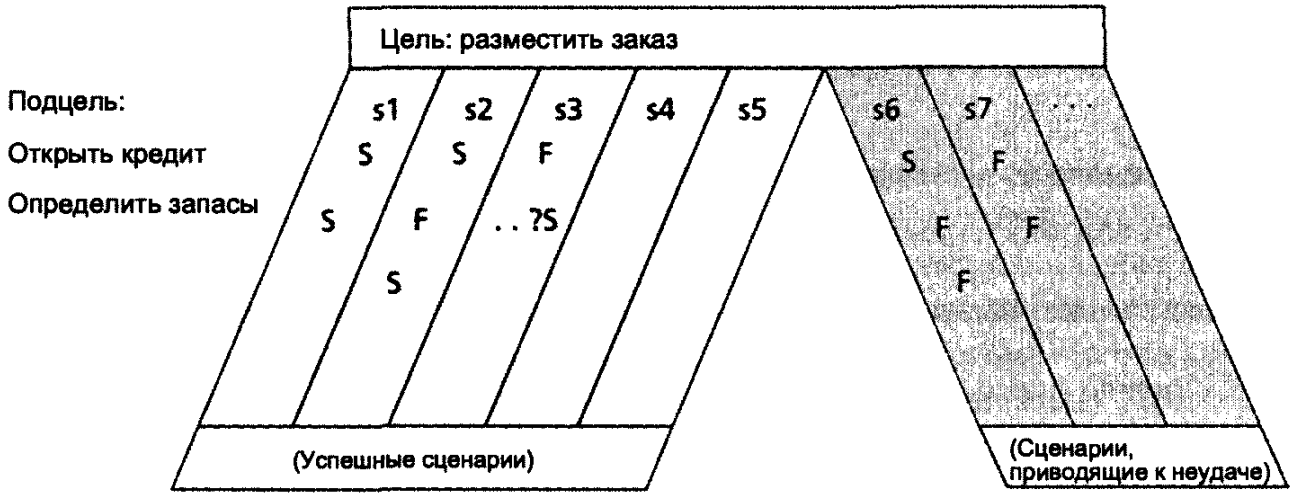


Рис. 2.2. *"Полосатые брюки" — сценарии приводят или не приводят к цели*

Каждая полоса соответствует одному сценарию и находится в части успехов или в части неудач. Назовем первую полосу на половинке успехов основным сценарием. Остальные полосы — это другие сценарии, в конце концов приводящие к успеху, некоторые альтернативными путями, другие — справившись с промежуточным отказом. Каждая полоса на половинке неудач — это сценарий, в котором возникает сбой, после которого он, возможно, восстановится, но завершится все-таки неудачей.

На самом деле мы не будем отдельно писать каждый сценарий от начала до конца. Это нерациональная стратегия, поскольку это утомительно и сложно поддерживать. "Полосатые брюки" помогают понять, что каждый вариант использования имеет два исхода, что цель основного действующего лица связывает все сценарии и что каждый сценарий — это простое описание цели, достигаемой или нет.

На рис. 2.3 я дополнил "полосатые брюки" подчиненными вариантами использования, входящими в основные, которые их вызывают. Заказчик хочет разместить заказ. Одна из подцелей заказчика — открыть кредит. Это подцель сложная и может быть достигнута или нет: вариант использования, который мы свернули в единственный шаг. Шаг "заказчик открывает кредит" — это ремень на других брюках. В полосе, или сценарии, этого шага подцель либо достигается, либо нет. В сценариях 1 и 2 на рисунке подцель достигается. В сценариях 3 и 7 — нет. В сценарии 3, однако, следующая подцель для определения запасов достигается, и вариант использования в целом завершается успешно. В сценарии 7 вторая попытка тоже не удается, и полный вариант использования для размещения заказа завершается неудачей.

Полоски на подчиненных вариантах использования (см. рис. 2.3) показывают, что внешнему варианту использования как бы все равно, какой подчиненный вариант использования выполняется в его рамках. Подчиненный вариант приводит к успеху либо к неудаче. Внешний, или вызывающий, вариант использования просто основан на успехе либо на неудаче шага, определяемого подчиненным вариантом использования.

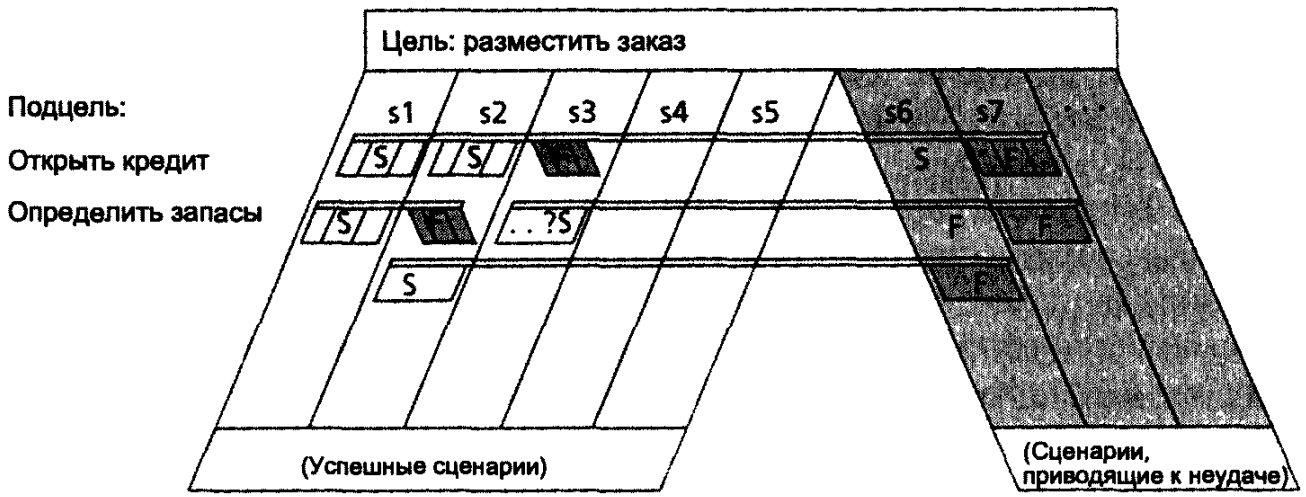


Рис. 2.3. *"Полосатые брюки" с подцелями*

Основные идеи, которые следуют из аналогии с "полосатыми брюками", таковы:

- Некоторые сценарии приводят к успеху, другие к неудаче.
- Вариант использования собирает вместе все сценарии, успешные и заканчивающиеся неудачей.
- Каждый сценарий — это непосредственное описание одного набора обстоятельств с одним исходом.
- Варианты использования содержат сценарии (полосы на брюках), а сценарий содержит подчиненные варианты использования в качестве своих шагов.
- На шаг сценария не влияет, какая полоса в подчиненном варианте использования была задействована. Влияет лишь результат — закончился вариант успехом либо неудачей.

Эти основные идеи прослеживаются на протяжении всей книги.

2.2. Соглашение между участниками, имеющими интересы

Модель Действующие лица и цели объясняет, как писать предложения в варианте использования, но не касается необходимости описывать внутреннее поведение разрабатываемой системы. По этой причине модель Действующие лица и цели следует расширить. Здесь важно, что варианты использования можно рассматривать как соглашение между участниками, имеющими свои интересы. Расширенную модель я буду называть концептуальной моделью *Участники и интересы*. Блок участников и интересов определяет, что включить в вариант использования и что нет.

SuD реализует соглашение между участниками, причем варианты использования детализируют поведенческую часть этого соглашения. Во время работы системы

присутствуют не все участники. Обычно присутствует основное действующее лицо, но не всегда. Другие участники отсутствуют, поэтому их можно назвать *действующими лицами вне сцены*. Система работает, чтобы удовлетворить интересы этих действующих лиц, включая сбор информации, проведение проверок достоверности и обновление журналов (рис. 2.4).

Банкомат должен вести журнал всех взаимодействий, чтобы защитить участников в случае разногласий. Он записывает и другую информацию, поэтому можно определить, как далеко зашла транзакция перед сбоем. Банкомат и банковская система проверяют, имеет ли держатель счета соответствующий капитал, прежде чем выдать ему наличные. Это гарантирует, что банкомат не выдаст больше денег, чем есть на счету этого клиента.

Вариант использования как соглашение о поведении охватывает в полном объеме поведение, связанное с удовлетворением интересов участников, и только его.

Чтобы тщательно выстроить вариант использования, перечислим всех участников, назовем их интересы в связи с функционированием варианта использования и установим, что значит для каждого участника успешное завершение варианта использования и какие гарантии они хотят получить от системы. Разобравшись с этим, напишем шаги варианта использования с гарантией, что все интересы удовлетворяются от момента инициирования варианта использования до его окончания. Так определяется, когда начинать и когда заканчивать описание, а также что включать или не включать в вариант использования.

Большинство людей не пишут варианты использования так кропотливо, и часто им это сходит с рук. Опытные писатели проделывают это упражнение в голове, когда пишут бессистемные варианты использования. Они, возможно, опускают некоторые вещи, но учитывают их при разработке программного обеспечения. Однако иногда это дорого обходится, как в истории, описанной в разделе 4.1.

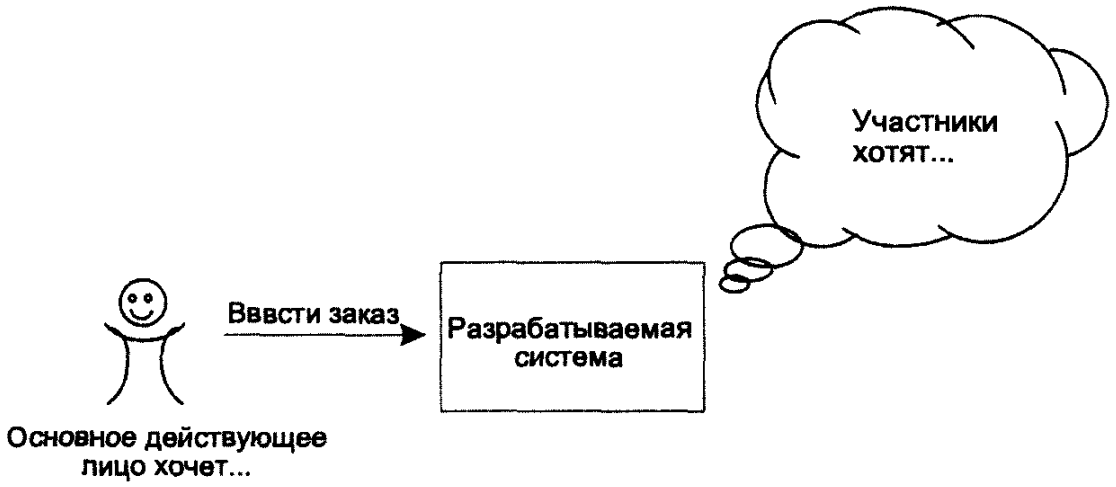


Рис. 2.4. *SuD обслуживает основное действующее лицо, защищая интересы действующих лиц вне сцены*

Чтобы удовлетворить интересы участников, должны быть описаны три вида действий:

- Взаимодействие между двумя действующими лицами (чтобы содействовать достижению цели)
- Проверка достоверности (чтобы защитить участника)
- Изменение внутреннего состояния (от имени участника)

Модель Участники и интересы вносит лишь небольшое изменение в общую процедуру создания варианта использования: перечень участников и их интересов и использование этого перечня в качестве двойной проверки для гарантии, что ничто не пропущено в тексте варианта использования. Это изменение существенно влияет на качество варианта использования.

2.3. Графическая модель

Этот раздел предназначен только для тех, кому нравится строить абстрактные модели; его можно пропустить.

Вариант использования описывает соглашение о поведении между участниками, имеющими интересы. Мы организуем поведение с помощью функциональных целей избранного множества участников (тех, кто попросит систему что-нибудь сделать для них). Этих участников мы называем *основными действующими лицами*. Название варианта использования — это цель основного действующего лица. В варианте использования содержатся все аспекты поведения, необходимые для описания этой части соглашения.

У системы есть обязанность удовлетворять оговоренные соглашением интересы обусловленных соглашением участников с их действиями. Действия бывают трех типов:

- Взаимодействие между двумя действующими лицами, при котором информация может передаваться назад или вперед.
- Проверка достоверности, чтобы защитить интересы одного из участников.
- Изменение внутреннего состояния для защиты интереса участника.

Сценарий состоит из шагов действия. В *сценарии успеха* все (обусловленные соглашением) интересы участников удовлетворяются вследствие обслуживания, которое сценарий обязан выполнять. В *сценарии неудачи* все интересы защищаются в соответствии с гарантиями системы. Сценарий завершается, когда все интересы участников удовлетворены или защищены.

Три триггера, которые запрашивают достижение цели, являются основным действующим лицом, инициирующим взаимодействие с системой, основным действующим лицом, использующим посредника для инициирования взаимодействия, или временным триггером либо триггером состояния.

Модель вариантов использования, описанная в этой главе, показана на рис. 2.5 — 2.8. Здесь используется унифицированный язык моделирования UML.

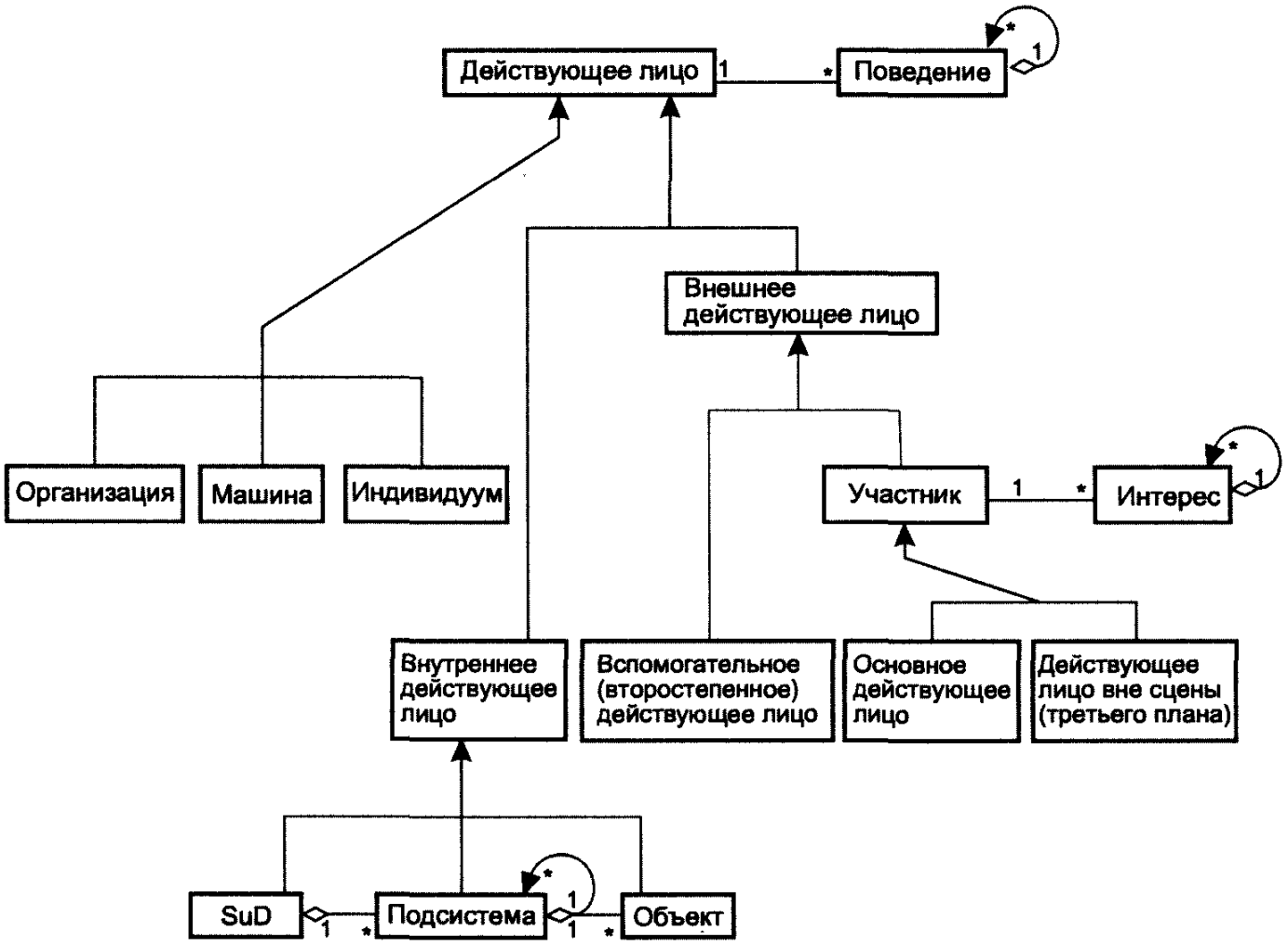


Рис. 2.5. *Действующие лица и участники. Участник имеет интересы. Действующее лицо характеризуется поведением. Основное действующее лицо одновременно является участником*

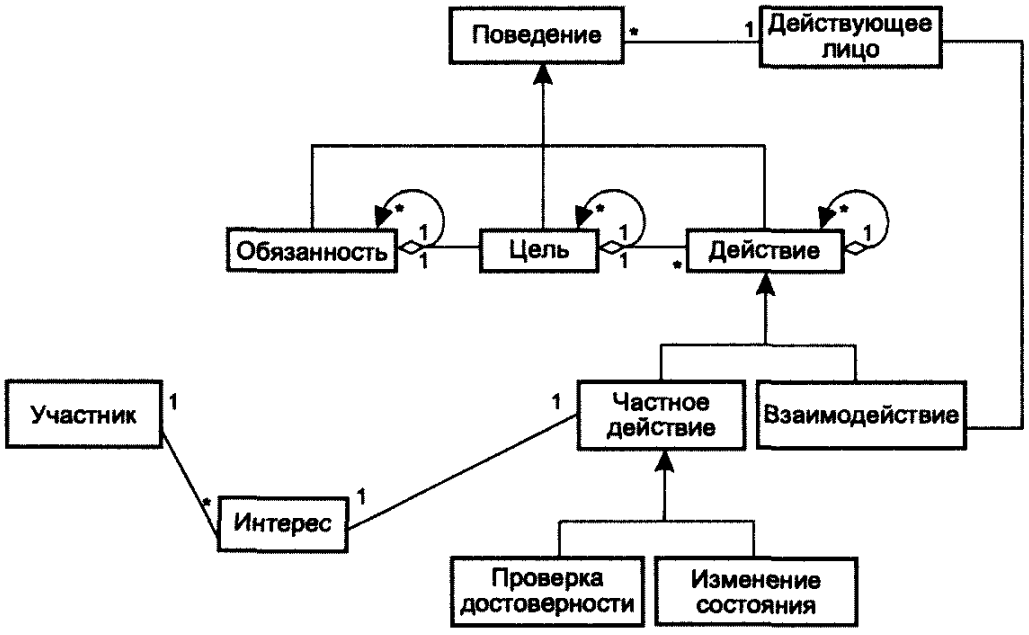


Рис. 2.6. *Поведение. Ориентированное на цель поведение включает обязанности, цели и действия. Частные действия, о которых мы пишем, содействуют или защищают интересы участников. Взаимодействия соединяют действия одного действующего лица с действиями другого*

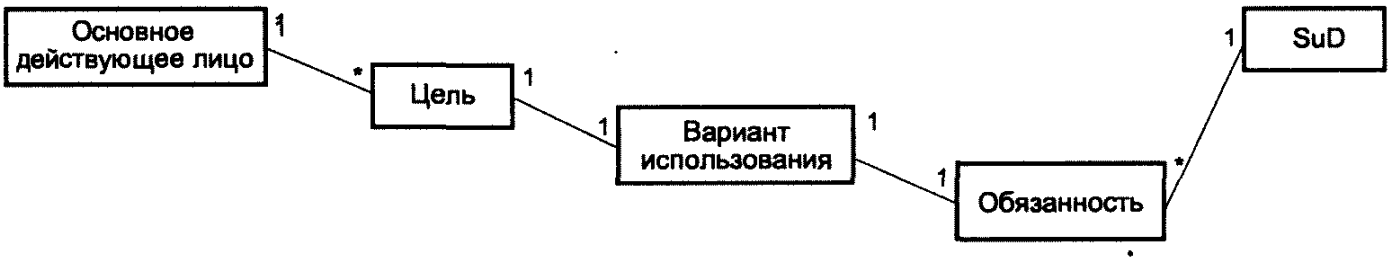


Рис. 2.7. *Вариант использования как инициирование обязанности. Вариант использования фиксирует цель основного действующего лица и обращается к SuD для выполнения ее соответствующей обязанности*

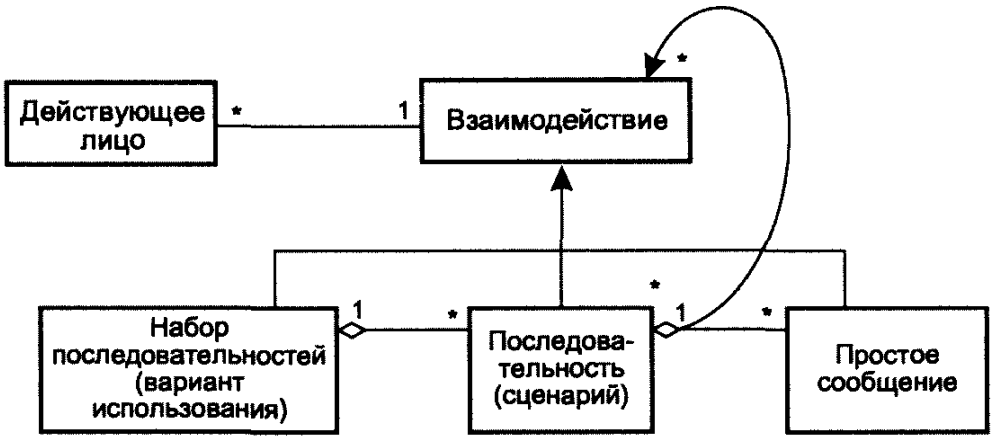


Рис. 2.8. *Взаимодействия как композиция. N действующих лиц взаимодействуют. Взаимодействия можно разложить на варианты использования, сценарии и простые сообщения. Термин "последовательность" употребляется для удобства*

Это абстрактная модель. Я не знаю, как ее отлаживать. Могу лишь предложить несколько лет тестировать ее на проектах, используя основанный на модели инструмент. Другими словами, она, вероятно, содержит несколько неуловимых ошибок. Я включил ее для тех, кто хочет поэкспериментировать и, возможно, создать такой основанный на модели инструмент.

Глава 3

Область действия

Термин *область действия* (scope) обозначает границы нашего проекта, в противоположность тому, что проектирует кто-то еще, или тому, что уже существует.

Отследить область действия проекта или даже только область рассуждения о нем может быть непросто. Консультант Роб Томсетт познакомил меня с инструментом для управления обсуждением области действия — списком *Внутри/Вне*. Этот простой и эффективный инструмент можно использовать для управления обсуждением области действия на обычных совещаниях, а также при обсуждении требований проекта.

Постройте таблицу с тремя колонками. Левая колонка содержит название предмета обсуждения, следующие две колонки называются Внутри и Вне. Всякий раз, когда вы сомневаетесь, входит ли предмет в область обсуждения, добавьте его в таблицу и посоветуйтесь с коллегами, внутри он или вне области обсуждения. Пока каждому сотруднику не станет совершенно ясно, где находится предмет, мнения расходятся. По словам Роба, иногда требуется обратиться в оргкомитет проекта, чтобы установить, действительно ли определенный предмет находится внутри области действия разработки. От того, где именно находится предмет, срок разработки может увеличиться или уменьшиться на много месяцев. Попробуйте применить этот метод в вашем проекте или, скажем, на следующем совещании.

В таблице 3.1 представлен пример списка Внутри/Вне, который мы составили для системы отслеживания запросов на покупку.

Используйте этот список с самого начала работы над требованиями или вариантами использования, чтобы отделить вопросы, входящие в область действия данного проекта, от тех, которые в нее не входят. Обращайтесь к списку всякий раз, когда обсуждение грозит уйти в сторону, или дискуссия разворачивается вокруг требования, отношение которого к данной области действия вызывает сомнение. Обновляйте список по ходу дела.

Используйте список Внутри/Вне для предметов, относящихся как к функциональной области действия, так и к области проектирования разрабатываемой системы.

Таблица 3.1. Пример списка Внутри/Вне

Предмет	Внутри	Вне
Выписывание счета в любой форме		Вне
Формирование отчетов о запросах (например, по поставщику, по детали, по заказчику)	Внутри	
Объединение запросов в один денежный перевод по почте	Внутри	
Частичные доставки, опоздавшие доставки, неправильные доставки	Внутри	
Все новые услуги системы, программное обеспечение	Внутри	
Любые части системы, не относящиеся к программному обеспечению		Вне
Определение любого ранее разработанного программного обеспечения, которое можно использовать	Внутри	
Заявки	Внутри	

3.1. Функциональная область действия

Функциональная область действия относится к услугам, которые предоставляет ваша система и которые в конце концов будут зафиксированы в вариантах использования. Поскольку вы только начинаете разрабатывать проект, вполне вероятно, что вы не знаете деталей. Вы определяете функциональную область действия одновременно с созданием вариантов использования, эти две задачи переплетены. Список Внутри/Вне помогает решить эти задачи, так как позволяет провести границу между тем, что внутри области действия и тем, что вне этой области. Два других инструмента — это список *Действующее лицо/Цель* и *Краткое описание вариантов использования*.

Список Действующее лицо/Цель

В списке Действующее лицо/Цель перечислены все цели пользователя, которые поддерживает система, иными словами, ее функциональное содержание. Список Внутри/Вне показывает, находятся ли его элементы в соответствующей области действия. В отличие от него список Действующее лицо/Цель включает лишь те услуги, которые действительно будут поддерживаться системой. В таблице 3.2 представлен список Действующее лицо/Цель для одного из проектов системы отслеживания запросов на покупки.

Чтобы создать этот список, постройте таблицу из трех колонок. Впишите имена основных действующих лиц (это те, кто имеет цели) в левую колонку, в среднюю введите цель каждого действующего лица по отношению к системе, а в третью колонку — приоритет, или первоначальное предположение, в какой версии система будет поддерживать эту цель. Обновляйте этот список непрерывно в течение всей работы над проектом, чтобы он всегда отражал состояние функциональных границ системы.

Иногда добавляют дополнительные колонки: *триггер*, указывающий варианты использования, которые будут запускаться в соответствующее время, а не каким-то лицом; бизнес-приоритет; сложность разработки; приоритет разработки, чтобы разделить потребности бизнеса и стоимость разработки для определения ее приоритета.

Таблица 3.2. *Пример списка Действующее лицо/Цель*

Действующее лицо	Цель уровня задачи	Приоритет
Любое	Проверять запросы	1
Уполномоченное лицо	Изменить полномочия	2
Покупатель	Изменить контакты с поставщиком	3
Инициатор запроса	Инициировать запрос	1
	Изменить запрос	1
	Отменить запрос	4
	Отметить доставку заказа	4
	Отказаться от доставленных товаров	4
Утверждающее лицо	Закончить оформление запроса для подачи	2
Покупатель	Завершить запрос для размещения заказа	1
	Инициировать перевод денег поставщику	1
	Сигнализировать о недоставке	4
Уполномоченное лицо	Проверить достоверность подпис и утверждающего лица	3
Приемщик	Зарегистрировать доставку	1

Список Действующее лицо/Цель — это отправная точка переговоров между представителем пользователя, финансовым спонсором и группой разработки. Он фокусирует внимание на планировании и содержании проекта.

Краткие описания вариантов использования

Я буду постоянно напоминать о том, как важно управлять своими силами и, когда возможно, работать с низким уровнем точности. Список Действующее лицо/Цель представляет собой низший уровень точности описания поведения системы. Он очень полезен для работы с общим описанием системы. Следующий уровень точности будет либо основным сценарием, либо кратким описанием варианта использования.

Краткое описания варианта использования — это описание поведения варианта использования в двух-шести предложениях, где упоминаются лишь наиболее значительные действия и сбои. Оно напоминает разработчикам о том, что делается в варианте использования, и полезно для оценки сложности работы. Группы, строящие новую систему из готовых коммерческих компонентов (COTS), используют это опи-

сание при выборе компонентов. Некоторые проектные группы, например, имеющие очень хорошо налаженный внутренний обмен информацией и постоянное общение с пользователями, никогда ничего не пишут сверх этих кратких описаний вариантов использования в качестве требований. Остальная информация для требований появляется в ходе непрерывных обсуждений, представлена прототипами и регулярно поставляемыми дополнениями.

Таблица 3.3. *Пример кратких описаний вариантов использования*

Действующее лицо	Цель	Краткое описание
Производственная группа	Модифицировать схему административного региона	Производственная группа добавляет описание административного региона (административное деление, денежное обращение, код языка, типы улиц и т.д.) в справочную базу данных. Контактная информация об источнике данных помещается в каталог. Это особый случай модификации справочных данных
Производственная группа	Подготовить цифровые картографические исходные данные	Производственная группа преобразует внешние цифровые данные в стандартный формат, проверяет и корректирует их, подготавливая для слияния с рабочей базой данных. Данные каталогизированы и хранятся в исходной цифровой библиотеке
Производственная и полевая группы	В рабочей базе данных зафиксировать транзакции обновления при совместной отладке	Группа применяет накопленные транзакции обновления к рабочей базе данных. Транзакции без конфликтов фиксируются в рабочей базе данных. Контекст приложения синхронизируется с рабочей базой данных. Фиксируемые транзакции очищаются от контекста приложений, сохраняя совместимость рабочей базы данных, причем конфликтующие транзакции можно урегулировать вручную либо интерактивным путем

Можно подготовить краткое описание варианта использования как таблицу, расширение списка Действующее лицо/Цель или непосредственно в виде части тела варианта использования в первом приближении. Пример краткого описания из таблицы 3.3 любезно предоставлен Полом Фордом, Стивом Янгом и Полом Бузайдом из компании Navigation Technologies.

3.2. Область действия проектирования

Область действия проектирования — это рамки системы, я бы сказал "пространственные рамки", если бы программное обеспечение занимало пространство. Это множество систем, аппаратных и программных, которые мы должны проектировать или обсуждать. Если мы разрабатываем банкомат, нам следует произвести

оборудование и программное обеспечение, которые находятся в корпусе. Корпус и все, что в нем — это то, что разрабатываем мы. Компьютерная сеть, с которой будет общаться банкомат, не входит в рамки нашей разработки, лежит вне нашей области действия проектирования.

Далее под *областью действия* подразумевается *область действия проектирования*, потому что функциональная область действия адекватно определяется списком Действующее лицо/Цель и вариантами использования. Область действия проектирования имеет отношение к каждому варианту использования.

Очень важно, чтобы писатель и читатель понимали область действия проектирования варианта использования одинаково и правильно. Ошибка может увеличить стоимость проекта в несколько раз и привести к катастрофическому результату контракта. Читатели варианта использования должны быстро понять, что вы включаете в рамки системы. Только из названия варианта использования или основного действующего лица это ясно не будет. Даже внутри одного и того же множества вариантов использования обнаруживаются системы различного масштаба.

Обычно писатели считают область действия системы настолько очевидной, что даже не упоминают ее. Однако это не так. Один автор думает, что областью действия является вся корпорация (см. рис. 3.1), другой считает областью действия все программные системы компании, третий подразумевает под ней новую систему клиент-сервер, тогда как четвертый думает только о клиенте либо только о сервере. Читатели, которые не знают об этих значениях по умолчанию, плохо ориентируются в таком документе или неправильно его понимают.

Невымышленная история

Чтобы помочь составить заявку на подряд для разработки большой системы в заданное время и с фиксированными затратами, мы разобрали некоторые примеры проектов. Я выбрал принтер и описал его функцию. Эксперт по информационным системам рассмеялся. "Вы, привыкшие к персональному компьютеру, с ума меня сведете", — сказал он. "Вы что думаете, мы используем маленький лазерный принтер, чтобы печатать наши счета? У нас огромная система печати с цепным печатающим устройством, очередью ввода/вывода и прочим. Мы производим счета коробками!"

Я был потрясен. "Вы хотите сказать, что принтер не входит в область действия системы?"

"Конечно, нет! Мы будем использовать систему печати, которая у нас уже есть".

В самом деле, мы обнаружили сложный интерфейс с системой печати. Наша система должна была подготавливать магнитную ленту с файлами для печати. Ночь напролет система печати читала бы ленту и делала распечатки. Она бы подготавливала еще одну ленту с описанием результатов печати и неверными записями, которые не смогла распечатать. На следующий день наша система читала бы результаты и отмечала, что распечатано неправильно. Работа по проектированию интерфейса к этой ленте была существенной и абсолютно не похожей на то, что мы ожидали.

Нам не надо было проектировать систему печати, но следовало использовать ее. Система не входила в область действия нашего проекта (она была вспомогательным действующим лицом; см. раздел 3.3). Если бы мы не обнаружили эту ошибку, то написали бы вариант использования, чтобы включить ее в нашу область действия проектирования, и нам бы пришлось разрабатывать больше систем, чем было необходимо.

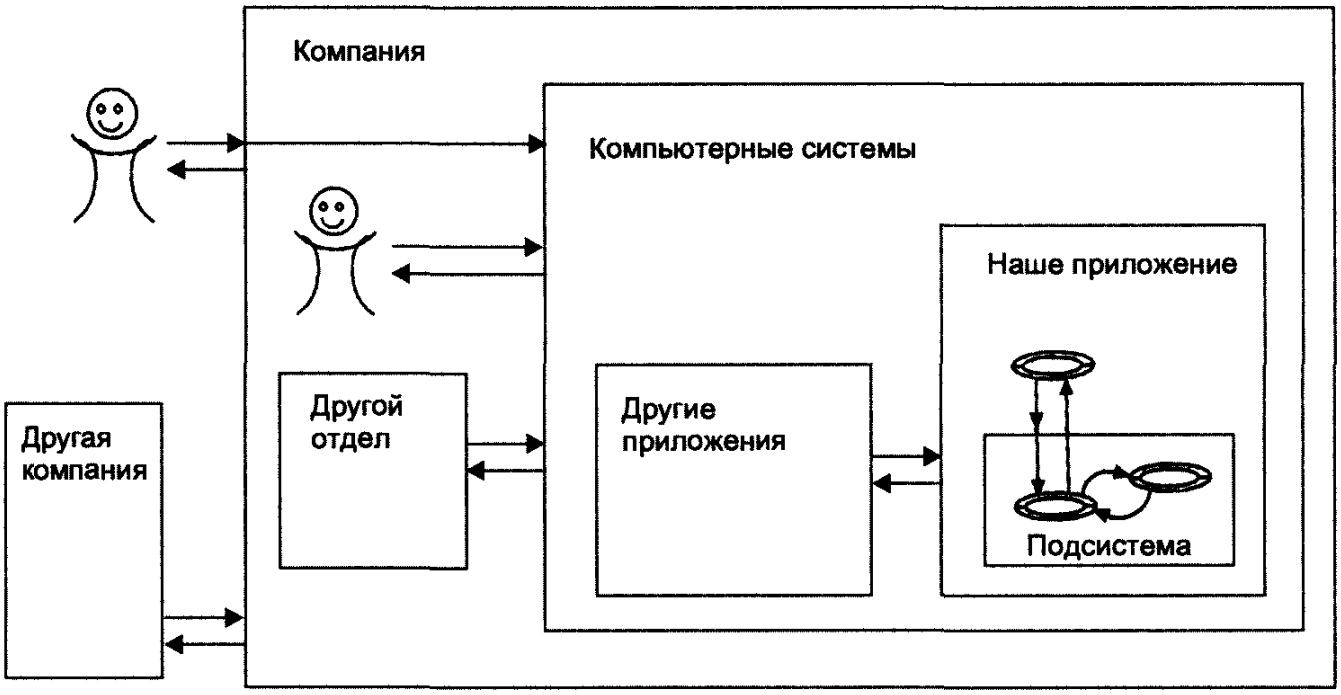


Рис. 3.1. *Область действия проектирования может быть любого размера*

Что можно сделать, чтобы правильно понять документ?

Единственный ответ, который я нашел — это снабдить каждый вариант использования меткой его области действия проектирования с помощью особых имен для наиболее значительных областей действия. Предположим, что компания MyTelCo проектирует систему NewApp, которая включает подсистему Searcher. Ниже следуют имена областей действия проектирования:

- *Предприятие (MyTelCo)* . Вы обсуждаете поведение целой организации или предприятия для достижения цели основного действующего лица. Запишите в поле Область действия варианта использования название организации (MyTelCo), а не просто "компания". Обсуждая отдел, употребляйте его название. Варианты использования для бизнеса пишутся в области действия предприятия.

- *Система (NewApp)* . Это оборудование или программное обеспечение, которое вам поручили разрабатывать. Вне системы находятся все части оборудования, программного обеспечения и людских ресурсов, с которыми система должна взаимодействовать.

- *Подсистема (Seacher)* . Вы раскрыли главную систему и собираетесь рассмотреть, как работает ее часть.

Использование графических пиктограмм для выделения области действия проектирования

Рассмотрим значки, присоединяемые слева к названию варианта использования для предварительного информирования читателя относительно области действия проектирования. Пока нет инструментов управления значками, но я считаю, что они уменьшают путаницу. В данной книге я снабдил каждый вариант использования соответствующей пиктограммой, чтобы вам легче было заметить его область действия.

Когда вы будете читать следующий список, помните, что в варианте использования "черный ящик" не обсуждается внутренняя структура разрабатываемой системы, в противоположность варианту использования "прозрачный ящик".

- Областью действия варианта использования для *бизнеса* является предприятие. Обозначается пиктограммой домика. Домик будет серым, если вы рассматриваете предприятие в целом как "черный ящик". Если вы говорите об отделах и о персонале предприятия, оставьте домик белым.
- Область действия варианта использования для *системы* — это компьютерная система. Значок — ящик: серый, если вы трактуете систему как "черный ящик", и белый, если вы касаетесь работы ее компонентов.
- Вариант использования для *компонента* относится к подсистеме или к компоненту разрабатываемой системы. В качестве графического символа компонента используется болт (см. варианты использования 13 — 17).

Примеры области действия проектирования

Рассмотрим три примера, иллюстрирующих системы с различными областями действия проектирования.

(1) Область действия предприятие — система

Предположим, мы работаем на телефонную компанию *MyTelCo*, которая разрабатывает новую систему *Acura* для приема заказов на обслуживание и модернизацию. Acura состоит из рабочей станции, подключенной к серверу. Сервер будет соединен с мэйнфреймом, на котором работает старая система BSSO. Система BSSO представляет собой обычный терминал, подключенный к мэйнфрейму. Нам не разрешили вносить в нее какие-либо изменения, мы можем только использовать ее интерфейсы.

К основным действующим лицам для Acura относятся клиент, служащий, различные администраторы и BSSO (последняя вне нашей области действия).

Определим цели, которые должна поддерживать система. Наиболее очевидная — "добавить новую услугу". Основным действующим лицом для этой цели будет служащий компании, действующий от имени заказчика. Создадим несколько вариантов использования.

Теперь нужно установить, что представляет собой разрабатываемая система. Оказывается, нас интересуют две вещи:

- *MyTelCo.* Нам хотелось бы знать, как выглядит для пользователя обслуживание компанией MyTelCo, как реализуются новые услуги в полной форме — от первоначального запроса до выполнения и предоставления. Этот вопрос интересен вдвойне. Администраторы компании захотят узнать, какой представляется новая система внешнему миру, а группа разработчиков — увидеть контекст, в котором будет существовать новая система.

  Этот вариант использования будет написан для области действия предприятия (🏠), причем в поле области действия будет записано MyTelCo, и в варианте использования не будут упомянуты внутренние действующие лица (служащие, отделы, компьютеры). Этот вид варианта использования часто называют вариантом использования для бизнеса, так как он описывает деятельность предприятия.

- *Acura.* Нам хотелось бы знать, что представляет собой обслуживание этой системой с точки зрения служащего или клиента, с одной стороны, и с точки зрения системы BSSO, с другой. К этому варианту использования разработчики относятся с наибольшим вниманием, так как он точно устанавливает, что они должны создавать. Вариант использования будет написан для области действия системы (📦), в поле области действия будет записано Acura. В нем будут свободно фигурировать служащие, отделы и другие компьютерные системы, но никак не подсистемы рабочей станции и сервера.

  Создадим два варианта использования. Чтобы не излагать одну и ту же информацию дважды, напишем вариант использования для предприятия на более высоком уровне (знак воздушного змея), показывая, как MyTelCo отвечает на запрос, предоставляет услугу, возможно, даже записывает стоимость на счет клиента и получает оплату. Задача варианта использования для предприятия — показать контекст, в котором будет работать новая система. Далее подробно опишем обработку запроса в течение 5-20 мин в варианте использования для цели пользователя и области действия Acura.

Вариант использования 6

🏠 Добавить новую услугу (предприятие)

Основное действующее лицо: клиент
Область действия: MyTelCo
Уровень: обобщенный
1. Клиент звонит в MyTelCo, запрашивает новую услугу...
2. MyTelCo предоставляет... и т.д. ...

Вариант использования 7

📦 Добавить новую услугу (Acura)

Основное действующее лицо: служащий (от имени внешнего клиента)
Область действия: Acura

Уровень: цель пользователя

1. Клиент звонит в компанию, служащий обсуждает запрос с клиентом.
2. Служащий находит клиента в системе Acura.
3. Acura представляет текущий пакет услуг данного клиента... и т.д. ...

Для областей действия рабочей станции Acura или сервера Acura никаких вариантов использования писать не будем, поскольку нам это не очень интересно. Позднее кто-нибудь из группы разработчиков может заняться документированием проекта подсистем Acura с помощью вариантов использования. Тогда они напишут два варианта использования — один для области действия рабочей станции Acura, другой для области действия сервера Acura. Мой опыт показывает, что до подобных вариантов использования дело не доходит, так как существуют другие адекватные способы документирования архитектуры подсистем.

(2) Несколько компьютеров на одно приложение

Следующая ситуация не столь тривиальна и весьма сложна. Сделаем надстройку над ситуацией MyTelCo.

> Acura постепенно заменит BSSO. Запросы новой услуги будут записываться в Acura, а затем преобразовываться с помощью BSSO. Со временем набор функций Acura расширится. Эти две системы должны сосуществовать и синхронизироваться друг с другом. Таким образом, варианты использования придется написать для обеих систем: для новой Acura и для модифицированной для синхронизации с ней BSSO.

Сложность данной ситуации в том, что существуют четыре варианта использования: два для Acura и два для BSSO. Есть один вариант использования для каждой системы со служащим в качестве основного действующего лица и другой, в котором основное действующее лицо — это еще одна компьютерная система. Сократить эту четверку никак нельзя, но она смущает читателей, поскольку выглядит избыточной.

Чтобы документировать данную ситуацию, сначала я напишу вариант использования обобщенного уровня, чьей областью действия являются обе компьютерные системы. Это даст мне возможность документировать их взаимодействие со временем. В этом варианте использования я буду обращаться к особым вариантам использования, которые включают каждое требование системы. Первый вариант использования будет иметь тип "прозрачный ящик" (обратите внимание на символ прозрачного ящика).

Ситуация достаточно сложная, так что я включил диаграммы области действия каждого варианта использования.

Вариант использования 8

Ввести и обновить запросы (объединенная система)

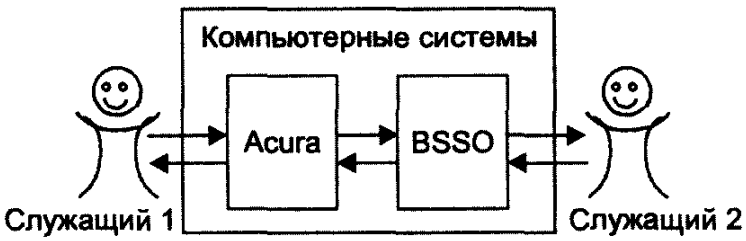
Основное действующее лицо: служащий (от имени внешнего клиента)

Область действия: компьютерные системы, включая Acura и BSSO (см. диаграмму)

Уровень: обобщенный

Основной сценарий:

1. Служащий добавляет новую услугу в Acura.
2. Acura записывает запрос новой услуги в BSSO.
3. Некоторое время спустя служащий обновляет запрос услуги в BSSO.
4. BSSO записывает обновленный запрос в Acura.



Все четыре подчиненных варианта использования имеют уровень цели пользователя и отмечены символом уровня моря. Хотя все они — системные варианты использования, но относятся к разным системам, отсюда и диаграммы. В каждой диаграмме я выделил основное действующее лицо и затенил разрабатываемую систему. На этот раз варианты использования являются "черными ящиками", так как представляют собой требования для новой разработки. Кроме того, я дал им различные названия с глаголом записать, чтобы указать на синхронизацию систем друг с другом.

Вариант использования 9

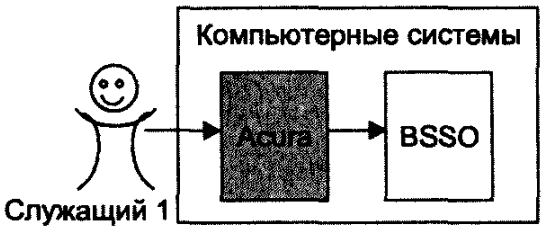
Добавить новую услугу (в Acura)

Основное действующее лицо: служащий (от имени внешнего клиента)

Область действия: Acura

Уровень: цель пользователя

... тело варианта использования ...



Вариант использования 10

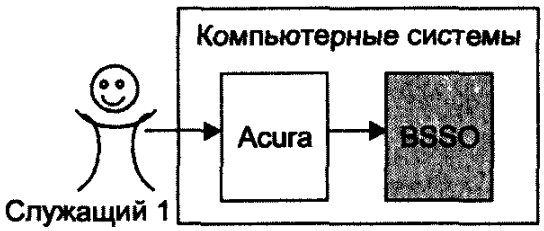
Записать запрос на новую услугу (в BSSO)

Основное действующее лицо: Acura

Область действия: BSSO

Уровень: цель пользователя

... тело варианта использования ...



Вариант использования 11

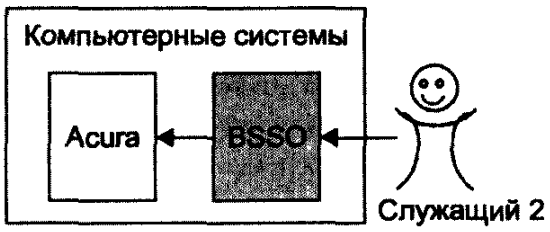
Обновить запрос на обслуживание (в BSSO)

Основное действующее лицо: служащий (от имени внешнего клиента)

Область действия: BSSO

Уровень: цель пользователя

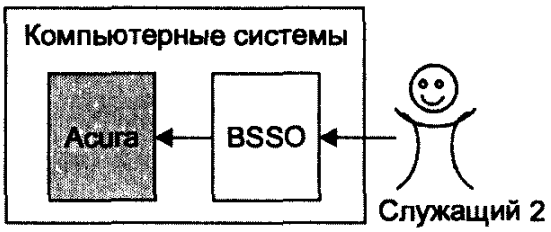
... тело варианта использования ...



Вариант использования 12

Записать обновленный запрос (в Acura)

Основное действующее лицо: BSSO
Область действия: Acura
Уровень: цель пользователя
... тело варианта использования ...



Если вы используете UML-диаграммы вариантов использования, можете не писать, а рисовать вариант использования обобщенного уровня. Это все же не уменьшает путаницы с четырьмя вариантами использования для цели пользователя, поэтому для каждого варианта следует записать основное действующее лицо, область действия и уровень. Может быть, стоит добавить диаграмму области действия.

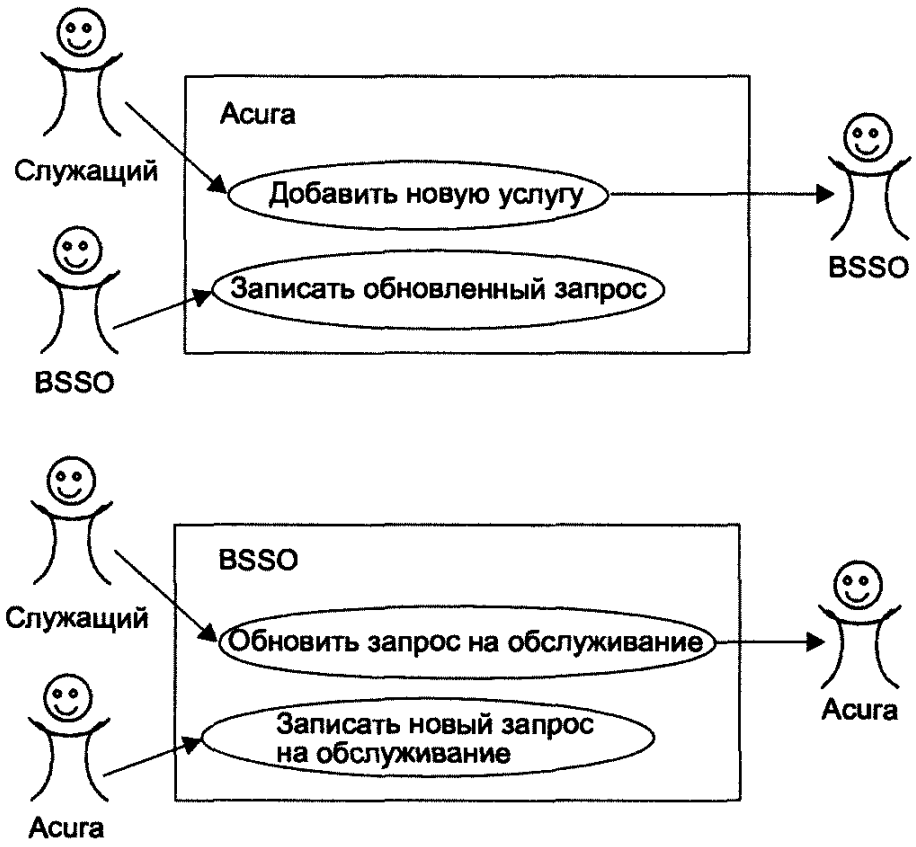


Рис. 3.2. *Диаграммы вариантов использования для Acura-BSSO. Это изображение взаимодействия двух систем в стиле UML. В верхнем блоке показано, что BSSO — это вспомогательное действующее лицо для одного варианта использования системы Acura и основное действующее лицо для другого варианта использования. В нижней диаграмме роли поменялись*

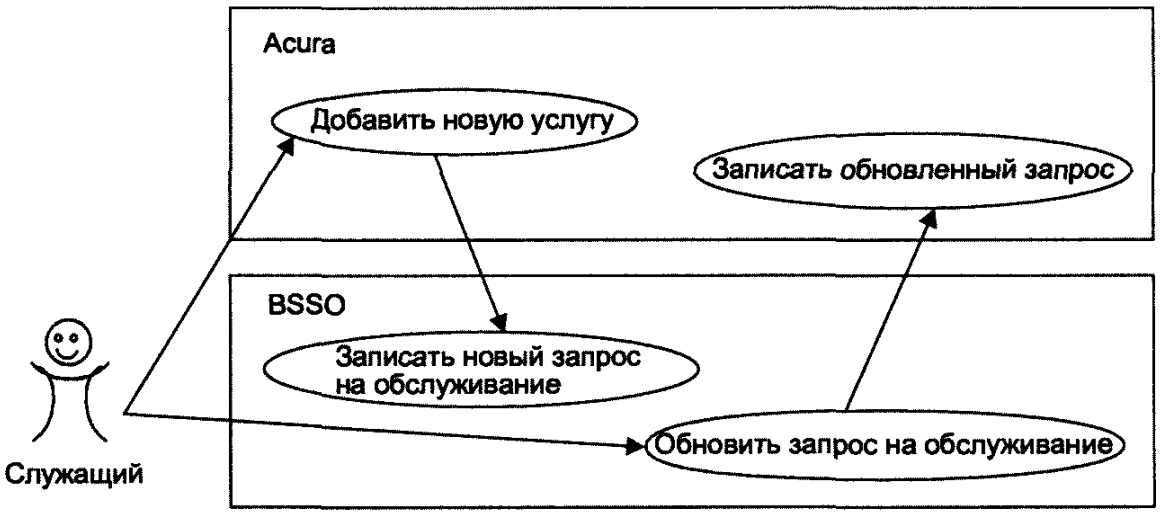


Рис. 3.3. *Комбинированная диаграмма варианта использования для Acura-BSSO. Здесь показаны связи четырех вариантов использования наиболее ясно, но эта диаграмма не является стандартной, поскольку на ней один вариант использования системы запускает другой*

Лично я не считаю, что это вносит ясность. Я бы предпочел нестандартную диаграмму варианта использования (см. рис. 3.3), чтобы показать связь двух систем. Эта диаграмма более наглядна, но ее труднее сопровождать. Диаграмма должна помогать вам и вашим читателям лучше понимать друг друга.

(3) Варианты использования "Гайки и болты"

Посмотрим, как документируют структуру проекта с помощью вариантов использования. Группа начала с 18-страничного, полного диаграмм описания правил для разрабатываемой системы. Было решено, что его слишком трудно читать, и начались эксперименты с вариантом использования как методом описания.

Группа решала эту задачу неделю. Сначала они набросали 40 вариантов использования, чтобы охватить все запросы, которые может обрабатывать главная ЭВМ. Используя расширения и список изменения данных, они сократили количество вариантов использования до шести.

Для большинства читателей эти варианты использования будут малопонятными. Полагаю, однако, что среди них есть технари-программисты, ищущие способы документирования своих проектов. Поэтому я включил эти варианты использования, чтобы показать, как эта группа документировала внутреннюю архитектуру и применяла список изменений. Обратите внимание, что подчиненные варианты использования подчеркнуты. Автор вариантов использования — Дэйл Маргел из Калгари.

Общее описание:

Общая архитектура должна уметь решать параллельные задачи. Для этого она должна поддерживать потоки процессов и блокировку ресурсов. Эти функции выполняет система параллельного обслуживания (Concurrency Service Framework, CSF). CSF используется объектами клиента для защиты критичных разделов кода от небезопасного доступа со стороны множества процессов.

Вариант использования 13

Организовать последовательный доступ к ресурсу

Основное действующее лицо: объект клиента CSF

Область действия: система параллельного обслуживания (CSF)

Уровень: цель пользователя

Основной сценарий:

1. Клиент CSF запрашивает блокировку ресурса, чтобы обеспечить к нему определенный доступ.
2. Программа блокировки ресурса возвращает управление клиенту CSF, чтобы он мог использовать этот ресурс.
3. Клиент CSF использует ресурс.
4. Клиент CSF информирует программу блокировки ресурса о завершении работы с ресурсом.
5. По завершении работы клиента CSF программа блокировки ресурса восстанавливает прежнее состояние ресурса.

Расширения:

2a. Программа блокировки ресурса обнаруживает, что клиент CSF уже имеет доступ к этому ресурсу:

2a1. Программа блокировки ресурса применяет к запросу политику преобразования блокировки (вариант использования 14).

2b. Программа блокировки ресурса обнаруживает, что ресурс уже используется:

2b1. Программа блокировки ресурса применяет политику совместимости (вариант использования 15), чтобы обеспечить доступ клиенту CSF.

2c. Время блокировки ресурса не истекло:

2c1. Программа блокировки ресурса запускает таймер занятости.

3a. Заданный таймером занятости интервал истекает, прежде чем клиент информирует программу блокировки ресурса, что он закончил работу:

3a1. Программа блокировки ресурса посылает процессу клиента сообщение об исключительной ситуации.

3a2. Отказ!

4a. Программа блокировки ресурса обнаруживает ненулевое значение счетчика блокировок у клиента CSF:

4a1. Программа блокировки ресурса уменьшает значение счетчика обращений для запроса.

4a2. Успех!

5a. Программа блокировки ресурса обнаруживает, что ресурс в настоящий момент не используется:

5a1. Программа блокировки ресурса применяет политику выбора доступа (вариант использования 16) для предоставления доступа одному из стоящих в очереди клиентов.

5b. Таймер занятости все еще работает:

5b1. Программа блокировки ресурса отменяет таймер занятости.

Список изменений в технологии и данных:

1. Указанный в запросе доступ может быть:
   - Монопольный
   - Совместный

2c. Тайм-аут в блокировке ресурса может определять:
   - Клиент CSF
   - Политика блокировки ресурса
   - Глобальное значение по умолчанию

Вариант использования 14

Применить политику преобразования блокировки

Основное действующее лицо: объект клиента CSF

Область действия: система параллельного обслуживания (CSF)

Уровень: подфункция

Основной сценарий:

1. Программа блокировки ресурса проверяет, является ли запрашиваемый доступ монопольным.
2. Программа блокировки ресурса проверяет, не имеет ли уже клиент CSF совместного доступа к ресурсу.
3. Программа блокировки ресурса проверяет, нет ли клиента CSF, ожидающего повышения уровня доступа.
4. Программа блокировки ресурса проверяет, нет ли других клиентов CSF, разделяющих данный ресурс.
5. Программа блокировки ресурса предоставляет клиенту CSF монопольный доступ к ресурсу.
6. Программа блокировки ресурса увеличивает значение счетчика блокировок для клиента CSF.

Расширения:

1a. Программа блокировки ресурса обнаруживает, что запрашивается совместный доступ:

1a1. Программа блокировки ресурса увеличивает значение счетчика блокировок для клиента CSF.

1a2. Успех!

2a. Программа блокировки ресурса обнаруживает, что клиент CSF уже имеет монопольный доступ:

2a1. Программа блокировки ресурса увеличивает значение счетчика блокировок для клиента CSF.

2a2. Успех!

3a. Программа блокировки ресурса обнаруживает, что существует другой клиент CSF, ожидающий повышения уровня доступа:

3a1. Сигнализировать клиенту, что запрошенный доступ не может быть предоставлен.

3a2. Отказ!

4a. Программа блокировки ресурса обнаруживает, что существуют другие клиенты CSF, использующие ресурс:

4a1. Программа блокировки ресурса переводит клиент в режим ожидания доступа к ресурсу (вариант использования 17).

Вариант использования 15

Применить политику совместимости доступа

Основное действующее лицо: объект клиента CSF

Область действия: система параллельного обслуживания (CSF)

Уровень: подфункция

Основной сценарий:

1. Программа блокировки ресурса проверяет, запрашивается ли совместный доступ.
2. Программа блокировки ресурса проверяет, используется ли в настоящий момент ресурс как совместный.

Расширения:

2a. Программа блокировки ресурса обнаруживает, что запрашивается монопольный доступ:

2a1. Программа блокировки ресурса переводит клиент в режим ожидания доступа к ресурсу (вариант использования 17). Процесс возобновляется позже в результате проведения стратегии обслуживания блокировки.

2b. Программа блокировки ресурса обнаруживает, что в данный момент ресурс используется монопольно:

2b1. Программа блокировки ресурса переводит клиент в режим ожидания доступа к ресурсу (вариант использования 17).

Изменения:

1. Критерий совместимости может изменяться.

Вариант использования 16

Применить политику выбора доступа

Основное действующее лицо: объект клиента CSF

Область действия: система параллельного обслуживания (CSF)

Уровень: подфункция

Основной сценарий:

Цель в контексте: программа блокировки ресурса должна определять, какой запрос в режиме ожидания (если таковые имеются) следует обслужить.

Примечание: эта стратегия может меняться.

1. Программа блокировки ресурса выбирает запрос, который ожидает дольше всех.
2. Программа блокировки ресурса предоставляет доступ выбранному запросу (выбранным запросам), делая соответствующий процесс работоспособным.

Расширения:

1a. Программа блокировки ресурса не находит ожидающих запросов:
    1a1. Успех!

1b. Программа блокировки ресурса обнаруживает запрос, ожидающий повышения уровня доступа от разделяемого до монопольного:
    1b1. Программа блокировки ресурса выбирает запрос на повышение уровня доступа.

1c. Программа блокировки ресурса выбирает запрос на разделяемый доступ.
    1c1. Программа блокировки ресурса повторяется (шаг 1), пока не выберет следующий запрос на монопольный доступ.

Изменения:

1. Критерий выбора может изменяться.

Вариант использования 17

Перевести клиент в режим ожидания доступа к ресурсу

Основное действующее лицо: объект клиента CSF

Область действия: система параллельного обслуживания (CSF)

Уровень: подфункция

Основной сценарий:

Используется: программой блокировки ресурса CC 2,4

1. Программа блокировки ресурса приостанавливает процесс клиента CSF.
2. Клиент CSF ждет возобновления процесса.
3. Процесс клиента CSF возобновлен.

Расширения:

1a. Программа блокировки ресурса обнаруживает, что был определен тайм-аут ожидания:
    1a1. Программа блокировки ресурса запускает таймер.

2a. Время ожидания истекает:
    2a1. Сигнализировать клиенту CSF, что запрошенный доступ не может быть предоставлен.
    2a2. Отказ!

Список изменений в технологии и данных:

1a1. Тайм-аут ожидания блокировки может определять:
    - Клиент CSF

- Политика блокировки ресурса
- Глобальное значение по умолчанию

3.3. Предельные варианты использования

В подразделе Область действия предприятие — система я рекомендовал писать два варианта использования: один для разрабатываемой системы, а другой — для внешней области действия. Теперь мы можем более конкретно поговорить о том, как для каждого варианта использования найти самую ближнюю (предельную, outermost) область из внешних областей действия проектирования, к которой вариант использования еще применим, и написать вариант использования обобщенного уровня для этой области действия.

Вариант использования написан для области действия проектирования. Обычно можно найти более широкую область действия проектирования, которая еще имеет основное лицо, действующее вне нее. Если вы продолжите расширение этой области действия, то достигнете точки, в которой дальнейшее расширение перенесет основное действующее лицо внутрь области действия. Это и есть *предельная область действия*. Иногда предельная область действия — это предприятие, отдел или просто компьютер. Часто компьютерный отдел является основным действующим лицом в вариантах использования, описывающих компьютерную безопасность, отдел сбыта — основным действующим лицом в вариантах использования, посвященных рекламе, а клиент — основным действующим лицом в вариантах использования для главных функций системы.

Обычно существуют только от двух до пяти предельных вариантов использования для системы в целом, так что не каждый вариант использования пишется дважды. Их так мало, потому что каждый объединяет основные действующие лица, имеющие сходные цели, в одной области действия проектирования и собирает все варианты использования более низкого уровня для этих действующих лиц.

Я рекомендую писать предельные варианты использования, поскольку это занимает мало времени и обеспечивает отличный контекст для набора вариантов использования. Такие варианты использования показывают преимущества, которые система предоставляет своим самым внешним пользователям. Они также обеспечивают оглавление для просмотра описания поведения системы.

Рассмотрим предельные варианты использования, разработанные для компании MyTelCo и ее системы Acura.

> MyTelCo решила предоставить Интернет-клиентам непосредственный доступ к Acura, чтобы разгрузить своих служащих. Acura будет также формировать отчеты об объеме продаж каждого служащего. Кому-то придется организовать уровни безопасности доступа для клиентов и служащих. У нас есть четыре варианта использования: *Добавить услугу (с помощью клиента)*, *Добавить услугу (с помощью служащего)*, *Сформировать отчет об объеме продаж* и *Организовать управление безопасностью доступа.*

Придется написать все четыре варианта использования с системой Acura в качестве области действия SuD. Нужно определить предельную область действия для каждого из них.

Клиент находится вне MyTelCo, поэтому мы имеем один предельный вариант использования с клиентом в качестве основного действующего лица и MyTelCo в качестве области действия. Это будет вариант использования обобщенного уровня, трактующий MyTelCo как "черный ящик", отвечающий на запрос клиента, предоставляющий услугу и т.д. Практически этот вариант использования намечен в варианте использования 6.

Служащий находится внутри MyTelCo. Предельной областью действия для варианта использования Добавить услугу (с помощью персонала) являются все компьютерные системы. Я предполагаю, что все варианты использования для служащих уровня цели пользователя находятся в этом предельном варианте использования, включая несколько вариантов использования для подфункций, например Войти в систему и Выйти из системы.

В варианте использования Сформировать отчет об объеме продаж конечным основным действующим лицом является отдел маркетинга. Предельный вариант использования, областью действия которого является отдел обслуживания, показывает взаимодействие отдела маркетинга со всеми компьютерными системами и отделом обслуживания для определения премий за продажи, для формирования отчетов об объеме продаж и т.д.

Для варианта использования Организовать управление безопасностью доступа конечным основным действующим лицом является отдел безопасности или отдел информационных технологий (IT), а предельной областью действия проектирования — отдел IT либо все компьютерные системы. Чтобы определить и отследить вопросы безопасности, этот вариант использования интенсивно обращается к отделу безопасности и ко всем компьютерным системам.

Эти четыре внешних варианта использования охватывают функции безопасности, маркетинга, обслуживания и клиентов, используя систему Acura во всех аспектах ее работы. Маловероятно, что для системы Acura понадобится написать что-то сверх этих четырех вариантов использования, даже если появится сотня вариантов использования более низкого уровня.

3.4. Использование рабочих результатов определения области действия

Вы определяете функциональную область действия будущей системы, тратите силы и мечетесь между несколькими рабочими результатами, записанными на доске. На одной стороне доски у вас список Внутри/Вне для отслеживания решений в функциональной области действия. У вас есть список действующих лиц с их целями. У вас есть схема области действия проектирования, показывающая людей, организации и системы, которые будут взаимодействовать с разрабатываемой системой.

Вы считаете, что выявляете всех их по мере того, как вырабатывая картину функционирования новой системы, анализируете ее взаимодействие с ними. Вы полагаете, что представляете себе эту область действия проектирования, однако изменения в списке Внутри/Вне раздвигают границы. Теперь вы имеете новое основное действующее лицо и поправки к списку целей.

Рано или поздно вы, вероятно, увидите, что вам необходим четвертый элемент — концепция новой системы. Концепция объединяет все аспекты обсуждения и помогает решить, что следует внести в область действия в первую очередь.

Когда вы это проделаете, у вас будет четыре рабочих результата, которые помогут сформировать область действия системы:

- Концепция
- Диаграмма области действия проектирования
- Список Внутри/Вне
- Список Действующее лицо/Цель

Эти результаты переплетены, и вы, скорее всего, будете изменять их по мере формирования области действия предстоящей разработки.

3.5. Упражнения

Область действия проектирования

3.1. Назовите по крайней мере пять областей действия проектирования системы, о которой, с точки зрения пользователя, можно было бы сказать: *"... Дженни стоит перед банкоматом своего банка. Темно. Она вводит свой личный номер и ищет кнопку Ввод..."*.

3.2. Создайте диаграмму нескольких областей действия для банкомата, включая оборудование и программное обеспечение.

3.3. Для какой системы вы пишете требования? Что представляет собой ее пространство? Что внутри нее? С какими внешними объектами она должна взаимодействовать? В какую систему входит эта система и что находится снаружи этой внешней системы, с чем она должна поддерживать связь? Дайте название внешней системе.

3.4. Создайте диаграмму нескольких областей действия для системы личного консультанта по финансам (PAF; см. упражнение 4.4).

3.5. Создайте диаграмму нескольких областей действия для web-приложения, при выполнении которого рабочая станция пользователя через Интернет подключается к web-серверу вашей компании, соединенному с унаследованной системой на мэйнфрейме.

3.6. В чем разница между двумя вариантами использования для бизнеса, если область действия предприятия рассматривается как *"черный ящик"* и как *"прозрачный ящик"*?

Глава 4

Участники и действующие лица

Участник — это один из тех, кто заключает контракт. Действующее лицо — это некто (нечто), обладающий поведением, оно должно быть способно выполнить предложение с if. Действующим лицом может быть индивидуум, компания или организация, компьютерная программа или компьютерная система — оборудование, программное обеспечение либо то и другое.

Действующими лицами могут быть:

- *Участники* системы
- *Основные действующие лица* вариантов использования
- *Участники разрабатываемой системы* (SuD)
- *Вспомогательные действующие лица* варианта использования
- *Внутренние действующие лица* (в компонентах SuD)

4.1. Участники

Участник — это кто-то (что-то), имеющий законный интерес в поведении, реализуемом в варианте использования.

Каждое основное действующее лицо является участником, но некоторые участники никогда не взаимодействуют с системой непосредственно, даже если имеют право проявлять интерес к поведению системы. Примером служат владельцы системы, правление компании и органы государственного управления, например Налоговое управление и Министерство страхования.

Студенты называют участников, которые никогда напрямую не появляются в шагах действий вариантов использования, *закулисными*, *третьестепенными* или *бессловесными* действующими лицами.

Уделяя внимание последним, мы значительно улучшаем качество варианта использования. Их интересы касаются проверок и подтверждений, которые производит

система, журналов, которые она создает, и выполняемых ею действий. Правила бизнеса документируются, так как система должна проводить их в жизнь от имени участников. Необходимо, чтобы варианты использования показывали, как система защищает интересы участников. Далее следует история, иллюстрирующая, во что обходится забывчивость в данном вопросе.

Невымышленная история

В первый год работы после продажи нескольких копий своей новой системы компания получила запросы на изменение системы. Это казалось естественным, пока дело не дошло до вариантов использования и не понадобилось предпринять "мозговой штурм" по поводу участников и интересов в недавно поставленной системе.

Обнаружилось, что во время такого штурма сотрудники компании сами выявили элементы последнего запроса на изменение. Очевидно, во время разработки системы они совершенно упустили из вида некоторые интересы нескольких участников. Эти участники довольно скоро заметили, что система не обслуживала их надлежащим образом, и, естественно, послали запрос на изменение.

С тех пор в компании стали стал неукоснительно именовать участников и интересы на ранней стадии во избежание повторения этой дорогостоящей ошибки.

Мои коллеги и я считаем, что, выявляя участников и их интересы, мы определяем важные требования на ранней стадии. Эти требования другим способом выяснить невозможно. Это не занимает много времени и сберегает много сил в будущем.

4.2. Основное действующее лицо

Основное действующее лицо варианта использования — это участник, который обращается к системе за одной из ее услуг. По отношению к системе он имеет цель, которая может быть достигнута в результате работы этой системы. Основное действующее лицо часто (но не всегда) является тем, кто запускает вариант использования.

Обычно вариант использования стартует вследствие того, что основное действующее лицо посылает сообщение, нажимает на кнопку, на клавишу или инициирует работу варианта использования каким-либо другим способом. Однако существуют две распространенные ситуации, когда инициатором варианта использования является не основное действующее лицо. В первом случае служащий компании или оператор на телефоне инициирует вариант использования от имени кого-то еще, во втором — вариант использования запускается по таймеру.

Наличие служащего компании или телефонного оператора часто удобно с технологической точки зрения для конечного основного действующего лица, которое действительно имеет свой интерес. По мере развития технологии становится более вероятным, что конечное основное действующее лицо будет инициировать или запус-

кать вариант использования непосредственно с помощью Интернета или автоматической телефонной системы. Примером служит клиент, который прямо сейчас звонит и выдает запрос. С системой, переделанной для работы в Интернете, клиент сможет вводить свой запрос напрямую (как в *Amazon.com*).

Подобным же образом подразделение маркетинга или аудита может настаивать на вариантах использования, с которыми работал бы служащий. Вариант использования как таковой служащему не нужен, просто он технологически удобен для менеджеров по продажам. При несколько иных условиях менеджеры сами работали бы с вариантами использования.

Сегодня я пишу "торговый представитель для клиента" или "служащий для отдела маркетинга", чтобы зафиксировать, что пользователь системы действует в интересах кого-то еще. Разработать интерфейс пользователя и определить уровень защиты необходимо для служащего, а в результатах заинтересованы клиент или отдел продаж.

Таймер — другой пример запуска без оператора. Вариант использования запускается каждую полночь или в конце месяца. В этом случае основное действующее лицо — это какой-либо участник, заинтересованный в том, чтобы вариант использования работал в это самое время.

Можно вступить в продолжительную дискуссию и сравнивать пользователей с конечными основными действующими лицами. Предлагаю вам не тратить слишком много времени на это. Если группа начинает исследовать вопросы проектирования интерфейса пользователя, она затрачивает много усилий на изучение реальных характеристик пользователя (или ей следует это делать). Когда разработчики рассмотрят требования, они поймут, что для каждого варианта использования полезно знать конечное основное действующее лицо, т.е. того, кто действительно заинтересован в результате.

Один сообразительный студент спросил: "Насколько велик будет ущерб, если мы на этом этапе неправильно определим конечное основное действующее лицо?" Ущерб будет не слишком большим, как показано в следующем разделе.

Почему основные действующие лица бывают несущественны (и существенны)

Основные действующие лица важны в начале сбора требований и непосредственно перед сдачей системы. Между этими двумя этапами они не важны.

В начале создания вариантов использования

Перечисление основных действующих лиц помогает нам составить мнение о системе в целом за короткий промежуток времени (это довольно скоро выскочит у нас из головы). Мы устраиваем "мозговые штурмы", выявляя сначала всех действующих лиц, а затем — их цели. Это цели, которые нас действительно интересуют, но если мы будем выявлять их напрямую, то слишком многое упустим. Выявление основных действующих лиц определяет рабочую основу. Затем ее можно досконально обсудить, чтобы получить более качественный список целей.

Построение чуть более длинного списка основных действующих лиц делу не мешает, поскольку в худшем случае одна и та же цель будет сформулирована дважды. Когда будем просматривать набор действующих лиц и целей, чтобы назначить приоритеты разработки, обнаружим и удалим дубликаты.

Однако даже при двойном "мозговом штурме" маловероятно, что мы перечислим все цели, достижение которых должна обеспечивать система. Новые цели имеют обыкновение появляться, когда мы пишем шаги обработки ошибок варианта использования, но это невозможно учесть на столь ранней стадии. Лучшее, что мы можем сделать, — это зафиксировать все цели, перечислив сначала всех основных действующих лиц.

Полный список основных действующих лиц дает еще три преимущества:

- Он фокусирует наши мысли на людях, которые будут использовать систему. В документе, содержащем требования, мы фиксируем, кого предполагаем в качестве основных действующих лиц, описание их работы, их типичную подготовку и навыки. Мы делаем это, чтобы разработчики системы и интерфейса пользователя могли обеспечить соответствие системы этим характеристикам.
- Он определяет структуру списка Действующее лицо/Цель, которая будет использована для расстановки приоритетов и распределения работы.
- Он нужен для разбиения огромного множества вариантов использования на пакеты, которые можно распределить между различными группами разработчиков.

Во время создания вариантов использования и во время проектирования

Когда мы начинаем разрабатывать варианты использования в деталях, основные действующие лица практически теряют важность, что удивительно. Дело в том, что со временем создатели вариантов использования обнаруживают, что в них могут быть разные типы действующих лиц. Например, кто-то (не простой служащий) может ответить на звонок и поговорить с клиентом. Поэтому писатели называют основное действующее лицо общим именем, используя такие ролевые имена, как "*приемщик утерянного*", "*приемщик заказов*" или "*производитель счетов*". Как следствие варианты использования сообщают, что производитель счетов производит счет, а приемщик заказов принимает заказ (что не слишком информативно).

Дроблением ролей можно управлять разными способами, каждый из них имеет преимущества и недостатки. Ни одна стратегия не является идеальной, поэтому вам просто придется выбрать одну из них.

АЛЬТЕРНАТИВА 1. Разбейте множество основных действующих лиц в соответствии с их ролями. Постройте таблицу Действующее лицо/Роль, в которой перечислены все люди и системы, являющиеся основными действующими лицами в каких-либо вариантах использования, и все их роли. Используйте названия роли в поле Основное действующее лицо. С помощью таблицы Действующее лицо/Роль перейдите от вариантов использования к людям и системам в реальной жизни.

Эта стратегия позволяет авторам игнорировать сложные названия работ и продолжать писать. Разработчик интерфейса пользователя или компоновщик пакета программного обеспечения применит таблицу Действующее лицо/Роль, чтобы обеспечить соответствие вариантов использования их конечным пользователям. Недостаток альтернативы 1 в том, что приходится поддерживать и читать отдельный список.

АЛЬТЕРНАТИВА 2. В начальном разделе вариантов использования пишем: **"Администратор может выполнять все те варианты использования, которые может выполнять служащий, и сверх того. Региональный управляющий может выполнять все те варианты использования, которые может выполнять администратор, и сверх того. Поэтому, если написано, что основное действующее лицо — это, к примеру, служащий, значит подразумевается, что любой индивидуум должностью выше, администратор и региональный управляющий в данном случае, способен выполнять этот вариант использования".**

Такую форму легче поддерживать, чем таблицу *Действующее лицо/Роль*, поскольку маловероятно, что она изменится. Ее недостаток в том, что люди будут затрачивать некоторое время, напоминая друг другу об этом примечании.

Обе альтернативы обеспечивают адекватные результаты. Я использую вторую, так как меня устраивает, что надо писать, анализировать и поддерживать на один продукт меньше.

Дело в том, что поле основного действующего лица шаблона варианта использования со временем обесценивается. Это нормально, и не стоит об этом беспокоиться.

Завершение проекта, подготовка к внедрению системы

Непосредственно перед сдачей системы основные действующие лица опять становятся важными. Нам необходим список всех людей и вариантов использования, которые они будут выполнять. Нам нужно:

- Разделить систему на модули, которые будут загружаться на различных пользовательских компьютерах.
- Определить уровни безопасности для каждого варианта использования (для Интернет-пользователей, внутренних пользователей, администратора и т.д.).
- Организовать обучение для различных групп пользователей.

Действующие лица в сравнении с ролями

Термин *действующее лицо* подразумевает *индивидуума* в действии. Иногда в варианте использования он означает индивидуум. Однако в других случаях он указывает на общую категорию индивидуумов, которые могут исполнять заданную роль.

Предположим, что Ким — клиент MyTelCo, Крис — служащий, а Пэт —менеджер по продажам. Каждый из них способен разместить заказ. На языке действующих лиц мы скажем, что Ким, Крис и Пат могут быть основными действующими лицами для варианта использования Разместить заказ. Мы также скажем, что клиент, слу-

жащий и менеджер по продажам — допустимые основные действующие лица для варианта использования Разместить заказ. Можно отметить, что менеджер по продажам может выполнять любой вариант использования, который может выполнять служащий. Все это правильно.

Используя язык ролей, мы скажем, что Ким, Крис и Пэт — это действующие лица-индивидуумы. Каждый из них может играть роль клиента, но только Крис и Пэт способны играть роль служащего, и одна только Пэт способна играть роль менеджера по продажам. Управляет вариантом использования Разместить заказ роль приемщика заказов. Этот способ описания более точен, чем предыдущий, и некоторые предпочитают его. Однако в мире вариантов использования он является нестандартным.

Вам следует употреблять такие термины, какие предпочитают разработчики вашей группы. Между тем, *действующее лицо* — термин, принятый в программной инженерии. Он вполне адекватен, поэтому я использую его в этой книге.

Диаграммы UML и специализация Действующее лицо/Роль

В UML используется незакрашенная стрелка, обозначающая, что одно действующее лицо является специализацией другого (см. рис. А.6).

Такая стрелка позволяет кратко выразить, что менеджер может делать все то, что может делать служащий. Достаточно нарисовать стрелку, направленную от *менеджера* к служащему.

Недостатком является то, что многим такая диаграмма кажется перевернутой. Они не считают менеджера *особым видом* служащего или служащего *особым видом* клиента, что, по-видимому, показывает диаграмма (в действительности она показывает, что *один может делать то, что может делать и другой*). Многие полагают, что менеджер — это больше, чем служащий. Это мнение не столь существенно, но вам придется иметь его в виду.

Стрелка специализации вовсе не помогает решить главную часть проблемы Действующее лицо/Роль. Для служащего отдела продаж и служащего отдела аудита наборы вариантов использования перекрываются, но между ними нельзя поставить стрелку специализации, так как никто из них не может делать все, что может другой. Таким образом, вы возвращаетесь к середине дискуссии о действующем лице и роли.

Характеристики основных действующих лиц

Имея только список действующих лиц, нельзя существенно помочь разработчикам, которым надо знать, какими навыками должны обладать пользователи, чтобы построить соответствующие поведение системы и интерфейс пользователя. Группы, создающие *таблицу профиля действующих лиц*, считают, что они лучше представляют себе, насколько их программное обеспечение удовлетворит потребности конечных пользователей, поскольку во время разработки у них перед глазами есть данные о навыках конечных пользователей.

Самая простая таблица профиля действующих имеет две колонки (см. таблицу 4.1). Иногда приводят и вторые имена, или псевдонимы, под которыми известны эти

действующие лица. Изменения в таблице профиля действующих лиц обсуждаются в книге Software for Use (Constantine and Lockwood, 1999).

Таблица 4.1. Пример таблицы профиля действующих лиц

Название	Профиль: подготовка и навыки
Клиент	Человек с улицы, умеющий пользоваться сенсорным дисплеем, но вряд ли имеющий опыт работы с графическим интерфейсом пользователя (GUI). Могут возникнуть трудности с чтением, возможны близорукость, дальтонизм и т.д.
Служащий, оформляющий возврат товаров	Лицо, постоянно работающее с этим программным обеспечением. Умеет работать с сенсорным дисплеем, опытный пользователь. Может захотеть модифицировать интерфейс пользователя.
Менеджер	Нерегулярный пользователь, работал с GUI, но не знаком с какой-либо определенной функцией программного обеспечения. Нетерпелив.

4.3. Вспомогательные действующие лица

Вспомогательное действующее лицо в варианте использования — это внешнее действующее лицо, предоставляющее некоторую услугу для разрабатываемой системы. Это может быть высокоскоростной принтер, служба Интернета или люди, которые должны провести для нас некоторое исследование (например, следователь по делам о насильственной или скоропостижной смерти дает страховой компании заключение о смерти клиента). Обычно мы называем их второстепенными действующими лицами, но некоторым этот термин кажется дезориентирующим. Теперь чаще используется термин *вспомогательное действующее лицо*.

Мы определяем вспомогательное действующее лицо, чтобы указать на внешние интерфейсы, которые будет применять система, и протоколы, лежащие в основе этих интерфейсов. Отсюда дополнительные требования: форматы данных и внешние интерфейсы (см. рис. 1.1).

Действующее лицо может быть основным в одном варианте использования и вспомогательным в другом.

4.4. Разрабатываемая система

Разрабатываемая система сама по себе является действующим лицом —особым. Обычно мы называем ее по имени, например Acura, или просто разрабатываемой системой, SuD. Она именуется или описывается в поле Область действия проектирования варианта использования.

SuD не является основным либо вспомогательным действующее лицом ни для какого варианта использования, несмотря на то, что она — действующее лицо (см. подраздел Область действия проектирования).

4.5. Внутренние действующие лица и варианты использования типа "прозрачный ящик"

Большую часть времени мы рассматриваем разрабатываемую систему как "черный ящик", содержимое которого мы не можем видеть. *Внутренние действующие лица* намеренно не упоминаются, что вполне оправдано, когда мы применяем варнанты использования, чтобы сформулировать требования к еще не разработанной системе.

Иногда нам требуется применить форму варианта использования для документирования взаимодействия частей системы (см. варианты использования 5 и 19). Мы могли бы проделать это, показывая более масштабное проектирование для мультикомпьютерной системы (см. вариант использования 8). При таких условиях компоненты системы проявляются как действующие лица.

Заглядывая внутрь системы и именуя компоненты и их поведение, мы трактуем систему как "прозрачный ящик". Все относительно написания вариантов использования пока работает, только теперь мы обсуждаем поведение как внутренних, так и внешних действующих лиц. В варианте использования "прозрачный ящик" существует более двух действующих лиц, поскольку наряду с внешними действующими лицами раскрыты и компоненты системы.

Исключительно редко и, как правило, по ошибке варианты использования типа "прозрачный ящик" пишут как требования к поведению компьютерной системы, которую предстоит разработать.

4.6. Упражнения

Действующие лица и участники

4.1. Определите вариант использования для торгового автомата, если его владелец является основным действующим лицом.

4.2. Вас наняли, чтобы документнровать требования для нового банкомата. Определите, является ли каждый элемент следующего списка участником, основным действующим лицом, вспомогательным действующим лицом, разрабатываемой снстемой или вовсе не относится к действующим лицам (либо к перечисленному множеству).

Банкомат
Клиент
Карточка для банкомата
Банк
Передняя панель банкомата
Владелец банка
Специалист по техническому обслуживанию
Принтер
Главная банковская компьютерная система
Банковский кассир
Грабитель банков

4.3. Банкомат — компонент более сложной системы. Практически, это часть множества более сложных систем. Повторите предыдущее упражнение для одной такой охватывающей системы.

4.4. Воображаемая компания "Персональные консультанты" выпускает новый продукт, который позволит людям анализировать инвестиционные стратегии, например вложения в пенсионные, образовательные фонды, землю и акции. Продукт, персональный консультант по финансам (PAF), поставляется на CD. Пользователь устанавливает его, а затем проигрывает различные финансовые сценарии, чтобы научиться оптимизировать свое финансовое будущее. PAF может также получать информацию о налоговых законах от различных налоговых пакетов, таких как Kiplinger Tax Cut. Компания заключает соглашение о прямом обмене информацией с компанией Kiplinger, а также соглашения с различными компаниями, оказывающими услуги в Интернете, например Vanguard или E*Trade, чтобы напрямую покупать и продавать ценные бумаги и акции через Интернет. В компании также считают, что необходима действующая в Сети версия PAF, за ее использование следует взимать плату.

Назовите и определите для PAF действующих лиц, основных действующих лиц, разрабатываемую систему и охватывающую систему (в которую PAF входит как компонент).

Глава 5

Три поименованных уровня цели

Цели и взаимодействия в сценарии можно развернуть в более детализированные и более разветвленные цели и взаимодействия. Это нормальное состояние, и мы хорошо с этим справляемся в повседневной жизни. Далее иллюстрируется тот факт, что наши цели содержат под- и подподцели.

> Мне нужен определенный договор о продаже. Чтобы его добиться, я должен пригласить управляющего компании на обед. А для этого мне нужна некоторая сумма наличными. Чтобы ее получить, мне придется извлечь деньги из данного банкомата. Для этого мне нужно, чтобы он признал мою идентичность. А для этого необходимо, чтобы он прочитал мою банковскую карточку. Чтобы сделать это, я должен найти щель для карточки.
>
> Я хочу найти клавишу табуляции, чтобы перевести курсор в поле адреса, чтобы записать мой адрес, чтобы я мог занести мои личные данные в эту программу расценок, чтобы я смог узнать цену, чтобы я смог купить автомобильный страховой полис, чтобы получить водительские права, чтобы водить автомобиль.

Хотя для повседневной жизни здесь нет ничего необычного, в качестве варианта использования это выглядит странно. Создавая варианты использования, мы в каждом предложении сталкиваемся с вопросом, какого уровня цели вариант использования следует описывать.

Помогает именование уровней целей. Далее приведены названия уровней цели и пиктограммы, которые я считаю полезными, а также описано, как определить уровень цели, необходимый в данный момент. На рис. 5.1 даны названия и зрительные образы, которыми я пользуюсь.

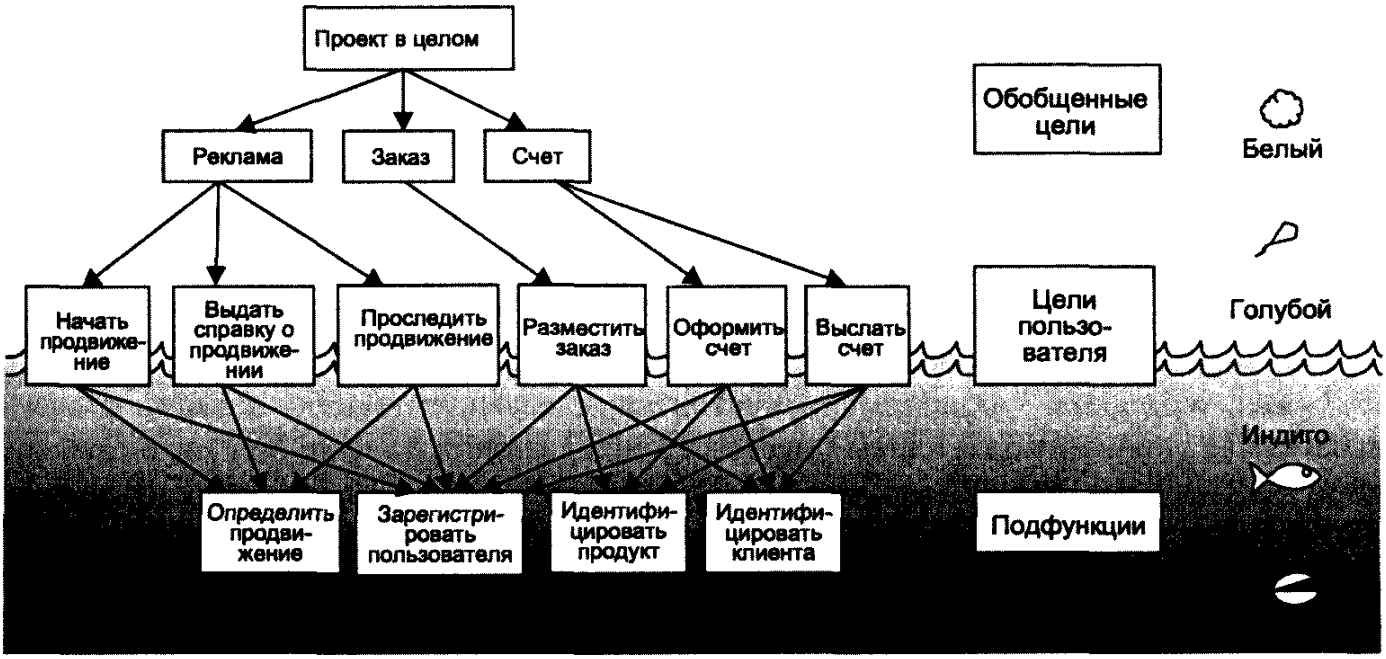


Рис. 5.1. *Уровни вариантов использования. Набор вариантов использования обнаруживает иерархию целей — непрерывно разворачивающуюся историю*

5.1. Цели пользователя (голубые, уровня моря)

Наибольший интерес представляет *цель пользователя*. Это та цель, которую преследует основное действующее лицо, пытаясь добиться от системы выполнения определенной работы, либо пользователь, работающий с системой. Она соответствует "элементарному бизнес-процессу" в технологии бизнесс-процессов.

Цель пользователя характеризуется вопросом: "Уйдет ли основное действующее лицо удовлетворенным, выполнив это?" Для служащего этот вопрос звучит так: "Зависит ли ваша производительность труда от того, сколько вы сегодня сделаете?" Цель можно также пропустить через тест "перерыва на кофе": "Когда это закончу, сделаю перерыв на кофе". В большинстве случаев цель пользователя проходит тест для одного клиента за один сеанс (2 — 20 мин).

Вообще цели "Совершить покупку на онлайновом аукционе" и "Войти в систему" не считаются целями пользователя. Онлайновые аукционы требуют несколько дней и поэтому не проходят односеансовый тест. Вход в систему 42 раза подряд не отвечает (как правило) должностным обязанностям индивидуума или цели использования системы.

"Зарегистрировать нового клиента" и "Купить книгу" вполне могут быть целями пользователей. Задача регистрации 42 новых клиентов не лишена смысла для агента по продажам. Покупку книги можно совершить за один сеанс.

Пока все как будто выглядит просто. Однако, столкнувшись со множеством фраз, написанных на доске, или вариантом использования, который по какой-то причине выглядит неправильно, легко впасть в сомнения. Я считаю, что большинство

людей будут чувствовать себя уверенно, изображая уровни целей в цвете либо применяя обозначения высоты над уровнем моря.

Цветовая гамма охватывает белый, голубой, индиго, черный цвета (в книге как оттенки серого). Цель пользователя — голубая. Долгосрочные цели, имеющие более высокий уровень, например "Завершить онлайновый аукцион" и "Получить страховое возмещение за автомобильную аварию" — белые. Краткосрочные цели, имеющие более низкий уровень, — цвета индиго. Черный указывает, что эта цель такого низкого уровня, что было бы ошибкой писать для нее вариант использования, например: "Нажать на клавишу Tab".

Идея метафоры уровня моря состоит в следующем. Небо находится очень высоко над уровнем моря, а вода опускается ниже уровня моря на большую глубину, но существует только один уровень, где встречаются небо и море, — уровень моря. То же самое справедливо и для целей. Много уровней цели находится над целью пользователя и много под ней, но цели пользователя достаточно важны, чтобы их записать. Поэтому им соответствует уровень моря (знак волн). Облако или воздушный змей обозначают более высокий уровень, чем уровень моря, рыбка или моллюск — это значок уровня цели ниже уровня моря.

Система характеризуется поддержкой целей уровня моря. Вот одна из таких целей:

> Вы служащий, сидящий на рабочем месте. Звонит телефон, вы берете трубку. Некто на другом конце говорит: "...". Вы поворачиваетесь к компьютеру. В этот момент вы думаете о том, что необходимо выполнить работу G. Вы некоторое время работаете с компьютером и клиентом и, наконец, завершаете работу G. Отворачиваетесь от компьютера, прощаетесь и вешаете трубку.

G — голубая (уровня моря) цель пользователя. Выполняя G, вы выполняете ряд целей более низкого уровня (индиго). Персона, позвонившая по телефону, вероятно, имеет в виду более высокий уровень цели, и выполнение G только один шаг к ней. Цели этой персоны, имеющие более высокий уровень, белые.

Цели пользователя уровня моря (голубые) чрезвычайно важны, поэтому стоит серьезно потрудиться, чтобы их понять и усвоить. Кратчайшее изложение функций системы — это список целей пользователя, которые она поддерживает. Это основа для расстановки приоритетов, сдачи системы, разделения работ, оценки и разработки.

Варианты использования будут создаваться на уровнях выше и ниже уровня моря. Можно представить себе огромное число целей более низкого уровня и варианты использования как "подводные", поскольку это подразумевает, что мы на самом деле не хотим описывать их или читать о них.

▪ Невымышленная история

Однажды мне переслали сотни страниц вариантов использования, все цвета индиго, или подводные. Этот документ, описывающий требования, был такой длинный и скучный, что не мог быть полезен ни его авторам, ни его читателям. Позднее отправитель выслал мне шесть вариантов использования уровня моря, которые заменили предыдущий документ, и сообщил, что все нашли новый документ более легким для понимания и работы с ним.

Два уровня голубого

Обычно голубой вариант использования имеет один белый вариант использования выше и несколько вариантов использования цвета индиго ниже. Однако иногда он обращается к другому голубому варианту использования. Я видел это только в одной ситуации, но эта ситуация возникает многократно.

Предположим, выполняя кое-какие поручения, я иду мимо магазина видеопроката. Размышляю: "Раз уж я здесь, зарегистрируюсь". Захожу и прошу оформить мне регистрацию. Такова моя цель пользователя, голубой вариант использования. На следующей неделе прихожу со своим членским билетом, чтобы взять напрокат видео. Я осуществил эти две цели пользователя в разные дни.

Однако вы пользуетесь видеопрокатом по-другому. Заходите в видеомагазин взять напрокат видео. Служащий спрашивает, зарегистрированы ли вы. Вы говорите, что нет, поэтому служащему приходится зарегистрировать вас в процессе выдачи напрокат первого видеофильма. Регистрация представляет собой шаг в рамках выдачи напрокат видео, хотя обе цели голубые.

Ситуация "Попутно зарегистрировать клиента" — единственная ситуация, в которой я видел один голубой вариант использования внутри другого голубого варианта использования. Когда меня об этом спрашивают, я иронически отвечаю, что оба варианта использования имеют уровень моря, но "Взять напрокат видео" сидит на гребне волны (см. рис. 5.1), а "Зарегистрировать клиента" — в ее подошве.

5.2. Обобщенный уровень (белый, облако ☁ или воздушный змей 🪁)

*Цели обобщенного уровня** включают несколько целей пользователя. При описании системы они выполняют три назначения:

- Показывают контекст, в котором выполняются цели пользователя.
- Показывают жизненный цикл последовательности связанных целей.
- Формируют перечень вариантов использования более низкого уровня, как белых, так и голубых.

Обобщенные варианты использования по шкале цветов белые. Белые варианты использования имеют шаги белого, голубого, а порой и цвета индиго ("Войти в систему" — это цель цвета индиго, которая, вероятно, будет обнаружена в белом варианте использования). Я не считаю полезным проводить различия между разнообразными уровнями белого, но иногда могут сказать что-нибудь вроде: "Этот вариант использования по-настоящему белый, белый до того, что поднялся в облака".

---

*. Ранее я использовал два термина: *"стратегический"* и *"обобщенный"*. Недавно я пришел к выводу, что "обобщенный" вызывает меньше путаницы, и выбрал его для этой книги.

В терминах уровня моря мы скажем, что большинство обобщенных вариантов использования похожи на воздушного змея как раз над уровнем моря, а другие поднялись до облаков.

Обобщенные варианты использования обычно выполняются в течение часов, дней, недель, месяцев или лет. Здесь представлен основной сценарий "долгоиграющего" варианта использования, задача которого — связать голубые варианты использования, разбросанные по годам. Обратите внимание, что графика помогает указать, что вариант использования касается компании, рассматриваемой как "черный ящик", и что уровень цели очень белый (высоко в облаках). Подчеркнутые фразы представляют собой варианты использования более низкого уровня. Знак "+" в конце имени — это альтернативный способ обозначения вариантов использования обобщенного уровня (подробнее см. в разделе 5.4).

Вариант использования 18

Выполнить операции со страховым полисом + ...

Основное действующее лицо: клиент
Область действия: страховая компания (MyInsCo)
Уровень: обобщенный (белый)
Шаги:

1. Клиенту назначают цену полиса.
2. Клиент покупает полис.
3. Клиент делает заявление об аннулировании полиса.
4. Клиент закрывает полис.

В этой главе приведены и другие варианты использования:

- Вариант использования 19, Обработать заявление (бизнес)
- Вариант использования 20, Оценить указанную в заявлении сумму
- Вариант использования 21, Обработать заявление (система)

Возвращаемся к предельным вариантам использования

Ранее я рекомендовал вам написать несколько предельных вариантов использования для разрабатываемой вами системы. Ниже представлен более точный процесс обнаружения этих предельных вариантов использования:

1. Начните с цели пользователя.
2. Спросите, какому (предпочтительно вне организации) основному действующему лицу, АА, служит эта цель. Действующее лицо АА — конечное действующее лицо вариантов использования, которые мы хотим собрать.
3. Найдите предельную область действия проектирования S, такую, чтобы АА еще находился вне S. Придумайте название для области действия S. У меня обычно три такие предельные области действия:

- Компания
- Объединяемые программные системы
- Конкретная разрабатываемая программная система

4. Определите все цели пользователей, для которых конечное основное действующее лицо — это AA, а область действия проектирования — S.
5. Выработайте обобщенную цель, GG, которую действующее лицо AA имеет в отношении системы S.
6. Напишите обобщенный вариант использования для цели GG действующего лица AA в отношении системы S. Этот вариант использования связывает ряд вариантов использования уровня моря.

Обычно таких вариантов использования наивысшего уровня (GG) набирается всего четыре-пять даже для самых больших систем. Они обобщают интересы трех-четырех конечных основных действующих лиц (AA):

- Клиента к компании
- Отдела маркетинга к комплексу программных систем
- Отдела безопасности к самой программной системе

Эти предельные варианты использования помогают определить границы работы, и я настоятельно рекомендую писать их по указанным выше причинам. Однако они не обеспечат вашу группу функциональными требованиями к системе, которую она будет разрабатывать, — эти требования заключены в вариантах использования для целей пользователя (голубых).

5.3. Подфункции (индиго или черный, подводный ⋈ или моллюск) ∈

Цели уровня подфункции — это цели, достижение которых требуется для реализации целей пользователей. Включайте их только по мере необходимости. Они иногда нужны для легкого прочтения или потому, что их применяют многие другие варианты использования. Примеры вариантов использования уровня подфункции: *Найти изделие, Найти заказчика и Сохранить в файле* (см., в частности, необычный вариант использования 23 цвета индиго).

Варианты использования уровня подфункции являются подводными, цвета индиго. Некоторые даже лежат на дне. Они окрашены в черный цвет, что означает, что это слишком низкий уровень. Не вздумайте возводить его в степень варианта использования (он даже не плавает... настоящий моллюск). Удобно иметь специальное название для подобных вариантов использования ультранизкого уровня. Если кто-нибудь таковой напишет, вы укажете, что его содержимое должно в свернутом виде войти в другой вариант использования.

Голубые варианты использования имеют шаги цвета индиго, а в вариантах использования индиго встречаются более глубинные шаги индиго, как показано ниже на рис. 5.2. Здесь для обнаружения более высокого уровня цели для вашей целевой фразы следует ответить на вопрос, почему действующее лицо это делает. Метод "как/почему" более подробно обсуждается в разделе 5.5.

Обратите внимание, что даже в самых глубинных подводных вариантах использования для подфункций самого низкого уровня основное действующее лицо находится вне системы. Я бы не стал об этом упоминать, если бы люди иногда не говорили о подфункциях так, будто их проектирование каким-то образом подлежит обсуждению, и если бы не отсутствовало основное действующее лицо. В случае подфункции действуют все правила для вариантов использования. Основным действующим лицом для подфункции может быть то же лицо, что и в варианте использования более высокого уровня, который обращается к данной подфункции.

Обобщение уровней целей

К настоящему моменту важны три пункта относительно уровней целей:

- Серьезно отнеситесь к выявлению вариантов использования уровня моря. Они действительно важны.
- Напишите несколько предельных вариантов использования, чтобы обеспечить контекст для других.
- Не слишком беспокойтесь по поводу того, представлена ли ваша любимая фраза среди описаний требований к системе как название варианта использования.

Быть названием варианта использования — не означает быть самым важным требованием, равно как и не быть названием — не значит быть не важным. Я понимаю, что люди огорчаются, если требование, которое они считают самым важным, стало только шагом в варианте использования и не было развернуто в вариант использования, который прослеживается сам собой.

Не волнуйтесь об этом. Одно из достоинств целевой модели состоит в том, что относительно небольшое изменение превращает сложный фрагмент текста в вариант использования или задвигает тривиальный вариант использования обратно в вариант использования более высокого уровня. Каждое предложение пишется как цель, и каждая цель может быть развернута в собственный вариант использования. Глядя на написанное, мы не можем сказать, какие предложения развернуты, а какие нет (если не следовать связям), и это хорошо, поскольку оберегает целостность написанного от несущественных изменений. Цели, которым гарантируются собственные варианты использования, — это цели только голубого цвета.

5.4. Использование пиктограмм для выделения уровней целей

Выше я продемонстрировал несколько пиктограмм, которые полезно помещать слева от названия варианта использования. Поскольку уровни цели не более одно-

значны, чем названия, я располагаю пиктограмму уровня цели справа вверху от названия. Это в дополнение к заполнению полей шаблона. Читатели (и писатели) используют любые подсказки, чтобы распознать уровень цели.

Придерживаясь высотной терминологии, я выделяю пять высот над уровнем моря. В большинстве случаев вам хватит трех средних.

- Варианты использования очень высокого обобщения (очень белые) обозначаются облаком,  . Используйте этот знак в тех редчайших ситуациях, когда вы видите, что шаги варианта использования сами являются белыми целями. Если нельзя добавить пиктограмму, добавьте знак плюса (+) в конце названия варианта использования (см. вариант использования 18).

- Обобщенные (белые) варианты использования обозначаются знаком воздушного змея,  . Это имеет место для большинства обобщенных вариантов использования, шаги которых представляют собой голубые цели. Добавьте + к названию варианта использования, если не можете применить пиктограмму.

- Варианты использования для цели пользователя (голубые, уровня моря) обозначаются знаком волн,  . Не добавляйте никакого суффикса или добавьте восклицательный знак (!) к названию, если нельзя использовать пиктограмму.

- Варианты использования для подфункций (индиго) обозначаются знаком рыбы,  . Используйте его для большинства вариантов использования индиго. Вместо пиктограммы можно поставить знак минуса (-).

- Некоторые подфункции (черные) никогда не следует описывать в виде вариантов использования. Используйте знак моллюска,  , чтобы пометить вариант использования, который нужно слить с вариантом использования, вызывающим его.

С помощью этих пиктограмм можно пометить область действия проектирования и уровень цели даже в UML-диаграммах вариантов использования, как только поставщики инструментов начнут их поддерживать. А суффиксы можно применять сразу же. Если ваши шаблоны уже содержат поля Область действия использования и Уровень цели, можете использовать их в качестве резервных меток. Если ваши шаблоны не содержат этих полей, добавьте их.

5.5. Выявление правильной цели пользователя

Выявление правильной цели пользователя — труднейшая вещь в вариантах использования. Сосредоточьтесь на следующих правилах:

- Выявите цель пользователя.

- Используйте от трех до 10 шагов на вариант использования.

Выявление цели пользователя

На всех уровнях цели только одна выделяется из других:

> Вы описываете систему (на уровне бизнес-процессов или компьютерную). Вы думаете о том, кто будет систему использовать. Это лицо хочет что-то получить от вашей системы сейчас. Получив это, лицо может продолжить и сделать что-то еще. Что же теперь оно хочет от вашей системы?

Этот уровень имеет много названий. При моделировании бизнес-процессов он называется *элементарным бизнес-процессом*. По-французски это raison d'etre (основание быть) системы. В вариантах использования это цель пользователя.

Спросите себя, является ли это тем, чего действительно сейчас хочет от системы основное действующее лицо? Для большей части первых наметок вариантов использования ответ будет отрицательным. Очень часто начинающие набрасывают подводные варианты использования, думая, что те находятся на уровне моря. Чтобы обнаружить цель более высокого уровня, задайте себе любой из этих вопросов:

- Чего на самом деле хочет действующее лицо?
- Почему действующее лицо это делает?

Ответ может быть настоящей целью действующего лица, но задавайте вопрос снова, пока не будете уверены в этом. Интересно, что даже если тесты для цели пользователя субъективны, люди вскоре приходят к согласию по этому вопросу. Специалисты дают удивительно похожие ответы на вопросы о целях пользователей. Кажется, это устойчивое понятие.

Повышение и понижение уровней целей

Шаги варианта использования описывают, как осуществляется процесс. Название варианта использования указывает, почему этот процесс представляет интерес. Можно обозначить эту связь "как/почему" во время поиска подходящего уровня цели для шага (см. рис. 5.2).

Чтобы повысить уровень цели одного или нескольких шагов, спросите, почему действующее лицо это делает. Ответом будет цель одним уровнем выше.

Способ оценить уровни целей — посмотреть на длину варианта использования. Большинство хорошо написанных вариантов использования имеют от двух до восьми шагов. Я никогда не видел варианта использования длиннее, чем 11 шагов, который бы не выигрывал от сокращения. Сомневаюсь, что в этих числах есть что-нибудь магическое, но я предполагаю, что люди не желают или не умеют думать в терминах процессов, которые требуют более 10 промежуточных шагов. Я дожидаюсь серьезного контрпримера, только чтобы доказать, что числа не оказывают глубокого влияния.

Какой бы ни была причина, имейте в виду это наблюдение, чтобы улучшить ваши варианты использования. Если у вас больше 10 шагов, вы, возможно, включили детали интерфейса пользователя или описали шаги действий на слишком низком уровне.

- Удалите подробности интерфейса пользователя. Покажите намерение действующего лица, а не его перемещение.
- Повысьте уровень цели, задавая вопрос "почему", чтобы определить следующий более высокий уровень цели.
- Объедините шаги.
- Сравните ваши варианты использования с образцами из раздела 5.6 и из главы 19.

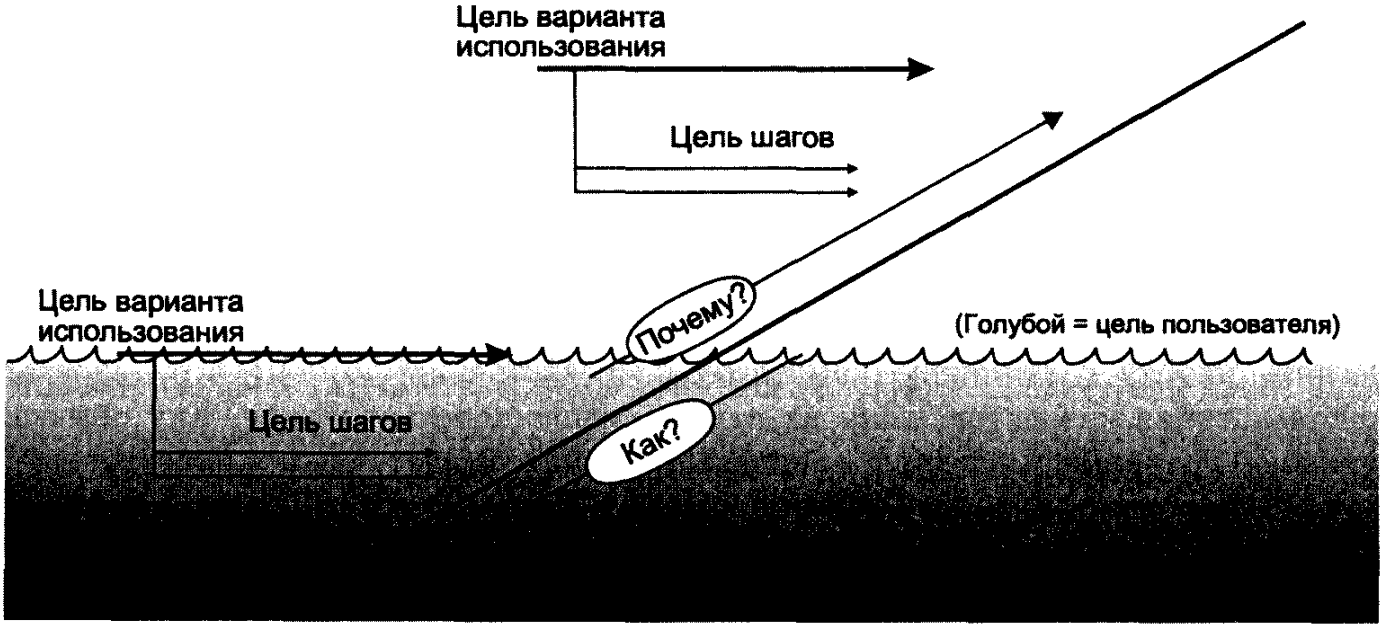


Рис. 5.2. *Спросите "почему", чтобы изменить уровни*

5.6. Более длинный пример: "Обработка заявления" на нескольких уровнях

Я выражаю благодарность сотрудникам страховой корпорации пожарных Fund Insurance Corporation из Новато, Калифорния, разрешивших мне включить варианты использования 19-23 в качестве примеров*. Они были написаны специалистами по обработке заявлений в этой области, работающими с бизнес-аналитиками из отдела IT и коллективом разработчиков. Специалисты предметной области глубоко понимают суть использования системы, о которой специалисты из IT не могли догадываться. В свою очередь, сотрудники из IT помогли экспертам предметной области четко сформулировать варианты использования. Оба коллектива обеспечили сочетание предметной, корпоративной и технической сторон.

Пишущая команда состояла из Кэрри Биэр, Эйлин Каррен, Брента Хаппа, Полы Айвей, Сьюзен Пассини, Памелы Пратт, Стива Сэмпсона, Джилл Шиктанц, Нэнси Джуэлл, Тришы Мадалено, Марка Гринберга, Николь Лазар, Дона Копполо и Эрика

---

* Копирайт © 1999 Fireman's Fund, Novato, CF. Печатается с разрешения.

Иванса. Они демонстрируют, что эксперты в предметной области без навыков программирования могут работать со специалистами из IT над требованиями к системе.

Я включаю пять вариантов использования для иллюстрации того, что мы к настоящему моменту обсудили, особенно областей действия проектирования и уровней целей. Эти варианты использования также демонстрируют хороший стиль написания для шагов и расширений. Каждый вариант использования я предваряю комментарием, указывая на некоторые интересные или спорные моменты.

Набор начинается с варианта использования для бизнеса уровня облака типа "прозрачный ящик". Посмотрите, как цели переходят на более низкие уровни и как область действия системы сжимается от "работы компании" до "всех компьютерных систем" и далее всего-навсего до "разрабатываемой системы". Подчеркнутые фразы представляют собой обращения к другим вариантам использования. Шаблон был немного изменен, чтобы основной сценарий успеха был ближе к началу и быстрее читался.

КОММЕНТАРИЙ К ВАРИАНТУ ИСПОЛЬЗОВАНИЯ 19. SuD — это работа компании. Обратите внимание, что компьютерная система даже не упоминается. Вариант использования будет применяться в бизнесе, чтобы скрепить бизнес-процедуры и облегчить работу с помощью компьютера. В настоящий момент этот вариант использования находится лишь на первом этапе проектирования. Как обычно, основной сценарий выглядит тривиальным. Он показывает, как все работает в самой благоприятной ситуации. Интересные моменты откроются в ситуациях отказа и когда компания будет использовать эту информацию для внесения поправок в поддержку эксплуатации системы. Обратите внимание на участников.

Вариант использования 19

Обработка заявления (бизнес)

Область действия: работа страховой компании
Уровень: обобщенный для бизнеса
Версия: будущая
Статус: набросок
Ревизия: текущая
Контекст использования: оценщик обрабатывает заявление.
Предусловия: понесен ущерб.
Триггер: заявление подано в страховую компанию.
Основной сценарий:

1. Сторона заявителя, знающая о событии, регистрирует ущерб в страховой компании.
2. Служащий принимает и распределяет заявление оценщику.
3. Назначенный оценщик
   проводит расследование
   оценивает сумму возмещения ущерба

определяет резервы
ведет переговоры по заявлению
договаривается о сумме возмещения и закрывает заявление

Расширения:

последуют в дальнейшем.

Гарантия успеха: устанавливается сумма возмещения и дело закрывается

Минимальная гарантия: нет.

Участники и интересы:

Подразделения страховой компании, продающие ее страховые полисы

Клиенты страховой компании, покупающие полисы

Министерство страхования, осуществляющее руководство рынком страховых услуг

Претенденты, понесшие ущерб в результате действия застрахованного лица

Отдел заявлений страховой компании

Будущие клиенты

КОММЕНТАРИЙ К ВАРИАНТУ ИСПОЛЬЗОВАНИЯ 20. Ниже следует еще один вариант использования, в котором SuD все еще является работой компании. Однако цель находится на более низком уровне, чем в варианте использования 19. Здесь описывается работа оценщика, которая может длиться дни, недели или месяцы. Это обобщенный вариант использования уровня воздушного змея, поскольку содержит много односеансовых действий.

Автор не говорит о компьютере прямо, а только называет цели оценщика. Разработчики должны придумать, в чем может заключаться помощь компьютера в этом процессе. Этот вариант использования служит им сырьем для изобретения.

Шаг 7 был добавлен в связи с интересами Министерства страхования.

Вариант использования 20

Произвести оценку ущерба по заявлению

Область действия: работа страховой компании

Уровень: белый (обобщенный, выше уровня цели единичного пользователя)

Контекст использования: оценщик производит всестороннюю оценку обстоятельств понесенного ущерба.

Основное действующее лицо: оценщик

Предусловия: документирование последует

Триггер: документирование последует

Основной сценарий:

Обратите внимание: идеально, когда перед оценкой проведено расследование, однако глубина расследования может изменяться от заявления к заявлению.

1. Оценщик просматривает и оценивает медицинские отчеты, документы на арестованное имущество, прибыль на данное число и другие документированные доказательства.
2. Оценщик оценивает постоянную неплатежеспособность, используя юридическую формулу для определения процента неплатежеспособности.
3. Оценщик определяет сумму долга при постоянной неплатежеспособности, беря кредит для авансированных сумм и платы за арестованное имущество, чтобы установить полную сумму по заявлению.
4. Оценщик определяет окончательный диапазон суммы возмещения.
5. Оценщик проверяет резервы, чтобы убедиться, что компания в состоянии уплатить сумму в установленных пределах.
6. Оценщик добивается утверждения установленной суммы выплаты и увеличения резерва, если это входит в его полномочия.
7. Оценщик приобщает к делу документы.
8. Оценщик посылает корреспонденцию и (или) документацию заинтересованным сторонам, если это необходимо.
9. Оценщик продолжает приобщать к делу документы, касающиеся всех действий по урегулированию.

Расширения: ...

Частота события: оценивается каждое заявление; это может происходить несколько раз в день.

Гарантия успеха: заявление оценивается и определяются пределы величины возмещения.

Минимальная гарантия: прежде чем начнется повторная оценка заявления для урегулирования претензии, будет произведено дополнительное расследование или составлено медицинское заключение.

Участники и интересы:

Претендент — хочет получить максимальную сумму возмещения.

Оценщик — хочет установить минимально допустимую сумму возмещения.

Страховая компания — то же самое.

Поверенный (защита и обвинение) застрахованных лиц.

Отдел страхования и органы государственного управления штатов (в каждом штате свой департамент, надзирающий за законностью ведения дел и выплачиваемых сумм) охраняют законность и строгое соблюдение процедур.

Открытые вопросы: при разработке бизнес-правил придется обращаться к вопросам юрисдикции.

КОММЕНТАРИЙ К ВАРИАНТУ ИСПОЛЬЗОВАНИЯ 21. Многим специалистам, занятым в проекте, этот системный вариант использования показался расплывчатым до бесполезности. Однако время на его создание было потрачено не зря, и тому есть несколько причин.

Во-первых, он объединил ряд вариантов использования для пользователя в соответствии с бизнес-правилами. Он описывает процессы закрытия, очистки и архивирования заявления, которые ряду участников проекта просто не известны. Хотя три последних шага не сильно загружают работой программистов, они являются частью процедуры обработки заявления, полезной контекстуальной информацией для любого читателя.

Во-вторых, он записывает в служебные файлы бизнес-правила, не знакомые некоторым разработчикам. Накануне группа потратила три рабочих часа, пытаясь выяснить, что это за правила. Этот вариант использования сберег гораздо больше времени (которое иначе ушло бы на обсуждение), чем было потрачено на его создание.

Этот вариант использования служит введением и оглавлением для Читателей, от руководителей до новых работников компании. Руководители увидят, что в него вошли ключевые процессы, а новички узнают, как работает компания и углубятся в варианты использования для целей пользователя. Интересно расширение *a1, так как оно вызывает вариант использования обработки отказов, который не может быть написан оценщиками, а должен быть создан в группе разработчиков.

Вариант использования 21

Обработка заявления (системы) +

Область действия: "система" как комплекс всех компьютерных систем

Уровень: обобщенный (белый)

Версия: 1-я

Статус: готов для анализа

Ревизия: текущая

Контекст использования: клиент хочет получить возмещение за страховое событие.

Основное действующее лицо: клиент

Предусловия: нет

Триггер: клиент подает заявление.

Основной сценарий:

1. Клиент подает заявление (на бумаге, по телефону или факсу) служащему.
2. Служащий находит полис, регистрирует заявление в системе и назначает оценщика.
3. Оценщик расследует заявление и обновляет его с помощью дополнительной информации.
4. Оценщик вводит информацию по ходу расследования.
5. Оценщик корректирует записи и резервирует сумму по ходу расследования.
6. Оценщик получает документацию, включая счета за период срока действия заявления, и вводит счета.

7. Оценщик оценивает ущерб по заявлению и документирует процесс переговоров в системе.
8. Оценщик решает вопрос с возмещением и закрывает дело в системе.
9. Система удаляет заявление из базы данных спустя 6 месяцев после закрытия дела.
10. Система архивирует заявление по прошествии установленного периода.

Расширения:

*а. В любое время система выходит из строя:

*а1. Группа системной поддержки восстанавливает систему.

1а. Представленная информация неполна:

1а1. Страховая компания запрашивает недостающую информацию.

1а2. Претендент предоставляет недостающую информацию.

1а2а. Претендент не предоставляет информацию в течение установленного периода.

1а2а1. Оценщик закрывает дело в системе.

2а. У претендента недействительный полис:

2а1. Страховая компания отклоняет заявление, уведомляет претендента, обновляет заявление, закрывает дело.

3а. На данный момент нет свободных агентов:

3а1. (Что мы здесь делаем?)

8а. Претендент уведомляет оценщика о новых действиях по поводу возмещения:

8а1. Служащий открывает дело повторно. Возвращается к шагу 3.

Список изменений в технологии и данных:

Частота события: документирование последует.

Гарантия успеха: дело закрыто, урегулировано и помещено в архив.

Минимальная гарантия: дело закрыто, но может быть открыто позднее.

Участники и интересы:

Компания — выплатить минимально допустимое возмещение.

Клиент — получить максимальное возмещение.

Министерство страхования — гарантировать корректность процедур.

Бизнес-правила:

Дата описаний: будет установлена в других вариантах использования.

Связи с интерфейсом пользователя (UI): предстоит документировать.

Открытые вопросы: период, по окончании которого следует помещать дело в архив.

КОММЕНТАРИИ К ВАРИАНТУ ИСПОЛЬЗОВАНИЯ 22. Это один из самых сложных вариантов использования, которые я видел. Он показывает, почему варианты использования следует описывать естественным языком.

Главный источник сложности — порядок следования. Служащий на телефоне, говорящий с обезумевшим клиентом, должен быть в состоянии вводить данные в лю-

бом порядке, в то же время пытаясь следовать стандартной последовательности вопросов. Одновременно компьютер использует эти данные по мере их ввода для возможной на данный момент обработки, например сбор записей по этому клиенту, присвоение заявлению номера и назначение оценщика. Авторы написали не меньше четырех полных версий этого варианта использования, добиваясь ясности, чтобы показать нормальное течение работы и то, что компьютер работает асинхронно. Возможно, на седьмой-восьмой ревизии они нашли бы что-то лучшее, но почувствовали, что дальнейшее умножение ревизий неэффективно, и остановились на этой версии.

Этот вариант использования вызывает вариант использования Найти что угодно несколько раз, задавая при каждом вызове различные объекты и критерии поиска. Разработчики придумали остроумное решение, чтобы не переписывать несколько раз стандартные шаги поиска (списки совпадений, критерии сортировки, повторная сортировка, повторный поиск, объект не найден и т.д.). Я прошу вас сделать то же самое в упражнении 5.4 в конце главы.

Обработка расширения ***a. Сбой питания** неожиданно стал причиной появления новых проблем с требованиями. Было введено понятие промежуточных сохранений, которые служащему пришлось потом отыскивать. Это стало сюрпризом для писателей, а дополнительная проблема хранения и поиска временно хранимых данных — сюрпризом для разработчиков. Все это вылилось в условие отказа **6b**, которое касалось тайм-аута для временно хранимых данных и ставило перед писателями очень конкретный вопрос: "Каковы бизнес-правила для временно введенных данных? Они не могут быть зафиксированы, поскольку недостает важной информации, но и удалить их нельзя, так как они удовлетворяют минимальному критерию ввода". Разработчики так и сяк вертели неприемлемые альтернативы (не делать промежуточных сохранений и удалять данные), пока не решили эту проблему.

Расширение **1c** демонстрирует отказы внутри отказов. Авторы могли бы превратить это в собственный вариант использования, но они решили, что это сильно усложнит набор вариантов использования (новый вариант использования пришлось бы отслеживать, анализировать и поддерживать). Вместо этого они сделали его расширением расширения. Многие применяют расширения варианта использования по этой причине, хотя в большинстве случаев это объясняется тем, что авторам удобнее для начала сделать собственный вариант использования расширением.

Расширение **2-5a** демонстрирует, как податлива среда. Условие может возникнуть на любом шаге от второго до пятого. Как его следует описывать — один раз для каждого случая возникновения? Это представляется слишком расточительным. Было решено просто написать понятные читателям расширения **2-5a** и **2-5b**.

Вариант использования 22

Зарегистрировать ущерб

Область действия: "система" означает компьютерную систему обработки заявлений

Уровень: голубой (цель пользователя)

Версия: 2

Статус: проанализирован

Ревизия: текущая

Контекст использования: фиксирование полной информации об ущербе.

Основное действующее лицо: служащий

Предусловия: служащий уже вошел в систему.

Триггер: служащий уже начал вводить данные об ущербе.

Гарантия успеха: данные об ущербе регистрируются и сохраняются.

Минимальная гарантия: ничего не происходит.

Участники и интересы: как выше

Основной сценарий: Чтобы ускорить работу служащего, система должна работать асинхронно, как только будут записаны необходимые данные. Служащий может вводить данные в любом порядке в соответствии с требованиями момента. Следующая последовательность представляется наиболее вероятной.

1. Служащий вводит номер полиса застрахованного лица и, возможно, его имя и дату страхового события. Система анализирует доступную информацию о полисе и указывает, что заявление соответствует полису.
2. Служащий вводит основные данные об ущербе. Система подтверждает, что не существует потенциально конкурирующих заявлений, и присваивает заявлению номер.
3. Служащий продолжает вводить информацию об ущербе, соответствующую заявлению данного типа.
4. Служащий использует систему для получения дополнительной информации по делу из других компьютерных систем.
5. Служащий выбирает и назначает оценщика.
6. Служащий подтверждает, что он закончил; система сохраняет данные и инициирует отправку уведомления агенту.

Расширения:

*a. Сбой питания во время обработки заявления:

*a1. Система периодически автоматически сохраняет данные (возможно, в определенных точках фиксации транзакции, открытый вопрос).

*b. Заявление не подлежит обработке в нашей компании:

*b1. Служащий указывает системе, что вводится заявление "только с целью записи" и продолжает либо завершает обработку заявления.

1a. Найденная информация о полисе не соответствует информации застрахованного лица:

1a1. Служащий вводит правильные номер полиса или имя застрахованного лица и просит систему проверить данные с новым индексом полиса.

1b. По имеющимся данным система не смогла найти полис:

1b1. Служащий возвращается к заявлению и вводит имеющиеся данные.

1c. Служащий изменил номер полиса, дату страхового события или тип заявления по сравнению с данными предыдущего поиска:

1c1. Система проверяет правильность изменений, анализирует страховое событие с правильной информацией о полисе, проверяет и подтверждает, что заявление соответствует полису.

1c1a. Система не может подтвердить соответствие полису:

1c1a1. Система предупреждает служащего.

1c1a2. Служащий отыскивает полис, используя параметры поиска для полиса.

1c2. Система предупреждает служащего о необходимости пересмотреть сумму возмещения.

1d. Служащий хочет повторно запустить обработку заявления, которая была прервана и данные по которой были сохранены или которую следует завершить:

1d1. Служащий находит дело с помощью параметров поиска для заявления.

1d2. Система открывает данные для редактирования.

2-5a. Служащий изменяет ранее введенный тип заявления и данные, относящиеся к новому типу, не вводятся:

2-5a1. Система представляет соответствующие поля дела, относящиеся к нововведенному служащим типу заявления.

2-5b. Служащий изменяет ранее введенный тип заявления, и в связанных с типом полях появляются данные:

2-5b1. Система предупреждает, что данные существуют, и предлагает служащему отменить изменения либо продолжить с новым типом заявления.

2-5b1a. Служащий отменяет изменение, система продолжает обработку заявления.

2-5b1b. Служащий настаивает на новом типе заявления, система стирает определяемые типом данные (но сохраняет все основные данные заявления).

2c. Система обнаруживает потенциальный дубликат заявления:

2c1. Система показывает список возможных заявлений-дубликатов из базы данных заявлений.

2c2. Служащий выбирает и просматривает заявление из списка. Этот шаг может повторяться несколько раз.

2c2a. Служащий обнаруживает, что это заявление — дубликат: Служащий открывает заявление-дубликат из списка заявлений для редактирования, если оно не помечено как обработанное (в соответствии с уровнем безопасности данных для служащего). Служащий может удалить любые данные в ранее сохраненном файле.

2c2b. Служащий обнаруживает, что заявление не является дубликатом: служащий возвращается к заявлению и завершает его обработку.

2d. Предварительная информация об ущербе изменяется после первой проверки заявления-дубликата:

2d1. Система снова проверяет заявление-дубликат.

2e. Служащий может сохранять данные по заявлению несколько раз, пока не завершит шаги 2-6. (Иногда он может сохранять данные просто для удобства или ввод приходится по какой-то причине прервать, например, заявление должно быть немедленно передано на обработку оценщику более высокого класса.)

2e1. Служащий сохраняет данные по заявлению, чтобы завершить обработку позднее.

4-5a. Тип заявления либо описание ущерба (см. бизнес-правила) изменились, с тех пор как служащий анализировал сумму возмещения:

4-5a1. Система предупреждает служащего, что надо пересмотреть сумму возмещения.

6a. Служащий подтверждает, что он закончил, не завершив ввод минимума информации:

6a1. Система предупреждает служащего, что не может принять заявление без даты страхового события, имени застрахованного лица или номера полиса и назначения оценщика.

6a1a. Служащий решает продолжить ввод данных по заявлению или сохранить данные без отметки о завершении.

6a1b. Служащий настаивает на выходе без ввода минимума информации, система аннулирует все сохраненные промежуточные данные и прекращает работу.

6a2. Система предупреждает служащего, что он не может присваивать заявлению номер, не заполнив обязательных полей (тип заявления, дату страхового события, номер полиса или имя застрахованного лица); система фокусирует внимание служащего на полях, требующих ввода.

6b. Тайм-аут: служащий временно сохранил данные по заявлению, намереваясь вернуться к обработке; система решила, что самое время зафиксировать данные по заявлению, но оказалось, что не введено имя оценщика:

6b1. Система назначает оценщика по умолчанию (см. бизнес-правила).

Частота события: ??

Бизнес-правила:

*. Когда сохраненные данные по заявлению передаются главной системе (временные диаграммы)?

1. Минимум попей, необходимых для сохранения данных по заявлению (и чтобы их можно было потом найти): ...
2. Номер заявления, однажды присвоенный системой, не может измениться.
3. Бизнес-правила для ручного ввода номера заявления — понадобится?
4. Описание ущерба состоит из двух полей: одно свободной формы, другое — ниспадающее меню.

5. Система должна знать, как определить сумму возмещения, исходя из префикса полиса.
6. Поля, которые необходимо заполнить, чтобы подтвердить ущерб: ...

6b. Правила назначения оценщика по умолчанию: ...

Используемые описания данных:

Параметры поиска полиса, информация об индексе полиса, предварительная информация об ущербе, информация об ущербе, обусловленная типом заявления, дополнительная информация.

Параметры поиска заявления, критерий проверки на заявление-дубликат, список возможных заявлений-дубликатов, заявление из списка.

Связи с UI: предстоит документировать

Владелец: Сьюзен и Нэнси

Рецензенты: Элистер, Эрик ...

Открытые вопросы:

Как часто происходит автосохранение?

Карточки официального уведомления агентов — где и как их печатают и т.д.?

Группа разработчиков решила, что было бы глупо писать почти идентичные варианты использования Найти клиента, Найти полис и т.д. Вместо этого они создали настраиваемый алгоритм, который применяла каждая команда писателей. Любое предложение в форме **Найти ...**, например **Найти клиента** или **Найти изделие**, означает вызов варианта использования Найти что угодно с неявным замещением "*чего-то*" специальным термином. Каждому варианту использования понадобится свой критерий поиска, сортировки и вывода на экран, поэтому писатель будет записывать данные и ограничения на различных листах с гипертекстовыми связями. Поэтому пример предложения будет выглядеть как Найти клиента, используя Параметры поиска клиента.

При этом четком соглашении логику варианта использования *Найти что угодно* можно написать только один раз и применять во многих похожих, но неидентичных контекстах. Это очень понравилось разработчикам, так как это означает, что для всех видов поиска им придется разработать только один алгоритм. Попробуйте на этом примере свои силы. Пока не буду показывать данное решение. Поработайте над упражнением 5.4, прежде чем заглядывать в решение, которое обсуждается в разделе 14.2.

Вариант использования 23

Найти что угодно (постановка задачи)

Заполнить в качестве упражнения.

5.7. Упражнения

Уровни цели

5.1. Дженни стоит перед банкоматом своего банка. Темно. Она ввела свой PIN-код и ищет кнопку "Ввод". Назовите для Дженни цели белую, голубую и цвета индиго.

5.2. Перечислите не менее десяти целей различных основных действующих лиц в отношении банкомата и укажите их уровни целей.

5.3. Перечислите обобщенные цели и цели пользователей всех основных действующих лиц для программного обеспечения PAF (персонального консультанта по финансам), описанного в упражнении 4.4. Определите самый высокий уровень, предельные комбинации "действующее лицо/сфера/цель".

5.4. Найдите что угодно. Напишите вариант использования Найти что угодно, триггером которого является желание пользователя локализовать нечто. Вариант использования должен позволить пользователю вводить параметры поиска и сортировки. Он также должен иметь дело со всеми ситуациями, которые могут возникнуть, и в случае успеха завершиться определением компьютером этого "чего-то" независимо от дальнейшего его применения в вызывающем варианте использования.

Глава 6

Предусловия, триггеры и гарантии

6.1. Предусловия

Предусловие варианта использования объявляет, выполнение какого условия гарантирует система перед тем, как разрешить запуск этого варианта использования. Так как условие выполняется системой, и известно, что оно истинно, то оно не будет проверяться снова в период выполнения варианта использования. Общеизвестный пример: пользователь уже вошел в систему и система идентифицировала его.

Вообще предусловие определяет, что другой вариант использования уже отработал, чтобы установить его. Будем говорить, что вариант использования *Разместить заказ* полагается на предусловие входа в систему. Я сразу же смотрю, какой вариант использования установил его (я бы поискал Войти в систему). Лично я обычно создаю вариант использования более высокого уровня, в котором упоминаются варианты использования *Разместить заказ* и *Войти в систему*, чтобы читатель видел, как они соотносятся друг с другом. В этом примере это может быть обобщенный вариант использования *Работать с приложением*. В этом случае мы применяем приведенную ниже структуру (я сокращаю шаблон, чтобы просто показать существенные части). Обратите внимание на уровни целей трех вариантов использования.

Вариант использования

Работать с приложением

Уровень: обобщенный (белый)

Предусловие: отсутствует

1. Служащий входит в систему.
2. Служащий размещает заказ.

Вариант использования

Войти в систему

Уровень: подфункция (индиго)
Предусловие: отсутствует
...

Вариант использования

Разместить заказ

Уровень: цель пользователя (голубой)
Предусловие: служащий вошел в систему. ...

Не все следуют моему правилу писать более высокий вариант использования, чтобы собрать варианты использования более низкого уровня. Таким образом, вы можете включить в предыдущий пример только варианты использования *Войти в систему* и *Разместить заказ*. Вы должны уметь вывести из документа, что предусловие правильно и будет обязательно выполнено.

Продолжим этот пример, чтобы показать вариант использования, устанавливающий условие в одном шаге и полагающийся на это условие в подчиненном варианте использования. И опять подчиненный вариант использования рассчитан на то, что условие истинно, и не будет поверять наличие ошибок.

Вариант использования

Разместить заказ

Уровень: цель пользователя (голубой)
Предусловие: служащий вошел в систему.
1. Служащий идентифицирует клиента, система находит данные пользователя.
2. Служащий вводит информацию о заказе.
3. Система рассчитывает расценки.

...

Вариант использования

Рассчитать расценки

Уровень: подфункция (индиго)
Предусловие: клиент установлен и известен системе; содержимое заказа известно.
1. Система рассчитывает базовую расценку для заказа.
2. Система рассчитывает скидку для клиента.

...

Этот пример иллюстрирует, как один вариант использования полагается на информацию, фиксируемую в вызывающем варианте использования. Автор варианта использования *Рассчитать расценки* объявил информацию, которая уже имеется, и теперь может двигаться вперед и обращаться к данным клиента.

Внимательный читатель недоверчиво отнесется к варианту использования *Рассчитать расценки*. Я сказал, что он цвета индиго, но до сих пор неясно даже, является ли он самостоятельным вариантом использования. Если я как автор не раскрою сложных взаимодействий с пользователем или интересных случаев отказа, я переклассифицирую его как черный (пиктограмма моллюска). Это послужит сигналом включить этот текст обратно в вариант использования Разместить заказ и уничтожить вариант использования *Рассчитать расценки*.

Пишите предусловие как простое утверждение о состоянии окружающей среды на момент открытия варианта использования. Вот некоторые примеры используемых предусловий:

Предусловие: пользователь вошел в систему.

Предусловие: клиент идентифицирован.

Предусловие: система уже локализовала данные полиса клиента.

Распространенной ошибкой является записывать в предусловие утверждение, которое часто, но не обязательно бывает истинным.

Допустим, мы пишем вариант использования *Затребовать сводку по вознаграждениям* с заявителем в качестве основного действующего лица. Логично предположить, что заявитель подал по крайней мере одно заявление или счет, прежде чем просить выдать ему сводку вознаграждений. Однако это не всегда так. Система не может это гарантировать, и на самом деле это несущественно. Заявителям следует предоставить возможность потребовать сводку их вознаграждений в любое время, поэтому неправильно было бы написать предусловие **Заявитель подал счет.**

6.2. Минимальные гарантии

Минимальные гарантии — это наименьшие обещания системы участникам, в Частности, если цель основного действующего лица не может быть достигнута. Конечно, они действительны и при достижении цели, но реальный к ним интерес возникает, если основная цель не может быть выполнена. Большую часть времени два или более участников вынуждены иметь дело с минимальными гарантиями, например пользователь, компания, поставляющая систему, и, возможно, органы государственного управления.

Не старайтесь перечислить в разделе Минимальные гарантии все возможные для данного варианта использования пути отказа. Существуют десятки путей, и они имеют мало общего. Все условия отказа и обработка отказов раскрываются в разделе расширений, и поддерживать синхронизацию этих двух списков утомительно и одновременно чревато ошибками. Задача этого раздела шаблона — объявить то, что обещает система.

Наиболее распространенной минимальной гарантией является следующая: "Система зарегистрировала точку, до которой она продвинулась". Регистрация сбоев транзакций не является ни очевидной, ни тривиальной. В описании требований часто забывают о системных журналах, а потом программисты могут о них вспомнить. Однако они чрезвычайно важны как для владельцев системы, так и для ее пользователей. Система использует журнал, чтобы продолжить транзакцию после восстановления нормальных условий работы; участникам он нужен для урегулирования спорных вопросов. Автор варианта использования должен определить, насколько необходимо использовать журнал, выяснив интересы участников либо хорошо обдумав условия отказов.

Минимальные гарантии записываются в виде ряда простых утверждений, которые будут истинны по окончании каждого варианта использования. Они показывают, что интересы каждого участника удовлетворены.

Минимальные гарантии: заказ будет инициирован только по получении оплаты.

Минимальные гарантии: если не был зафиксирован минимум информации, неполное заявление отклоняется, и регистрация заявления не проводится; если был зафиксирован минимум информации (см. бизнес-правила), неполное заявление будет сохранено и зарегистрировано.

Тест "прошел/не прошел" для раздела минимальной гарантии состоит в том, что участники соглашаются с тем, что их интересы защищены при возникновении условий отказа для данной цели.

6.3. Гарантия успеха

Гарантия успеха устанавливает, что интересы участников удовлетворяются при успешном завершении варианта использования в конце основного сценария или в конце успешного альтернативного пути. Обычно она пишется в добавление к минимальным гарантиям: предоставляются минимальные гарантии и выполняются некоторые дополнительные условия. Эти дополнительные условия включают по крайней мере цель, объявленную в заголовке варианта использования.

Подобно минимальным гарантиям, гарантия успеха записывается как набор простых утверждений, которые применяются в конце успешного выполнения варианта использования и показывают, что интересы каждого участника удовлетворены. Например:

> **Гарантия успеха:** претенденту будет выплачено возмещение, соответствующее договоренности, заявление закрыто, соглашение зарегистрировано.
>
> **Гарантия успеха:** файл будет сохранен.
>
> **Гарантия успеха:** система инициирует для клиента заказ, получив информацию об оплате и зарегистрировав запрос на заказ.

Тест "прошел/не прошел" для раздела гарантии успеха состоит в том, что участники соглашаются с тем, что их интересы удовлетворены.

Лучший способ раскрыть гарантию успеха — это спросить: "Что сделало бы этого участника несчастным по окончании успешного выполнения варианта исполь-

зования?" На этот вопрос обычно легко ответить. Затем напишите отрицание ответа. Чтобы посмотреть пример, выполните упражнение 6.4, а потом прочтите обсуждение этой темы в приложении В.

6.4. Триггеры

Триггер определяет событие, которое запускает вариант использования. Иногда триггер предшествует первому шагу варианта использования, в другой раз сам является первым шагом. До настоящего времени я не встречал убедительного доказательства, что одну форму можно применять во всех случаях. Также я не замечал, чтобы возникала путаница, если предпочесть один способ другому. Вам придется обучать ваш персонал или совершенствовать стиль проектирования.

Будем считать, что система банкомата начинает работать, только когда пользователь вставляет в банкомат свою карточку. Таким образом, не имеет смысла говорнть, что триггер имеет место, когда кто-то решает использовать банкомат. Триггер Клиент вставляет карточку является также первым шагом варианта использования.

Вариант использования

Работать с банкоматом

Триггер: клиент вставляет карточку.

1. Клиент вставляет карточку с идентификатором банка, номером счета и закодированным PIN-кодом.
2. Система проверяет...

Теперь рассмотрим пример со служащим, сидящим целый день за рабочей станцией, на экране которой имеется набор пиктограмм для запуска различных прикладных программ. Триггер — это то, что клиент вызывает с помощью конкретного запроса, который можно выразить в любой форме. Для иллюстрации я напишу его вторым способом.

Вариант использования

Зарегистрировать жалобу

Триггер: клиент обращается с рекламацией.

1. Служащий вызывает приложение.
2. Система выводит последний вариант списка рекламаций для служащего.

...

6.5. Упражнения

Минимальные гарантии

6.1. Напишите минимальную гарантию для извлечения денег из банкомата.

6.2. Напишите минимальную гарантию для главного варианта использования системы PAF (персональный консультант по финансам описан в упражнении 4.4).

6.3. Напишите минимальную гарантию для варианта использования уровня моря вашей текущей системы. Покажите ее коллегам и предложите им проанализировать ее с точки зрения интересов участников.

Гарантии успеха

6.4. Напишите гарантию успеха для извлечения денег из банкомата.

6.5. Напишите гарантию успеха для главного варианта использования системы PAF.

6.6. Напишите гарантию успеха для варианта использования уровня моря вашей текущей системы. Покажите ее коллегам и предложите им проанализировать ее с точки зрения интересов участников.

Глава 7

Сценарии и шаги

Набор вариантов использования — это непрерывно разворачивающаяся история преследования основными действующими лицами своих целей. Каждый вариант использования имеет ветвящуюся сюжетную линию, которая показывает, как система обеспечивает достижение цели или отвергает ее. Эта сюжетная линия представлена основным сценарием и набором фрагментов сценария в качестве расширений к нему. Каждый сценарий или фрагмент стартует при выполнении условия запуска, которое указывает, когда происходит запуск, и остается истинным, пока сценарий не завершится или цель не будет отвергнута. Все цели разные по сложности, поэтому мы используем одинаковую форму для описания процесса достижения цели любой сложности на любом уровне сценария.

7.1. Основной сценарий

При объяснении какого-либо сложного понятия мы начинаем с простых и доступных вещей, а потом добавляем: "Ну, на самом деле, это немного сложнее. Когда случается то-то и то-то, в действительности происходит следующее...".

Такой стиль объяснения очень хорош, и таким способом мы описываем ветвящуюся сюжетную линию, которая становится вариантом использования. Сначала мы создаем от начала до конца описание одного простого для понимания и довольно типичного сценария, в котором достигается цель основного действующего лица и удовлетворяются интересы всех участников. Это *основной сценарий*. Все другие способы преуспеть в достижении цели и обработка всех неудач описаны в расширениях к этому варианту использования.

Общая структура

Основной сценарий и все его расширения находятся внутри структуры, в которую входят следующие части:

- *Условие, при котором работает сценарий.* Для основного сценария это предусловие вместе с триггером. Для сценария расширения это условие расширения (возможно, с номером шага или местом в сценарии, к которому относится условие).
- *Цель.* Для основного сценария это название варианта использования. Цель обязательно соответствует интересам участников. Для сценария расширения целью является либо достижение цели варианта использования, либо воссоединение с основным сценарием после обработки условия.
- *Набор шагов действия.* Формирует тело сценария и следует одинаковым правилам в каждом сценарии или фрагменте сценария.
- *Условие окончания.* Цель достигается в конце основного сценария. Фрагмент сценария может заканчиваться достижением цели либо отказом от нее.
- *Возможный набор расширений, написанный в виде фрагментов сценария.* Расширения к основному сценарию помещаются в раздел расширений шаблона варианта использования. Расширения к расширениям располагают в той же строке, внутри тела расширения или сразу за ним.

Здесь представлены две выдержки из варианта использования 1, который я раскрыл, чтобы показать сходство общих структур.

Основной сценарий:

Вариант использования

Купить ценные бумаги через Интернет

Предусловие: программа PAF у пользователя уже открыта.

Триггер: пользователь выбрал "покупку ценных бумаг".

1. Пользователь решил купить ценные бумаги через Сеть.
2. PAF получает от пользователя название нужного сайта (E*Trade, Schwab и т.д.).
3. PAF подключается к сайту, сохраняя управление процессом.
4. Покупатель выбирает и покупает ценные бумаги на этом сайте.
5. PAF перехватывает ответы web-сайта и обновляет портфель покупателя.
6. PAF показывает покупателю новое состояние портфеля.

Расширение к данному варианту использования:

3а. Отказ Сети любого рода во время установки:

3а1. Система сообщает пользователю о неудаче, дает совет и возвращается на предыдущий шаг.

3а2. Пользователь либо отказывается продолжать этот вариант использования, либо пробует снова.

В этой главе мы подробно рассмотрим тело сценария, состоящее из шагов действия.

Тело сценария

Каждый сценарий или его фрагмент записывается в виде последовательности действий разных действующих лиц, направленных на достижение цели. Я употребляю термин *"последовательность"* для удобства, но мы можем вносить пометки, показывающие, что шаги можно выполнять параллельно, выбирать в различном порядке, повторять, а некоторые даже могут быть необязательными.

Как отмечалось в разделе 2.2, любой отдельный шаг будет описывать:

- Взаимодействие двух действующих лиц ("Клиент вводит адрес")
- Шаг подтверждения для защиты интереса участника ("Система подтверждает PIN-код")
- Внутреннее изменение для удовлетворения интереса участника ("Система выводит сумму из баланса")

Ниже приведен пример типичного основного сценария, позаимствованный из варианта использования 22. Обратите внимание, что шаги 1, 3, 5-8 — это взаимодействия, шаг 4 —проверка, а шаги 2 и 9 — внутренние изменения.

1. **Служащий вводит номер полиса застрахованного лица и, возможно, его имя и дату страхового события.**
2. **Система анализирует доступную информацию о полисе и указывает, что заявление соответствует полису.**
3. **Служащий вводит основные данные об ущербе.**
4. **Система подтверждает, что не существует конкурирующих заявлений, и присваивает заявлению номер.**
5. **Служащий продолжает вводить информацию об ущербе, соответствующую заявлению данного типа.**
6. **Служащий использует систему для получения дополнительной информации по делу из других компьютерных систем.**
7. **Служащий выбирает и назначает оценщика.**
8. **Служащий подтверждает, что он закончил.**
9. **Система сохраняет данные и инициирует отправку уведомления агенту.**

Любой шаг действия пишется для того, чтобы показать простое действие. Я уподобляю это описанию футбольного матча: игрок *1 пасует игроку 2; игрок 2 ведет мяч; игрок 2 пасует игроку 3.*

Приобретение навыков написания трех видов шагов действия благотворно скажется на вашем стиле описания вариантов использования. Тот же стиль употребляется для шагов действия в каждой части любого варианта использования высокого

либо низкого уровня, будь то основной сценарий, расширение, бизнес-процесс или система.

7.2. Шаги действия

Шаги действия, которые составляют хорошо написанный вариант использования, представлены в одной грамматической форме, простом действии, когда одно действующее лицо выполняет задачу или передает информацию другому действующему лицу.

Пользователь вводит имя и адрес.
В любое время пользователь может потребовать деньги назад.
Система подтверждает подлинность имени и счета.
Система обновляет остаток на счету клиента, чтоб отразить оплату.

Обычно временные соотношения можно опустить, так как шаги, как правило, следуют один за другим.

В способах изложения шагов действия существует масса менее важных вариаций, как видно из примеров вариантов использования. Тем не менее старайтесь писать, придерживаясь следующих правил.

Правила

Правило 1. Используйте простые предложения

Структура предложения должна быть предельно простой:

Подлежащее...сказуемое...прямое дополнение... предложный оборот.

Например:

Система... удерживает ...сумму... из остатка на счете.

И это все. Я говорю об этом, поскольку многие непредумышленно опускают подлежащее. В этом случае неясно, кто же производит действие (у кого, так сказать, мяч). Если предложение плохо сформулировано, трудно следить за развитием сюжета.

Правило 2. Ясно укажите, "кто владеет мячом"

Весьма наглядный пример — приятели, перебрасывающиеся футбольным мячом. Игрок 1 передает мяч игроку 2, игрок 2 немножко ведет мяч, а затем перебрасывает игроку 3. Мяч может загрязниться, и один из игроков его вытрет.

У сценария та же структура. На каждом шаге одно из действующих лиц "владеет мячом". Это действующее лицо будет подлежащим (первое названное действующее лицо), вероятно, первым или вторым словом в предложении. "Мяч" — это сообщение и данные, которые одно действующее лицо передает другому.

Действующее лицо с мячом будет либо владеть им само, либо передавать кому-то еще, либо очищать от грязи. В половине случаев шаг заканчивается тем, что

мяч получает другое действующее лицо. Спросите себя, кто владеет мячом в конце предложения. Ответ в варианте использования всегда должен быть ясным.

Правило 3. Пишите, глядя на вариант использования с высоты птичьего полета

Начинающие авторы вариантов использования, особенно программисты, создающие его описание, часто пишут сценарии с точки зрения системы, которая смотрит на мир и беседует сама с собой. У них получаются примерно такие предложения: "Получить карточку банкомата и PIN-код. Снять сумму с остатка на счете".

Вместо этого пишите вариант использования следующим образом:

Клиент вводит в банкомат карточку и PIN-код.

Система снимает сумму с остатка на счете.

Некоторым писателям нравится стиль драматического произведения, когда поведение действующих лиц описывается, как в пьесе.

Клиент: Вводит в банкомат карточку и PIN-код.

Система: Снимает сумму с остатка на счете.

Заметьте, что смысл в обоих случаях остается неизменным.

Правило 4. Покажите продвижение процесса

Прогресс отдельного шага связан с тем, насколько приблизилась цель. В обобщенных, или белых, вариантах использования шаг, вероятно, приближает цель в полном объеме. В варианте использования уровня подфункции приближение цели менее различимо. Если мы видим шаг Пользователь нажимает на клавишу tab, это значит, что мы заглянули в вариант использования цвета темного индиго (или черного), либо автор просто решил описывать слишком незначительные действия.

Ошибочный выбор очень мелких шагов сказывается на длине варианта использования. Если вариант использования состоит из 13-17 шагов, почти наверняка его предложения не слишком продвигают дело к цели. Вариант использования будет более читабельным и понятным, если объединить эти мелкие шажки. При этом объем существенной информации не уменьшится. Я редко встречал хорошо написанный вариант использования, основной сценарий которого включал бы более девяти шагов.

Чтобы найти для шага цель чуть более высокого уровня, спросите себя, почему действующее лицо делает это (как описано в главе 5 в подразделе Повышение и понижение уровней целей). Ответ на этот вопрос, скорее всего, окажется нужной для вашего шага целью, хотя, возможно, вам придется задавать его несколько раз, прежде чем вы получите желаемый ответ (см. далее пример повышения уровня цели с помощью вопроса "почему").

Пользователь нажимает на клавишу tab

Почему пользователь нажимает на клавишу tab? Чтобы попасть в поле адреса.

Почему он пытается попасть в поле адреса? Потому что он должен ввести свои имя и адрес, прежде чем система начнет что-то делать.

Он хочет, чтобы система что-то сделала (скорее всего, выполнила этот самый вариант использования), а для этого ему нужно ввести свое имя и адрес.

Таким образом, предложение, описывающее действие, заметно продвигающее процесс, выглядит так:

Пользователь вводит имя и адрес.

Правило 5. Показывайте намерение, а не движения действующего лица

Описание детальных действий пользователя в процессе работы с пользовательским интерфейсом системы — одна из распространенных и серьезных ошибок при создании варианта использования. Я называю это *описанием деталей интерфейса*. Такое описание ухудшает качество документа, излагающего требования к системе, делая его более длинным, неустойчивым и перегруженным связями.

- Длинные документы труднее читать и сложнее поддерживать.
- Описываемый диалог, возможно, не является требованием, но показывает, как автор представляет себе в данный момент интерфейс пользователя.
- Диалог неустойчив в том смысле, что небольшие изменения в проекте системы делают его описание недействительным.

Работа проектировщика интерфейса пользователя и заключается в создании эффективного интерфейса, который позволит пользователю достигнуть цели варианта использования. Описание детальных действий соответствует этой задаче проектирования, но не документу, излагающему функциональные требования.

В документе, излагающем требования, нас интересует описание концепции интерфейса, т.е. то, что объявляет намерение пользователя и суммирует информацию, передаваемую от одного действующего лица другому. Авторы книги *Software for use* посвятили ее часть именно этой теме, пользуясь термином *важнейшие варианты использования* (essential use cases) для обозначения системных вариантов использования уровня моря, описывающих концепцию интерфейса.

Обычно все данные, передаваемые в одном направлении, собираются в одном шаге действия. Ниже приведены распространенный фрагмент неправильного описания и способ его исправления.

Первоначальный вариант:

1. **Система запрашивает имя.**
2. **Пользователь вводит имя.**
3. **Система запрашивает адрес.**
4. **Пользователь вводит адрес.**
5. **Пользователь щелкает по кнопке "OK".**
6. **Система представляет параметры пользователя.**

Исправленный вариант:

1. **Пользователь вводит имя и адрес.**
2. **Система представляет параметры пользователя.**

Если передается более трех элементов данных, вы можете поместить каждый элемент в отдельной строке в табличном списке. Оба способа хороши, но первый работает лучше, если вы впервые набрасываете варианты использования, поскольку его можно быстрее написать и прочитать. Второй предпочтительней, если вы хотите обеспечить высокую точность для трассировки или тестирования.

Допустимый вариант 1:

1. **Пользователь вводит имя, адрес, номер телефона, конфиденциальную информацию, номер контактного телефона для чрезвычайных ситуаций.**

Допустимый вариант 2:

1. **Пользователь вводит:**
   - **имя**
   - **адрес**
   - **номер телефона**
   - **конфиденциальную информацию**
   - **номер контактного телефона для чрезвычайных ситуаций**

Правило 6. Включайте "рациональный" набор действий

Ивар Якобсон описал шаг варианта использования как представление транзакции. Для этой формулировки он зафиксировал четыре части сложного взаимодействия (рис. 7.1):

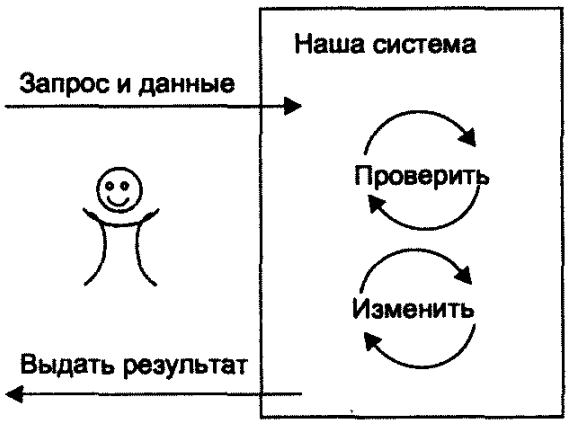


Рис. 7.1. *Четыре части транзакции*

1. Основное действующее лицо посылает системе запрос и данные.
2. Система подтверждает запрос и данные.
3. Система изменяет внутреннее состояние.
4. Система выдает действующему лицу результат.

Можно записать четыре части как отдельный шаг действия или скомбинировать их различными способами, в том числе включив в один шаг действия все четыре части. Как лучше — зависит от сложности каждой части и от того, где происходят естественные перерывы в процессе обработки.

Вашему вниманию предлагается пять версий. Все они правильные, хотя версию 1 я считаю слишком сложной для восприятия. Мне нравится версия 2 с понятными частями, но я нахожу их слишком длинными для работы на этом этапе. Для данного примера я предпочитаю версии 3, и 4. Полагаю, что шаги действия в версии 5 мелковаты, сценарий становится слишком длинным и тяжеловесным, но достоинство этой версии в том, что шаги являются отдельно тестируемыми единицами, а это может пригодиться при более формальном подходе к разработке.

Версия 1:

1. Клиент вводит номер заказа. Система обнаруживает, что он совпадает с выигравшим номером месяца, регистрирует пользователя и номер заказа как победителя этого месяца, посылает электронной почтой сообщение менеджеру по продажам, поздравляет клиента и выдает инструкцию, как получить приз.

Версия 2:

1. Клиент вводит номер заказа.
2. Система обнаруживает, что он совпадает с выигравшим номером месяца, регистрирует пользователя и номер заказа как победителя этого месяца, посылает электронной почтой сообщение менеджеру по продажам, поздравляет клиента и выдает инструкцию, как получить приз.

Версия 3:

1. Клиент вводит номер заказа.
2. Система обнаруживает, что он совпадает с выигравшим номером месяца.
3. Система регистрирует пользователя и номер заказа как победителя этого месяца, посылает электронной почтой сообщение менеджеру по продажам, поздравляет клиента и выдает инструкцию, как получить приз.

Версия 4:

1. Клиент вводит номер заказа.
2. Система обнаруживает, что он совпадает с выигравшим номером месяца.
3. Система регистрирует пользователя и номер заказа как победителя этого месяца, посылает электронной почтой сообщение менеджеру по продажам.
4. Система поздравляет клиента и выдает инструкцию, как получить приз.

Версия 5:

1. Клиент вводит номер заказа.
2. Система обнаруживает, что он совпадает с выигравшим номером месяца.
3. Система регистрирует пользователя и номер заказа как победителя этого месяца.

4. Система посылает электронной почтой сообщение менеджеру по продажам.
5. Система поздравляет клиента и выдает инструкцию, как получить приз.

Правило 7. "Подтвердить", а не "Проверить"

Одним из трех видов шагов действия является подтверждение системой, что соблюдено некоторое бизнес-правило. Часто пишут, что система проверяет условие. Это неудачный глагол для действия. Он не отражает продвижение процесса вперед, не является на самом деле целью и не указывает явно, каков результат проверки. Вам тотчас же придется написать: "Если поверка прошла удачно..." и "Если проверка закончилась неудачей...".

Применим метод "Спросить почему", чтобы найти более подходящее предложение. Почему система проверяет это условие? Ответ: чтобы *установить*, или *подтвердить*, или *обеспечить* что-то. Эти глаголы лучше выражают действие для достижения цели. Замените предложение "Система проверяет, правильный ли пароль" предложением:

Система подтверждает, что пароль правильный.

Увидев слово если, вспомните это правило. Всякий раз, когда вы видите **"Если (условие)... тогда..."**, взгляните на предыдущее предложение. Вероятно, его сказуемое — проверяет. Замените в первом предложении проверяет на подтверждает и сделайте второе предложение простым описанием действия без если. Ниже представлен пример до и после преобразования.

Первоначальный вариант:

2. Система проверяет, правилен ли пароль.
3. Если да, система представляет пользователю допустимые действия.

Исправленный вариант:

2. Система подтверждает, что пароль правилен.
3. Система представляет пользователю допустимые действия.

Обратите внимание, что во втором случае сценарий описан как успешный. Он также побуждает читателя спросить на шаге 2, что произойдет, если пароль неверный. Читатель обратится к соответствующему разделу расширения и найдет расширение, начинающееся со слов Пароль неверен. Это придает варианту использования последовательный ритм, что облегчает чтение и просмотр.

Правило 8. В некоторых случаях вводите временные ограничения

Большинство шагов следуют непосредственно из предыдущего. Иногда приходится писать так:

В любое время между 3 и 5 шагами пользователь...

или

Как только пользователь..., система...

Вставлять временные ограничения можно, но только когда это необходимо. Обычно временные параметры очевидны, и нет необходимости их упоминать.

Правило 9. Идиома: "Пользователь использует систему А для получения данных от системы В"

Это ситуация, с которой вы можете столкнуться. Вы хотите, чтобы разрабатываемая система А получила информацию от системы В или как-то иначе осуществила с ней взаимодействие. Ей следует это делать, только когда основное действующее лицо определяет, что настал нужный момент. Нельзя написать: **"Пользователь нажимает на кнопку "Получить", и в этот момент система получает данные от системы В"**. Это потребует описания деталей интерфейса.

Можно использовать два шага:

4. **Пользователь дает системе сигнал получить данные от системы В.**
5. **Система получает исходные данные от системы В.**

Это приемлемо, однако громоздко и избыточно. Лучше написать так:

4. **Пользователь применяет систему для получения исходных данных от системы В.**

С помощью этого небольшого изменения мы показываем, что пользователь сам выбирает время, и мяч переходит от пользователя к системе А, затем к системе В. Кроме того, мы определяем обязанности всех трех систем. Подробности инициирования пользователем действия не определены, как и должно быть.

Правило 10. Идиома: "Делать шаги х-у, пока не возникнет условие"

Иногда мы хотим отметить, что некоторые шаги могут повторяться. И опять нам повезло, что мы пишем обычной прозой, а не пользуемся формализмами программирования. Просто напишите, что шаг или шаги будут повторяться.

Если повторяется только один шаг, можно записать указание о повторении прямо в шаге:

Пользователь выбирает один или более продуктов.

Пользователь просматривает различные каталоги товаров, пока не найдет тот, которым захочет воспользоваться.

Если повторяется несколько шагов, можно указать на повторение до или после повторяющихся шагов. Я пишу о повторениях после шагов, чтобы чуть облегчить чтение сценария, но подойдет любой способ.

1. **Клиент предоставляет учетный идентификатор либо имя и адрес.**
2. **Система спрашивает о предпочтениях клиента.**
3. **Пользователь выбирает товар для покупки, отмечает его для покупки.**
4. **Система добавляет товар к тележке для покупок клиента. Клиент повторяет шаги 3-4, пока не укажет, что выбор закончен.**
5. **Клиент покупает товары, находящиеся в тележке для покупок.**

Заметьте, что нам не нужно нумеровать предложение о повторении, и нет необходимости в предложении, открывающем повторение. И то и другое загромождает описание.

В варианте Делать шаги x-y, пока не возникнет условие, шаги x-y могут быть в любом порядке. Думаю, имеет смысл помещать условие перед соответствующими шагами.

1. **Клиент входит в систему.**
2. **Система представляет доступные продукты и услуги. Шаги 3-5 могут происходить в любом порядке.**
3. **Пользователь выбирает продукты для покупки.**
4. **Пользователь указывает предпочтительную форму оплаты.**
5. **Пользователь дает адрес для доставки.**
6. **Пользователь указывает, что сделал все покупки.**
7. **Система инициирует заказ, причем выбранные продукты будут оплачены по выбранной форме оплаты и посланы по адресу назначения.**

Нумеровать или нет

Номера выделяют шаги и позволяют ссылаться на них в разделе расширений. Однако их труднее поддерживать. Без хороших инструментов автоматической перенумерации при вставке или добавлении шагов утомительно постоянно перенумеровывать шаги одного и того же варианта использования. Хорошие варианты использования можно написать с нумерацией шагов и без нее, поэтому этот вопрос надо решить для проекта в целом.

Группы, в которых я работал, испытывали оба метода и последовательно выбирали нумерацию предложений по всем абзацам, поэтому нумерованные предложения стали для меня стандартом. Другие привыкли писать в форме простых абзацев и не думают переключаться.

Я включаю шаблоны, как для бессистемных, так и для написанных по всей форме вариантов использования, чтобы подчеркнуть правомерность выбора.

Абзацы являются выбором по умолчанию и в обычных шаблонах, и в шаблоне технологии Rational Unified Process (см. вариант использования 32). Если хотите, можете нумеровать шаги, как в этом варианте использования. Я почти всегда применяю нумерацию, даже если все остальное в варианте использования написано не по форме, поскольку считаю, что нумерация помогает исследовать поведение.

Полная форма требует нумерации.

7.3. Упражнения

Шаги действия

7.1. Напишите сценарий одного из ваших вариантов использования двумя способами: с подробным описанием интерфейса и с помощью описания намерений. Обсудите различия между сценариями.

7.2. Напишите основной сценарий для варианта использования уровня задачи Извлечение денег с помощью режима Быстрое получение наличных.

7.3. Напишите основной сценарий для одного стратегического варианта использования и одного варианта использования уровня задачи для системы персонального консультанта по финансам (PAF).

7.4. Молодой коллега прислал вам нижеследующий вариант использования. Учитывая изученный к настоящему моменту материал, пошлите ему в ответ критические замечания и исправления.

Вариант использования

Войти в систему

Этот вариант использования описывает процесс, с помощью которого пользователи входят в систему обработки заказов. В нем также устанавливаются права доступа для различных категорий пользователей.

Поток событий:

Основной путь:

1. **Вариант использования запускается, когда пользователь запускает приложение.**
2. **Система покажет экран для входа.**
3. **Пользователь вводит имя и пароль.**
4. **Система проверит информацию.**
5. **Система установит права доступа.**
6. **Система покажет основной экран.**
7. **Пользователь выберет функцию.**
8. **Пока пользователь не выберет Выйти из цикла**
9. **Если пользователь выберет Разместить заказ, использовать Разместить заказ.**
10. **Если пользователь выберет Вернуть товар, использовать Вернуть товар.**
11. **Если пользователь выберет Снять заказ, использовать Снять заказ.**
12. **Если пользователь выберет Получить статус заказа, использовать Получить статус.**
13. **Если пользователь выберет Выслать каталог, использовать Выслать каталог.**
14. **Если пользователь выберет Зарегистрировать жалобу, использовать Зарегистрировать жалобу.**
15. **Если пользователь выберет Выдать отчет по продажам, использовать Выдать отчет по продажам. Конец проверки если**
16. **Пользователь выберет функцию. Конец цикла**
17. **Вариант использования завершается.**

Глава 8

Расширения

В вариант использования должны входить все сценарии, как успешные, так и приводящие к неудаче. Мы также обсудили, как писать основной сценарий. Теперь надо узнать, как добавить к нему все остальные.

Можно написать каждый сценарий отдельно. Это соответствовало бы "полосатым брюкам" (см. рис. 2.2). Этот подход время от времени пропагандируют, и некоторые инструменты подталкивают писателя идти этим путем. Однако сопровождение "полосатых брюк" — трудная задача (см. главу 2). Каждое изменение в сценарии должно быть скопировано во все остальные сценарии, содержащие тот же текст.

Второй способ — вставлять предложения с если на протяжении всего текста: **"Если пароль верный, система делает так..., в противном случае этак..."**. Это абсолютно законно, некоторые так и пишут. Но читателям приходится нелегко с этими если, особенно когда одно предложение с если находится внутри другого. Читатели теряют нить уже после двух веток с если, а большинство вариантов использования имеют много точек ветвления.

Третий способ создать этот текст — написать основной сценарий как простую последовательность действий от запуска до завершения, а затем написать *расширение* сценария для каждой точки ветвления. Это, пожалуй, лучший из трех способов.

8.1. Расширение. Основные положения

Расширения работают так. Вслед за основным сценарием для каждой точки, в которой поведение может идти по различным сценариям вследствие определенного условия, запишите это условие, а потом управляющие им шаги. Часто расширение заканчивается воссоединением с основным сценарием.

Расширение на самом деле — это спуск вниз по варианту использования. Оно запускается при возникновении условия, которое делает его релевантным. Оно содержит последовательность шагов действия, описывающих, что происходит при этом условии, и заканчивается достижением цели или отказом от нее. По пути могут встретиться и расширения к расширениям, обрабатывающие предложения с если и но.

Эти расширения возникают там, где находятся наиболее интересные требования к системе. Основной сценарий разработчикам часто известен. При обработке отказов нередко используются бизнес-правила, которые разработчики не знают. Авторам описания требований приходится выяснять правильные реакции системы, и довольно часто при таком исследовании вводится новое действующее лицо, новый вариант использования или новое условие расширения.

Ниже приведена типичная дискуссия, иллюстрирующая это утверждение.

"Предположим, что сеть неожиданно отказала. Что мы делаем?"

"Система регистрирует это".

"Принято. Но если сеть прекратила работать, что будет, когда она заработает снова?"

"Мы должны добавить новый вариант использования для повторного запуска системы после отказа сети. Система восстановится, откроет журнал и либо завершит транзакцию, либо ее отменит".

"Но что если журнал будет поврежден? Что тогда будет делать система?"

"Не знаю. Давайте запишем это как открытый вопрос и продолжим".

В этом диалоге обнаруживается новый вариант использования, а также необходимость исследовать стратегию бизнеса. Не обманывайте себя, полагая, что не нужно обсуждать эти расширения. Какой-нибудь программист в проекте, разрабатывая программу, натолкнется на те же условия, и очень дорогое время будет потрачено на открытие и изучение новых бизнес-правил. Лучше всего этим заниматься на этапе сбора требований к системе.

Рекомендую организовать работу в три этапа:

1. Хорошо поразмыслите и включите каждую возможность, которую вы только можете вообразить.
2. Оцените, исключите и объедините идеи в соответствии с правилами, изложенными в разделе 8.2.
3. Продумайте, как пробраться через все эти системные обработки условий.

8.2. Условия расширения

Условия расширения — это условия, при которых изменяется поведение системы. Мы говорим расширение в противоположность неудаче или исключительной ситуации, чтобы наряду с условиями неудачи включить альтернативный сценарий успеха.

Здесь представлены два фрагмента, иллюстрирующие путь успеха и путь неудачи в расширениях.

Пример 1

...

4. Пользователь дает системе команду сохранить наработанное на текущий момент.

...

Расширения

4a. Система автоматически обнаруживает необходимость немедленного сохранения:

4a1. ...

4b. Сохранение завершается неудачей:

4b1. ...

Вышеприведенный пример можно истолковать так: вместо того чтобы сохранить текущее состояние данных по требованию пользователя, система может автоматически обнаружить необходимость немедленного сохранения. В этом случае ... (что-то делается). Сохранение (по требованию пользователя или автоматическое) может закончиться неудачно. В этом случае ... (делается что-то еще).

Пример 2

...

3. Система сканирует документ, сверяя правописание каждого слова со словарем.

4. Система обнаруживает орфографическую ошибку, выделяет слово и предоставляет на выбор пользователю альтернативы.

5. Пользователь выбирает одну из замен. Система замещает выделенное слово другим, выбранным пользователем для замены.

...

Расширения

4a. Система не обнаруживает больше орфографических ошибок до конца документа:

4a1. Система уведомляет пользователя и завершает выполнение варианта использования.

5a. Пользователь решает сохранить первоначальное написание:

5a1. Система оставляет слово и продолжает работу.

5b. Пользователь набирает новое написание, не входящее в список:

5b1. Система проверяет новое написание и возвращается на шаг 3.

...

Выявляйте все возможные ошибки и альтернативные пути

Очень важно максимально охватить условия расширения, прежде чем писать, как их обрабатывать. Это утомительная работа, как и документирование обработки расширений. На то, чтобы продумать только одно расширение и сделать правильные выводы о работе системы, уходит много сил, не говоря уже о 3-4 расширениях. Это значит, что вы не будете обдумывать следующую порцию условий расширения. А это следует сделать.

Если вы выявите все альтернативные ситуации успехов и неудач, у вас будет список, который послужит основой для вашей дальнейшей работы в течение неско-

льких часов или дней. Вы можете сделать перерыв и вернуться к тому месту, где закончили.

Выявите все возможные пути, которые приводят к неудачному завершению сценария, и альтернативные пути, ведущие к успеху. Убедитесь, что не пропустили ни один из них:

- Альтернативный путь к успеху **(служащий использует клавишу быстрого вызова).**
- Основное действующее лицо действует некорректно **(неверный пароль).**
- Бездействие основного действующего лица **(истечение времени ожидания пароля).**
- Каждое появление предложения "система подтверждает" означает, что последует расширение, обрабатывающее неподтверждение **(неверный учетный номер).**
- Несоответствующая реакция второстепенного действующего лица или ее отсутствие **(истечение времени ожидания ответа).**
- Внутренняя ошибка в разрабатываемой системе, которая должна быть обнаружена и обработана в обычном порядке **(заблокирован автомат для выдачи наличных).**
- Неожиданная и необычная ошибка, которую необходимо обработать, и которая будет иметь видимый результат **(обнаружено повреждение журнала транзакций).**
- Критически важные недостатки в производительности системы, которые вы должны обнаружить **(время реакции не укладывается в 5 с).**

Часто люди анализируют шаги от начала сценария до его конца, чтобы гарантировать максимальный охват всех условий. Если вы поступите так же, количество вещей, которые могут выйти из строя, удивит вас. Упражнение 8.1 даст вам возможность попытаться выявить ряд ошибок. Ответ вы найдете в приложении В. Двое учащихся моего курса назвали почти в три раза больше ошибок, чем дано в ответе. Насколько преуспеете вы?

Правило 11. Пусть условие само указывает, что было обнаружено

Запишите, что обнаружила система, а не просто, что произошло. Не пишите: **"Клиент забывает PIN-код"**. Система не может обнаружить это. Возможно, клиент ушел, с ним случился сердечный приступ или он успокаивает плачущего младенца. Система в этом случае обнаруживает бездействие, что означает превышение временного предела. Напишите: **"Предел ожидания ввода PIN-кода превышен"** или **"Истекло время ожидания PIN-кода"**.

Условием часто служит выражение, описывающее то, что было обнаружено. Мне нравится ставить двоеточие (:) в конце условия, чтобы читатель не принимал

условие за шаг действия. Это соглашение предотвращает много ошибок. Вот несколько примеров:

Неверный PIN-код:
Сеть не работает:
Клиент не отвечает (время ожидания ответа истекло):
Деньги не выдаются надлежащим образом:

Если вы используете нумерованные шаги, напишите номер шага, где может быть обнаружено условие, и сопроводите номер буквой (скажем, **4a**). Буквы не связаны какой-либо последовательностью, поэтому условие **4b** не обязательно должно следовать за **4a**. Это позволяет присоединить сколько угодно условий расширения к любому шагу действия.

2a. Недостаточно финансовых средств:
2b. Сеть не работает:

Если условие может возникнуть на нескольких шагах и вам представляется важным указать на это, перечислите эти шаги:

2-5a. Пользователь неожиданно прекратил работу:

Если условие может возникнуть в любое время, используйте знак звездочки (*) вместо номера шага. Сначала перечисляйте условия со звездочкой, затем нумерованные.

***a. Сеть перестает работать:**
***b. Пользователь ушел без уведомления (время ожидания ответа истекло):**
2a. Недостаточно финансовых средств:
2b. Сеть не работает:

Не беспокойтесь о том, в каком участке происходит ошибка: на шаге ввода пользователем данных или на следующем, где система подтверждает данные. Некоторые утверждают, что ошибка возникает в любом месте, но на этот довод не стоит тратить время. Обычно я связываю ошибки с шагом подтверждения, если таковой присутствует.

Когда вы пишете обычными абзацами без нумерации шагов, вы не можете обращаться к определенному шагу, в котором возникает условие. Поэтому сделайте условие наглядным, чтобы читатель знал, когда оно может возникнуть. Перед каждым условием пропустите строку или сделайте пробел и выделите условие, например *курсивом* (см. вариант использования 32).

Невымышленная история

Когда-то я участвовал в работе над серьезным проектом. К сожалению, мы не слишком тщательно продумали расширения.

Подобно многим разработчикам, мы не захотели рассматривать, что случится, если программа натолкнется в базе данных на неверную информа-

цию. Каждый из нас надеялся, что другая команда позаботится об этом. Вы догадываетесь, что произошло? Через неделю после первой сдачи первой очереди проекта вице-президент решил посмотреть, как один из его заказчиков использует новые способы продвижения товара. Он запустил свою новенькую систему и запросил данные по этому крупному заказчику. Система ответила, что данные не найдены. Его состояние трудно было описать.

Высшее звено персонала в полном составе было собрано на экстренное совещание, чтобы решить, что делать с ошибками в базе данных. Мы обнаружили, что в базе была только одна неверная секция, поэтому сообщение об ошибке должно было быть таким: "Некоторые данные пропущены". Хуже было другое — мы упустили из вида, что реакция системы при обнаружении неверных внутренних данных на самом деле есть часть внешних требований к ней.

Мы переделали проект системы, чтобы она оптимально справлялась с неполными данными и выдавала как наилучшие возможные результаты, так и сообщение о том, что некоторые данные пропущены.

Вывод из этой истории состоит в том, что необходимо рассматривать внутренние ошибки, такие как пропущенные данные.

О списке выявленных условий

Ваш список выявленных условий будет содержать больше идей, чем вы будете в конечном счете использовать, и это нормально. Смысл упражнения в том, чтобы попытаться охватить все ситуации, которые когда-либо встретит система в этом варианте использования. Позже вы сократите список.

В этой точке ваш список, вероятно, еще не полон. Вы, возможно, думаете о новом условии ошибки, когда пишете сценарий расширения или добавляете новый шаг подтверждения где-то внутри варианта использования. Не беспокойтесь об этом. Тщательно выполняйте работу на этой стадии анализа и пополняйте список в процессе работы.

Совершенствуйте список расширений

Цель совершенствования — сделать список расширений максимально коротким. Идеальный список условий расширения показывает все ситуации, которые система должна обрабатывать. Помните, что длинный документ, содержащий требования к системе, всегда тяжело читается, а избыточные описания сложно сопровождать. Объединяя условия расширения, вы сокращаете документ и время его чтения.

Проведя тщательный анализ, вы должны получить короткий и простой основной сценарий и длинный список условий, которые предстоит рассмотреть. Внимательно просмотрите список, удаляя те условия, которыми система не должна управлять, и объединяя те, которые оказывают на систему одинаковое результирующее воздействие. Используйте два критерия:

- Система должна быть способна обнаруживать это условие.
- Система должна обрабатывать обнаружение условия.

Попробуйте сделать необнаруживаемые условия обнаруживаемыми, прежде чем их удалять. Удалите условие **"Клиент забыл карточку для банкомата"**, поскольку не существует эквивалентного условия, которое система способна обнаружить. Преобразуйте необнаруживаемое условие **"Клиент забыл PIN-код"** в условие **"Истекло время ожидания ввода PIN-кода"**, которое система сможет определить.

Затем объедините эквивалентные условия. Вы могли бы написать, что карточка поцарапана, считывающее устройство неисправно, карточка не является карточкой для банкомата. С точки зрения требований поведение банкомата будет одинаковым: **"Возвратить карточку и уведомить клиента"**. Поэтому попробуйте объединить эти условия. Постарайтесь найти одно осмысленное предложение для всех, можно построить распространенное предложение: **"Карточка не читается или предназначена не для банкомата"**.

Свертывайте ошибки

Как часть объединения условий, производящих одинаковое воздействие, объединяют ошибки вариантов использования более низкого уровня, имеющие одинаковый результирующий эффект на вариант использования более высокого уровня. Это свертывание ошибок более низкого уровня дает возможность избежать лавины условий расширения на более высоких уровнях.

Предположим, что вы работаете над пакетом PAF и пишете вариант использования уровня цели *Обновить инвестиции*. Пусть один из последних шагов таков:

Вариант использования

Обновить инвестиции

...

7. Пользователь дает команду PAF сохранить текущее состояние данных.

8. ...

Это обращение вызывает вариант использования Сохранить текущее состояние данных, который содержит условия такого рода:

Вариант использования

Сохранить текущее состояние данных

...

Расширения:

3a. Файл уже существует (пользователь не желает его перезаписывать):

...

3b. Директория не найдена: ...

4a. Не хватает места на диске: ...

4b. Файл защищен от записи: ...

... и т.д. ...

Вариант использования Сохранить текущее состояние данных завершается успехом либо неудачей, возвращающей выполнение в конец шага 7 варианта использования *Обновить инвестиции*. В случае успеха выполнение продолжается с шага 8. Но что если сохранение заканчивается неудачей? Что нужно написать в расширении **7а**? Читателя варианта использования Обновить инвестиции не интересует, почему сохранение не удалось (все неудачи имеют один и тот же результирующий эффект), поэтому в вариант использования Обновить инвестиции включите только одно расширение, описывающее, что произошло.

Вариант использования

Обновить инвестиции

...

7. Пользователь дает команду PAF сохранить текущее состояние данных.

8. ...

Расширения:

7а. Сохранение произвести не удалось:

7а1. ... все, что должно произойти потом...

Лучше всего в свертывании ошибок то, что даже на самом высоком уровне варианта использования терминология сообщений об ошибках соответствует этому уровню. Даже занятые руководители найдут время прочитать их, потому что сообщения в варианте использования очень высокого уровня тоже вполне высокого уровня.

8.3. Обработка расширений

В простейшем случае условие обрабатывает базовая последовательность шагов. Однако обычно расширение представляет собой миниатюрный вариант использования. Триггером служит условие расширения; цель —достичь цели варианта использования либо возобновить выполнение после произошедшей ошибки. Тело является набором шагов действия и, может быть, их расширениями. Расширение может завершаться достижением цели или отказом от нее, совсем как в варианте использования. Сходство не случайно, и это очень удобно с точки зрения рациональной формы описания сложных расширений.

Начинайте описывать обработку условия с шага действия, который следует за обнаруживаемым условием. Нет необходимости повторять, что было обнаружено условие. Продолжайте повествование точно так же, как и при создании основного сценария, соблюдая все правила для уровня цели, стиля и предложений, которые мы рассматривали ранее. Пишите, пока не достигнете места, где можно воссоединиться с основным сценарием, или пока вариант использования не закончится неудачей.

Обычно фрагмент сценария завершается одним из следующих способов:

Шаг, порождающий расширение, зафиксирован и заменен. В конце обработки расширения положение таково, как если бы шаг был успешным.

3. Пользователь активизирует URL web-сайта.
4. ...

Расширения:

3a.Нет доступного URL:

3a1. Пользователь ищет web-сайт.

3b. ...

Система предоставляет действующему лицу еще один шанс. В конце обработки расширения действие возвращается в начало того же шага. В следующем примере обратите внимание, что система повторно подтверждает пароль:

3. Пользователь вводит пароль.
4. Система подтверждает пароль.
5. ...

Расширения:

4a. Неверный пароль:

4a1. Система уведомляет пользователя, снова запрашивает пароль.

4a2. Пользователь повторно вводит пароль.

4b. ...

Вариант использования завершается ввиду полного отказа.

3. Пользователь вводит пароль.
4. Система подтверждает пароль.
5. ...

Расширения: ...

4c. Превышен предел попыток ввода пароля:

4c1. Система уведомляет пользователя, прекращает сеанс.

5a. ...

Чтобы достигнуть цели, действие развивается по совершенно другому пути.

3. Пользователь делает...
4. Пользователь делает...
5. ...

Расширения:

3a. Пользователь запускает персональную макрокоманду, чтобы завершить процесс:

3a1. Вариант использования завершается.

В двух первых вариантах использования не нужно указывать, что происходит потом в расширении, так как для читателя очевидно, что этот шаг будет запущен снова или продолжен. В последних двух случаях достаточно сказать о неудаче или о завершении варианта использования, поскольку шаги показывают, что интересы участников соблюдены.

В большинстве расширений не говорится о том, по какому пути пойдет действие по окончании расширения. Обычно это очевидно, а если писать после каждого рас-

ширения "Перейти к шагу N", текст будет перегружен, что плохо для читателя. В редких случаях, когда не ясно, что действие перепрыгивает на другой участок основного сценария, в конечном шаге можно записать: "Перейти к шагу N".

Примеры каждой из таких ситуаций можно найти в этой книге в различных образцах вариантов использования.

Правило 12. Делайте отступы в тексте обработки условия

Используя стиль с нумерацией, демонстрируемый в этой книге, делайте отступы для шагов действия, которые указывают, как обрабатывается условие, и начинайте повторную нумерацию с 1 после буквы. Шаги действия следуют всем ранее описанным правилам стиля.

Расширения:

2а. Недостаточно финансовых средств:

2а1. Система уведомляет клиента, предлагает ввести другую сумму.

2а2. Клиент вводит новую сумму.

Перед сдачей этой книги в печать Волкер Хеймер из компании Software Futures ССН описал соглашение, упрощающее нумерацию. Разработчики его фирмы опускают лидирующую комбинацию номер-буква, начиная непосредственно с точки, за которой следует номер. Приведенный выше пример будет выглядеть так:

2а. Недостаточно финансовых средств:

.1. Система уведомляет клиента, предлагает ввести другую сумму.

.2. Клиент вводит новую сумму.

Преимущество этого способа в том, что при изменении нумерации, Например, когда шаг 2 станет шагом 3, перенумеровать шаги расширения значительно проще. Если вы пишете без нумерации, делайте отступы либо начинайте каждый шаг действия с абзаца. Шаблон Rational Unified Process имеет специальный уровень заголовка для условия расширения, причем шаги действия записываются как текст под этим заголовком.

Ошибки внутри ошибок

Внутри фрагмента сценария обработки расширения вы можете встретить новое условие ветвления, возможно, ошибку. Если вы придерживаетесь стиля форматирования с отступами, как в данной книге, сделайте еще один отступ и продолжайте называть условия и писать сценарий, как прежде. В какой-то точке отступы и нумерация станут такими сложными, что вы решите превратить все расширение в еще один вариант использования. Большинство моих корреспондентов говорят, что это случается примерно на третьем уровне отступов. Ниже приведен пример, взятый из варианта использования 22.

6а. Служащий решает выйти, не получив минимума информации:

6а1. Система предупреждает служащего, что он не может выйти и закрыть форму без даты, имени или номера полиса или записи имени оценщика.

6a1a. Служащий решает продолжить ввод данных по ущербу.

6a1b. Служащий сохраняет "промежуточный" отчет и выходит из системы.

6a1c. Служащий настаивает на выходе без записи минимума информации:

Система отвергает сохранение любых промежуточных версий и завершает выполнение.

Обратите внимание, что в этом примере разработчик не присвоил номер последней строке. На номере 6a1c1 он решил, что расширение уже порядком загромождено и короткий фрагмент простого текста сделает чтение более удобным.

Вообще стоимость создания нового варианта использования достаточно высока, чтобы слишком долго не выделять расширение в самостоятельный вариант использования. Компромиссное решение состоит в том, что вышеприведенный пример существует, пока имеет смысл делать отступ, не выделяя его в самостоятельный вариант использования.

Создание из расширения нового варианта использования

Чтобы выделить расширение в новый подчиненный вариант использования, определите цель основного действующего лица и соответственно назовите вариант использования, обозначьте его уровень (возможно, *подфункция*), откройте шаблон для нового варианта использования и запишите детали, которые вы перетащили из вызывающего варианта использования.

Возьмем для примера вариант использования 32. Он когда-то содержал шаг:

Пользователь может сохранить или распечатать отчет в любой момент.

В нем есть ряд расширений, описывающих альтернативы и ситуации отказов, но их число постоянно росло: **непоименованный отчет, уже существующее имя (заменять или не заменять), пользователь прекращает сохранение на середине**, и т.д. Наконец, разработчики решили выделить *Сохранить отчет* в отдельный вариант использования.

В первоначальном варианте использования вы тем не менее должны учитывать, что новый подчиненный вариант использования может закончиться неудачей, поэтому ваш вариант, вероятно, будет включать условия как успеха, так и неудачи.

С точки зрения теории и затраченных усилий перемещать расширение в самостоятельный вариант использования и обратно достаточно просто. Модель варианта использования обеспечивает легкое решение. Переместить расширения *Сохранить отчет* из *Управления отчетами* не составляет труда, а чтобы вернуть их назад, нужно несколько минут работы в текстовом редакторе.

Однако создание варианта использования не ограничивается механическими усилиями. Новый вариант использования требуется помечать, отслеживать, планировать, тестировать и сопровождать. Для проектной группы это недешевые операции.

Вообще сохранение расширения внутри варианта использования определяется в основном экономическими соображениями. Две ситуации могут побудить вас создать новый вариант использования для расширения:

- *Расширение используется в нескольких местах.* Выделение его в собственный вариант использования означает, что отслеживать и сопровождать его можно будет только в одном месте. В идеальном случае это вообще единственная причина для создания варианта использования, уровень которого ниже уровня моря.
- *Расширение делает вариант использования трудным для чтения.* Я считаю, что предел читабельности — около двух страниц текста варианта использования и три уровня отступов. (Мои варианты использования в основном короче, чем у других, поэтому ваша страница может быть длиннее.)

8.4. Упражнения

Расширения

8.1. Перечислите то, что может отказать во время функционирования банкомата.

8.2. Перечислите то, что может отказать в первом варианте использования уровня цели пользователя для системы обработки расширений PAF, персонального консультанта по финансам (см. упражнение 4.4).

8.3. Напишите вариант использования Извлечь наличные, содержащий обработку ошибок с помощью предложений с если. Напишите его снова, на этот раз с расширениями сценария. Сравните два варианта.

8.4. Найдите файл требований, написанный в форме, отличной от вариантов использования. Как он учитывает условия ошибок? Что вам нравится в каждом методе разработки, что полезного можно извлечь из этих наблюдений?

8.5. Напишите полный вариант использования для системы PAF, описывая шаги обработки ошибок (система PAF упоминается в упражнении 4.4).

Глава 9

Изменения в технологии и данных

Расширения говорят о том, что система выполняет действия по-разному. То, что происходит, не меняется, но могут меняться способы выполнения действий. Почти всегда это объясняется какими-нибудь изменениями в технологии или различиями в данных, которые необходимо зафиксировать. Записывайте это в список изменений технологии и данных, а не в раздел расширений.

Пример 1

Ваша система должна кредитовать заказчика за возвращенные товары. Вы записываете этот шаг действия:

7. Вернуть заказчику деньги за возвращенные товары.

Уплатить заказчику можно с помощью чека, компьютерной системы электронных платежей или кредита на следующую покупку. Таким образом, вы добавляете

Список изменений в технологии и данных:

7а. Вернуть заказчику деньги с помощью чека, компьютерной системы электронных платежей или кредита на будущие покупки.

Пример 2

Вы детально описываете новый банкомат. Технология усовершенствована до такой степени, что клиенты могут быть идентифицированы по банковской карточке, радужной оболочке глаза или отпечаткам пальцев. Вы записываете:

Основной сценарий:

...

2. Пользователь идентифицирует себя, банк и номер счета.

...

Список изменений в технологии и данных:

2а. Использовать магнитную банковскую карточку, сканирование радужной оболочки глаза или отпечатки пальцев.

Эти пункты изменений не являются расширениями для данного варианта использования. Каждый развертывается в собственное расширение на каком-то более низком уровне этого варианта использования, который вы, возможно, никогда не напишете. Каждое изменение оказывает заметное влияние на смету и план работ, поэтому изменения надо фиксировать и отслеживать. Вы регистрируете возможности с помощью списка изменений в технологии и данных.

В списке изменений нет шагов действий. Записывать туда условия и шаги действия некорректно.

Список изменений в технологии и данных содержится в варианте использования 13.

Если вы решили применить UML-диаграммы вариантов использования, создайте пустой обобщенный базовый вариант использования для основного шага и специализированный вариант использования для каждого изменения. Пустой обобщенный базовый вариант использования указывает на "*что*", но не говорит "*как*". В каждом специализированном варианте использования определяются его собственные шаги, раскрывающие, как это делается. Нотация UML объясняется в приложении А. На рис. 9.1 представлен пример нотации.

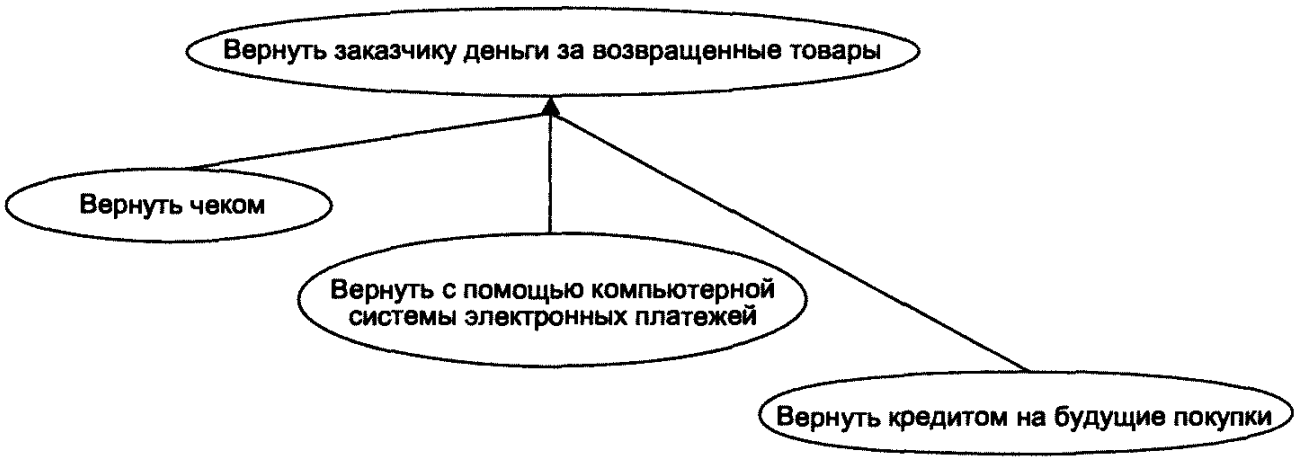


Рис. 9.1. *Изменения в технологии, показанные в нотации UML с помощью специализации*

Глава 10

Связывание вариантов использования

10.1. Подчиненные варианты использования

Шаг действия может быть простым шагом или названием другого варианта использования. Предложение:

Пользователь сохраняет отчет

без выделения или аннотации означает простой, элементарный шаг. Любой из шагов:

Пользователь сохраняет отчет
Пользователь *сохраняет отчет* (UC 35 Сохранить отчет)

указывает на вариант использования, называемый *Сохранить отчет*. Это настолько естественно, что едва ли требуется дополнительное объяснение. Даже случайные читатели вариантов использования понимают, что подчеркнутое (обычно гиперссылка) или *выделенное курсивом* действие означает, что вызывается другой вариант использования, который расширяет указанное действие.* Удобно писать варианты использования, при необходимости свертывая или развертывая действие. Научиться этому недолго, а описание получается компактным.

---

* В терминах UML другой вариант использования называется *включаемым* (included). Я считаю, что начинающим писателям и неопытным читателям будет понятнее, если сказать, что один вариант использования обращается к другому или вызывает другой. Выбор термина — дело ваше.

10.2. Варианты использования расширений

Иногда необходим другой вид связи между вариантами использования, который хорошо согласуется с механизмом расширения. Рассмотрим следующий пример.

> Вы разрабатываете новую программу обработки текста, названную *Wapp*. Основная деятельность пользователя — набор текста. Однако он может изменить масштаб изображения или размер шрифта, запустить программу проверки орфографии или сделать еще что-нибудь из того, что с набором непосредственно не связано. Вы хотите, чтобы никакие события, происходящие с документом, не касались набора текста.
>
> Вы также хотите, чтобы различные команды разработчиков программного обеспечения придумывали новые услуги, не переделывая основной вариант использования для каждой новой возможности. Вы хотите расширить набор требований, избежав проблем.
>
> Ситуация характеризуется следующим образом:
>
> - Имеется основная деятельность, которую можно прервать.
> - Основная деятельность может быть прервана рядом способов и не управляется прерываниями.

Это не то же самое, что основное меню, перечисляющее услуги системы пользователю на выбор. Основным меню управляет выбор пользователя, тогда как в нашем примере основная деятельность не контролируется, а только прерывается другими действиями.

Вы не хотите, чтобы основной вариант использования в этом примере явно называл все прерывающие варианты использования. Иначе сопровождение станет настоящей головной болью, поскольку группе разработчиков, добавляющей новый прерывающий вариант использования, придется редактировать основной вариант использования. А это может привести к повреждению последнего или размножению его версий, необходимости просмотра и т.д.

Применяйте тот же механизм, что описан для расширений сценария, но создавайте новый вариант использования, который называется *вариантом использования расширения* (extension use case) и подобен расширению сценария, за исключением того, что он самостоятелен. Как и расширение сценария, он начинается с условия, которое обращается к ситуации основного варианта использования, где может возникнуть условие. Поместите все это в раздел шаблона Триггер.

Ниже приведено несколько отрывков для текстового процессора *Wapp*. На рис. 10.1 отображена ситуация с *Wapp* в виде UML-диаграммы. С помощью особого стиля коннектора, крючка, я показываю, как вариант использования расширяет другой (или цепляется к другому). Подробности см. в приложении А.

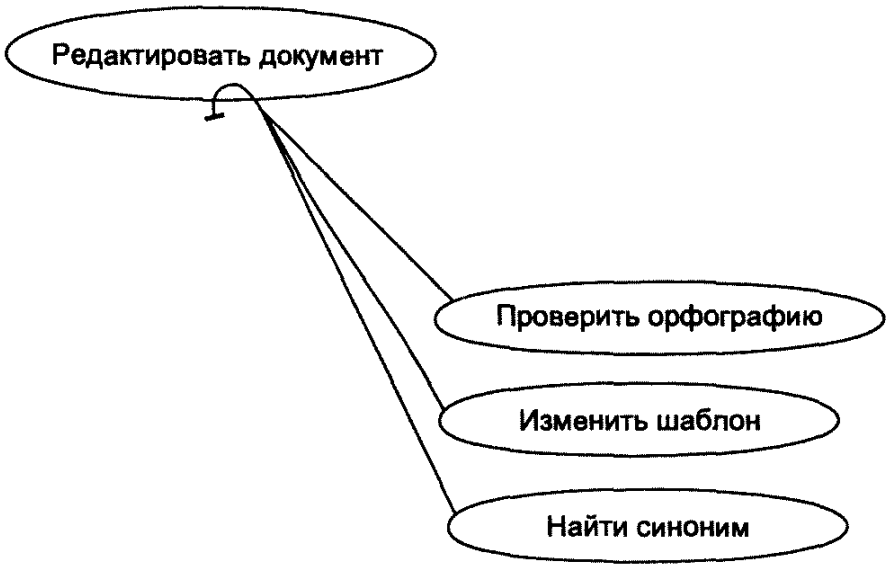


Рис. 10.1. *UML-диаграмма вариантов использования расширения*

Вариант использования

Редактировать документ

Основное действующее лицо: пользователь
Область действия: Wapp
Уровень: цель пользователя
Триггер: пользователь открывает приложение.
Предусловие: отсутствует.
Основной сценарий:
1. Пользователь открывает документ для редактирования.
2. Пользователь вводит и модифицирует текст.
...
... Пользователь сохраняет документ и выходит из приложения.

Вариант использования

Проверить орфографию

Основное действующее лицо: пользователь
Область действия: Wapp
Уровень: подфункция!
Предусловие: документ открыт.
Триггер: любой момент при выполнении варианта использования Редактировать документ, когда документ открыт, и пользователь решает запустить программу проверки орфографии.
Основной сценарий успеха:
... и т.д. ...

Вариант использования

Найти синоним

Основное действующее лицо: пользователь
Область действия: Wapp
Уровень: подфункция!
Предусловие: документ открыт.
Триггер: любой момент при выполнении варианта использования Редактировать документ, когда курсор стоит на слове и пользователь решает запустить тезаурус.
Основной сценарий успеха:
... и т.д. ...

Вариант использования

Изменить шаблон документа

Основное действующее лицо: пользователь
Область действия: Wapp
Уровень: подфункция!
Предусловие: документ открыт.
Триггер: любой момент при выполнении варианта использования Редактировать документ, когда документ открыт и пользователь выбирает эту функцию.
Основной сценарий:
... и т.д. ...

Когда применять варианты использования расширения

Создавайте варианты использования расширения, только когда это необходимо, так как их труднее понимать и поддерживать. В двух ситуациях их создание оправдано.

Наиболее распространенная ситуация — когда пользователь может применить много асинхронных или прерывающих выполнение услуг, которые не должны мешать основному варианту использования. Часто эти услуги разрабатываются разными командами и проявляют себя при работе с "коробочным" программным продуктом, таким как текстовый редактор.

Другая ситуация — когда вы пишете дополнения к документу, излагающему зафиксированное описание требований. Сьюзен Лилли из S.R.A. написала мне:

> Вы работаете над проектом с итеративным процессом и множественными проходами. У вас есть базовые требования для одного прохода. В следующем проходе вы *расширяете* основной вариант использования, добавляя новые функции. Вы не трогаете основной вариант использования.

Если основной вариант использования не зафиксирован, расширение будет работать ненадежно. Изменения в основном варианте использования могут нарушить

условие расширяющего варианта использования, поэтому остерегайтесь вариантов использования в таких случаях. Упражнение 10.2 дает вам возможность поэкспериментировать с вариантами использования расширения этого рода.

Алан Уильямс изложил правило, помогающее решить, должен ли вариант использования вызывать подчиненный вариант использования или подчиненный вариант использования должен расширять основной:

> Если триггер включает аспекты, за которые отвечает основной вариант использования (т.е. основной вариант использования знает, когда, или где, или почему должен выполняться второй вариант использования), значит основной вариант использования включает (вызывает) другой или обращается к другому.
>
> Если триггер включает аспекты, за которые отвечает подчиненный вариант использования (т.е. подчиненный вариант использования знает когда, или где, или почему он должен последовать), подчиненный вариант использования расширяет основной вариант использования.

Обратите внимание, что когда основной вариант использования *вызывает* подчиненный вариант использования, основной называет его, говоря, когда и где он выполняется. Вариант использования, к которому происходит обращение, не содержит названия основного варианта использования. Когда подчиненный вариант использования *расширяет* основной, последний не называет имени или на самом деле ничего не знает о расширяющем (прерывающем) варианте использования. Расширяющий вариант использования называет вариант использования, который он прерывает, и определяет среду, в которой он выполняется.

10.3. Упражнения

Связывание вариантов использования

10.1. Найдите условие в варианте использования уровня цели пользователя для системы PAF (описанной в упражнении 4.4), которое требует выделения подчиненного варианта использования. Напишите этот подчиненный вариант использования и установите с ним связь из варианта использования уровня цели пользователя, уделяя внимание как его успеху, так и неудаче.

10.2. Рассмотрите ситуацию с АТМ, когда вы находитесь не в своем родном банке, а в другом. Напишите подчиненный вариант использования для запроса наличных "банк/банк" и свяжите его с предыдущим вариантом (вариантами) использования, описывающими получением наличных. Сделайте это двумя способами: в виде подчиненного варианта использования, к которому обращается ваш основной вариант использования, и в виде варианта использования расширения. Обсудите с коллегами, какой способ лучше и почему.

Глава 11

Форматы вариантов использования

11.1. Разновидности форматов

Полный формат

Большая часть примеров в этой книге представлена в моем любимом стиле, т.е. в полном формате:

- Одна колонка текста (не таблица)
- Нумерованные шаги
- Отсутствие предложений с если
- Соглашение о нумерации в разделе расширений, включающей комбинации цифр и букв (например, 2a, 2a1, 2a2 и т.д.)

Альтернативные формы — это свободный формат, формат в две колонки и шаблон Rational Unified Process. Все они описаны в следующих разделах. Когда я показываю группе разработчиков один и тот же вариант использования, представленный в различных стилях, они почти всегда выбирают версию в одну колонку с нумерованными шагами, поэтому я продолжаю применять ее сам и рекомендую другим. Ниже приведен базовый шаблон, который группы разработчиков по всему миру ввели в Lotus Notes, DOORS, Word, Access и другие инструменты работы с текстами.

Вариант использования 24

Полный формат шаблона варианта использования <название>

<название должно быть в виде краткой фразы с глаголом в неопределенной форме совершенного вида и отражать цель>

Контекст использования: <более длинное предложение цели, при необходимости условия ее нормального достижения>

Область действия: <область действия проектирования, в которой разрабатываемая система рассматривается как "черный ящик">

Уровень: <один из следующих: обобщенный, цели пользователя, подфункции>

Основное действующее лицо: <ролевое имя для основного действующего лица или описание>

Участники и интересы: <список участников и ключевых интересов в данном варианте использования>

Предусловие: <то, что, как ожидается, уже имеет место>

Минимальные гарантии: <как защищаются интересы участников при всех исходах> Гарантии успеха: <что имеет место, если цель достигнута>

Триггер: <то, что запускает вариант использования, возможно, временное событие>

Основной сценарий:

<запишите здесь шаги сценария от триггера до достижения цели и затем какую-либо очистку> <номер шага> <описание действия>

Расширения:

<запишите здесь по одному расширению, каждое из которых обращается к шагу основного сценария>

<изменяемый шаг> <условие>: <действие или подчиненный вариант использования>

<изменяемый шаг> <условие>: <действие или подчиненный вариант использования>

Список изменений в технологии и данных:

<запишите здесь изменения, которые в итоге вызовут ветвление в сценарии>

<номер шага или изменения> <список изменений>

<номер шага или изменения> <список изменений>

Вспомогательная информация:

<то, что необходимо для проекта в качестве дополнительной информации>

Свободный формат

Пример варианта использования в свободном формате, взятый из рукописи Гради Буча, контрастирует с вариантом использования в полном формате. Я добавил основное действующее лицо, область действия и уровень, чтобы оформить этот пример. Обратите внимание, что во втором абзаце все-таки присутствуют расширения.

Вариант использования 25

Действительная регистрация в системе (версия в свободном формате)

Основное действующее лицо: пользователь

Область действия: приложение

Уровень: подфункция

Когда пользователь обнаруживает себя, система запрашивает его имя и пароль. Система подтверждает, что введенное имя ей знакомо и пароль правильный. После этого пользователь получает доступ ко всем остальным командам пользователя.

Если имя пользователя входит в группу администраторов, пользователь получает доступ ко всем командам, как уровня пользователя, так и уровня администратора. Если введенное имя пользователя не зарегистрировано либо пароль не соответствует этому имени, пользователь отвергается.

Таблица в одну колонку

Некоторым разработчикам нравится помещать шаги сценария в таблицу. Мне кажется, что линии таблицы распыляют внимание, однако, есть много приверженцев табличного стиля. Таблица 11.1 представляет собой такой шаблон.

Таблица 11.1. *Вариант использования формата таблицы в одну колонку*

Вариант использования #	<название — это цель в виде короткой фразы с глаголом неопределенной формы несовершенного вида>	
Контекст использования	<при необходимости более длинное предложение контекста использования>	
Область действия	<то, в чем система рассматривается как "черный ящик">	
Уровень	<один из следующих: обобщенный, цели пользователя, подфункции>	
Основное действующее лицо	<ролевое имя для основного действующего лица или описание>	
Участники и интересы	Участник	Интерес
	<имя участника>	<запишите сюда интерес этого участника>
	<имя участника>	<запишите сюда интерес этого участника>
Предусловия	<то, что, как ожидается, уже имеет место>	
Минимальные гарантии	<интересы, защищенные при всех исходах>	
Гарантии успеха	<интересы, удовлетворенные при успешном завершении>	
Триггер	<действие, запускающее вариант использования>	

Таблица 11.1 (продолжение)

Описание	Шаг	Действие
	1	<запишите здесь шаги сценария от триггера до достижения цели и затем необходимую очистку>
	2	<...>
	3	
Расширения	Шаг	Ветвящееся действие
	1а	<вызывающее ветвление условие><действие или название подчиненного варианта использования>
Изменения в технологии и данных	1	<список изменений>

Таблица в две колонки

Ребекке Вирс-Брок принадлежит идея *диалога*, отличительной чертой которого является использование двух колонок, причем действия основного действующего лица записываются в левой колонке, а действия системы — в правой. Диалоги чаще всего записываются в процессе подготовки к проектированию интерфейса пользователя, поэтому они могут содержать больше деталей действий пользователя.

Вариант использования можно записать как таблицу из двух колонок. Такая форма отличается ясностью, но часто бывает довольно длинной, порой более трех страниц (например, вариант использования 36). Обычно к моменту, когда мы модифицируем текст, чтобы он укладывался в 3-9 шагов на соответствующих уровнях цели, описание настолько простое и ясное, что потребность в колонках отпадает.

Авторы книги *Software for Use* (1999) пользуются диалоговым форматом при формировании требований к интерфейсу пользователя в существенных вариантах использования. Разница с предыдущим подходом в том, что в существенном варианте использования все движения пользователя (детали интерфейса) опущены, поэтому он получается очень коротким.

Трудность использования двухколоночного формата для фиксации требований к поведению состоит в том, что негде написать о вспомогательных действующих лицах. Можно было бы добавить для них третью колонку, но у меня нет сведений о таком варианте. *Диалоги* и *существенные варианты использования* подходят скорее к требованиям к интерфейсу пользователя, чем к поведению системы в целом.

Принимая все это во внимание, многие все же считают двухколоночную форму привлекательной. Поэкспериментируйте с ней. Таблица 11.2 представляет фрагмент сценария в двухколоночном формате.

Таблица 11.2. Двухколоночная таблица

Клиент	Система
Вводит номер заказа	
	Определяет, что номер заказа соответствует выигравшему номеру месяца.
	Регистрирует пользователя и номер заказа как победителя данного месяца.
	Посылает электронной почтой сообщение менеджеру по продажам.
	Поздравляет клиента и инструктирует его, как получить приз.
Выходит из системы	

Стиль RUP

Технология Rational Unified Process (RUP) использует шаблон, весьма похожий на шаблон полной формы. Нумеровать шаги не обязательно. Расширениям, которые называются *альтернативными потоками событий*, отведены собственные озаглавленные разделы. Все, о чем идет речь в этой книге, хорошо согласуется с этим шаблоном. Несмотря на переизбыток номеров заголовков, он хорошо смотрится и прост в употреблении. Вот его базовая форма:

1. **Название варианта использования**
   - **1.1. Краткое описание**
     ...текст...
   - **1.2. Действующие лица**
     ...текст...
   - **1.3. Триггеры**
     ...текст...
2. **Поток событий**
   - **2.1. Основной поток**
     ...текст...
   - **2.2. Альтернативные потоки**
     - 2.2.1. Условие 1
       ...текст...
     - 2.2.2. Условие 2
       ...текст...
     - 2.2.3. ...
3. **Специальные требования**
   - **3.1. Платформа**
     ...текст...
   - **3.2. ...**

4. **Предусловия**

   ...текст...

5. **Постусловия**

   ...текст...

6. **Точки расширения**

   ...текст...

Компания Rational Software Corporation прислала мне в качестве примера вариант использования, приведенный ниже. Обычно в наборе инструментов ему сопутствуют диаграмма варианта использования и другие рабочие продукты. Этот вариант использования кажется мне вполне очевидным, думаю, вы придете к тому же мнению. Обратите внимание, что и простые абзацы, и нумерованные шаги используются в соответствии с тем, как разработчик представляет себе лучший вариант. Для совместимости с примерами в этой книге я добавил к заголовку две пиктограммы, но сам шаблон оставил без изменений.

В варианте использования 32 также используется шаблон RUP.

Вариант использования 26

Записаться на курсы

1. **Название варианта использования:** Записаться на курсы

   1.1. Краткое описание

   Этот вариант использования позволяет студенту записаться на курс, предлагаемый в текущем семестре. Студент может также модифицировать или удалить выбранные курсы, если изменения сделаны в пределах периода добавления/удаления в начале семестра. Система каталога курсов предоставляет список всех курсов, предлагаемых в текущем семестре.

   Основное действующее лицо этого варианта использования — студент. Система каталога курсов — действующее лицо внутри варианта использования.

2. **Поток событий:**

   Вариант использования начинается, когда студент выбирает функцию "работа с графиком" из основной формы. [См. прототип интерфейса пользователя для формата экрана и полей.]

   2.1. Основной поток

   2.1.1. Создать график

   2.1.1.1. Студент выбирает функцию "создать график".

   2.1.1.2. Система выводит пустую форму графика [см. прототип интерфейса пользователя для формата экрана и модель предметной области для требуемых полей].

   2.1.1.3. Система выводит список предлагаемых курсов из системы каталога курсов. [Как он выбирается и отображается? Текст? Ниспадающие списки?]

2.1.1.4. Студент выбирает 4 курса в качестве основных и 2 в качестве альтернативных из списка доступных курсов. Когда выбор закончен, студент использует функцию "зафиксировать выбор". [Определить термины "основной курс" и "альтернативный курс" в глоссарии проекта. Должно быть выбрано точно 4 и 2 курса или "до 4..." и т.д.?]

2.1.1.5. На этом шаге подчиненный поток Добавить курс выполняется для каждого выбранного курса.

2.1.1.6. Система сохраняет график. [Когда обновляется главный график? Немедленно? По ночам (в пакетном режиме)?]

2.2 Альтернативные потоки

2.2.1. Модифицировать график

2.2.1.1. Студент выбирает "модифицировать график".

2.2.1.2. Система находит и отображает текущий график студента (т.е. график на текущий семестр). [Это все, что можно получить для текущего семестра?]

2.2.1.3. Система отыскивает список всех имеющихся на текущий семестр курсов из системы каталога курсов. Система показывает список студенту.

2.2.1.4. Студент может затем модифицировать свой выбор курсов, удаляя курсы и добавляя новые. Студент выбирает новые курсы из списка имеющихся курсов. Студент также может удалить любой курс из имеющегося графика. Когда редактирование завершается, студент использует функцию "зафиксировать выбор".

2.2.1.5. На этом шаге подчиненный поток Добавить курс выполняется для каждого выбранного курса.

2.2.1.6. Система сохраняет график.

2.2.2. Удалить график

2.2.2.1. Студент выбирает функцию Удалить график.

2.2.2.2. Система отыскивает и показывает текущий график студента.

2.2.2.3. Студент выбирает "удалить".

2.2.2.4. Система запрашивает подтверждение удаления.

2.2.2.5. Студент подтверждает удаление.

2.2.2.6. Система удаляет график. [В какой момент освобождается место студента в общем графике?]

2.2.3. Сохранить график

В любой момент студент может сохранить график, выбрав функцию "сохранить". Текущий график сохраняется, но студент не попадает в список ни одного из выбранных курсов. В его графике курс помечается как "выбранный".

2.2.4. Добавить курс

Система подтверждает, что необходимые предварительные условия (пройденные ранее курсы) студентом соблюдены, и что этот курс открыт для записи студентов. Затем система добавляет студента в список выбранного курса. В графике курс получает пометку "зарегистрированный".

2.2.5. Не выполнены предварительные условия или набор на курс закончен

Если в подчиненном потоке Добавить курс система определяет, что студент не выполнил необходимые предварительные условия или что набор на выбранный курс уже закончен, выводится сообщение об ошибке. Студент может либо выбрать другой курс, либо отменить операцию. В этой точке вариант использования перезапускается.

2.2.6. График не найден

Если в подчиненных потоках Модифицировать график или Удалить график система не смогла отыскать график студента, выводится сообщение об ошибке. Студент подтверждает прием сообщения об ошибке и вариант использования перезапускается.

2.2.7. Система каталога курсов недоступна

Если система не может осуществить взаимодействие с Системой каталога курсов после определенного количества попыток, система выдаст студенту сообщение об ошибке. Студент подтверждает прием сообщения об ошибке, и вариант использования завершается.

2.2.8. Запись на курсы закрыта

Если студент выбирает функцию "работа с графиком", а запись на текущий семестр уже закрыта, студенту выдается сообщение, и вариант использования завершается. Студенты не смогут записаться на курсы после того, как запись на текущий семестр закрылась.

3. Специальные требования

В настоящее время для данного варианта использования специальные требования не определены.

4. Предусловия

4.1. Вход в систему

До запуска этого варианта использования студент должен войти в систему.

5. Постусловия

Постусловия, связанные с этим вариантом использования, отсутствуют.

6. Точки расширения

Точки расширения, связанные с этим вариантом использования, отсутствуют.

Стиль с предложениями с если

Программисты просто не могут не включить в текст предложения с если. Многие говорят, что чем учиться писать расширения, лучше писать так:

> **Если заказ соответствует выигравшему номеру, то <все, что полагается делать для выигравшего номера>, иначе сообщить клиенту, что его номер не выиграл.**

Если бы в варианте использования было только одно предложение с если, я бы не возражал. В самом деле, в модели варианта использования нет ничего, что бы препятствовало употреблению предложения **"если ... то ... иначе"**. Когда таких предложений два или больше, текст становится запутанным. А внутри предложения с если может быть другое подобное предложение.

Когда кто-либо настаивает на употреблении предложений с если, я предлагаю ему так и сделать и рассказать мне о приобретенном опыте. Все приходили к выводу, что предложения с если делают вариант использования нечитабельным, и возвращались к стилю с расширениями. Поэтому я настоятельно рекомендую не писать в сценарии такие предложения.

Стиль с использованием языка Оккам

Если вы решили построить вариант использования в соответствии с формальной моделью, обратитесь сначала к языку Оккам (Occam), изобретенному Тони Хоаром. По сравнению с другими известными мне языками, Оккам облегчает комментирование альтернативных, параллельных и необязательных последовательностей, которые вам понадобятся. Однако я не знаю, как Оккам обрабатывает исключительные ситуации, необходимые для стиля с расширениями.

На Оккаме вы напишете так:

```
ALT
    Альтернатива 1
    Альтернатива 2
    TLA (этим кончаются альтернативы)
PAR
    Параллельное действие 1
    Параллельное действие 2
    RAP (этим кончаются параллельные действия)
OPT
    Необязательное действие
    TPO
```

Если вы все-таки решили создать или употребить формальный язык для вариантов использования, примените вариант использования 22 в качестве первой пробы, так как он содержит параллельные, асинхронные, исключительные действия и совместную обработку. Думаю, вы увидите, насколько хорошо естественный язык справляется с этими действиями, оставаясь простым и понятным.

Стиль диаграмм

Вариант использования детализирует действия и взаимодействия действующих лиц в процессе достижения цели. Существует ряд нотаций в виде диаграмм, способных выразить это: диаграммы последовательности, кооперативные диаграммы, диаграммы деятельности и сети Петри. Если вы используете одну из этих нотаций, вы еще можете применить большинство идей этой книги, чтобы сделать ваши тексты и схемы информативными.

Графические нотации имеют два неудобства в использовании. Во-первых, конечные пользователи и бизнесмены-руководители вряд ли знакомы с ними и не проявляют достаточно терпения, чтобы освоить нх. Графические нотации отталкивают ценных читателей.

Во-вторых, диаграммы не показывают всего, что необходимо написать. CASE-средства, которые я видел, реализуют варианты использования с помощью диаграмм взаимодействия и вынуждают писателя прятать текст за всплывающим окном диалога, связанным со стрелками взаимодействия. При этом невозможен беглый просмотр варианта использования: читатель должен дважды щелкнуть по каждой стрелке, чтобы увидеть, что за ней спрятано. Я неоднократно был свидетелем того, как разработчики и читатели вариантов использования отказывались от применения CASE-средств для построения диаграмм. Они делали выбор в пользу простой текстовой обработки документов.

В следующем разделе обсуждается стиль на основе диаграмм, не подходящий для вариантов использования.

UML-диаграмма варианта использования

UML-диаграмма варианта использования, состоящая из эллипсов, стрелок и человекоподобных фигурок, не является нотацией для фиксации вариантов использования. Эллипсы и стрелки показывают объединение и декомпозицию вариантов использования, а не их содержание.

Помните, что вариант использования обозначает цель. Он состоит из сценариев, состоящих, в свою очередь, из шагов действия, каждый из которых формулируется как цель. Поэтому его можно раскрыть, чтобы превратить в самостоятельный вариант использования. Можно сделать цель варианта использования эллипсом, развернув каждый шаг действия как эллипс и проведя стрелку от эллипса варианта использования к эллипсу шага действия, показывая, что первый включает последний. Можно продолжить декомпозицию от самого высокого до самого низкого уровня варианта использования, производя гигантскую диаграмму полной декомпозиции поведения системы.

Однако диаграмма с эллипсами упускает существенную информацию, например, какое действующее лицо выполняет каждый шаг и следит за порядком шагов. Она полезна в качестве оглавления и с этой целью ее следует сохранить (см. памятку 24 и раздел А.1).

Не пытайтесь заменить эллипсами текст варианта использования. Один студент на лекции спросил меня: “Когда вы начинаете писать текст? На уровне листа декомпозиции эллипса?”

Ответ состоит в том, что вариант использования существует в тексте и все (или какие-либо) схемы являются лишь иллюстрациями, помогающими читателям локализовать текст, который необходимо прочитать.

Многие находят полезной диаграмму вариантов использования самого высокого уровня (которая показывает внешних действующих лиц и варианты использования уровня цели пользователя). Она обеспечивает контекстную диаграмму, подобную тем, что используются разработчиками уже не один год. Ценность диаграмм вариантов использования быстро падает со снижением уровня. Более подробно это обсуждается в приложении А.

11.2. Факторы, влияющие на стиль варианта использования

На конференции OOPSLA в 1998 г. двенадцать опытных разработчиков и преподавателей вариантов использования собрались, чтобы обсудить общие вопросы, связанные с вариантами использования и факторами, которые побуждают людей писать их по-разному. Ниже описаны эти факторы.

Возможно, сочетание некоторых аспектов, изложенных ниже, заставит вас задуматься о том, что вы должны работать не так, как предполагалось. Будьте терпеливы и пишите варианты использования так, как вам это нужно. Учитывая все эти факторы, вы поймете, как писать удобочитаемые варианты использования.

Уравновешивающие факторы: установки бизнеса, общественные отношения, конфликтующие подходы

Вы хотите ввести в обиход варианты использования, но попадаете в подобную ситуацию или вам навязывают дискуссию:

Мы всегда делали это другим способом.

При разнообразии подходов вы можете решить, что:

Разработчики имеют предубеждение.

Существуют различные подходы, и приверженцы разных подходов делают одни и те же вещи по-разному.

Разработчики вариантов использования применяют словарь, отличный от словаря их читателей.

Уровень понимания

В разное время в разных местах разные люди понимают одно и то же по-своему. Основанием для изменения стиля, рекомендованного для вариантов использования, могут быть следующие факторы.

Ваши знания о:

- Предметной области
- Вариантах использования в общем

Этап жизненного цикла, к которому относятся эти знания

Наличие необходимости в определении содержания или затрат

Необходимое в данный момент рассмотрение (вширь или вглубь)

- Присутствие скрытого или затянувшегося анализа
- Люди склонны выделять вещи, которые знают
- Планирование или глубина знания либо знание предметной области

Потребности участников

Какая точка зрения вам ближе:

Заказчик является читателем, потребителем варианта использования, который весьма доволен описанием высокого уровня.

Работник фирмы (отдела IT) — писатель или разработчик, заинтересованный в детальном описании.

Несколько групп представляют несколько точек зрения на варианты использования или расходятся во мнении о построении полной либо неполной модели.

Участвуют ли разные читатели? Если да, кто они?

Опыт против формализма

Опыт: каждая группа разработчиков включает людей, незнакомых с вариантами использования, но они быстро становятся "опытными" писателями. Имеющие опыт знают, как экономить усилия; новичкам нужны ясные указания и непротиворечивые инструкции.

Формализм: возможно, руководитель или принятая в отделе методология диктует формальный (или неформальный) стиль описания, несмотря на любой опыт или его отсутствие.

Охват

Широта охвата зависит от состава группы, писательского мастерства ее членов, их взаимодействия и необходимости охвата всей проблемы либо передачи информации читателям.

На изменение охвата влияют:

Эксперты предметной области (они могут узко сфокусировать предметную область)

Количество писателей

Количество читателей

Количество разработчиков

Знание (или его отсутствие) бизнесменами своих целей

Решение работать над общей моделью

Географическая разбросанность членов группы

Непротиворечивость

Необходимость непротиворечивости содержания может вступить в конфликт с неопределенностью или противоречивостью требований заказчика:

Непостоянство требований

Постоянство формата

Сложность

На сложность варианта использования влияют следующие факторы:

Полнота описания

Полное описание проблемной области

Несколько точек зрения

Упрощение взгляда на систему

Упрощение представления

Подробное представление

Узкий взгляд или широкий взгляд

Сложность проблемы

Желание добавить технические подробности (особенно если проблема трудна)

Следующие факторы влияют на сложность системы:

Бессилие анализа (сложность системы ошеломляет аналитика)

Количество профилей действующих лиц

Количество функциональных точек

Тип системы

Степень требовательности пользователей

Режим реального времени

Встроенная система (она должна быть устойчива к ошибкам)

Противоречие

Необходимо разрешить противоречие заказчика, но неопределенность маскирует это противоречие.

Завершенность

Завершенности могут препятствовать следующие факторы:

Требования, необходимые для разработки, неполны.

Писатели не имеют доступа к пользователям (пользователи не являются заказчиками).

Цели в сравнении с задачами — что выполнять или как это выполнять

Пользователи часто определяют требования, а не способ использования.

Контекст может противоречить способу использования.

Действия и задачи описывают, что происходит в системе, а не почему это происходит.

Ресурсы

Для создания хороших вариантов использования требуется время, но время проекта ограничено. Руководству приходится покупать незаконченный проект.

Другие факторы

Другие факторы, влияющие на процесс создания вариантов использования:

Требования к инструментам (поддержка)

Неизвестная цель

Необходимость разбивать список требований на части для последующего анализа

Проектирование без ограничений в сравнении с заданным уровнем проектирования

Чистый стиль или понятное проектирование

Абстрактные или конкретные варианты использования

Возможность трассировки

Общая скорость работы

Получился большой список. Вы можете использовать его, чтобы принять решение о степени формализма; о том, следует ли начать с малого, а потом увеличить темпы; как много надо написать или как организовать создание вариантов использования; насколько широко или до какой степени точности следует разработать вариант использования, прежде чем его детализировать.

11.3. Стандарты для пяти типов проектов

Вы работаете над проектом. Прочитав эту книгу, вы с коллегами знакомы с основными идеями. Теперь вопрос в том, каким стандартам следовать. Ответ зависит от того, кто вы, какими навыками обладаете и какова ваша задача в данный момент. Сопоставьте ваш ответ с приведенным выше списком факторов.

Далее следуют стандарты для пяти ситуаций. В каждой из них основной выбор происходит между свободным стилем и полной формой описания.

1. Выявление требований, даже если варианты использования не будут применяться в окончательном документе, излагающем требования.
2. Моделирование бизнесс-процессов.
3. Формулирование или определение объема требований к системе.
4. Составление функциональных требований к небольшому проекту, который следует завершить в очень сжатые сроки.

5. Формулирование детальных функциональных требований в начале длительного или объемного проекта.

Вы можете применять эти стандарты неизмененными или настроить их в соответствии с вашими корпоративными нуждами или потребностями момента либо согласно приведенным в этой книге принципам.

Ниже я использую примеры с компанией MyCo по разработке новой системы Acura взамен старой системы BSSO. Напоминаю: следует писать действительные названия, а не слова *корпорация* или *система*.

Для выявления требований

Вариант использования 27

Шаблон выявления — стрямкать новую бутявку

Область действия: Acura

Уровень: уровень моря

Контекст: потребности бокра стрямкать новую бутявку, так как старая подудонилась (описание цели в контексте функционирования)

Основное действующее лицо: бокр (или кто-то еще)

Участники и интересы: бокр, MyCo, ... (кто-то или что-то еще)

Предусловия: что должно быть истинным, прежде чем можно будет запустить данный вариант использования?

Триггеры: бокр выбирает функцию трямканья (все, что может этим быть).

Основной сценарий:

... Абзац сочинения, описывающего успешное трямканье бутявки бокром с помощью Acura... действующие лица делают одно, другое и еще кое-что. ... Еще один абзац сочинения, описывающего условия и альтернативные пути в попытке стрямкать бутявку или потерпеть неудачу... действующие лица делают то, се и еще кое-что.

Частота события: несколько раз в день

Открытые вопросы: ... полезно заполнять этот раздел...

Используйте этот шаблон, если ваша конечная цель — раскрыть требования (см. вставку в главе 1). Имейте в виду, что требования можно написать в форме, отличной от варианта использования. Таким образом, игра заключается в том, чтобы быстро проскочить через варианты использования, проектируя их в ходе работы.

Этот шаблон написан в свободной форме. Сохраняйте в шаблоне участников с интересами, чтобы напоминать каждому об их требованиях, но не включайте гарантии системы.

Ваши варианты использования будут, скорее всего, "черными ящиками" , и большинство из них — уровня цели пользователя Можете создать варианты использования более высокого уровня для контекста, но не опускайтесь ниже уровня моря.

Для моделирования бизнес-процесса

Вариант использования 28

Шаблон бизнесс-процесса — достигнуть успешного функционирования компании

Область действия: работа MyCo
Уровень: обобщенный
Контекст: описание цели в контексте работы
Основное действующее лицо: основное действующее лицо, кем бы оно ни было
Участники и интересы: кто-либо или что-либо им соответствующее
Минимальные гарантии: какими бы они ни были
Гарантии успеха: какими бы они ни были
Предусловия: что должно быть истинным, прежде чем можно будет запустить данный вариант использования.
Триггеры: все, что может этим быть.
Основной сценарий:
1. ... шаги действия...
2. ...
Расширения:
1а. ... Условия расширения:
    1а1. ... шаги действия...
Частота события: какая-нибудь
Открытые вопросы: ... полезно заполнять этот раздел...

Это шаблон предназначен для перепроектирования бизнесс-процесса или процесса управления новым программным продуктом. Такие варианты использования читают высшие руководители, начальники отделов и высшие должностные лица, поэтому старайтесь сделать их читабельными и не акцентируйте внимание на деталях данных. Нумеруйте шаги, чтобы виден был порядок следования. Не забудьте описать обработку ошибок в расширениях, так как при этом обнаруживаются важные правила бизнеса.

Варианты использования высшего уровня, предельные, будут множеством "черных ящиков", , показывающих взаимодействие компании с внешними партнерами. Они нужны либо как спецификации, с которыми будет сверяться бизнес-процесс, либо для определения контекста для вариантов использования типа "прозрачный ящик". Варианты использования типа "прозрачный ящик", , работающие с системами, подобными *MyCo Operations*, демонстрируют организацию в действии с людьми и отделами, работающими совместно для обеспечения должных характеристик функционирования организации. Цели пользователя будут от облака до уровня моря. Для примера шаблона я выбрал обобщенную цель уровня воздушного змея.

Для определения объема требований к системе

Вариант использования 29

Шаблон для определения объема — определить объем требований к системе

Область действия: Acura

Уровень: голубой

Контекст: поместите здесь предусловия или условия нормального использования.

Основное действующее лицо: кто-нибудь

... поместите здесь несколько предложений, описывающих успешное осуществление действующими лицами необходимой цели в основном сценарии ...

... поместите здесь несколько предложений, касающихся некоторых альтернативных путей и обработок...

Частота события: как часто

Открытые вопросы: ... всегда полезно отметить...

Этот шаблон предназначен для проектирования требований к системе, чтобы оценить ее объем и конфигурацию. Позднее вы сможете детализировать эти требования, преобразуя их в соответствии с полной формой. Можно проектировать и непосредственно на основе вариантов использования в свободном формате, если ваш проект удовлетворяет профилю (см. раздел 3.1 и памятку 19).

Шаблон дан в свободном формате, так как соответствует раннему этапу работы на среднем уровне точности, а разрабатываемая система может быть как системой, так и бизнес-процессом. Цели могут быть любого уровня, включая подфункции, поскольку затраченные на разработку усилия зависят в основном от сложности вариантов использования уровня подфункции. В предыдущем примере я использовал систему Acura и цель пользователя (см. вариант использования 25).

Для небольшого проекта, который следует завершить в очень сжатые сроки

Вариант использования 30

Очень сжатый шаблон: добыть бананан

Область действия: Acura

Уровень: цель пользователя

Контекст:

Основное действующее лицо:

Участники и интересы:

Минимальные гарантии и гарантии успеха:

Предусловия:
Триггеры:
Основной сценарий:
... Абзац, в котором описывается действующее лицо, добивающееся успеха в основном сценарии...
Расширения:
... Абзац, в котором упоминаются все альтернативные пути и обработки...
Частота события:
Открытые вопросы: ...

Используйте этот шаблон, если необходимы требования в письменном виде, но проект небольшой и должен быть завершен в очень сжатые сроки. Из соображений экономии избегайте накладных расходов, связанных с нумерацией и полным форматом шаблона. Поэтому подойдет свободная форма, но все же не забывайте фиксировать предусловия, гарантии и расширения. Полагаю, вы будете тщательно работать над улучшением внутренних связей проекта, как описано в памятке 19.

Для детальных функциональных требований

Вариант использования 31

Название варианта использования — Оризовать позвение

Область действия: Acura
Уровень: цель пользователя
Контекст использования:
Основное действующее лицо:
Участники и интересы:
Минимальные гарантии:
Гарантии успеха:
Предусловия:
Триггеры:
Основной сценарий:
1. ...
2. ...
Расширения:
1a. ...
    1a1. ...
Частота события:
Открытые вопросы: ...

Употребляйте этот шаблон, если ваша задача — собрать требования к поведению, применяя все возможности полного формата варианта использования. Это по-

дойдет для более объемных или дорогих проектов, проектов с фиксированной ценой, в случае географически распределенной группы, в начале этапа расширения и исследования объема вариантов использования, созданных ранее.

Разрабатываемая система может быть чем угодно, как и действующие лица и цели. Здесь также применяется система Acura и уровень цели пользователя для примера шаблона.

11.4. Заключение

Все форматы варианта использования выражают приблизительно одинаковую основную информацию. Рекомендации и правила в этой книге применимы к каждому. Не слишком беспокойтесь о том, какой формат использовать в вашем проекте, выбирайте тот, который будет удобен и авторам, и читателям.

11.5. Упражнения

Предложения с если

11.1. Перепишите следующий вариант использования, избавляясь от предложений с если и употребляя предложения цели на соответствующих уровнях и альтернативные сценарии или расширения.

Оказать услугу по чистке свечей зажигания

Условия: свечи грязные или клиент просит оказать услугу.

1. Откройте капот.
2. Найдите свечи зажигания.
3. Накройте щиток защитными материалами.
4. Удалите свечи.
5. Если свечи треснули или изношены, замените их.
6. Почистите свечи.
7. Прочистите зазор в каждой свече.
8. Настройте свечу, если необходимо.
9. Проверьте свечу.
10. Замените свечи.
11. Подключите провода зажигания к соответствующим свечам.
12. Проверьте характеристику двигателя.
13. Если она в порядке, переходите к шагу 15.
14. Если нет, проделайте указанные шаги.
15. Промойте инструмент и оборудование.
16. Сотрите смазку с автомобиля.
17. Закончите необходимую бумажную работу.

Итог: двигатель работает ровно.

Часть 2

Часто обсуждаемые темы

Глава 12

Когда считать работу завершенной

Вы "закончили" работу, если:

- Назвали *всех основных действующих лиц и все цели пользователя* по отношению к системе.
- Зафиксировали *все условия запуска* для системы в качестве триггеров вариантов использования либо как условия расширения.
- Написали *все варианты использования уровня цели* наряду с обобщенными и уровня подфункции, необходимыми для их поддержки.
- Каждый вариант использования написан достаточно ясно, чтобы:
  - Спонсоры определили, действительно ли достигается желаемое.
  - Пользователи согласились, что это именно то, что они хотят или могут принять в качестве поведения системы.
  - Разработчики смогли разработать систему с данным набором функций.
- Спонсоры *согласны* с тем, что набор вариантов использования удовлетворяет всем их потребностям (в настоящий момент).

ВСЕ ОСНОВНЫЕ ДЕЙСТВУЮЩИЕ ЛИЦА И ИХ ПОЛЬЗОВАТЕЛЬСКИЕ ЦЕЛИ. Они определяют границы того, что должна выполнять система. Вам не с чем сравнить список основных действующих лиц и цели пользователей, есть только мнения людей, которые должны принимать систему. Поэтому вы не можете быть уверены, что сделали всю работу. Стоит несколько раз просмотреть список в режиме "мозгового штурма".

ВСЕ УСЛОВИЯ ЗАПУСКА. Они представляют собой тонкую настройку границ. Системе придется реагировать на каждое из них. В вариантах использования некоторые события запуска появляются как триггеры вариантов использования, например **Пользователь вставляет карточку в автомат, Клиент обращается с заявлением добавить статью в страховой полис или удалить из него статью, или Пользователь выбирает функцию обновления программного обеспечения.** Другие включены в расширения сценария, например Пользователь нажал на клавишу Cancel, Система обнаруживает падение мощности и т.д.

Один из способов повторно исследовать набор системных триггеров —определить все элементы, которые имеют жизненный цикл, и затем пересмотреть эти жизненные циклы. Ищите все события, которые приводят к изменению состояния элемента в течение его жизненного цикла.

ОБОБЩЕННЫЕ ВАРИАНТЫ ИСПОЛЬЗОВАНИЯ И ВАРИАНТЫ ИСПОЛЬЗОВАНИЯ УРОВНЯ ПОДФУНКЦИИ. Обобщенные варианты использования создают контекст для вариантов использования уровня цели пользователя. Они отвечают на часто задаваемый вопрос: "Как все эти варианты использования (уровня цели пользователя) совмещаются друг с другом?" Я предпочитаю, чтобы каждый вариант использования находился внутри варианта использования более высокого уровня, вплоть до единственного корневого. Этот корневой вариант использования представляет собой лишь оглавление почти без сюжетной линии, однако для новичков удобно иметь единую точку, из которой можно запустить каждый вариант использования системы.

Варианты использования уровня подфункции поддерживают варианты использования уровня цели пользователя. Они необходимы, только когда их вызывают из нескольких других вариантов использования или для выделения сложного участка поведения системы.

СОГЛАШЕНИЕ О ВАРИАНТАХ ИСПОЛЬЗОВАНИЯ. Варианты использования завершены только тогда, когда и спонсоры, и эксперты по использованию могут их прочесть и с ними согласиться, а также сказать, что варианты использования содержат все, что они хотели. Разработчики должны быть в состоянии построить систему в соответствии с этими спецификациями. Это непростая задача, но это та самая задача составления требований.

Когда вы заканчиваете

Слово "заканчивать" создает впечатление, что вам следует написать все варианты использования от начала до конца, прежде чем начать выполнение задач проектирования. Это не так. В разделе 17.1 рассматривается разработка плана проекта с частичной реализацией вариантов использования; см. также книгу Surviving Object-Oriented Projects (Cockburn, 1998), где подробно описаиа пошаговая разработка программного обеспечения, и статью VW-Staging *(http://members.aol.com/acockburn/papers/vw-stage.htm)*, в которой сжато излагаются методы пошаговой и итеративной разработки.

Разные проектные группы в зависимости от ситуации применяют разные стратегии. Некоторые сразу же проектируют все варианты использования, возможно, чтобы быть готовыми предложить цену в случае контракта с фиксированной ценой. Им следует знать, что их варианты использования будут нуждаться в тонкой настройке в процессе проектирования. Некоторые группы лишь намечают действующих лиц и цели пользователей, откладывая разработку вариантов использования до соответствующего этапа. Другие создают варианты использования в течение каждых шести-девяти месяцев работы, откладывая рассмотрение всех других требований почти до конца работы. Остальные пишут варианты использования прямо перед началом этапа разработки. Каждая из этих стратегий имеет своих приверженцев.

Глава 13

Как работать с большим количеством вариантов использования

Существуют два способа работы с большим количеством вариантов использования: сделать каждый из них покороче или разделить их на отдельные группы. Следует использовать оба метода.

Сделать каждый вариант использования покороче (представление с низким уровнем точности)

Само название варианта использования уже полезно, вот почему оно считается первым уровнем точности. Собрание названий вариантов использования — отличное средство манипулирования полным набором вариантов использования, особенно для оценки, планирования и контроля. Перенесите список названий в электронную таблицу и примените возможности последней для сортировки, упорядочивания и суммирования различных интересующих вас свойств каждого варианта использования. Этот подход обсуждается в разделах 1.5 и 17.1.

Второй уровень точности — краткое описание варианта использования, изложение варианта использования в двух-трех предложениях. Для просмотра краткое описание также можно ввести в электронную таблицу или табличную форму (см. таблицу 3.3). Это полезно для обзора системы и организации работы.

Краткость в описании каждого варианта использования имеет смысл, если вы хотите просмотреть полный набор вариантов использования за один раз. Однако придет время, когда вам понадобится сгруппировать их.

Создать группы вариантов использования

Если вы работаете с инструментами обработки текстов, такими как Lotus Notes, электронные таблицы или текстовые редакторы, вы можете сформировать группы

(кластеры) вариантов использования с помощью обычных методов расстановки меток. В инструментах, поддерживающих UML, такие группы называются пакетами. Ниже описаны четыре общих и достаточно эффективных метода группировки.

ПО ДЕЙСТВУЮЩЕМУ ЛИЦУ. Наиболее очевидный способ группировки вариантов использования — по основному действующему лицу. Однако при 80-100 вариантах использования этот способ неэффективен. На этом уровне у вас будет либо слишком много вариантов использования на одно основное действующее лицо, либо слишком много основных действующих лиц, либо слишком много пересечений между действующими лицами и вариантами использования.

ПО ОБОБЩЕННОМУ ВАРИАНТУ ИСПОЛЬЗОВАНИЯ. Некоторые варианты использования естественно группируются по своему жизненному циклу или (в больших проектах) по их месту в жизненном цикле. Такие связанные варианты использования обнаруживаются в обобщенных вариантах использования. Если вы не пишете обобщенные варианты использования, вам все же может понадобиться сгруппировать варианты использования, чтобы создавать, обновлять и удалять информацию определенного вида, а это и есть группа. В одном проекте система поддерживала эквивалент чековой книжки для клиентов. Мы назвали все изменяющие чековую книжку варианты использования вариантами использования чековой книжки. Эта группа разрабатывалась одной командой разработчиков, и все ее члены двигались вперед таким образом, что руководителям проекта было легко управлять разработкой.

ПО КОМАНДЕ РАЗРАБОТЧИКОВ И ВЕРСИИ. Группировка вариантов использования по команде разработчиков, которая должна реализовывать проект, и номеру версии упрощает контроль над выполнением работ. В таком случае правомерен вопрос: "Собирается ли команда, отвечающая за группу профилей пользователей, разработать ее в срок?" Группировка по команде разработчиков возможна даже в больших проектах.

ПО ПРЕДМЕТНОЙ ОБЛАСТИ. Если в проекте более 100 вариантов использования, разработчики автоматически разделяют их по предметным областям. Обычно именовать естественные предметные области не составляет труда. Один проект использовал информацию клиента, продвижения, выписку счетов и рекламную деятельность. В другом были такие предметные области, как прием заказов, маршрутизация, отслеживание, доставка и выписка счета. Зачастую внутри различных предметных областей существуют подпроекты. Каждая предметная область может содержать от 20 до 100 вариантов использования.

Несмотря на то, что отслеживать 240 вариантов использования довольно сложно, отслеживать 15-20 групп вполне приемлемо. В большом проекте я бы в первую очередь сгруппировал варианты использования по предметной области, чтобы получить 3-6 групп по 20-100 вариантов использования в каждой, а затем сгруппировал бы их по версии и команде разработчиков. В зависимости от количества вариантов использования в этих группах, я бы подытожил прогресс в работе с помощью соответствующих групп или обобщенных вариантов использования.

Глава 14

CRUD и параметризованные варианты использования

14.1. Варианты использования CRUD

До сих пор нет общего мнения относительно того, как организовать все маленькие варианты использования типа *Создать завиток, Найти завиток, Обновить завиток* и *Удалить завиток*. Они известны как варианты использования CRUD (аббревиатура названий операций над базами данных Create, Retrieve, Update и Delete). Вопрос в том, входят ли все они в один вариант использования, чей размер больше варианта использования Управлять завитками, либо существуют самостоятельно?

В принципе они самостоятельны, поскольку каждый представляет собой отдельную цель, возможно, осуществляемую другим лицом с другим уровнем допуска. Однако они загромождают набор вариантов использования и могут утроить число единиц, которые нужно отслеживать.

Мнения о том, как лучше всего действовать в отношении вариантов использования CRUD, разделились. Сьюзен Лилли из S.R.A. выступает за то, чтобы держать их отдельно, так как это поможет отследить, какое из основных действующих лиц имеет доступ к различным функциям. Я склонен начать с одного, *Управлять завитками*, чтобы уменьшить загромождение вариантов использования. Если писать варианты использования становится сложно, я выделяю одну часть, как описано в подразделе Создание из расширения нового варианта использования (раздел 8.3). Я прослеживаю доступ пользователя к системным данным и функциям с помощью отдельных рабочих таблиц. Все способы правомочны, и я не нашел достаточных оснований возводить в правило лишь один из них.

Ниже следует вариант использования, написанный Джоном Колацци и Аленом Максвеллом из Empower IT*. Они начали писать обоими способами, но затем решили объединить эти варианты использования в один, обобщенного уровня, под названием *Управление*. В конце концов они выделили подчиненный вариант использования Сохранить, чтобы упростить работу. Их варианты использования показывают, насколько они следовали собственному стилю в данном образце. Начав с шаблона Rational Unified Process, они пронумеровали шаги и расширения.

Вариант использования 32

Управление отчетами

1. Краткое описание

Этот вариант использования описывает операции создания, сохранения, удаления, печати, выхода и отображения отчетов и управляет ими. Этот вариант использования имеет очень низкий уровень точности и обращается к другим вариантам использования, чтобы отвечать своей цели (своим целям). Другие варианты использования содержатся в документах, перечисленных в разделе Специальные требования.

1.1. Действующие лица

Пользователь (основное).

Файловая система: обычная файловая система PC или сетевая файловая система с доступом на уровне пользователей (второстепенное).

1.2. Триггеры

Пользователь явно выбирает операции, используя интерфейс Explorer.

1.3. Поток событий

1.3.1. Основной поток — открыть, отредактировать, распечатать, сохранить и выйти из отчета

a. Пользователь выбирает Отчет, дважды щелкнув по слову Отчет в Explorer, и выбирает его открытие (открытие также запускается двойным щелчком по слову Отчет в Explorer).

b. Система выводит отчет на экран.

c. Пользователь задает формат отчета и пр. с помощью варианта использования Определить спецификации отчета. Система выводит измененный отчет.

d. Шаги c и d повторяются, пока пользователь не удовлетворится видом отчета.

e. Пользователь выходит из отчета с помощью варианта использования Выйти из отчета.

f. Пользователь может сохранить или распечатать отчет в любой момент после шага c, обратившись к варианту использования Сохранить отчет или к приведенному ниже альтернативному потоку Распечатать отчет.

---

* Использован с разрешения авторов.

1.3.2. Альтернативные потоки

1.3.2.1. Создать новый отчет

a. Пользователь выбирает в Explorer функцию "создать новый отчет", щелкнув правой кнопкой и выбрав соответствующую строку во всплывающем меню.

Система создает новый отчет с именем по умолчанию и устанавливает статус для имени "не установлено", а статус отчета — "изменяемый".

b. Поток варианта использования продолжается, соединившись с основным потоком на шаге b.

1.3.2.2. Удалить отчет

a. Пользователь выбирает отчет, щелкнув по слову Отчет в Explorer, и затем выбирает "удалить".

b. Система открывает отчет (или делает его текущим, если он уже открыт) и запрашивает у пользователя подтверждение на удаление.

c. Отчет закрывается и ресурсы освобождаются.

d. Система удаляет имя отчета из списка отчетов и данные по отчету с носителя информации.

1.3.2.3. Распечатать отчет

a. Пользователь выбирает Отчет, щелкнув в Explorer по слову Отчет, и указывает на "распечатать" ИЛИ пользователь выбирает возможность распечатки текущего отчета (отчет, редактируемый либо отображаемый в основном потоке этого варианта использования).

b. Пользователь выбирает принтер для печати и режимы печати, определенные для принтера (диалог печати и т.д., управляемый операционной системой) ИЛИ пользователь выбирает "распечатать отчет в файл ...".

c. Система загружает отчет и форматы. Система посылает задание на печать операционной системе или распечатывает отчет в указанный файл отчета. Система закрывает отчет.

1.3.2.4. Копировать отчет

a. Пользователь выбирает Отчет, щелкнув в Explorer по слову Отчет, и затем выбирает "копировать".

b. Система запрашивает новое имя отчета и подтверждает, что такого имени еще не было.

c. Система повторяет шаг b до тех пор, пока пользователь не введет правильное (несуществующее) имя, не решит заместить существующий отчет или окончательно прервать операцию копирования.

d. Система сохраняет отчет под указанным именем как новый отчет.

e. Если копия замещает существующий отчет, последний удаляется.

1.3.2.5. Переименовать отчет

a. Пользователь выбирает Отчет, щелкнув в Explorer по слову Отчет, и затем выбирает "переименовать".

b. Пользователь вводит новое имя, система подтверждает, что имя правильное (нестарое, несуществующее, с правильной орфографией и т.д.).

c. Система повторяет шаг b до тех пор, пока не будет принято правильное имя либо пользователь не прекратит операцию с помощью команды "отменить".

d. Система обновляет список отчетов, вводя новое имя для выбранного отчета.

1.3.3. Специальные требования

1.3.3.1. Платформа

Для управления операциями вывода отчета и других действий, связанных с интерфейсом пользователя, необходимо знать тип платформы.

1.3.4. Предусловия

Элемент данных существует в компьютере и выбран как текущий элемент.

1.3.5. Постусловия

1.3.5.1. Постусловие(я) успеха [примечание: это гарантия успеха].

Система ожидает взаимодействия с пользователем. Отчет может быть загружен и выведен, либо пользователь может выйти из отчета (закрыть отчет). Все изменения сохраняются по запросу пользователя; твердая копия выдается по запросу, и список отчетов надлежащим образом обновляется при необходимости.

1.3.5.2. Постусловие(я) неудачи [примечание: это минимальная гарантия].

Система ожидает действий пользователя. Ниже приведены возможные состояния:

Отчет может быть загружен

Список отчетов все еще правилен

1.3.6. Точки расширения

Отсутствуют

Вариант использования 33

Сохранить отчет

1. Краткое описание

Этот вариант использования описывает процесс сохранения отчета. Этот вариант использования вызывается из варианта использования Управление отчетами и из варианта использования Выйти из отчета.

1.1. Действующие лица

Пользователь (основное)

Файловая система: типичная файловая система PC или сетевая файловая система с доступом на уровне пользователей (второстепенное)

1.2. Триггеры

Чтобы вызвать этот вариант использования, пользователь выбирает операции как задачи в варианте использования Управление отчетами или вариант использования Выйти из отчета (который включен в вариант использования Управление отчетами).

1.3. Поток событий

1.3.1. Основной поток — Сохранить новый отчет

a. Вариант использования начинается, когда пользователь выбирает Сохранить отчет.

b. Система обнаруживает, что статус имени отчета — "не установлено", и запрашивает имя нового отчета. Пользователь выбирает имя отчета; система подтверждает, что это имя пока отсутствует в списке отчетов. Добавляет запись в список отчетов.

c. Пользователь прекращает операцию сохранения... Вариант использования завершается.

d. Система обновляет список отчетов, внося информацию об отчете. Система создает уникальное имя файла отчета, если оно еще не установлено, и сохраняет спецификации отчета в файловой системе.

e. Статус отчета получает значение "неизменяемый", а статус его имени — значение "установлено".

f. Вариант использования заканчивается выводом отчета на экран.

1.3.2. Альтернативные потоки

1.3.2.1. Альтернативный подчиненный поток — Имя отчета существует — заменить

a. Система обнаруживает имя в списке, запрашивает у пользователя указания заменить файл. Пользователь решает заменить.

Система использует имя файла существующего отчета и запись в списке отчетов. Вариант использования продолжается с шага d основного потока.

1.3.2.2. Альтернативный подчиненный поток — Имя отчета существует — отменить

a. Система обнаруживает имя в списке, запрашивает у пользователя указания заменить файл. Пользователь решает отменить действие. Вариант использования заканчивается выводом отчета на экран.

1.3.2.3. Альтернативный поток — Сохранить отчет как ...

a. Пользователь выбирает "сохранить отчет как ..."

b. Пользователь вводит имя нового отчета, система проверяет, есть ли данное имя в списке отчетов.

Имени в списке пока нет. Система находит имя в списке, запрашивает у пользователя указание заменить файл. Пользователь отвечает "НЕТ". Вариант использования продолжается с шага b. c. Вариант использования продолжается с шага d основного потока.

1.3.2.4. Альтернативный подчиненный поток — Имя отчета существует —заменить

a. Система находит имя в списке, запрашивает у пользователя указание заменить файл. Пользователь дает указание заменить. Система использует имя файла существующего отчета и запись списка отчетов. Вариант использования продолжается с шага d основного потока.

1.3.2.5. Альтернативный подчиненный поток — Имя отчета существует — отменить

a. Система обнаруживает имя в списке, запрашивает у пользователя указание заменить файл. Пользователь решает отменить действие. Вариант использования заканчивается выводом отчета на экран.

1.3.2.6. Альтернативный поток — Сохранить существующий отчет

a. Пользователь выбирает для текущего отчета функцию "сохранить отчет" (текущий отчет прежде был сохранен и его имя присутствует в списке отчетов).

b. Система находит запись в списке отчетов, обновляет при необходимости данные в списке, сохраняет спецификации отчета в файле отчета.

c. Статусу отчета присваивается значение "неизменяемый".

d. Вариант использования заканчивается выводом отчета на экран.

1.3.3. Специальные требования

Отсутствуют

1.3.4. Предусловия

Элемент данных существует в компьютере и выбран как "текущий элемент".

Отчет в настоящее время выведен на экран и установлен в качестве "текущего отчета".

Статус отчета — "изменяемый".

1.3.5. Постусловия

1.3.5.1. Постусловие(я) успеха [примечание: это гарантия успеха]. Система ожидает взаимодействия с пользователем. Отчет загружен и выведен на экран. Список отчетов обновлен (внесено имя отчета и т.д.) в соответствии с требованиями операции "сохранить". Статус отчета — "неизменяемый". Статус имени отчета — "установлено".

1.3.5.2. Постусловие(я) неудачи [примечание: это минимальная гарантия].

Система ожидает взаимодействия с пользователем.

Отчет загружен и выведен на экран.

Статус отчета — "изменяемый". Статус имени отчета тот же, что и при запуске варианта использования.

Список отчетов все еще правилен. (Список отчетов при необходимости приводится в порядок, когда сохранение заканчивается неудачей.)

1.3.6. Точки расширения

Отсутствуют

14.2. Параметризованные варианты использования

Время от времени мы сталкиваемся с созданием почти одинаковых вариантов использования. Наиболее распространенные примеры — *Найти клиента, Найти продукт, Найти данные о продвижении* и т.д. Скорее всего, только одна группа разработчиков будет создавать общий механизм поиска, а другие команды будут его использовать.

Написать полдюжины похожих вариантов использования в свободной форме не составляет большого труда. Однако писать шесть подобных вариантов использования по полной форме довольно трудоемкое и нудное занятие. Здесь необходим метод, экономящий усилия. Я опишу такой метод с помощью варианта использования 23.

Для этого варианта использования, прежде всего, характерно следующее:

- Именование цели в шаге очень похоже на вызов подпрограммы.
- Варианты использования будут читать люди, а не компьютеры.

При дальнейшем рассмотрении мы отметили, что при поиске предмета, чем бы он ни был, нужна одна и та же логика:

1. **Пользователь определяет предмет, который необходимо найти.**
2. **Система производит поиск, выдает список возможных вариантов.**
3. **Пользователь выбирает, возможно, повторно сортирует список и изменяет критерий поиска.**
4. **Система находит предмет (или не находит).**

При каждом использовании изменяются:

- Название предмета, который нужно найти
- Критерии поиска (поля поиска) предмета, который нужно найти
- Набор выводимых атрибутов предмета, который нужно найти (показываемые значения одно за другим)
- Критерий сортировки списка найденных вариантов

Мы создали параметризованный вариант использования, т.е. такой, который работает с кратким именем для каждого из перечисленных элементов. Мы решили назвать этот варианта использования *Найти что угодно*. Мы решили, что немного попрактиковавшись, читатель поймет, что фраза "Служащий находит клиента" означает вызов "Найти что угодно" для поиска клиента.

Мы сделали то же самое для всех кратких имен в параметризованном подчиненном варианте использования: полей поиска, выводимых значений и критериев сортировки. Мы только определили, где писатель будет указывать подробности.

Для значений данных мы определили три уровня точности. Первым была фраза из текста варианта использования — *краткое имя данных*, такое как *Профиль клиента*. Вторым были списки полей, связанных с кратким именем данных, именующим всю собранную под ним информацию, например имя, адрес, служебный телефон, домашний телефон и т.д. Третий уровень точности был точным определением поля: длина, критерии оценки и т.д.

Вариант использования содержал только первый уровень точности. Описания данных, критерии поиска и сортировки были помещены на отдельной странице, гиперссылка на которую содержалась внутри шага варианта использования.

Результатом стал шаг действия, выглядевший примерно так:

Служащий находит клиента с помощью параметров поиска клиента.

На странице *Параметры поиска клиента* определялись поля поиска, последовательность вывода полей и критерии сортировки. Читатель варианта использования щелкал по подчеркнутому выражению, чтобы увидеть содержимое страницы. Читатели, авторы и разработчики легко и быстро освоили этот механизм.

Вариант использования *Найти что угодно* теперь выглядит так:

1. **Пользователь определяет параметры поиска чего-либо.**
2. **Система находит все, что соответствует параметрам поиска, и выводит список их отображаемых значений.**
3. **Пользователь может пересортировать их согласно критерию сортировки.**
4. **Пользователь выбирает интересующий его вариант.**

Этот изящный прием избавляет вызывающие варианты использования от загромождения многочисленными деталями (более низкого уровня) поиска, сортировки, пересортировки и т.д. Общее поведение при поиске локализуется и записывается лишь один раз. Постоянство для каждого, кто ведет поиск, обеспечено, и те, кому необходимо знать подробности для целей программирования, могут найти их. Разработчикам очень понравилось, что им дали только одну спецификацию механизма поиска, поэтому они не интересовались, все ли виды поиска действительно одинаковы.

Довольно советов. Закончите упражнение 5.4. Обратите особое внимание на гарантии и расширения, потому что и те, и другие важны для вызывающих вариантов использования.

Глава 15

Моделирование бизнес-процессов

Все написанное в этой книге применимо и к бизнес-процессам, и к разработке программных систем. Любую систему, в том числе и бизнес-систему, которая предлагает набор услуг внешним действующим лицам, защищая в то же время интересы других участников, можно описать с помощью вариантов использования. Читабельные текстовые варианты использования особенно полезны в моделировании бизнес-процессов.

В этой книге вариантами использования для бизнеса являются следующие:

- Вариант использования 2.
- Вариант использования 5.
- Вариант использования 18.
- Вариант использования 19.
- Вариант использования 20.

15.1. Моделирование или проектирование

Утверждение "Мы применяем варианты использования для реинжиниринга бизнес-процессов" может иметь одно из значений:

"Мы используем их для документирования старого процесса, перед тем как перепроектировать его".

"Мы используем их для создания соответствующих проекту внешних требований к поведению".

"Мы используем их для документирования нового процесса после повторного проектирования".

Все это правильно. Вы должны решить, какое значение вы имеете в виду.

Рассуждая о вариантах использования, я предусмотрительно употребляю термин *моделирование* или *документирование бизнес-процессов* вместо реинжиниринга или проектирования. Вариант использования только документирует процесс. Он не модернизирует и не перепроектирует его. В процессе проектирования разработчики восходят к вершинам изобретательности. Варианты использования не сообщают им, как это делать. Каждый уровень документирования служит требованием к поведению, которое должен удовлетворить следующий уровень проектирования (на самом деле мы говорим, что данный проект удовлетворяет этим требованиям к поведению).

Внедрение новой технологии часто вносит изменения в бизнес-процессы. Вы можете заново их настроить в направлении от основного бизнеса к технологии, от нового процесса к технологии или от технологии непосредственно (одновременно порождая процесс). Работает каждый из этих методов.

Работа в направлении от основного бизнеса

Применяя этот метод работы (сверху вниз), вы начинаете с точного определения основного бизнеса (основного вида деятельности) организации, как описано в книге *Reengineering the Corporation* (Hammer, 1984). Когда вы сделаете это, вам будут известны:

Участники, заинтересованные в поведении организации

Внешние основные действующие лица, чьи цели, как вы предполагаете, удовлетворяет организация

События, на которые организация должна реагировать

Услуги, предоставляемые данным бизнесом, с положительными результатами для участников

Обратите внимание, что без единого слова о том, *как* ваша организация будет работать, вы теперь имеете информацию, которая устанавливает граничные условия ее поведения. Это также граничная информация для варианта использования: участники и интересы, основные действующие лица с их целями, гарантии успеха.

Контекст для проектирования бизнес-процесса можно документировать с помощью вариантов использования для бизнеса типа "черного ящика", рассматривая компанию или организацию как разрабатываемую систему (рис. 15.1).

На этом этапе вы по-новому группируете ваши ресурсы и создаете новые процессы для наилучшего использования текущей технологии. В настоящее время компьютерные системы служат для организации активной памяти и активных каналов связи в корпорации. В книге *Reengineering the Corporation* приводится много примеров того, как изобретательная деятельность приводит к различным бизнес-проектам и соответствующему эффекту. Итогом является новый проект корпорации или организации (рис. 15.2).

Далее результат переосмысления и перепроектирования процесса документируется с помощью вариантов использования типа "прозрачного ящика", показывающих людей и отделы (и, возможно, компьютеры), взаимодействующие, чтобы обеспечить внешне видимое поведение организации.

Полностью разработанные варианты использования типа "прозрачного ящика" должны демонстрировать обработку организацией всех ошибок и исключительных состояний, подобно тому, как это делает какой-либо набор вариантов использования или полная модель бизнес-процесса. В варианте использования вы можете записать (или не записать) название технологии в зависимости от вашей задачи.

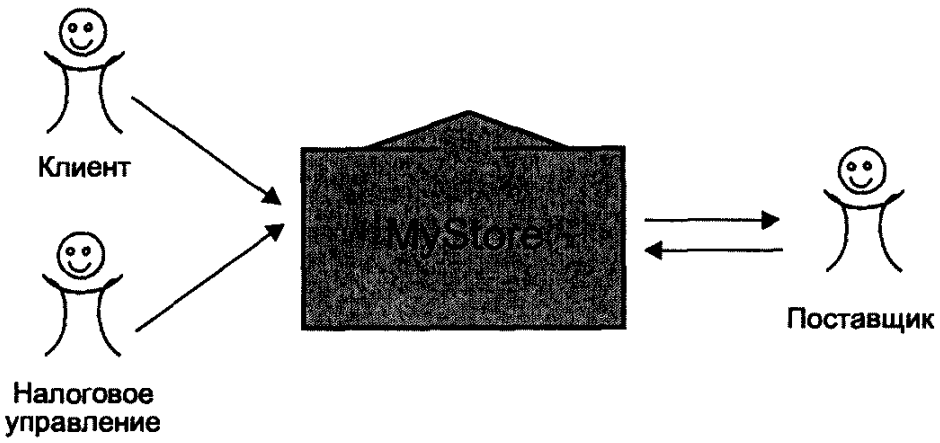


Рис. 15.1. *"Черный ящик" основного бизнеса*

Рис. 15.2. *Новый бизнес-проект как "прозрачный ящик"*

Работа в направлении от бизнес-процесса к технологии

С этой промежуточной стартовой точки вы не исследуете основную деятельность организации, а скорее определяете новый бизнес-процесс, чтобы приспособиться к новой технологии. Вы, вероятно, уже определили некоторую новую технологию, возможно, новый комплекс программных продуктов или мобильных устройств. Вам требуется поставить граничные условия для нововведений технологов.

Чтобы документировать новые предлагаемые бизнес-процессы, не упоминая новую технологию, вы должны писать варианты использования типа "прозрачного ящика" (рис. 15.3). Упоминание новой технологии в данной ситуации так же неуме-

стно, как описание интерфейса пользователя в варианте использования системы (см., например, вариант использования 21).

Рис. 15.3. *Новый бизнес-проект как "прозрачный ящик" (снова)*

В принципе компьютер в описании можно заменить корзинами бумаг, перевозимыми от одного индивидуума к другому. Задача вашей группы —понять, насколько активные каналы, например большие компьютеры или множество карманных компьютеров и компьютеров с радиосвязью, улучшают процесс.

Разработчики теперь знают, какой процесс должно поддерживать их изобретение. Они напишут системные варианты использования типа "черного ящика", чтобы показать соответствие новой системы технологически независимым вариантам использования для бизнеса типа "прозрачного ящика". Эти системные варианты использования становятся требованиями к разработке системы (см. рис. 15.4).

Хотя на бумаге все выглядит просто, это трудоемкий и дорогостоящий процесс. Технология развивается быстро и оказывает большое воздействие на бизнес. Часто не хватает времени работать по такой схеме. Много раз я обнаруживал, что деловые эксперты, являющиеся классными специалистами в своей области, способны мысленно построить модель нового бизнес-процесса, позволяя вам сэкономить время и деньги. Это дает вам возможность идти по третьему пути, описанному ниже.

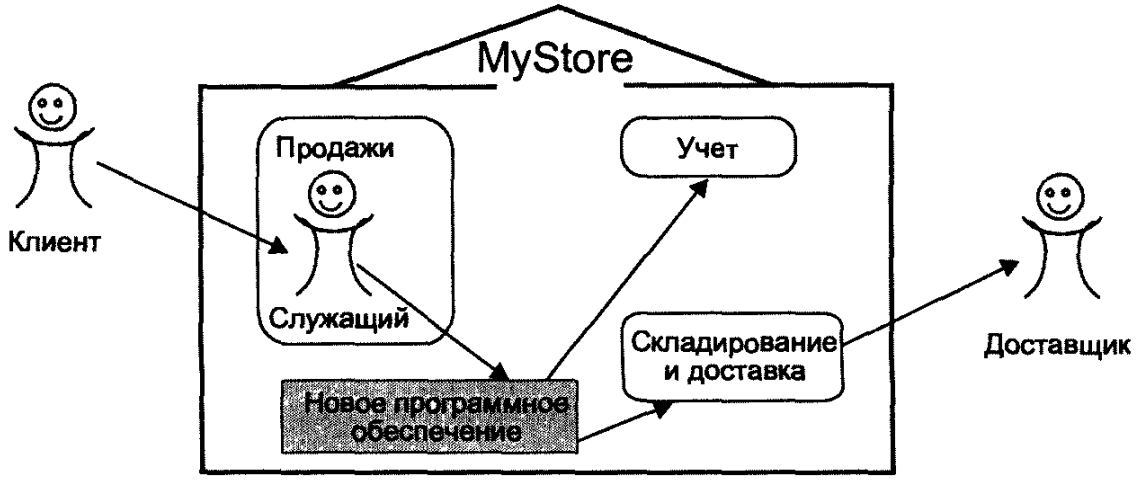


Рис. 15.4. *Новый бизнес-процесс в виде вариантов использования типа "черного ящика"*

Работа в направлении от технологии к бизнес-процессу

Во-первых, соберите несколько опытных экспертов, серьезно настроенных повысить качество технологии. Убедитесь, что у вас есть по два представителя каждой части бизнес-процесса, с которым ваша система будет иметь дело.

Во время переговоров эксперты по технологии назовут возможности системы, которые смогут улучшить процесс. Будьте готовы к тому, что они назовут больше, чем ваша группа разработчиков способна реализовать (это нормально, технологам необходимо мыслить с запасом).

Пусть эксперты напишут варианты использования типа "черного ящика" для системы, которую они себе представляют. В этих вариантах использования не будет упоминаться интерфейс пользователя. Они должны описать, что система сделает, чтобы максимально эффективно поддерживать основных действующих лиц в достижении их целей в бизнесе. Разделы расширения должны включать все виды критичных или редко обсуждаемых бизнес-правил. Экспертам, возможно, придется поговорить со своими коллегами, чтобы прояснить тонкие моменты поведения системы. В ходе работы они, конечно, создадут несколько вариантов использования обобщенного уровня и вариантов использования для бизнеса, показывающих контекст и связи целей во времени.

Другими словами, в результате будет создана документация бизнес-процесса в облегченной форме как часть требований к системе. Эксперты по применению смоделируют в уме новый бизнес-процесс, обсуждая в то же время, как следует себя вести действующим лицам и новой системе в различных обстоятельствах. Я считаю такой способ работы эффективным.

- Вариант использования 3 иллюстрирует, как в документации поведения системы можно завершить изложение фрагмента бизнес-процесса описанием исключительных состояний.
- Обобщенный (уровня воздушного змея) вариант использования 21 показывает контекст бизнес-процесса для системы.

15.2. Связывание вариантов использования для бизнеса и для системы

Вариант использования для бизнеса имеет такой же вид, как и системный вариант использования, поэтому все уроки по созданию и анализу вариантов использования применимы к обеим разновидностям. Постепенно развертывающаяся история начинается в вариантах использования для бизнеса и разворачивается в системные варианты использования. Это совместный эффект, который обеспечивают варианты использования для бизнеса и для системы.

Недостаток в том, что писатели и читатели могут случайно смешать их, привнося поведение системы в варианты использования для бизнеса, а бизнес-операции — в системные варианты использования. Это полезно, если они совершают это намерен-

но, но часто это делается неосознанно. Читатель, полагая, что имеет дело с системным вариантом использования, критикует вариант использования для бизнеса за слишком высокий уровень описания, не учитывая, что тот не предназначен для детализации поведения системы. Автор, работающий над вариантом использования для бизнеса, может случайно включить много подробностей поведения системы, в результате чего руководство быстро потеряет интерес к чтению этих перегруженных деталями документов.

Чтобы уменьшить путаницу, всегда записывайте область действия и уровень в поля шаблона варианта использования. Обучите своих авторов делать это, а читателей — начинать чтение с этих полей. По возможности применяйте пиктограммы. Употребляйте различающиеся шаблоны и нумеруйте варианты использования по-разному (в одной проектной группе нумерация вариантов использования для бизнеса начиналась с 1.000, а системных вариантов использования — с 1). Придумайте что-нибудь наглядное. Тогда вы получите преимущество совместного использования без путаницы в документах и потери нужных читателей.

Другой недостаток заключается в том, что редко имеет смысл полностью и надлежащим образом связывать варианты использования для бизнеса и для системы. Обычно те, кто создают варианты использования для бизнеса, описывают бизнес-процесс вплоть до применения системы, не включая последнее. Им не хватает времени, денег, сил и стимулов, чтобы написать, как используется новая система в повседневной жизни деловых людей. Авторы системных вариантов использования иногда добавляют для контекста одно-два предложения, описывающие бизнес-процесс, но у них отсутствуют побудительные мотивы переписывать варианты использования для бизнеса, чтобы включить новые функциональные возможности системы.

В результате возникает разрыв между вариантами использования для бизнеса и системными вариантами использования. Расти Уолтерс из FirePond комментирует это так:

> Я должен все же испытывать удовлетворение от полностью завершенных вариантов использования для бизнеса, которые развертываются в системные варианты использования. По моему опыту, весьма распространенной является ситуация, когда имеются три уровня вариантов использования для бизнеса. Несколько вариантов использования для бизнеса типа "черного ящика" уровня облака создаются вначале. Они быстро превращаются в варианты использования для бизнеса типа "прозрачного ящика" уровня облака, которые раскрываются, показывая тем самым, что они именуют варианты использования для бизнеса типа "прозрачного ящика" и уровня воздушного змея.
>
> Однако я не видел ясного соединения вариантов использования для бизнеса и системных вариантов использования.

Это не слишком хорошо для тех, кто ищет способ получить требования к системе из бизнес-процессов автоматически. Не думаю, что автоматический вывод требований возможен (см. раздел 15.1).

Некоторых это беспокоит. Меня нет. Большинство людей, с которыми я работал в организациях, способны совершить переход мысленно, чтобы связать вариант ис-

пользования для бизнеса самого низкого уровня с системным вариантами использования уровня воздушного змея или уровня моря. Кроме того, я должен быть уверен, что писать этот окончательный набор вариантов использования, который полностью привязывает варианты использования для бизнеса к системным вариантам использования, стоит усилий и денег, которые придется затратить. Эти две работы финансируются отдельно, и каждая соответствующим образом завершается, когда ее цель достигнута.

Пересмотрите варианты использования, начиная с варианта использования 19. Системные варианты использования действительно упоминаются в вариантах использования для бизнеса, но они были написаны специально, чтобы обеспечить контекст для группы разработчиков требований к системе, а не в качестве начала отдельной работы по реинжинирингу бизнес-процесса.

Далее Расти Уолтерс из FirePond рассказывает о своем опыте работы с вариантами использования для бизнеса.

▪ Расти Уолтерс:

Моделирование бизнес-процессов и требования к системе

Прочитав вашу книгу, я смог усовершенствовать проблемные области с помощью прежнего опыта и вновь обретенных знаний.

Мой предыдущий опыт

До прочтения этой книги я помогал документировать функциональные требования к нескольким приложениям комплекса программных продуктов.

Для одного приложения мы разработали системные варианты использования на обобщенном, пользовательском уровне, а также на уровне подфункции. Мы цепиком сконцентрировались на функциях системы и были удовлетворены окончательной моделью вариантов использования, так как она довольно легко читалась. Мы решили, что нет необходимости разрабатывать какие-либо варианты использования для бизнес-процесса, чтобы показать контекст. Нас вполне устраивали системные варианты использования обобщенного уровня.

Для другого приложения из комплекса ситуация отличалась, несмотря на то, что за эту модель вариантов использования и за предыдущую отвечала одна группа. Оглядываясь назад, я понимаю, что основной проблемой были сотрудники, рассматривающие эту задачу под разными углами зрения. Я работал в направлении от бизнес-процесса к технологии, а другие — от технологии к бизнес-процессу. Надо ли говорить, что область действия проектирования каждого варианта использования не была согласована между сторонниками двух направлений.

Сторонники направления от бизнес-процесса к технологии так никогда и не добрались до создания системных вариантов использования, а приверженцы направления от технологии к бизнес-процессу так и не взялись за написание вариантов использования для бизнеса. Вместо этого они перетасовывали варианты использования друг друга, "сталкивая их лбами". При этом

каждая группа пыталась непосредственно применять варианты использования другой группы. Поскольку разработчики не обладали в то время ни интуицией, ни должным пониманием, чтобы правильно пометить область действия и уровень вариантов использования, модель вариантов использования превратилась в настоящую мешанину. В конечном счете разработчикам никогда не везло с этой моделью вариантов использования. Они чувствовали, что с ней что-то не так, но точно не знали, в чем дело.

Опыт, полученный после прочтения книги

Работа в направлении от основного бизнеса, похоже, вызывает меньшую неразбериху. Я понял это в группе, чьей целью было понять и документировать бизнес-процессы.

Всем было ясно, что нас собрали обсудить бизнес-процессы и способы их работы в этой отрасли бизнеса, а не системы программного или аппаратного обеспечения. Возникшие неясности были связаны с выбором области действия проектирования: бизнес-процесс либо подразделение.

Мы начали с бизнеса, создавая весьма обобщенные (уровня облака) варианты использования типа "черного ящика". Это было понятно каждому, несмотря на то, что некоторые члены группы хотели погрузиться в детали более низкого уровня. Мы быстро продвигались в создании очень обобщенных (уровня облака) вариантов использования типа "прозрачного ящика". Когда мы подошли к вариантам использования следующего, более низкого уровня, сразу возникло сомнение по поводу области действия проектирования. Говорим мы о бизнесе либо об определенном подразделении? К тому же это тесно переплелось с вопросом о том, что составляет успех варианта использования. В одном случае мы закончили тем, что удалили два последних шага после того, как осознали, что эти шаги в действительности были в вызывающем варианте использования и не должны были находиться в текущем. В настоящее время у группы нет намерения когда бы то ни было преобразовывать варианты использования для бизнеса в варианты использования, определяющие требования к системе.

После совещания проблемные области стали понятными. Документируя результаты, я употреблял контекстные диаграммы области действия проектирования, пометив область действия и уровень каждого варианта использования пиктограммами. Простые и наглядные, они действительно влияют на читабельность вариантов использования и существенно помогают последним закрепиться в памяти читателей.

Глава 16

Пропущенные требования

Хорошо давать совет писать о намерении действующего лица, употребляя краткое имя для передаваемых даиных. Однако любому программисту ясно, что этой информации недостаточно для разработки. Программисту и разработчику интерфейса пользователя необходимо точно знать, что подразумевается под адресом, какие поля входят в адрес, длину этих полей и правила подтверждения для адреса, почтового индекса, номера телефона и т.д. Вся эта информация содержится где-то в требованиях, но не в вариантах использования.

Варианты использования — это только "Глава 3" документа о требованиях, требования к поведению. Они не содержат требований к функционированию, бизнес-правил, проекта интерфейса пользователя, описаний данных, описания поведения конечного автомата, приоритета и т.д.

"Где же тогда эти требования?" — спрашивают разработчики. Мы можем сказать, что варианты использования не обязаны их содержать, но где-нибудь они должны быть документированы.

Некоторые данные можно присоединить к каждому варианту использования в качестве связанной информации. Туда могут входить:

- Приоритет варианта использования
- Ожидаемая частота появления
- Потребности функционирования
- Дата сдачи
- Второстепенные действующие лица
- Бизнес-правила (возможно)
- Открытые вопросы

В разных проектах этот список регулируется в соответствии с потребностями. Во многих случаях эту информацию удобно хранить в электронной или простой таблице. Часто электронную таблицу используют в начале проекта для обзора информации варианта использования. В крайнем столбце слева записывают название варианта использования, а в других столбцах содержатся:

- Основное действующее лицо
- Триггер
- Приоритет реализации
- Оценка сложности
- Предполагаемая версия
- Требование к функционированию
- Статус завершенности
- Все, что еще необходимо

Здесь, однако, пропущен раздел, необходимый программистам почти так же, как требования к поведению: требования к данным, включая поверку полей.

16.1. Точность в требованиях к данным

Рекомендация дозировать усилия при достижении точности требований к данным остается в силе (см. 1.5). Я делю требования к данным на три уровня точности:

- Краткие имена данных
- Списки полей или описания данных
- Атрибуты и проверки полей

КРАТКИЕ ИМЕНА ДАННЫХ. Мы пишем, что служащий собирает информацию о клиенте или служащий записывает *имя клиента, его адрес и номер телефона*. Мы предполагаем расширить отдельные описания имени, адреса и номера телефона в каком-либо другом месте.

В варианте использования уместны краткие имена. Писать больше — значит замедлить темп сбора требований и сделать варианты использования намного длиннее, а также менее читабельными и менее устойчивыми (т.е. чувствительными к изменениям в требованиях к данным). К тому же весьма вероятно, что многие варианты использования будут обращаться к одним и тем же кратким именам данных.

В силу этих причин уберите подробности требований к данным из варианта использования и определите ссылку данного варианта использования на соответствующий *список полей*. Это можно сделать с помощью гиперссылки во многих инструментальных средствах или номера требования в средствах, поддерживающих перекрестные ссылки нумерованных элементов.

СПИСКИ ПОЛЕЙ. На некоторых этапах пишущим требования специалистам придется согласовывать значения каждого краткого имени данных. Состоит ли имя клиента из двух частей (имени и фамилии) или более? Что точно должно входить в адрес? В мире много различных форматов адресов. Здесь вполне уместно расширить описания данных до второго уровня точности. Это удобно делать параллельно с созданием вариантов использования или позднее, скажем, взаимодействуя с разработчиком интерфейса пользователя.

Существует много стратегий работы со вторым уровнем точности. Я назову две, а другие вы найдете в книгах *Software for Use* (Constantine and Lockwood, 1999) и *GUIs with Glue* (Hohman, на момент выхода данной книги находилась в печати). Впрочем, вы можете самостоятельно попрактиковаться в этой области.

- Первая стратегия заключается в том, чтобы работая в инструментальном средстве создавать отдельную запись для каждой единицы, имеющей краткое имя. Под именем клиента определите три поля: первое имя, второе имя, фамилию. Это все. Далее вы будете уточнять эту запись атрибутами и проверками полей до тех пор, пока она не будет содержать все атрибуты этих полей, как описано в пункте *"Атрибуты полей и проверки полей"*.

- Вторая стратегия основана на том факте, что вы записали имя, адрес и номер телефона в одном шаге варианта использования. Это означает, что для вас важно, чтобы эти три части данных вводились и выводились вместе. Это полезно для разработчика интерфейса пользователя, который может проектировать окно или расположение полей. Поэтому вы создаете одну запись в вашем инструментальном средстве для "имени, адреса и номера телефона". В ней вы перечисляете, какие поля необходимы для имени, какие для номера телефона и какие поля требуются для адреса. Далее этот список вы не расширяете.

Различие двух стратегий в том, что для второй вы записываете кластеры данных, имеющих краткие имена, на каждой странице списка полей. Если вам надо уточнить описание данных, вы не расширяете существующую запись, а создаете отдельную запись для каждого поля.

При любой стратегии данные второго уровня точности будут изменяться по мере продвижения проекта и по мере того, как разработчики расширяют свои знания о специфике данных. Кроме того, определять второй и третий уровни точности данных могут разные люди. Вероятно, удобнее хранить второй уровень точности данных отдельно от самого варианта использования.

АТРИБУТЫ ПОЛЕЙ И ПРОВЕРКИ ПОЛЕЙ. Программисты и разработчики базы данных обязательно должны решить, сколько знаков отводится на имя клиента и каковы ограничения на дату ущерба. Другими словами, типы полей, длину полей и проверки полей.

Некоторые группы разработчиков записывают эти сведения в главу документа о требованиях под названием "Требования к данным" или "Форматы внешних данных". Другие, использующие средство или базу данных с гиперссылками, помещают их в отдельные записи, классифицируемые как "Определения полей". Третьи зано-

сят атрибуты данных непосредственно в требования к интерфейсу пользователя и в документ проекта.

Что бы вы ни выбрали, имейте в виду:

- Вам необходимо расширить описание атрибутов и проверок полей до третьего уровня точности.
- Вариант использования — не место для такого расширения.
- Вариант использования должен ссылаться на это расширение.
- Атрибуты полей со временем могут изменяться независимо от деталей варианта использования.

16.2. Перекрестные ссылки между вариантами использования и другими требованиями

Форматы данных не являются частью варианта использования, но имена вариантов использования необходимы для данных, и поэтому мы можем установить гиперссылку между вариантом использования и описаниями данных. Сложные бизнес-правила не очень хорошо вписываются в повествовательный текст варианта использования, но и в этом случае мы можем употребить ссылку на содержащие их записи. Этот вид связей делает варианты использования центром документа о требованиях даже для многих нефункциональных требований (см. рис. 16.1).

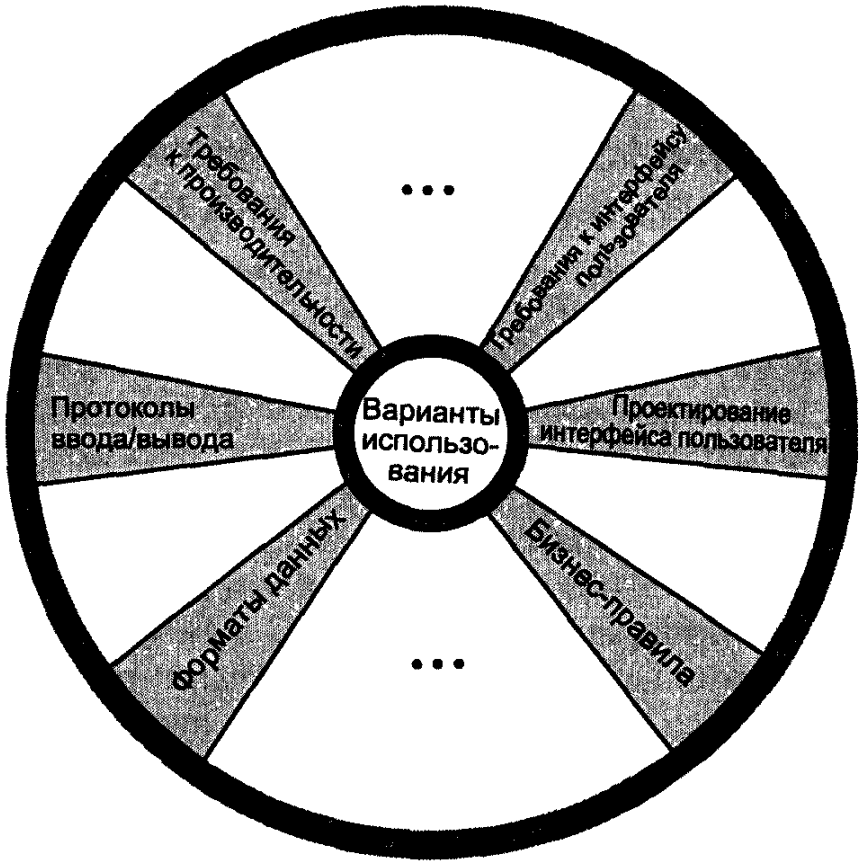


Рис. 16.1. *Модель требований в виде колеса (повторение рис. 1.1).*

Будьте осторожны и не помещайте в варианты использования те требования, которые не соответствуют формату вариантов использования. Последние лучше всего подходят для фиксации взаимодействия действующих лиц.

Временами жалуются, что тяжело описывать требования для операции объединения лент или компилятора с помощью вариантов использования. Всецело согласен. В таких случаях лучше всего применить алгебраическую или табличную форму.

Варианты использования годятся лишь для части набора требований. Получается, что часть, описывающая взаимодействия, находится в центре и связана со многими другими требованиями.

Глава 17

Роль вариантов использования в общем процессе

17.1. Варианты использования в организации проекта

Варианты использования дают руководству возможность контролировать реализацию каждой функции, предназначенной для пользователей. Название каждого варианта использования указывает на цель, которую основное действующее лицо найдет в числе целей, поддерживаемых системой. В теле каждого варианта использования содержатся сведения о том, что система требует и что предоставляет.

Используйте названия вариантов использования в качестве основы для организации проекта

На ранней стадии планирования проекта создайте таблицу с названиями вариантов использования и основных действующих лиц в двух левых столбцах. В третий столбец заказчики проекта занесут приоритет каждого варианта использования. В четвертую колонку поместите степень сложности реализации данной функции. Это естественное развитие списка Действующее лицо/Цель (см. раздел 3.1).

В книге *Extreme Programming Explained: Embrace Change* (1999) есть хорошая вариация на эту тему: разработчики оценивают стоимость разработки, в то время как заказчики решают, какой приоритет присвоить каждому варианту использования. Заказчики имеют возможность изменить приоритеты, посмотрев, как оценена работа, но не могут изменить оценки. В свете этой идеи вы можете заполнить столбцы таблицы планирования в два этапа.

Вы можете добавить дополнительные столбцы к таблице планирования, включив приоритет разработки варианта использования, предполагаемый номер версии или даже группу, которая будет его разрабатывать (см. таблицу 17.1).

Вы можете проанализировать эту таблицу и манипулировать ею в процессе проектирования.

Таблица 17.1. *Пример таблицы планирования*

Действующее лицо	Цель уровня задачи	Деловая потребность	Техническая сложность	Приоритет	N варианта использования
Любое	Проверить запросы	Высшая	Большая работа в общем случае	1	2
Уполномоченное лицо	Изменить полномочия	Высокая	Невысокая	2	3
Покупатель	Изменить контакты с поставщиком	Средняя	Невысокая	3	4
Клиент	Инициировать запрос	Наивысшая	Средняя	1	5
	Изменить запрос	Высшая	Невысокая	1	6
	Отменить запрос	Низкая	Невысокая	4	7
	Пометить запрос как выполненный	Низкая	Невысокая	4	8
	Отказаться от доставленных товаров	Низкая	Неизвестна	4	9
Утверждающее лицо	Завершить оформление запроса	Наивысшая	Высокая	2	10
Покупатель	Завершить запрос для заказа	Высшая	Высокая	1	11
	Инициировать почтовый перевод денег поставщику	Высшая	Высокая	1	12
	Сигнализировать о недоставке	Низкая	Средняя	4	13
Уполномоченное лицо	Подтвердить подпись утверждающего лица	Средняя	Высокая	3	14
Приемщик	Зарегистрировать доставку	Высшая	Средняя	1	15

Со временем вы завершите оценку для каждого варианта использования, распределите их по группам и будете отслеживать работу по каждой версии каждого варианта использования.

Далее приведена история о применении таблицы планирования для измерения, оценки, расстановки приоритетов и сокращения возможного рабочего набора вариантов использования. Преимущества такого способа работы:

- Список вариантов использования ясно показывает значимость системы для бизнеса.
- Список названий обеспечивает структуру для работы с учетом приоритета разработки и временных рамок.

Невымышленная история

Разработчик должен был решить, какие бизнес-процессы поддерживать в следующих версиях программного продукта. Он выдал список Действующее лицо/Цель на 7 страницах! Сметная стоимость, сложность, триггер и прочее. При участии руководителя проекта список был урезан до половины странички. Затем разработчик написал основной сценарий для этих бизнес-процессов и вместе с руководителем проекта сократил перечень шагов, чтобы проанализировать примерно полстраницы целей уровня системы. С этой информацией разработчик посетил филиалы компании. После этого у него сложилась ясная картина того, какие части бизнес-процессов наиболее разумно адресовать работникам филиалов. Разработчик и заказчик наметили подготовить в течение полугода четыре варианта использования.

Управление пересекающимися версиями вариантов использования

Было бы правильнее сказать, что конкретное множество законченных вариантов использования отображается в каждой версии.

Такой вариант использования, как *Заказать товар*, выявляет все виды специальной обработки, которая может быть предоставлена со временем. Типичная поэтапная стратегия:

- Реализовать простой случай в версии 1.
- Добавить управление рисками в версии 2.
- Реализовать предпочтительную для клиента обработку в версии 3.

Вы либо создаете один вариант использования и часть его закладываете в каждую версию, либо пишете три варианта использования. Оба способа будут работать, и оба имеют свои недостатки.

Некоторые группы разработчиков расщепляют варианты использования на единицы, которые целиком входят в версии. Они пишут вариант использования *Заказать товар*, затем *Заказать товар (управление рисками)* и *Заказать товар (дополнительные предпочтения клиента)*. Они или повторяют тело варианта использования, дописывая курсивом новые элементы, или создают второй вариант использования как расширение первого, а третий — как расширение второго. Расщепление вариантов использования упрощает отслеживание, однако число вариантов использования, подлежащих отслеживанию, утраивается, и создание дополнительных вариантов использования может усложниться.

Другие (я в том числе) предпочитают, чтобы варианты использования были максимально удобными для чтения и создавались с расчетом, что будут реализовываться частями. Порции, которые будут реализованы первыми, выделены желтым цветом или курсивом и называются выделенными порциями варианта использования 5. Этот способ я видел во многих проектах, и он неплохо работает. И все же он не так хорош, чтобы нас удовлетворить.

Возможен другой путь. Начните с создания полного варианта использоваиия, включающего несколько версий. Затем в начале некоторой итерации напишите его версию, включающую только те функции, реализовать которые вы планируете на этой итерации. На следующей итерации выделите части варианта использования, которые уже реализованы, обычным шрифтом, а новые части, предназначенные для следующей версии, — жирным шрифтом. При этой методике количество вариантов использования также увеличивается, но более контролируемым образом. Если в начале имеются 80 вариантов использования, разработчики подозревают, что придется управляться не только с иими, но и еще с каким-то числом вариантов использования на этой конкретной итерации. Множество вариантов использования расширяется, но этот процесс локализован и находится под контролем.

Вам придется выбрать методику работы, учитывая, что каждая имеет недостатки.

Имейте в виду, что все подходы предполагают, что организация использует поэтапную (итерационную) разработку и сдачу системы с этапами длительностью до четырех месяцев. Итерациониая разработка стала стандартной в современном проектировании программного обеспечения и подробно обсуждается в моей книге *Surviving Object Oriented Projects* (1998) и в статье VW Staging *(http://members.aol.com/acockburn/papers/vwstage.htm)*.

Реализация полных сценариев

Невымышленная история

Менеджер по тестированию и интеграции очень большого проекта расспрашивал меня об опасностях итерационной разработки. Оказалось, что группа разрабатывала и сдавала приложение, используя наборы функций (features), а не варианты использования. Его группа не могла тестировать эти куски, так как они не поддерживали никакого "сюжета", представляя собой кучу алгоритмов. В конце концов руководители проекта перестроили план так, чтобы результатом работы на каждом этапе была приемлемая сюжетная линия.

Часто случается, что не все варианты использования разрабатываются одновременно. Несмотря на это, каждый должен содержать полный сценарий. В варианте использования говорится, что хочет осуществить пользователь, и перечисляются многие необходимые для этого сцеиарии. Производимое вами программное обеспечение должно содержать некоторые из этих сценариев от начала до конца, или ваше приложение ие будет обеспечивать выполнение должной функции.

Плаиирование должно согласовываться с проектированием, чтобы создать набор функций, полезных для конечного пользователя. Это полные сценарии из вариантов использования. Специалисты по функциональному тестированию будут проверять совместимость варианта использования. Ввод в действие можно осуществлять только для полностью законченных сценариев вариантов использования.

Это кажется тривиальным, но если это проигнорировать, возможен ущерб, как показывает вышеприведенная история.

17.2. Варианты использования и списки задач или функций

Группы разработчиков, у которых налажено общение, формируют задачи проектирования на основе вариантов использования. Это хорошо, так как в противном случае у руководителей проекта будут трудности при отображении вариантов использования в задачи проектирования и актуализации последних.

Для иллюстрации приведу следующее электронное письмо от коллеги.

> В последние две недели двое из нас посещали клиента, чтобы получить от него требования и установить взаимодействие с разработчиками. Мы сфокусировались на вариантах использования и поняли, что их точность составляет 90-95%, а этого вполне достаточно для проектирования и реализации, поэтому мы были спокойны. Существовал документ относительно области действия, описывающий, какие функции или наборы функций были в ней или вне ее. Это похоже на приведенный вами пример области действия проектирования. Вначале этот документ был очень коротким, но мы удовлетворили запросы на "традиционное" описание требований. Поэтому пришлось его немного расширить, но мы постарались свести детали к минимуму.
>
> На первой встрече разработчики хотели пересмотреть "требования", которые были представлены для них в этом документе относительно области действия проектирования. Мы поняли, что в документе недостает иеобходимых подробностей, так как мы надеялись сконцентрироваться на вариантах использоваиия. Следующие три дня мы провели, разрабатывая пересмотренный документ, который вы увидите во вложеином файле. Это можно назвать "разжевыванием" вариантов использования. Надеясь повлиять на культуру разработки и помочь компании осуществить переход к требованиям на основе вариаитов использования, мы часто употребляли название каждого варианта использования в качестве набора функций, а каждый шаг сценария — как функцию. Мы буквально копировали шаги из вариантов использования и вставляли их в документ области действия под соответствующим заголовком варианта использования.
>
> Проблема, с которой мы столкнулись и которая в конце концов вынудила иас осуществить это удвоенное сопровождение, состояла в том, что точный текст сценариев в качестве четко выделенной функции не был вполне независимым. Даже если бы мы копировали текст сценариев, нам пришлось бы перефразировать его или окружать контекстом.

Таково начало. Разработчики зачастую хотят работать, исходя из нумерованных абзацев требований или списков функций. В других подобных историях сначала создавались "подробные требования" или документ с нумерованными абзацами. Кто-то тогда решил написать варианты использования, исходя из этих документов.

Это основное противоречие между поведением системы, описанным в вариантах использования, и задачами проектирования, которые ставятся перед отдельным разработчиком. Разработчик трудится над одним элементом строки или функцией, определенной в глубине одного варианта использования, или, возможно, проходящей через несколько вариантов использования. Это могут быть алгоритмы "отменить" (Undo), "регистрация транзакций в журнале" (Ttransaction Logging) или "структура экрана поиска" (Search Screen Framework). Разработчики не могут описывать свою работу на языке вариантов использования, поскольку они не имеют дела с поведением системы. В лучшем случае разработчик скажет, что он работает над частью варианта использования 5, посвященной структуре экрана поиска.

В то же время руководители проекта хотят видеть приложение, полностью реализующее поведение системы и удовлетворяющее потребности пользователей. Отдельные задачи проектирования сами по себе не представляют для них интереса.

Синхронизация документации вариантов использования и списка задач проектирования требует много времени и труда. Работу, описанную моим коллегой, приходится повторять в процессе проектирования много раз. Я считаю, этого следует избегать. До сих пор мне не приходилось разбивать варианты использования на отдельные строки в проектах, над которыми работали до 50 специалистов. Может быть, дело в том, что в своих проектах я уделял особое внимание личному общению, сотрудничеству и соответствующей культуре проектирования. Нам удавалось выделять элементы строк в уме или с помощью желтого маркера, а также записывать ключевые элементы в список задач для планирования без больших накладных расходов.

Альтернативой является формирование двух документов и упорный труд по их актуализации. Если вы решили следовать этой стратегии, разбейте текст варианта использования на фрагменты, каждый из которых можно поручить отдельному разработчику или отдельной группе. Каждый фрагмент становится реализованной функцией программы, алгоритмом или задачей проектирования, которая будет распределяться по разработчикам, отслеживаться и контролироваться. Подробная смета разработки программного обеспечения складывается из смет всех задач проектирования. Отслеживание состояния проекта состоит в том, чтобы отмечать начало и завершение осуществления каждой задачи проектирования.

Далее приведен пример преобразования варианта использования в список работ.

Вариант использования 34

Зафиксировать встречную продажу

Цель в контексте: у покупателя есть тележка с товарами и он хочет произвести встречную продажу, чтобы посмотреть, как это повлияет на стоимость.

Область действия: программная система торговли

Уровень: подфункция

Предусловия: тележка должна содержать товар (товары).

Гарантии успеха: встречная продажа оценена, добавлена к товарам в тележке, стоимость товаров в тележке сокращена.

Минимальная гарантия: если цель не достигнута, встречная продажа не фиксируется и не добавляется к содержимому тележки.

Основное действующее лицо: покупатель (любой пользователь Сети)

Триггер: покупатель решает зафиксировать встречную продажу.

Основной сценарий:

1. Покупатель решает зафиксировать встречную продажу.
2. Система оценивает встречную продажу, предоставляя информацию покупателю и задавая ряд вопросов, чтобы определить стоимость встречной продажи. Вопросы и представляемая информация зависят от ответов покупателя. Стратегия ответов и информации определена заранее, чтобы выделить практику выполнения относительно встречной продажи.
3. Система регистрирует навигационную информацию и информацию о встречной продаже в рабочем порядке.
4. Покупатель просматривает и анализирует сводку и стоимость встречной продажи.
5. Покупатель добавляет ее к тележке.
6. Система добавляет к тележке встречную продажу и навигационную информацию.
7. Система показывает обзор тележки со всеми ее товарами и встречной продажей, а также пересчет общей стоимости с учетом встречной продажи (продаж).

   Покупатель повторяет вышеприведенные шаги столько раз, сколько хочет зафиксировать и оценить встречных продаж, по желанию добавляя их к тележке.

Расширения:

2a. В любой момент, прежде чем добавить встречную продажу, покупатель может вернуться и изменить предыдущий ответ.

5a. Покупатель решает не добавлять встречную продажу к тележке; система сохраняет навигационную информацию на будущее.

5b. Покупатель хочет, чтобы встречная продажа применялась к определенной единице в тележке; покупатель указывает находящийся в тележке товар, к которому он желает применить встречную продажу.

Таблица 17.2 представляет сформированный список работ.

Таблица 17.2. *Список работ для фиксации встречной продажи*

Ссылка	Функция	Деловая потребность	Версия
EC10	Зафиксировать встречную продажу	Должна быть	1.0
EC10.1	Обеспечить покупателю возможность ввести встречную продажу в тележку.	Должна быть	1.0

Таблица 17.2 (продолжение)

Ссылка	Функция	Деловая потребность	Версия
EC10.2	Обеспечить возможность представлять сгенерированные (на основе шаблонов) формы интерфейса пользователя и пробираться через них, чтобы собрать информацию о встречной продаже для определения ее стоимости.	Должна быть	1.0
EC10.3	Обеспечить возможность обращаться к внешней системе встречных продаж (или сайту) для определения стоимости встречной продажи. Связанная с покупателем информация о встречной продаже передается на внешний сайт, который оценивает встречную продажу и возвращает ее стоимость и важные характеристики.	Должна быть	1.0
EC10.4	Обеспечить возможность представить покупателю сводку встречных продаж, включая их стоимость.	Должна быть	1.0
EC10.5	Обеспечить возможность покупателю добавлять или удалять встречную продажу. Добавив встречную продажу к товарам в тележке, покупатель может связать ее с отдельным продуктом или всеми продуктами в тележке.	Должна быть	1.0
EC10.6	Обеспечить возможность пересчитать общую стоимость содержимого тележки, учитывая встречную продажу (продажи).	Должна быть	1.0
EC10.7	Обеспечить возможность редактирования имеющейся встречной продажи, возвращаясь для редактирования к процессу вопрос/ответ.	Должна быть	1.0
EC10.8	Обеспечить возможность удалять существующую встречную продажу из тележки и пересчитывать общую стоимость.	Должна быть	1.0
EC10.9	Обеспечить возможность регистрировать любую информацию о встречной продаже или шаги на основе предварительно сконфигурированных триггеров.	Должна быть	1.0

17.3. Варианты использования и проектирование

Варианты использования обеспечивают все требования к поведению типа "черный ящик", какие только могут встретиться в проекте (и только их). Требования должны называть все, что система должна делать. При этом не должна подавляться

свобода разработчиков. Дело разработчиков — создать хороший проект, отвечающий этим требованиям. Предполагается, что соответствие между требованиями и проектом этим и ограничивается.

Переход от вариантов использования к проектированию имеет как хорошие, так и плохие стороны. Плохо то, что:

- В проекте отсутствует группирование по вариантам использования.
- Слепое следование структуре варианта использования ведет к проектам функциональной декомпозиции (это реально касается тех команд разработчиков, которые занимаются объектно-ориентированным и компонентным проектированием).

Хорошо то, что:

- Некоторые методы проектирования используют преимущества всех сценариев.
- Варианты использования выявляют понятия, необходимые при моделировании предметной области.

Рассмотрим сначала недостатки.

В ПРОЕКТЕ ОТСУТСТВУЕТ ГРУППИРОВАНИЕ ПО ВАРИАНТАМ ИСПОЛЬЗОВАНИЯ. Задачи проектирования не отображаются точно в единицы вариантов использования. Задача проектирования приводит в результате к бизнес-объекту или к механизму поведения, который будет применяться в нескольких вариантах использования. Вариант использования, который планируется выпустить в более поздний срок, вероятно, будет содержать важную информацию для задачи проектирования, выполненной ранее. Это означает, что когда к разработчикам поступит эта информация, им придется изменять проект в более поздних версиях.

Существуют три способа решить этот вопрос. Первый заключается в том, что разработчики просматривают все варианты использования на предмет ключевой информации для своих задач проектирования. Ясно, что это осуществимо только для небольших проектов. Если вы сможете с этим справиться, это окупится.

Второй подход состоит в сканировании всех вариантов использования, чтобы найти *рискованные элементы* (ключевые функции), которые, вероятно, должны оказывать сильное воздействие на процесс проектирования. Группа создает проект, чтобы реализовать эти ключевые функции, надеясь, что остальные функции не слишком нарушат проект.

Суть третьего способа, для меня предпочтительного, в том, чтобы осознать, что программное обеспечение будет изменяться в течение своего жизненного цикла, и успокоиться. Группа разрабатывает каждую версию настолько хорошо, насколько это имеет смысл. Разработчики понимают, что когда-нибудь могут всплыть новые требования, которые вызовут изменения.

Этот путь может не подойти некоторым разработчикам, особенно тем, кто занимался проектированием баз данных. Часто в этих средах добавление новых полей к таблице и повторная оптимизация базы данных обходятся дорого. Экономические соображения заставляют разработчиков вначале определять все атрибуты, к которым

когда-либо будет обращение. Они разрабатывают и выпускают программное обеспечение, которое будет затем называться на 20%, 40% и 100% завершенным относительно общего набора атрибутов.

Однако в наиболее современных средах с итерационной разработкой добавление атрибута к классу или таблице — рядовая операция, достаточно дешевая, чтобы разработчики определяли только те части класса или категории, которые нужны были немедленно. В результате классы и компоненты всегда "полны" только по отношению к заданному набору функций. По мере того, как реализуются новые функции, понятие "полноты" изменяется.

Невымышленная история

Эта мысль пришла в голову во время работы над одним проектом, когда руководитель группы пожаловался, что ему не позволили "завершить" классы после 20-процентной отметки готовности, хотя разработанное приложение выполняло все, что хотел пользователь! Нам потребовалось много времени, чтобы понять, что он говорил с точки зрения культуры проектирования баз данных, а работал над проектом, где применялся метод итерационной разработки.

Будьте готовы к подобной дискуссии. Если бы не исключительно жесткие экономические рамки, я бы предложил вашей группе работать в соответствии с моделью "полноты в отношении набора названных функций".

ВАРИАНТЫ ИСПОЛЬЗОВАНИЯ И ФУНКЦИОНАЛЬНАЯ ДЕКОМПОЗИЦИЯ. Если вы применяете методы структурной декомпозиции, будет полезна функциональная декомпозиция в вариантах использования. Однако если вы занимаетесь объектно-ориентированным проектированием, примите во внимание следующее.

Особые заметки для специалистов объектно-ориентированного проектирования

Наборы вариантов использования, составляющих функциональную декомпозицию или функциональную иерархию, демонстрировались как эффективное средство взаимодействия при составлении требований к поведению. Написанное легче воспринимать, и поведение аккуратно свертывается во все более высокие уровни. Это облегчает жизнь тем, кому приходится согласовывать задачи системы.

Такая функциональная декомпозиция годится для требований, но это не значит, что она подходит для проектирования программного обеспечения. Совместная инкапсуляция данных и поведения позитивно зарекомендовала себя, упрощая разработку и сопровождение программных проектов. Однако не очевидно, что она предоставляет хорошую структуру для сбора или обсуждения требований. Согласно моему опыту, она не так удачна, как структура на основе функций. Другими словами, сбор требований выигрывает от функциональной декомпозиции и вместе с тем проектирование этого программного обеспечения выигрывает от заключения данных и поведения в один компонент.

Разработчики должны прочитать варианты использования, подумать и обсудить их, а затем выдать полезные абстракции. Это их работа, а не пользователя.

Некоторый риск заключается в том, что неопытный или неосмотрительный разработчик создает классы, которые станут отражением функциональной декомпозиции документа о требованиях, просто отобрав каждый вариант использования как класс (объект, компонент). Практика показала, что это неверная стратегия, и многие эксперты по объектно-ориентированному проектированию явно предостерегают от ее использования.

Кто-то, наверное, может отстаивать включение варианта использования уровня цели пользователя в собственный класс, так как он охватывает полную транзакцию с четкой фиксацией и откатом. Он может быть подходящим кандидатом на инкапсуляцию. Однако варианты использования уровня подфункции редко имеют эти характеристики. Они обычно являются частичными алгоритмами, фрагментарно представленными в разных классах.

Противоположная опасность подстерегает специалистов по объектно-ориентированному проектированию, которым требуется моделировать предметную область непосредственно, не заботясь о функциях, которые необходимо поддерживать. Эти разработчики упускают из вида роль функциональных требований. Варианты использования указывают тому, кто моделирует предметную область, *какие аспекты представляют интерес для анализа*. Без этой информации потребуется слишком миого времени для моделирования частей предметной области, не существенных для данной системы. Варианты использования обеспечивают граничные условия надежного и полного моделирования (подробнее см. в статье An Open Letter to Newcomers to OO, *http://members.aol.com/humansandt/papers/oonewcomers.htm*).

Ясно, что во всех случаях варианты использования и объекты разделяют среду на различные организующие схемы. Имейте это в виду при проектировании.

Теперь перейдем к преимуществам.

ПРОЕКТЫ ПОЛЬЗУЮТСЯ СЦЕНАРИЯМИ. Варианты использования служат легкодоступными сценариями при проектировании программы. Они особенно полезны при проектировании на основе обязанностей* (*Responsibility-Driven Design*), когда проектирование осуществляется в соответствии с шагами сценария. Варианты использования также весьма полезны при других методах проектирования, показывая, когда проектирование завершено, и обрабатывая все ситуации (расширения).

ВАРИАНТЫ ИСПОЛЬЗОВАНИЯ ВЫЯВЛЯЮТ ПОНЯТИЯ ПРЕДМЕТНОЙ ОБЛАСТИ. Варианты использования довольно четко называют имена объектов предметной области. Рассмотрим фразу варианта использования:

---

* Прочтите статью K. Beck и W. Cunningham A Laboratory for Object-Oriented Thinking, ACM SIGPLAN, или книгу R. Wirfs-Brock, B. Wilkerson и L. Wiener *Designing Object-Oriented Software*. Познакомьтесь с информацией, находящейся по адресу *http://c2.com/ cgi/wiki?CrcCards* или *http://members.aol.com/humansandt/ papers/crc.htm*.

Система выдает *счет*, заполняет *поля строки счета* значением стоимости, добавляет налог и стоимость доставки и подсчитывает сумму. Нижний колонтитул *счета* формулирует условия доставки.

Не требуется большого воображения, чтобы представить счет, поле строки счета и нижний колонтитул счета с атрибутами стоимости, налога, доставки и суммы. Это не обязательно окончательные элементы проекта, но они являются хорошим начальным набором бизнес-объектов. Я видел, как проектные группы проделывали путь непосредственно от понятий в вариантах использования до набросков проекта. Такой метод делает проект сжатым и четким.

17.4. Варианты использования и проектирование интерфейса пользователя

В книгах *Software for Use* и *GUIs with Glue* вопросы проектирования интерфейса пользователя изложены Лучше, чем это могу сделать я. Однако при написании вариантов использования большинство проектных групп спрашивает, как осуществить переход от вариантов использования без интерфейса пользователя к действительному проектированию интерфейса пользователя.

Некоторые специалисты в состоянии изобрести приятный в использовании интерфейс. Эти люди прочитают варианты использования и придумают такое представление, которое придерживается шагов вариантов использования, сведя к минимуму усилия, требуемые от пользователя. Их проект интерфейса пользователя будет удовлетворять требованиям, заданным вариантами использования. С этой точки зрения проект будет рассмотрен пользователями и программистами.

Создатели вариантов использования показывают, что они вводят данные в форму для сбора данных или заполняют бумажную форму, чтобы было понятно, какая информация должна быть введена, и существуют ли ограничения на порядок ввода. Обеспечьте, чтобы эти формы соответствовали взглядам экспертов по использованию, а не интерпретировались как часть требований.

Хотя описание интерфейса пользователя не предназначено для документа о требованиях, полезно расширить документ примерами проектирования интерфейса пользователя. Такая информация по проектированию может улучшить восприятие документа о требованиях, так как дает и текстуальное (абстрактное), и визуальное (конкретное) описание поведения системы.

Проект интерфейса пользователя имеет три уровня точности (низкий, средний и высокий):

- Описание интерфейса пользователя *низкой точности* представляет собой экранно-навигационную схему, выполненную в виде конечного автомата или диаграммы состояний. Каждое состояние — это имя экрана, с которым будет иметь дело пользователь. Конечный автомат показывает, какие события, происходящие с пользователем, вызывают переход от одного экрана к другому.

- Описание интерфейса пользователя *средней точности* — это рисунок или уменьшенный снимок экрана. Поместите его в конце варианта использования, чтобы читатели могли и читать о предлагаемом проекте, и видеть его.

- Описание *высокой точности* перечисляет типы, длину и проверки всех полей для каждого экрана и никак не соответствует документу о требованиях.

17.5. Варианты использования и тестовые варианты

Варианты использования обеспечивают готовое описание функционального теста системы. Большинство тестировщиков приветствует работу с вариантами использования. Ведь они впервые получают что-то, с чем так легко работать. Более того, этот комплект тестов предоставляется им как раз во время сбора требований! Лучше всего, если они помогают писать варианты использования.

В формальной группе разработки команде тестирования приходится разбивать варианты использования на пронумерованные тесты и составлять план, определяющий отдельные параметры, которые будут обеспечивать различные пути выполнения программы. Затем они строят все тестовые примеры, которые устанавливают эти параметры и выполняются с этими установками. Кроме того, они используют все наборы данных, представляющие различные их комбинации, необходимые для тестирования, а также проектируют характеристики системы и тесты загрузки. Двум последним задачам варианты использования не содействуют.

Все это обычно касается команды тестирования. Таблицы 17.3 и 17.4 предоставлены Питом Мак-Брином (*http://www.mcbreen.ab.ca/papers/TestsFromUseCases.html*). Первым идет вариант использования, а затем — набор тестов приемки. Я оставляю этот пример вашей команде тестирования. Проанализируйте его для совершенствования навыков работы.

Обратите внимание, как Пит использует участников и интересы, чтобы построить тестовые примеры. Его тесты содержат специальные тестовые значения.

Вариант использования 35

Заказать товары, сформировать счет (пример тестирования)

Контекст: клиент размещает заказ на товары, счет формируется и отправляется вместе с заказанными товарами.

Минимальные гарантии:

В случае неудачи товары не будут выделены клиенту, учетная информация клиента останется неизменной и попытка транзакции будет записана в журнал.

Гарантии успеха:

Товары будут выделены клиенту.

Будет создан счет (применяется правило выписки счета клиенту).

Список выбора будет отослан для распределения.

Осноаной сценарий:

1. Клиент выбирает товары и указывает их количество.

2. Система выделяет для клиента требуемое количество каждого товара.
3. Система добывает заверенное разрешение на выписку счета.
4. Клиент указывает место доставки.
5. Система посылает инструкции по выбору товаров для распределения.

Расширения:

2a. Копичество товара на скпаде не достаточно для выполнения заказа:
   2a1. Клиент отменяет заказ.

2b. Товара нет на складе:
   2b1. Клиент отменяет заказ.

3a. Высокий кредитный риск у клиента (используйте тест приемки для этого исключительного состояния).

4a. Неправильное место доставки: ??

Для каждого расширения, приведенного выше, необходим по крайней мере один тестовый пример. Для полного тестирования нужно больше тестов для разных наборов значений данных. Тест основного сценария (таблицы 17.3 и 17.4) следует прогонять первым, поскольку полезно показать, как работает система в полном объеме. Часто его можно создать прежде, чем станут известны все условия расширения и пути восстановления.

Таблица 17.3. *Тесты основного сценария (низкий кредитный риск)*

Начальное состояние системы/входные данные	Клнент Пит, низкий кредитный риск, заказов — один, товар #1 по цене $10,00. Количество товара #1 в налнчин — 10.
Ожидаемое состояние системы/выходные данные	Количество товара #1 в налични — 9. Формируются инструкцни по доставке. Выдается счет для клиента Пита за один из товаров #1 по цене $10,00. Транзакцня регистрируется.

Таблица 17.4. *Тесты основного сценария (высокий кредитный риск)*

Начальное состояние системы/входные данные	Клиент Джо, высокий кредитный риск, заказов — однн, товар #1 по цене $10,00. Количество товара #1 в наличии — 10.
Ожидаемое состояние снстемы/выходные данные	Колнчество товара #1 в наличии — 9. Инструкции по доставке указывают доставку за наличные. Транзакция регистрируется.

17.6. Реальное создание вариантов использования

Вашей группе придется освоить определенные навыки работы. В следующем разделе показан процесс ветвления и соединения (branch-and-join), который является

моим излюбленным способом работы. Энди Краус из IBM описывает с удивительной ясностью опыт своей команды по координации работы большой разнородной группы пользователей. Весьма полезно почитать его отчет и поучиться.

Процесс ветвления и соединения

Два аспекта команда выполняет лучше, чем один человек: "мозговой штурм" и выработку соглашения (урегулирование). Однако когда группа разделяется, производится много документации. По этой причине выгоднее, если люди работают в полной группе при поиске соглашения или при "мозговом штурме", а остальное рабочее время — по двое или по трое. Ниже описан весь процесс.

1. Сформировать описание системной функции низкой точности:
   - Согласовать описание использования в свободной форме (группа).
   - Согласовать область действия и выявить путем "мозгового штурма" действующих лиц и цели (группа).
   - Разработать описания (раздельно).
   - Объединить описания (группа).
2. Сформировать представление высокой точности в виде вариантов использования:
   - Создать варианты использования методом "мозгового штурма" (группа).
   - Согласовать форму варианта использования (группа).
   - Написать варианты использования (раздельно).
   - Просмотреть варианты использования (раздельно).
   - Просмотреть варианты использования (группа).

Этап 1. Сформировать представление системной функции низкой точности

Этап 1 выполняется за четыре цикла.

ЦИКЛ 1.1. СОГЛАСОВАТЬ СТИЛЬ ОПИСАНИЯ ИСПОЛЬЗОВАНИЯ (ГРУППА). Члены команды вместе изучают описание (см. раздел 1.6). Каждый член команды пишет один документ, возможно, все документы на одну тему. Затем группа читает и обсуждает написанное, чтобы выработать общий взгляд на то, как должно выглядеть приличное описание, каков его объем и какие подробности оно должно содержать. Это может занять несколько часов. В конце цикла команда получает конкретное представление о том, что проектируется (или его части).

ЦИКЛ 1.2. СОГЛАСОВАТЬ ОБЛАСТЬ ДЕЙСТВИЯ И ВЫЯВИТЬ ПУТЕМ "МОЗГОВОГО ШТУРМА" ДЕЙСТВУЮЩИХ ЛИЦ И ЦЕЛИ (ГРУППА). Группа тратит столько времени, сколько нужно, для выявления общей цели, области действия и основных действующих лиц

системы. Сотрудники составляют концептуальное изложение, список ввода/вывода, диаграмму области действия проектирования, список основных действующих лиц и участников, а также список важнейших начальных наборов целей пользователей. Каждый из этих элементов касается других, поэтому обсуждение одного влияет на восприятие других. Таким образом, все элементы создаются одновременно. Если группа знает, что она собирается проектировать, это может занять от нескольких часов до одного дня. Если пока не знает, может потребоваться несколько дней. В итоге достигается согласие относительно предмета обсуждения, того, что надо создавать и состава основных действующих лиц.

ЦИКЛ 1.3. РАЗРАБОТАТЬ ОПИСАНИЯ (РАЗДЕЛЬНО). Группа разделяется, чтобы создать описания использования для выбранных функций разрабатываемой системы. Члены группы работают индивидуально и потом обмениваются документами. Затем они предоставляют результаты на суд группы в полном составе.

ЦИКЛ 1.4. ОБЪЕДИНИТЬ ОПИСАНИЯ (ГРУППА). Команда собирается для обсуждения содержания (не стиля) описаний. Задается вопрос: "Это действительно пример того, что мы хотим построить?" Может состояться еще одна дискуссия о природе системы и пройти еще один цикл создания описания, пока члены группы не решат, что описание отражает их взгляды на систему.

В этот момент первая фаза работы завершается. Группа располагает пакетом, который можно передать заказчикам. В этом пакете содержится эскиз (низкой точности) новой системы:

- Концепция системы
- Список того, что лежит в функциональной области действия, области действия проектирования и вне их
- Диаграмма системы в ее окружении
- Список ключевых основных действующих лиц
- Список участников системы и их основных интересов
- Список наиболее важных целей пользователей
- Набор описаний (каждое менее, чем в полстраницы)

Этап 2. Сформировать представление высокой точности в виде вариантов использования

Этап 2 выполняется за пять циклов.

ЦИКЛ 2.1. "МОЗГОВОЙ ШТУРМ" С ЦЕЛЬЮ СОЗДАНИЯ ВАРИАНТОВ ИСПОЛЬЗОВАНИЯ (ГРУППА). В первом цикле создается более точный список вариантов использования, которые следует написать. Группа использует облегченные методы анализа, "мозгового штурма" и выявления всех основных действующих лиц, которые могут встретиться системе на протяжении ее жизненного цикла. Затем группа методом "мозгового штурма" создает список всех целей пользователей, которые только

можно вообразить, для всех основных действующих лиц. Для этой работы группа может разделиться на подгруппы.

Полезная практика для больших разнородных групп — разделиться на рабочие группы по 3-5 человек. Обычно существует несколько предметных областей или групп по интересам, знания которых необходимо объединить, чтобы рабочие группы имели по одному специалисту в каждой области. Каждая группа обладает всеми необходимыми знаниями для решения задач, и каждый специалист имеет возможность высказаться. Небольшая группа способна продвигаться быстрее, чем крупная. Разделение основной группы на несколько маленьких групп означает, что за одно и тоже время охватывается большая область.

Если полный список основных действующих лиц и целей пользователей строится с помощью подгрупп, последние собираются снова для объединения результатов. Они производят групповой анализ, чтобы закончить и принять список. В конце периода группа имеет предварительный полный перечень вариантов использования уровня цели пользователя, которые надо написать. Со временем они почти наверняка обнаружат новые цели пользователей.

Группа выпускает список основных действующих лиц и целей пользователей. В это время может состояться дополнительное обсуждение приоритетов разработки, оценок сложности, времени разработки и т.д.

ЦИКЛ 2.2. СОГЛАСОВАТЬ ФОРМУ ВАРИАНТОВ ИСПОЛЬЗОВАНИЯ (ГРУППА). Этот цикл начинается с написания группой варианта использования. Можно писать индивидуально, а потом сдать личные версии в группу. Члены группы обсуждают уровень и стиль изложения, шаблон, участников и интересы, минимальные гарантии и пр. По окончании сеанса группа имеет начальную версию стандарта вариантов использования.

ЦИКЛ 2.3. НАПИСАТЬ ВАРИАНТЫ ИСПОЛЬЗОВАНИЯ (РАЗДЕЛЬНО). В этом цикле группа реорганизуется в подгруппы по специализации, возможно, по 2-4 человека на подгруппу. Для каждой специализированной подгруппы отбираются варианты использования.

В течение нескольких дней или недель подгруппы пишут варианты использования, индивидуально или в паре (я считаю, что большее количество партнеров по созданию варианта использования неэффективно). Варианты использования циркулируют внутри подгруппы для внесения комментариев и улучшений до тех пор, пока вариант не будет признан правильным. Затем они создают обобщенные варианты использования. Они почти наверняка расщепляют некоторые варианты использования, создают варианты использования уровня подфункции, добавляют основных действующих лиц, новые цели и т.д.

Удобно иметь двух людей, связанных с каждым вариантом использования, даже если один назначен главным автором. Будет возникать много вопросов о бизнес-правилах, т.е. что является реальным требованием, а что — лишь пережитком прежних дней. Хорошо, если есть человек, которого можно спросить о бизнес-правилах. Второй специалист может осуществлять двойную проверку, чтобы убедиться, что детали графического интерфейса пользователя не просочились в описание и что цели находятся на правильных уровнях.

ЦИКЛ 2.4. АНАЛИЗ ВАРИАНТОВ ИСПОЛЬЗОВАНИЯ (РАЗДЕЛЬНО). Писатели пускают по кругу черновики, электронные или написанные на бумаге. Интересно, что бумажный вариант имеет своеобразное преимущество: можно собрать комментарии каждого читателя, и автор за один редакторский проход варианта использования учтет все предложения. В одной группе рассказывали, что когда они пытались работать с замечаниями в режиме реального времени, им пришлось сделать намного больше проходов — сначала редактируя текст, чтобы реализовать предложение одного лица, затем аннулируя это изменение, чтобы учесть предложение другого. В любом случае, экспертная группа должна проверить уровень цели вариантов использования и бизнес-правила внутри них.

Авторы посылают варианты использования для просмотра и разработчику системы, и эксперту по ее использованию. Технический специалист удостоверяется, что вариант использования содержит достаточно подробностей для реализации (исключая описания данных и проект интерфейса пользователя). Эксперт по использованию убеждается, что все требования являются истинными и что бизнес-процесс действительно работает при таком поведении системы.

ЦИКЛ 2.5. АНАЛИЗ ВАРИАНТОВ ИСПОЛЬЗОВАНИЯ (ГРУППА). Наконец, проводится групповой анализ, на котором присутствуют разработчики программного обеспечения, эксперты по бизнесу, эксперты по использованию и разработчики интерфейса пользователя. То, что происходит после того, как создатели вариантов использования закончили свой лучший черновик, зависит от проекта, его политики и механизма анализа. Писатели должны быть уверены, что шаги понятны, правильны и достаточно детализированы для реализации. Это можно сделать с помощью официальных и неофициальных просмотров, просмотров пользователей или просмотров разработчика.

Вариант использования достигает своей первой официальной базовой формы, когда черновик прошел пользовательскую и техническую проверку. Теперь начинайте разработку и изменяйте вариант использования только для исправления ошибок, а не для редактирования формулировок.

Вы быстро почувствуете разницу между созданием черновика варианта использования и его завершением. Заканчивая вариант использования, писатели должны:

- Назвать все условия расширения.
- Продумать политику, связанную с обработкой ошибок.
- Удостовериться, что интересы участников соблюдены.
- Удостовериться, что все названия вариантов использования являются только действительными требованиями.
- Обеспечить, чтобы каждый вариант использования был читабельным для пользователей или экспертов по использованию и достаточно ясным, чтобы разработчики знали, что реализовывать.

Время на разработку одного варианта использования

Я считаю, что на черновик варианта использования требуется несколько часов, и несколько дней уйдет на продумывание обработки расширений. Одна группа из десяти человек выдала 140 кратких описаний вариантов использования за неделю (2,8 кратких описаний вариантов использования на человека в день) и затем потратила еще четыре недели, работая над ними и добавляя новые требования. Это увеличило время на один вариант использования и связанные с ним расширения до двух рабочих недель. Группа, работающая над разными проектами, затрачивала на один вариант использования в среднем 3-5 рабочих недель. Окончательные варианты использования были исключительно высокого качества.

Как создать варианты использования, работая с большими группами

Иногда приходится работать с большой разнородной группой экспертов, не включающей технических специалистов. Это очень непросто. Ниже я привожу выдержки из отчета, который написал Энди Краус о своем успешном опыте сеансов, облегчающих создание вариантов использования. Было задействовано 25 различных специалистов. Этот отчет был напечатан в *Object Magazine**.

▪ Энди Краус

Как создать варианты использования, работая с большой разнородной непрофессиональной группой

Не экономьте на оборудовании для проведения конференций. Вы будете проводить на конференциях недели, и вам необходимо владеть оборудованием во время сеансов... Мы столкнулись со значительными проблемами материально-технического обеспечения, переходя во время конференции из одной комнаты в другую.

Вы не сможете создать правильные варианты использования при отсутствии нужных для этого людей. Пусть лучше будет избыток людей и точек зрения, чем их недостаток... Люди, в чьих идеях и опыте мы нуждались, были аналитиками, служащими, детективами, операторами ввода данных и управляющими из 23 отделов. Заранее не зная действующих лиц реальной системы, невозможно было предсказать, какие пользователи могли понадобиться на определенном сеансе, — настоящая проблема, когда пытаешься собрать на какое-то время столько людей. Проблему мы решили, устроив встречу "Вариант использования — вбрасывание" с представителями всех пользовательских сфер, на которой мы

---

* Из Kraus, A., Use case Blue, Object Magazine, May 1996 (pp. 63-65). SIGS Publications, New York. Перепечатка с разрешения.

сообща определили предварительный график будущих сеансов.

Большим группам содействовать труднее, чем маленьким. Вам придется делать все, на что вы способны. В первую очередь будьте организованными. Так как процесс развивался, мы вынуждены были проводить сеансы с группами от 8 до 25 человек. Только с помощью представителей различных специальностей мы избавили требования от нюансов, обусловленных различными подходами участников конференции.

Не проводите на сеансе с пользователями более половины рабочего дня. Наши сеансы показали, что мы неправильно оценивали объем материала, который можно было охватить на сеансах, и (или) нашу способность обработать этот материал перед сеансами и после них. Обрабатывать исходный материал, добытый на сеансах, вам придется так же долго, как и вытягивать его. Вам придется выполнять множество рутинных организационных операций, административной работы, работы по планированию и подготовке к следующему сеансу во второй половине дня.

Плывите в одной лодке с руководством. Административное лицо вместе с руководителем проекта присутствовало на различных стадиях процесса сбора информации.

Ответственные за архитектуру системы лица должны присутствовать во время работы над вариантом использования. Архитектор участвует в процессе разработки как эксперт... Эксперт по предмету обсуждения обеспечивает экспертизу предметной области, чтобы быстро начать процесс и поддерживать его продвижение.

Присутствие на сеансах специалистов, поддерживающих приложение, которое должно быть заменено, поможет вам достигнуть цели. В сеансах принимали участие представители организаций и департамента информационных услуг. Они разъясняли прежние требования относительно внешних интерфейсов и знакомили с историей данной системы.

Никто не заменит подлинных пользователей, вовлеченных в процесс создания варианта использования. Мы были в состоянии обеспечить участие в сеансах настоящих служащих, аналитиков, исследователей, операторов ввода данных и руководителей этих групп работников.

Нужен секретарь. Важность быстрой аккуратной записи трудно переоценить... У нас было несколько секретарей, работавших на ранних сеансах и доказавших свою бесценность...

Показывайте все наработанные варианты использования, чтобы ускорить работу. Варианты использования разрабатывались в режиме взаимодействия и записывались организатором на лекционном плакате и секретарем в текстовом редакторе. Лекционные плакаты затем развешивались на стенах комнаты, где проходила конференция. Хотя иметь дело с громоздкими плакатами было неудобно, мы открыли в них несколько неожиданных достоинств. Мы развешивали варианты использования не в порядке очередности. Как только появлялся новый вариант использования, участники были вынуждены из-за неправильной последовательности просматривать вариант использования, разработанный ранее. Такая итерация, похоже, помогала присутствующим лучше познакомиться с существующими вариантами использования и быстрее разработать новые, которые

были подобны другим вариантам использования.

Название работы не может быть действующим лицом. Нашим пользователям было крайне трудно воспринять идею, что роль действующего лица может отличаться от названия работы. В конце дискуссии мы составили пробный список действующих лиц, но присутствующих он не удовлетворил. Как мы могли помочь им освоиться с ним?

Приложению все равно, кто вы такой; важно, какую шляпу вы носите. Поняв, что исполнение роли действующего лица в компьютерной системе похоже на "ношение шляпы", мы купили бейсбольные кепки и на каждой вышили имя действующего лица. На следующем сеансе кепки были выложены в ряд на столе организатора. Как только началась работа над вариантом использования, организатор взял кепку с надписью "Задающий вопросы" и надел ее на голову. Результаты были весьма благоприятны. Пользователи поняли, что не имеет значения, как называется их работа. Когда они используют систему, они носят определенную "шляпу" (играют определенную роль).

Вы можете пропустить некоторых действующих лиц в первоначальном списке. Прошло уже нескольких недель работы над вариантами использования, когда мы (организаторы) осознали, что варианты использования, связанные с контролем за пользователями системы, оказались пропущенными. Несмотря на все наши усилия обеспечить широкий диапазон участников, в сеансах не было никого из специалистов по контролю. Участники проекта не имели представления ни о каких контролирующих вариантах использования.

"Вариант использования для повседневной работы" [другими словами, описание использования] способствует успешному "мозговому штурму" при создании вариантов использования... Нашим пользователям недоставало контекста для вариантов использования, поэтому мы попросили их описать (на очень высоком уровне) шаги, которым они следуют, выполняя повседневные задачи, например, останавливая процесс. В какой-либо точке описания этого шага должна использоваться система, поэтому пользователи могут не обдумывать ее применение. Разработка вариантов использования повседневной деятельности на четвертый день принесла урожай в 20 вариантов использования.

Не надейтесь получать варианты использования в больших количествах. Как любая творческая деятельность, процесс создания вариантов использования имеет подъемы и спады. Попытки торопить творческий процесс не приносят успеха.

Будьте готовы к пробуксовке; используйте ускорители процесса. Всплывающие подсказки и справочные тексты и (или) вопросительные знаки, относящиеся к применению разрабатываемой системы, оказались прекрасными ускорителями. Мы иногда умышленно вводили спорные темы и точки зрения в качестве возбудителей дискуссии. Мы обнаружили, что когда люди сталкиваются с точкой зрения, которую не могут разделить, они способны выражать собственное мнение быстрее и яснее.

Создание вариантов использования — это общественная деятельность. Чувства были задеты, идеи разгромлены, и участники сплотились

против организатора, только чтобы защитить его позднее на том же сеансе. Несколько часов общения по окончании дискуссии сплачивали нас. В конце концов, все мы были связаны, так как собрались, чтобы узнать мнение друг друга и оказать друг другу поддержку в решении общих задач.

Стандартные "дескрипторы" облегчают процесс... Стандартные дескрипторы поддерживают атрибуты новой системы, разделенные на несколько направлений. Такими направлениями являются, например, люди, должности, местонахождения. Наборы дескрипторов обеспечили путь к непротиворечивому представлению информации... Наборы были поименованы и каталогизированы, чтобы их можно было использовать на сеансах обсуждения, а также при совершенствовании последующих вариантов использования. Стандартные обязанности системы и сценарии успеха и неудачи позволили нам сконцентрироваться на исключительных состояниях.

Стройте, сопровождайте и показывайте список предположений и допущений. Во время определенных периодов работы мы считали необходимым начинать сеансы с прочтения допущений во избежание споров об уже рассмотренных моментах.

Будьте минималистом. Сохраняйте свой шаблон варианта использования в минимальном объеме.

Глава 18

Краткие описания вариантов использования и экстремальное программирование (Extreme Programming, XP)

Экстремальное программирование (XP), использует еще более облегченную форму описания требований к поведению системы, чем методология, о которой идет речь в этой книге (см. *Extreme Programming Explained*, Beck, 1999). Согласно одному из положений XP, бизнес-эксперты и эскперты по использованию действуют с разработчиками в одной команде. Поскольку они рядом, группа пишет не подробные требования к программному обеспечению, а пользовательские рассказы (user stories), своего рода обязательство для будущего обсуждения требований части функциональных возможностей.

Пользовательский рассказ XP при своей сжатости может выглядеть либо как *краткое описание* варианта использования (см. таблицу 3.3), либо как функциональная возможность системы (см. таблицу 17.2).

Каждый пользовательский рассказ XP необходимо существенно детализировать и для бизнес-экспертов, и для технических специалистов, чтобы они поняли его смысл и оценили объем. Он должен быть небольшим произведением, чтобы разработчики могли осуществить его проектирование, кодирование и тестирование, а также подготовить его к использованию не более чем за три недели. Поскольку он удовлетворяет этим критериям, он может быть настолько сжатым и произвольным, насколько это устраивает группу. Его часто записывают на учетной карточке.

Когда начинается работа над пользовательским рассказом, разработчик просто передает карточку бизнес-эксперту и просит разъяснений, чтобы составить себе ясную картину функциональных возможностей.

В редких случаях небольшая команда разработчиков с пользователями, которые пребывают в группе в течение полного рабочего дня, применяет в качестве требова-

ний описания использования или краткие описания вариантов использования. Это возможно, только если владельцы требований находятся рядом с разработчиками системы. Разработчики сотрудничают непосредственно с владельцами требований в период проектирования системы. Как и в случае с пользовательскими рассказами XP, это осуществимо, если удовлетворяются условия завершения обязательства. Большинство проектов этим условиям не соответствует, и поэтому лучше всего рассматривать описание использования как разминку в начале сеанса создания варианта использования, а краткое описание варианта использования — как часть общего описания проекта.

Глава 19

Распространенные ошибки

Наиболее распространенные ошибки при создании вариантов использования состоят в пропусках подлежащих в предложениях, попытках проектирования интерфейса пользователя и в слишком низком уровне используемых целей. Здесь приведены некоторые примеры этих ошибок. Цель этой главы не протестировать вас, а дать возможность потренироваться.

В основном примеры короткие, а последний — длинный. Он взят из реального проекта. Попрактикуйтесь сначала на небольших примерах.

19.1. Отсутствует система

Первоначальный вариант

Вариант использования:

Снять наличные со счета

Область действия: банкомат
Уровень: цель пользователя
Основное действующее лицо: клиент

1. Клиент вводит карточку и PIN.
2. Клиент вводит "снять деньги" и количество.
3. Клиент забирает наличные, карточку и квитанцию.
4. Клиент уходит.

Замечания

Этот вариант использования показывает все, что делает основное действующее лицо, но не отражает поведения системы. Просто удивительно, насколько часто люди пишут такого рода варианты использования, создавая у читателя впечатление, что системе на самом деле не пришлось ничего делать.

Исправление состоит в указании всех действующих лиц и их действий.

Исправленный вариант

Теперь вы сможете написать вариант использования для банкомата, даже если вас разбудить среди ночи.

Вариант использования:

Снять наличные со счета

Область действия: банкомат

Уровень: цель пользователя

Основное действующее лицо: владелец счета

1. Клиент пропускает карточку для банкомата через считывающее устройство.
2. Банкомат считывает с карточки ID банка, номер счета, зашифрованный PIN, подтверждает ID банка и номер счета с помощью главной банковской системы.
3. Клиент вводит PIN. Банкомат сверяет его с зашифрованным PIN, считанным с карточки.
4. Клиент выбирает "быстрое снятие наличности" (Fast Cash) и указывает сумму в купюрах по $5.
5. Банкомат уведомляет главную банковскую систему о номере счета клиента, требуемой сумме и получает в ответ подтверждение и новый баланс.
6. Банкомат выдает деньги, карточку и квитанцию, показывающую новый баланс.
7. Банкомат регистрирует транзакцию.

19.2. Отсутствует основное действующее лицо

Первоначальный вариант

Ниже приведен фрагмент варианта использования для извлечения денег из банкомата.

Вариант использования:

Снять наличные со счета

Область действия: банкомат

Уровень: цель пользователя

Основное действующее лицо: клиент

1. Получает карточку для банкомата, PIN.
2. Получает в качестве типа транзакции "снять деньги".
3. Получает требуемую сумму.

4. Удостоверяется, что на счете достаточно денег.
5. Выдает деньги, квитанцию, карточку.
6. Возвращается в исходное состояние.

Замечания

Этот вариант использования написан строго с точки зрения системы. Он показывает все, что делает банкомат, но ничего не говорит о поведении основного действующего лица. Такого рода текст трудно воспринимать, проверять и корректировать. В некоторых случаях критически важная информация о поведении действующего лица опускается, вынуждая устанавливать очередность.

Исправление однозначно: указать каждое действующее лицо и действие.

Исправленный вариант

То же, что в разделе 19.1.

19.3. Слишком много деталей интерфейса пользователя

Первоначальный вариант

Вариант использования:

Совершить покупку

Область действия: приложение для электронной торговли

Уровень: цель пользователя

Основное действующее лицо: клиент

1. Система устанавливает экран для ввода ID и пароля.
2. Клиент вводит ID и пароль, щелкает по кнопке "OK".
3. Система подтверждает ID и пароль, отображает персональные данные.
4. Клиент вводит имя и фамилию, номер дома, название улицы, города, штата, почтовый индекс, номер телефона и щелкает по кнопке "OK".
5. Система подтверждает, что пользователь ей известен.
6. Система выводит список товаров, имеющихся в наличии.
7. Клиент щелкает по изображению товара, который хочет купить, вводит рядом с каждой картинкой количество, по окончании щелкает по кнопке "Готово".
8. Система удостоверяется с помощью системы управления складами, что на складе есть достаточное количество требуемых товаров. ... и т.д.

Замечания

Это, может быть, самая распространенная ошибка. Писатель слишком много пишет об интерфейсе пользователя, что делает вариант использования не документом

о требованиях как таковых, а скорее руководством для пользователя. Излишние подробности интерфейса пользователя ничего не добавляют к сюжету, но затрудняют чтение и делают документ о требованиях неустойчивым к изменениям.

Исправление состоит в том, чтобы найти способ описать намерения пользователя, не предлагая специального решения. Иногда это требует творческого подхода к формулировкам.

Исправленный вариант

Вариант использования:

Совершить покупку

Область действия: приложение для электронной торговли
Уровень: цель пользователя
Основное действующее лицо: клиент

1. Клиент регистрируется в системе с помощью ID и пароля.
2. Система подтверждает ID и пароль пользователя.
3. Клиент вводит имя, адрес, номер телефона.
4. Система подтверждает, что пользователь ей известен.
5. Клиент указывает товар и его количество.
6. Система удостоверяется с помощью системы управления складами, что на складе есть достаточное количество требуемых товаров. ... и т.д.

19.4. Очень низкие уровни цели

Первоначальный вариант

Вариант использования:

Совершить покупку

Область действия: приложение для электронной торговли
Уровень: цель пользователя
Основное действующее лицо: клиент (пользователь)

1. Пользователь регистрируется в системе с помощью ID и пароля.
2. Система подтверждает ID и пароль.
3. Пользователь вводит имя.
4. Пользователь вводит адрес
5. Пользователь вводит номер телефона.
6. Пользователь указывает товар.
7. Пользователь указывает количество.
8. Система подтверждает, что пользователь ей известен.
9. Система подключается к системе управления складами.
10. Система запрашивает у складской системы текущие уровни запасов.

11. Система управления складами возвращает текущие уровни запасов.
12. Система удостоверяется, что на складе есть достаточное количество требуемых товаров. ... и т.д.

Замечания

Ясно, что это будет длинный и нудный вариант использования. Мы не можем упрекнуть автора в том, что эти шаги описывают интерфейс пользователя слишком тщательно, но нам определенно нужно сократить текст и прояснить, что происходит.

Для сокращения текста:

- Объедините элементы данных (шаги 3-5) из разных шагов. Установив, что пользователь пытается обеспечить персональные данные, мы получим хорошее краткое имя для всех элементов информации об индивидууме, которые будут собираться. Я считаю, одно краткое имя слишком неопределенно, поэтому я указываю на список полей, который будет расширяться где-то в другом месте, не затрагивая этот вариант использования.
- Сосредоточьте все данные, идущие в одном направлении, в одном шаге (шаги 3-7). Это не всегда лучший способ. Бывает, что получение персональной информации существенно отличается от указания товара и количества, поэтому автор помещает их в разных строках. Дело вкуса. Я предпочитаю объединять все данные, идущие в одном направлении. Если это выглядит слишком громоздко или для расширений необходимо, чтобы они были разделены, я их вновь разделяю.
- Найдите более высокий уровень цели (шаги 8-11). Выясните, зачем система совершает все эти действия. Она пытается удостовериться при помощи системы управления складами, что на складе есть достаточное количество требуемого товара. Эта цель более высокого уровня и фиксирует требования так же ясно, но описание значительно сокращается.

Исправленный вариант

Вариант использования:

Совершить покупку

Область действия: приложение электронной торговли

Уровень: цель пользователя

Основное действующее лицо: клиент (пользователь)

1. Пользователь регистрируется в системе с помощью ID и пароля.
2. Система подтверждает ID и пароль пользователя.
3. Пользователь вводит персональные данные (имя, адрес, номер телефона), указывает товар и количество.
4. Система подтверждает, что пользователь ей известен.
5. Система удостоверяется при помощи системы управления складами, что на складе есть достаточное количество требуемого товара. ... и т.д.

19.5. Цель и содержание не соответствуют друг другу

Вспомните упражнение 7.4, в котором вы должны были исправить неверный вариант использования *Зарегистрироваться в системе*.

Если вы еще этого не сделали, найдите эти три ошибки:

- Тело варианта использования не соответствует намерению, заявленному в названии и описании. Практически это не менее двух сложенных вместе вариантов использования.
- Здесь описываются подробности интерфейса пользователя.
- В тексте используются логические структуры программирования, а не обычный язык.

Если вы решили не утруждать себя, обратитесь к приложению В, в котором содержатся разбор и решение.

19.6. Развитый пример варианта использования с чрезмерной детализацией интерфейса пользователя

Компания FirePond Corporation любезно позволила мне использовать следующие примеры *первоначальных* и *исправленных* вариантов. Первоначальный вариант занимает несколько страниц, часть из которых ушла на основной сценарий и альтернативы. Исправленный вариант втрое короче. Он содержит ту же базовую информацию, но без усложняющих подробностей интерфейса пользователя.

Внимательно прочитайте основной сценарий и подумайте, как сделать этот длинный вариант использования более читабельным, не жертвуя содержанием. Особенно обратите внимание на детали интерфейса пользователя, которые обнаруживаются в тексте, а также на некоторые расширения. Я удалил некоторые из них, но оставил значительную часть, чтобы вы почувствовали, как трудно работать с таким объемным вариантом использования. Как бы вы сократили его?

Несколько слов о названии варианта использования. Для языка маркетинга информационных технологий считается недостаточным сказать, что покупатель выбирает товар. Столкнувшись с множеством товаров, покупатель ищет решение для своей ситуации. Я мог бы переименовать этот вариант использования в *Выбрать товар*, но это не моя привилегия. Название *Найти решение* считается правильным в среде, где был написан и будет читаться этот вариант использования, поэтому оно остается.

Приношу благодарность Дэйву Скотту и Расселу Уолтерсу из FirePond Corporation.

Первоначальный вариант

Вариант использования 36

Найти решение (до)

Область действия: наша web-система

Уровень: цель пользователя

Основное действующее лицо: покупатель — потребитель или агент, желающий исследовать товары, которые он мог бы купить

Основной сценарий:

Действие действующего лица	Реакция системы
1. Этот вариант использования начинается, когда покупатель посещает web-сайт электронной торговли.	2. Система может получить информацию о типе покупатепя, посещающего web-сайт. 3. Система запросит установление личности покупатепя, инициирует установление личности. Если система не устанавливает личность сейчас, она должна быть установлена перед сохранением принятого решения. 4. Система предоставит покупателю следующие возможности: создать новое решение, восстановить сохраненное решение.
5. Покупатель выбирает "создать новое решение".	6. Система задаст первый вопрос для определения потребностей и интересов покупателя.
7. Покупатель может повторить добавление к тележке выбранных товаров: 8. Пока имеются вопросы по определению потребностей и интересов: 9. Покупатель ответит на вопросы.	10. Система на основе предыдущих ответов с помощью варьирующихся номеров и типов вопросов запросит информацию для определения потребностей и интересов покупателя, вместе с тем предоставив ему дополнительную информацию о продукции, ее возможностях и преимуществах, а также сравнительные данные и цены.
11. Покупатель отвечает на последний вопрос.	12. Поспе последнего вопроса о потреб ностях и интересах система выведет рекомендации относительно продуктовой линии и сопутствующую информацию, такую как данные о продукции, возможности и преимущества, сравнительные данные и цены.
13. Покупатель выберет продуктовую пинию.	14. Система задаст первый вопрос для опредепения необходимой модели изделия.

15. Пока имеются вопросы по определению рекомендаций относительно модели изделия: 16. Покупатель ответит на вопросы.	17. Система будет задавать вопросы, зависящие от предшествующих ответов, чтобы определить потребности и интересы покупателя относительно моделей изделий, вместе с тем предоставив ему допопнительную информацию о продукции, ее возможностях и преимуществах, а также сравнительные данные и цены.
18. Покупатель отвечает на последний вопрос.	19. После последнего вопроса о необходимой модели изделия система представит рекомендации по модепи изделия и соответствующую информацию, такую как данные о продукции, ее возможности и преимущества, сравнительные данные и цены.
20. Покупатель выберет модель изделия.	21. Система определит стандартные варианты модели изделия и затем задаст первый вопрос, чтобы определить основные варианты изделия.
22. Пока имеются вопросы по определению рекомендаций относительно варианта изделия: 23. Покупатель ответит на вопросы.	24. Система будет задавать вопросы, зависящие от предшествующих ответов, чтобы определить потребности и интересы покупателя относительно основных вариантов изделия, вместе с тем предоставив ему дополнительную информацию о продукции, ее возможностях и преимуществах, а также сравнительные данные и цены.
25. Покупатель отвечает на последний вопрос.	26. После последнего вопроса о желательном варианте изделия система представит пользователю выбранную модель и выбранные варианты для подтверждения.
27. Покупатель анализирует свой выбор товара, одобряет его и решает добавить товар к своей тележке.	28. Система добавит выбранный товар и сопутствующую информацию (навигационные данные и ответы) к тележке с товарами. 29. Система представпяет обзор тележ ки и всех товаров в ней.
30. Конец повторяющихся шагов для тележки с товарами.	31.
32. Покупатель пошлет запрос на персональное предложение относительно товаров своей тележки.	33. Система задаст первый вопрос, чтобы определить, какое содержимое тележки следует использовать для персонального предложения.

34. Пока имеются вопросы по определению содержимого тележки для персонального предложения: 35. Покупатель ответит на вопросы.	36. Система будет задавать покупателю вопросы, зависящие от предшествующих ответов, чтобы определить содержимое тележки для персонального предложения, вместе с тем предоставив ему дополнительную информацию о продукции, ее возможностях и преимуществах, а также сравнительные данные и цены.
37. Покупатель отвечает на последний вопрос.	38. После ответа на последний вопрос о содержимом тележки для персонального предложения система сформирует и представит персональное предложение.
39. Покупатель рассмотрит предложение и решит его распечатать.	40. Система распечатает предложение.
41. Покупатель запросит сохранение своего решения.	42. Если личность покупателя еще не установлена, инициируется установление личности. 43. Система запросит пользователя об идентификационной информации по решению.
44. Покупатель введет идентификационную информацию по решению и сохранит решение.	45. Система сохранит решение и свяжет его с покупателем.

Расширения:

*a. В любой точке в процессе выполнения варианта использования Принять решение, если покупатель не выполняет никаких действий в течение заранее определенного периода простоя, система уведомит его об отсутствии активности и спросит, не хочет ли он продолжить работу. Если покупатель не отвечает в течение разумного промежутка времени (30 с), вариант использования прекращает работу. В противном случае покупатель продолжит работу.

*b. В любой точке серии вопросов и ответов покупатель может вернуться к любому вопросу, изменить свой ответ и продолжить серию.

*c. В любой точке после представления рекомендации по товару покупатель может посмотреть результаты расчета характеристик, чтобы определить, насколько они соответствуют его потребностям. Система выполнит расчет и представит покупателю данные. Покупатель продолжит процесс поиска решения с той точки, где его оставил.

*d. В любой точке серии вопросов и ответов система может взаимодействовать с системой инвентаризации запасных частей, чтобы получить информацию о наличии запасных частей и (или) с системой технологической подготовки производства, чтобы получить перечень комплектующих. С помощью данных о наличии запасных частей и перечня можно отфиль-

тровать представленную информацию о выборе товара или показать наличие запасных частей покупателю во время процесса поиска решения. Инициировать варианты использования Получение данных о наличии запасных частей и Получить перечень комплектующих.

*e. В любой точке серии вопросов и ответов система представляет соответствующую информацию о производственных каналах для запасных частей. Покупатель выбирает канал. Система может передать по этому каналу связанную с товаром или другую информацию для наилучшего размещения или представить подходящее содержимое. Завершив работу с промышленным web-сайтом, покупатель возвратится к точке, откуда ушел, возможно, с возвращением требований к товару, которые система будет подтверждать. Инициировать обзорную информацию по товару.

*f. В любой точке процесса поиска покупатель может выдать запрос на соединение: вариант использования Инициировать запрос на соединение.

*g. В любой точке серии вопросов и ответов система может установить точки триггера фиксации данных о рынке, в которых система будет регистрировать навигационные данные, информацию о выборе товара, а также вопросы и ответы, которые будут использованы вне этой системы для анализа рынка.

*h. В заранее определенной точке поиска система может сформировать директиву для фиксации данных, накопленных к этому моменту. Вариант использования Инициировать формирование директивы.

*i. В любой точке покупатель может выйти из приложения электронной торговли: Если было найдено новое решение или изменено текущее с момента последнего сохранения, система спросит покупателя, не хочет ли он сохранить решение. Инициировать Сохранить решение.

*j. В любой точке после нахождения нового решения или изменения текущего покупатель может запросить сохранение решения. Инициировать Сохранить решение.

1a. Покупатель посещал web-сайт производителя товара и установил требования к изделию. Web-сайт производителя позволяет выбрасывать на рынок новые товары в эту систему электронной торговли для дальнейшего поиска решения:

1a1. Покупатель входит в систему электронной торговли с требования ми к товару и, может быть, данными по идентификации.

1a2. Система получает требования к товару и, возможно, данные по идентификации пользователя.

1a3. Система проверит, в какой точке процесса поиска находится покупатель, и установит стартовую точку серии вопросов для продолжения процесса поиска.

1a4. Основываясь на установленной стартовой точке, мы можем продолжать с шага 5, 12 или 18.

3a. Покупатель хочет работать с ранее сохраненным решением:

3a1. Покупатель выбирает восстановление решения.

3a2. Система представляет список сохраненных решений этого покупателя.

3a3. Покупатель выбирает решение, с которым будет работать.

3a4. Система восстанавливает выбранное решение.

3a5. Продолжать с шага 26.

23a. Покупатель хочет изменить некоторые рекомендованные варианты: {Создать вариант использования Выбрать вариант, поскольку существуют альтернативы основному потоку. Систему можно установить так, чтобы показывать все варианты, даже несовместимые. Если покупатель выбирает несовместимый вариант, система выведет сообщение об этом и, может быть, указание, как получить товар, сконфигурированный так, чтобы вариант был совместимым.} ... пока покупателю необходимо изменить вариант:

23a1. Покупатель выбирает вариант, который желает изменить.

23a2. Система представляет имеющиеся совместимые варианты.

23a3. Покупатель выбирает желаемый вариант.

26a. Покупатель хочет изменить количество выбранных товаров в тележке для покупок:

26a1. Покупатель выбирает продукт из тележки и изменяет его количество.

26a2. Система пересчитает цену, учитывая скидки, налоги, сборы и специальные цены, основываясь на информации о покупателе, а также на ответах на вопросы.

26b. Покупатель хочет добавить к тележке встречную продажу товара:

26b1. См. раздел Встречная продажа.

26c. Покупатель хочет вызвать сохраненное решение:

26c1. Система представляет список сохраненных решений данного покупателя.

26c2. Покупатель выбирает решение, с которым будет работать.

26c3. Система восстанавливает указанное решение.

26c4. Продолжать с шага 26.

26d. Покупатель хочет заплатить за продукты в тележке для покупок с помощью имеющихся финансовых планов:

26d1. Покупатель выбирает финансирование продуктов в тележке.

26d2. Система задаст ряд вопросов, зависящих от предшествующих ответов, чтобы определить рекомендации по финансовому плану. Система взаимодействует с финансовой системой, чтобы получить подтверждение кредитоспособности. Инициировать оценку кредитоспособности.

26d3. Покупатель выберет финансовый план.

26d4. Система задаст ряд вопросов, зависящих от предшествующих ответов, чтобы определить подробности выбранного финансового плана.

26d5. Покупатель рассмотрит подробности финансового плана и решит следовать ему.

26d6. Система разместит заказ на финансовый план с помощью финансовой системы. Инициировать размещение финансового заказа.

Замечания

В данном случае был выбран формат в две колонки, чтобы разделить шаги действующих лиц. Авторы не представляли себе отчетливо проекта интерфейса пользователя, кроме модели вопрос/ответ, поэтому описали вопросы и ответы в варианте использования.

С самого начала я избавился от формата в две колонки и создал простое повествование в одну колонку. Я хотел, чтобы сюжетная линия хорошо просматривалась и для этого не надо было переворачивать страницы.

Просматривая результат, я искал предположения относительно проектирования интерфейса пользователя, а также цели, уровень которых можно было бы повысить. Ключ содержится в предложении "Система задаст вопросы, число и тип которых будут изменяться в зависимости от предшествующих ответов". Здесь не упоминается что-либо столь же очевидное, как щелчки мышью, поэтому предполагается интерфейс, основанный именно на том, что пользователь набирает на клавиатуре ответы на вопросы. Я хотел представить совершенно другой интерфейс, чтобы посмотреть, как это повлияет на текст варианта использования. Я имел в виду проект, в котором пользователь только щелкал бы по картинкам. Тогда можно было бы уловить намерения пользователя и устранить зависимость от пользовательского интерфейса. Это тоже послужило сокращению документа.

В каждом предложении я перепроверял уровень цели, чтобы понять, не слишком ли низок уровень цели шага и можно ли его повысить в таком случае.

В начале этой работы я не был уверен, что формат в одну колонку будет ясно показывать разработчикам обязанности системы. Авторы первоначального варианта рассеяли мои сомнения. Практически они были довольны, так как:

- Документ стал короче и удобнее для чтения.
- Все действительные требования в документе сохранились и остались ясно выделенными.
- Проект расширил границы.

Исправленный вариант

Вариант использования 37

Найти возможные решения (после)

Область действия: web-система программного обеспечения

Уровень: цель пользователя

Предусловия: нет

Минимальная гарантия: нет действия, нет покупок

Гарантия успеха: покупатель имеет ноль или более товаров, подготовленных к покупке, в системе есть журнал регистрации выбора и навигационных данных, записаны также характеристики пользователя.

Основное действующее лицо: покупатель (любой путешествующий по Сети)

Триггер: покупатель собрался найти решение.

Основной сценарий:

1. Покупатель инициирует поиск нового решения.
2. Система помогает покупателю выбрать продуктовую линию, модель и ее варианты, представляя ему информацию и задавая серии вопросов для определения потребностей и интересов покупателя. Серии вопросов и выводимая на экран информация зависят от ответов, которые покупатель дает по ходу дела. Система выбирает их в соответствии с запрограммированными путями выбора, чтобы выделить и рекомендовать то, что, вероятно, заинтересует покупателя. Представляемая информация содержит данные о производстве, возможностях, преимуществах, сравнительных данных и т.д.
3. Система регистрирует навигационную информацию в ходе процесса.
4. Покупатель выбирает окончательную конфигурацию товара.
5. Покупатель добавляет его к тележке.
6. Система добавляет к тележке для покупок выбранный товар и навигационную информацию.
7. Система выводит на экран обзор тележки для покупок со всем ее содержимым. Покупатель повторяет предыдущие шаги столько раз, сколько пожелает, чтобы добраться до различных специализированных товаров и выбрать их, по желанию добавляя к тележке.

Расширения:

*a. В любой момент покупатель может запросить соединение.

1a. По соглашению между владельцем web-сайта и владельцем посылающего запрос компьютера в запросе может содержаться информация о типе покупателя:

1a1. Система выделяет из web-запроса любую пользовательскую и навигационную информацию, добавляет ее к данным регистрации и начинает с некоторой точки в глубине серии вопросов и ответов.

1a1a. Информация, приходящая с другого сайта, неверна или не понятна. Система делает максимум возможного, регистрирует всю поступившую информацию и по возможности продолжает.

1b. Покупатель желает продолжить работу над прежним сохраненным неполным решением.

1b1. Покупатель устанавливает личность и сохраняет решение.

1b2. Система восстанавливает решение и возвращает покупателя туда, где он остановился, перед тем как сохранить решение.

2a. Покупатель решает обойти одну или все серии вопросов:

2a1. Покупателю предоставляются рекомендации по товару на основе ограниченных (или нет) персональных характеристик, и он делает выбор.

2a2. Система регистрирует этот выбор.

2b. В любой момент перед добавлением товара к тележке покупатель может вернуться и изменить любой предыдущий ответ, чтобы получить новые рекомендации по товару и (или) новую возможность выбора.

2c. В любой момент перед добавлением товара к тележке покупатель может запросить расчет по имеющемуся тесту (характеристики), например, потянет ли данная конфигурация с этим весом на трейлер: Система производит вычисления и выводит ответ.

2d. Покупатель проходит точку, которую владелец web-сайта предварительно определил для формирования директив по продажам (динамическое бизнес-правило): Система формирует директиву по продажам.

2e. Система установлена для запроса у покупателя удостоверения личности: Покупатель устанавливает личность.

2f. Система установлена для взаимодействия с другими известными системами (инвентаризации запасных частей, технологической подготовки производства), которое повлияет на наличие и выбор товара:

2f1. Система взаимодействует с другими известными системами (инвентаризации запасных частей, технологической подготовки производства), чтобы получить необходимую информацию. (Получить данные о наличии запасных частей, Получить перечень комплектующих.)

2f2. Система использует результаты для фильтрации или показа данных о наличии товара и (или) вариантов (запасных частей).

2g. Покупатель выбирает канал из представленного множества каналов к web-сайтам производителей: Покупатель просматривает другой web-сайт.

2h. Система устанавливается для взаимодействия с известной информационной системой заказчика:

2h1. Система получает информацию для заказчика от информационной системы для заказчиков.

2h2. Система использует результаты, чтобы начать с некоторой точки внутри серии вопросов и ответов.

2i. Покупатель хочет узнать оценку кредитоспособности, поскольку она повлияет на процесс выбора товара: Покупатель получает оценку кредитоспособности.

2j. Покупатель указывает, что он купил товар прежде, и система устанавливается для взаимодействия с известной системой финансового учета.

2j1. Система получает последовательность выставления счетов.

2j2. Система использует последовательность выставления счетов покупателя, чтобы повлиять на процесс выбора товара.

2k. Покупатель решает изменить некоторые из рекомендованных вариантов: Система позволяет покупателю изменить столько вариантов, ско-

лько он пожелает, по ходу дела выводя на экран данные о допустимых вариантах и предупреждая о несовместимости других вариантов.

5a. Покупатель решает не добавлять товар к тележке для покупок: Система оставляет навигационную информацию на будущее.

7a. Покупатель хочет изменить содержимое тележки для покупок: Система позволяет покупателю изменить количество, удалить элементы или вернуться к более ранней точке в процессе выбора.

7b. Покупатель просит сохранить содержимое тележки для покупок:

7b1. Система запрашивает у покупателя имя и ID для сохранения и сохраняет содержимое.

7b1a. Личность покупателя не установлена: Покупатель устанавливает личность.

7c. Покупатель имеет товар для встречной продажи: Система фиксирует встречную продажу.

7d. Покупатель решает оплатить товары в тележке: Покупатель получает финансовый план.

7e. Покупатель выходит из системы электронной торговли, когда тележка для покупок не сохранена:

7e1. Система запрашивает у покупателя имя и ID для сохранения и сохраняет содержимое тележки.

7e1a. Покупатель решает выйти без сохранения: Система регистрирует навигационные данные и завершает сеанс с покупателем.

7f. Покупатель выбирает элемент из тележки для покупок, чтобы узнать о наличии соответствующего товара (нового или используемого) от системы инвентаризации.

7f1. Покупатель выдает запрос локализовать соответствующий товар в системе инвентаризации.

7f2. Покупатель заменяет элемент в тележке для покупок соответствующим товаром из системы инвентаризации.

7f2a. Покупатель не хочет приводить товар в соответствие с элементом из системы инвентаризации:

Система оставляет в тележке для покупок первоначальный элемент, которым интересовался покупатель, самостоятельно приводя его в соответствие с элементом из системы инвентаризации.

7f3. Система гарантирует, что в тележке для покупок содержится один элемент, сконфигурированный в соответствии с элементом из системы инвентаризации.

Вызывающие варианты использования: Интернет-магазин, Продажа связанных продуктов

Открытые вопросы:

Расширение 2c. Какие вопросы законны? Какие еще системы задействованы? Каковы требования к интерфейсам?

Часть 3

Памятки для занятых

Глава 20

Памятки для каждого варианта использования

Памятка 1. Вариант использования — прозаическое эссе

Как сказано в предисловии, создание вариантов использования можно сравнить с написанием прозаического эссе, где необходимы четкие формулировки, которые свойственны письменной прозе.

Рассел Уолтерс из компании FirePond написал:

> Я думаю, вышеприведенное утверждение точно отражает суть вопроса. Это проблема еще не получила нужной оценки. Для автора вариантов использования изучить этот вопрос необходимо. Однако я не уверен, что практикант может самостоятельно освоить данную тему, по крайней мере пока не прочтет эту книгу. Я не считал эту проблему главной и работал с концепцией вариантов использования четыре года, пока не появилась возможность сотрудничать с вами. И даже тогда все оставалось по-прежнему, пока я не проанализировал первоначальные и исправленные версии варианта использования 36. Единственное, на что придется читателям этой книги затратить усилие, — это осознать, что основной проблемой является создание эффективных вариантов использования. (Печатается с разрешения.)

Эта памятка поможет вам сосредоточиться на тексте, а не на схемах, и принимать к сведению стили изложения, с которыми вы будете встречаться.

Памятка 2. Старайтесь, чтобы вариант использования был читабельным

Ваш документ о требованиях должен быть коротким, ясным и читабельным.

> Я иногда чувствую себя учителем словесности в восьмом классе, который ходит туда-сюда и говорит:

Используйте глаголы действия в настоящем времени. Употребляйте действительный залог, а не страдательный. Где в предложении подлежащее? Говорите о том, что действительно является требованием; не упоминайте то, что таковым не является.

Следующие рекомендации помогут вам сделать ваш документ о требованиях коротким, четким и легким для чтения:

1. Старайтесь писать короче и ближе к делу. Длинные варианты использования ведут к длинному описанию требований, которое мало кому нравится читать.
2. Начните с вершины и создайте логически последовательный сюжет. На вершине будет стратегический вариант использования. От него будут ветвиться варианты использования уровня цели пользователя и, в конечном счете, уровня подфункции.
3. Именуйте варианты использования с помощью коротких глагольных конструкций, объявляющих цель, которая должна быть достигнута.
4. Начинайте с триггера и продолжайте, пока цель не будет достигнута или отвергнута и система не зарегистрирует необходимую информацию о транзакциях.
5. Используйте полные предложения с глагольными конструкциями, которые описывают подлежащие выполнению подцели.
6. Обеспечьте, чтобы на каждом шаге действующее лицо и его намерение были четко определены.
7. Выделяйте условия отказа и старайтесь, чтобы действия по восстановлению были удобочитаемыми. Должно быть ясно, что происходит потом, чтобы даже не пришлось указывать номера шагов.
8. Для описания альтернативного поведения применяйте расширения, а не предложения с если в теле главного варианта использования.
9. Создавайте варианты использования расширения только в особых случаях.

Памятка 3. Только одна форма предложения

Существует только одна форма предложения, используемая для описания шагов действия:

- Предложение в настоящем времени
- С глаголом действия в действительном залоге
- Описывающее, как действующее лицо успешно достигает цели, продвигающей весь процесс

Примеры:

Клиент вводит карточку и PIN.

Система удостоверяется в полномочиях клиента и достаточности остатка на счете.

PAF (персональный консультант по финансам) перехватывает ответы с web-сайта и обновляет портфель пользователя.

Служащий находит заявление об ущербе с помощью деталей поиска заявления об ущербе.

Употребляйте такие предложения в вариантах использования для бизнеса, системных, обобщенных, а также вариантах использования уровня подфункции, написанных как по полной форме, так и произвольно. Это относится и к основному сценарию, и к фрагментам сценария расширения. Овладейте этим стилем.

В блоках условий в расширениях должна быть другая грамматическая форма, чтобы не путать расширение с шагами действий. Лучше (но не всегда) использовать фрагмент предложения (или, может быть, полное предложение) в прошедшем времени. В конце условия ставьте двоеточие (:), а не точку.

Тайм-аут ожидания ввода PIN:

Неверный пароль:

Файл не найден:

Пользователь вышел, не закончив:

Представленные данные не полны:

Памятка 4. "Включайте" подчиненные варианты использования

Абсолютно естественное действие (если никто не велел вам сделать по-другому) — написать шаг, который вызывал бы цель более низкого уровня или вариант использования:

Служащий *находит заявление* об *ущербе* с помощью параметров поиска ущерба.

В терминах UML вызывающий вариант использования просто включал бы подчиненный вариант использования. Это столь очевидно, что не стоило бы упоминания, если бы не находились последователи применения связей расширения и специализаций UML (см. приложение А).

В качестве первого практического приема всегда применяйте между вариантами использования *связь включения* (includes). В этом случае затруднений с документам будет меньше, чем с документами, в которых смешаны связи включения со *связями расширения* (extends) и *специализациями* (specializes). См. раздел 10.2.

Памятка 5. Кто владеет мячом?

Иногда пишут, используя страдательный залог, или с точки зрения самой системы, смотрящей на мир. Тогда и получаются такие предложения, как "Вводится кредитный лимит". В этом предложении не упомянут тот, кто осуществляет ввод.

Пишите как бы с высоты птичьего полета, наблюдая за сценой и описывая ее, либо пишите в форме пьесы, объявляя, какое действующее лицо собирается действовать. Или представьте, что вы комментируете футбольный матч, в котором действующее лицо 1 владеет мячом, ведет его, далее следует передача игроку 2; игрок 2 пасует игроку 3 и т.д.

Пусть первое или второе слово предложения будет именем действующего лица, совершающего действие. Позаботьтесь о том, чтобы при всех обстоятельствах было ясно, кто владеет мячом.

Памятка 6. Уровень цели должен быть правильным

- См. раздел 5.5, где вопрос обсуждается подробно.
- Удостоверьтесь, что для варианта использования указан правильный уровень цели: обобщенный, пользователя или подфункции.
- Периодически проверяйте, знаете ли вы, где "уровень моря" для ваших целей и насколько ниже (или выше) уровня моря находятся шаги. Напомним тесты для цели уровня моря:
  - Цели добивается одно лицо в одном месте в одно время (2-20 мин).
  - Действующее лицо может уйти удовлетворенным, как только цель будет достигнута.
  - Добившись многих целей, действующее лицо (если работает по найму) может попросить повышения.
- Помните, что большинство вариантов использования имеет 3-9 шагов в основном сценарии, и уровень цели шага обычно ниже уровня цели варианта использования. Если в варианте использования более девяти шагов, поищите шаги, которые можно объединить, например:
  - Если одно и то же действующее лицо "владеет мячом" на протяжении нескольких шагов подряд.
  - Если описаны движения пользователя — типичное описание интерфейса пользователя, нарушающее правило 5 (см. раздел 7.2).
  - Если много простых передач между двумя действующими лицами (спросите, не пытаются ли они на самом деле совершить что-нибудь одним уровнем выше с помощью этих передач).
- Поинтересуйтесь, почему пользователь (действующее лицо) делает это. Ответ, который вы получите — это более высокий уровень цели. Вам, возможно, удастся с помощью этой цели объединить несколько шагов. На рис. 20.1 показано, как цели шагов размещаются внутри целей вариантов использования на различных уровнях.

Спросите "почему", чтобы сдвинуть уровни

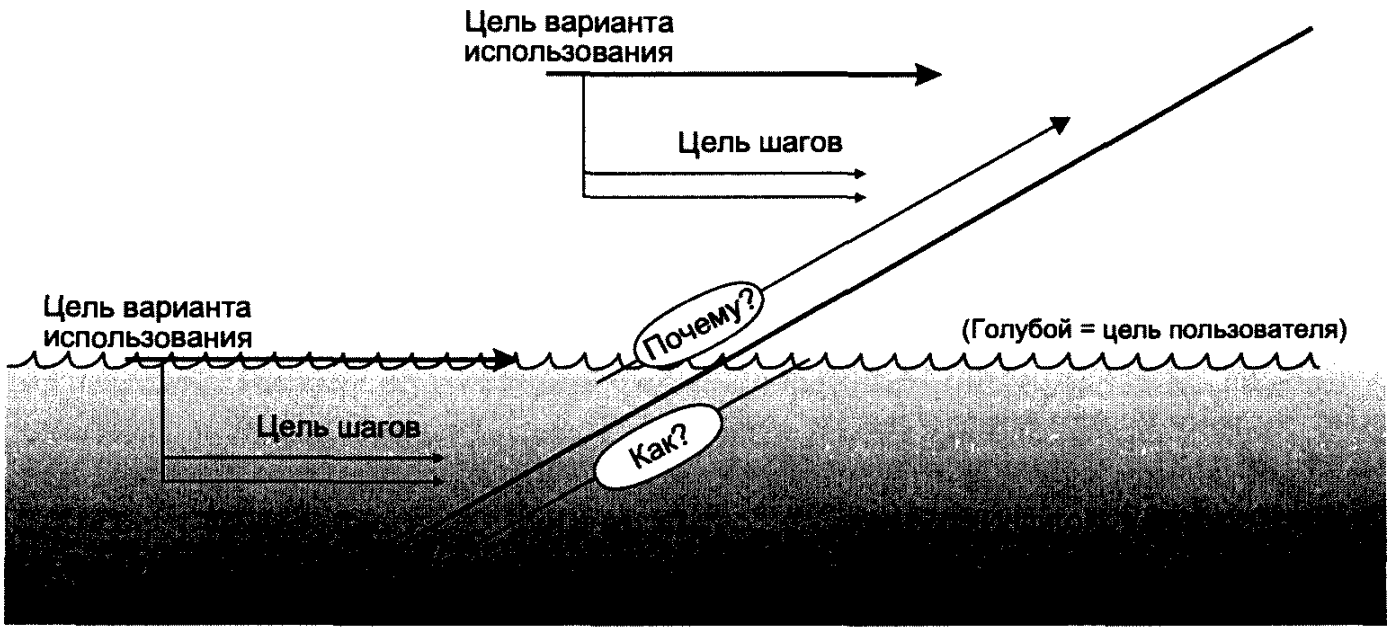


Рис. 5.2. *Спросите "почему", чтобы изменить уровни*

Памятка 7. Не допускайте деталей графического интерфейса пользователя в варианте использования

Убедитесь, что шаг, который вы только что описали, фиксирует истинное намерение действующего лица, а не просто движения в рамках интерфейса пользователя. Этот совет применим, когда вы пишете функциональные требования, так как ясно, что вы можете создавать варианты использования, чтобы документировать и сам интерфейс пользователя.

Описывать движения пользователя при манипулировании интерфейсом в документе о требованиях не следует, так как:

- Документ становится излишне длинным.
- Требования становятся неустойчивыми в том смысле, что небольшие изменения в проекте интерфейса вызывают изменение в "требованиях" (которые все-таки не были в чистом виде требованиями).
- Это задача разработчика интерфейса пользователя, которому вы должны доверять как специалисту.

Варианты использования в этой книге по большей части должны служить вам хорошими примерами. Вот выдержка из одного из них, варианта использования 20:

1. **Оценщик просматривает и оценивает отчеты ...**
2. **Оценщик оценивает постоянную неплатежеспособность, используя ...**
3. **Оценщик определяет сумму долга ...**
4. **Оценщик определяет окончательный диапазон суммы возмещения.**

Следующие шаги написаны неправильно:

1. **Система устанавливает экран Login (регистрации) с полями для имени и пароля пользователя.**
2. **Пользователь вводит имя и пароль и щелкает по кнопке "OK".**
3. **Система проверяет имя и пароль.**
4. **Система устанавливает основной (main) экран, содержащий меню функций.**
5. **Пользователь выбирает функцию и щелкает по кнопке "OK".**

Очень легко незаметно перейти к описанию подробностей интерфейса пользователя, поэтому будьте настороже.

Памятка 8. Два итога

Каждый вариант использования имеет два возможных итога: успех и неудачу.

Имейте в виду, что шаг действия вызывает подчиненный вариант использования, последний может завершиться успехом либо неудачей. Если он из расширения, опишите обработку как успеха, так и неудачи в том же расширении (см., например, вариант использования 22).

На самом деле у вас есть две обязанности по отношению к успеху или неудаче в достижении цели. Во-первых, убедитесь, что вы имеете дело с неудачей каждого вызываемого варианта использования. Во-вторых, обеспечьте, чтобы ваш вариант использования удовлетворял интересам каждого участника, особенно если цель не достигается.

Памятка 9. Участникам необходимы гарантии

Вариант использования содержит не только явные взаимодействия основного действующего лица и системы. Если бы он включал только это, то представлял бы не приемлемые требования к поведению, а лишь описание интерфейса пользователя.

Система подталкивает к соглашению между участниками. Один из них является основным действующим лицом, другие отсутствуют и не могут себя защитить. Вариант использования описывает, как система защищает все их интересы в различных обстоятельствах, когда пользователь управляет сценарием. Вариант использования описывает гарантии, которые им обеспечивает система.

Назовите всех участников с их интересами в каждом варианте использования. Найдите от двух до пятерых участников: основное действующее лицо, владельца, какое-либо руководящее ведомство и кого-нибудь еще. Возможно, персонал по тестированию или сопровождению заинтересован в функционировании данного варианта использования.

Обычно одни и те же участники задействованы в большинстве вариантов использования, и, как правило, их интересы во всех вариантах использования одинаковы. Поэтому нетрудно перечислить их имена и интересы. Об интересах можно сказать следующее:

- Интерес основного действующего лица сформулирован в названии варианта использования. Он обычно состоит в том, чтобы что-то получить.

- Интерес компании, как правило, состоит в том, чтобы основное действующее лицо не получило что-либо бесплатно или заплатило за то, что получило.
- Интерес руководящего ведомства обычно состоит в том, чтобы удостовериться, что компания следует правилам и регистрирует соответствующую информацию.
- Один из участников обычно заинтересован в восстановлении после отказа посреди транзакции, т.е. в новой регистрации.

Следите, чтобы основной сценарий и его расширения обращались к интересам участников. Это совсем не трудно. Прочитайте текст варианта использования, начиная с основного сценария, и посмотрите, присутствуют ли эти интересы. Вы будете удивлены, увидев, как часто они опускаются. Очень часто автор не учитывает, что ошибка может произойти в середине основного сценария, и не оставляет ни записи об этом, ни данных для восстановления. Проверьте, во всех ли случаях обработка ошибки защищает все интересы всех участников.

Часто новое условие расширения обнаруживает в основном сценарии пропущенное подтверждение. Иногда выполняется так много подтверждений, что автор помещает множество проверок в отдельную область документа, возможно, даже создавая раздел бизнес-правил.

Пит Мак-Брин из компании Roshi рассказал мне о том, как его группа в первый раз составила список интересов участников для системы, которую они уже разработали. В этом списке они нашли все запросы на изменения, поступившие в течение первого года работы их программного обеспечения. Они успешно создали и сдали систему, не удовлетворив некоторых потребностей отдельных участников. Поэтому от последних пришли запросы на изменения. Если бы список участников и интересов был составлен раньше, запросов на изменения (во всяком случае, некоторых из них) не было бы. В результате Пит стал строго проверять, зафиксированы ли интересы участников в вариантах использования. Такая проверка занимает мало времени, но при этом можно обнаружить очень многое.

Раздел гарантий шаблона документирует, как вариант использования удовлетворяет эти интересы. Вы могли бы сэкономить на описании гарантий для менее важного и менее формального проекта, если в группе хорошо налажено взаимодействие с нужными специалистами. Вы можете уделить гарантиям больше внимания в более критичных проектах, где возможен больший ущерб. Однако в обоих случаях ваша группа должна по крайней мере мысленно прокрутить как успешное, так и неудачное завершение варианта использования.

Разумно описывать гарантии до создания основного сценария, так как при этом вы задумываетесь о необходимых подтверждениях на первом проходе, вместо того чтобы обнаруживать их позднее и возвращаться назад для изменения текста.

В разделах 2.2 и 6.2 эти темы освещаются более подробно.

Памятка 10. Предусловия

Предусловия варианта использования объявляют правильные условия его функционирования. Предусловия должны быть таковы, чтобы система могла обеспечить

их истинность. Вы записываете предусловия в документе, поскольку не будете проверять их снова в варианте использования.

Существуют две распространенные ситуации, порождающие предусловия. Чаще всего это ситуация, когда пользователь регистрируется в системе и последняя подтверждает его имя и пароль. В другом случае второй вариант использования становится активным (отчасти благодаря первому варианту использования), ожидая, что первый вариант установит отдельное условие, на которое он может полагаться. Например, пользователь выбирает или частично выбирает товар в первом варианте использования, а второй при выполнении применяет информацию об этом выборе.

Когда я вижу предусловие, я знаю, что всегда существует вариант использования более высокого уровня, в котором это предусловие устанавливается.

Памятка 11. Тесты "прошел/не прошел" для одного варианта использования

Простые тесты "прошел/не прошел" позволяют узнать, правильно ли вы разработали часть варианта использования. В таблице 20.1 приведено несколько разработанных мною тестов. Все они должны давать ответ "да".

Таблица 20.1. *Тесты "прошел/не прошел" для одного варианта использования*

Поле	Вопрос
Название варианта использования	1. Является ли предложение, называющее цель основного действующего лица, предложением с глаголом действия?
	2. Может система удовлетворить эту цель?
Область действия и уровень	3. Заполнены ли поля?
Область действия	4. Рассматривает ли вариант использования упомянутую в разделе области действия систему как "черный ящик"? (Ответ должен быть "да", если это документ, описывающий требования к системе, но может быть и "нет", если это вариант использования для бизнеса типа "прозрачного ящика".)
	5. Если система в разделе Область действия является той, что должна разрабатываться, должны ли разработчики создавать только то, что есть в ней, и ничего, что в нее не входит?
Уровень	6. Соответствует ли содержание варианта использования заявленному уровню цели?
	7. Действительно ли цель находится на заявленном уровне цели?
Основное действующее лицо	8. Присуще ли ему поведение?
	9. Имеет ли оно цель в отношении разрабатываемой системы, которая должна быть услугой разрабатываемой системы?
Предусловия	10. Обязательны ли они и может ли их установить разрабатываемая система?
	11. Они действительно никогда не проверяются в варианте использования?

***Таблица 20.1** (продолжение)*

Поле	Вопрос
Участники и интересы	12. Названы ли они, и должна ли система удовлетворять их интересы, как установлено? (Использование варьируется установленными нормами и допусками.)
Минимальные гарантии	13. Все ли интересы участников защищены?
Гарантии успеха	14. Все ли интересы участников соблюдены?
Основной сценарий	15. В нем 3-9 шагов?
	16. Выполняется ли он от триггера до обеспечения гарантии успеха?
	17. Допускает ли он правильные изменения в последовательности?
Каждый шаг в любом сценарии	18. Он описывает достижение цели?
	19. Заметно ли продвигает процесс его успешное выполнение?
	20. Ясно ли, какое действующее лицо достигает цель, кто "ударяет по мячу"?
	21. Ясно ли намерение действующего лица?
	22. Уровень цели шага ниже уровня цели варианта использования в целом? Он чуть ниже (что предпочтительнее) уровня цели варианта использования?
	23. Вы уверены, что шаг не описывает проект интерфейса пользователя?
	24. Ясно ли, какая информация передается на шаге?
	25. Условие в шаге "подтверждается", а не "проверяется"?
Условие расширения	26. Может ли и должна ли система и обнаруживать, и обрабатывать его?
	27. Это действительно то, что нужно системе?
Список изменений в технологии и данных	28. Вы уверены, что это не обычное поведенческое расширение основного сценария?
Общее содержание варианта испопьзования	29. Заказчикам и пользователям: "Это то, что вам требуется?"
	30. Заказчикам и пользователям: "Вы сможете сказать по завершении сдачи, что получили то, что хотели?"
	31. Разработчикам: "Вы можете это реализовать?"

Глава 21

Памятки для набора вариантов использования

Памятка 12. Бесконечно развивающийся сюжет

В проекте системы существует один вариант использования на вершине стека, называемый примерно так: *Использовать систему ZZZ*. Этот вариант использования есть нечто большее, чем оглавление, включающее действующих лиц и их цели высшего уровня. Он служит отправной точкой для любого, кто впервые смотрит на систему. Он не является обязательным, так как в нем почти нет сюжета, но большинству людей просто удобно знать место, откуда следует начинать чтение.

Этот вариант использования наивысшего уровня вызывает предельные варианты использования, которые показывают обобщенные цели "крайних" основных действующих лиц системы. Для корпоративной информационной системы обычно существует внешний клиент, отдел маркетинга или информационных технологий, либо отдел безопасности. Эти варианты использования показывают взаимоотношения вариантов использования уровня моря, которые определяют систему. Для большинства читателей "история" начинается с одного из них.

Предельные варианты использования развертываются в варианты использования уровня цели пользователя или уровня моря. В вариантах использования уровня цели пользователя областью действия проектирования является разрабатываемая система. Шаги показывают взаимодействие действующих лиц и системы, направленное на достижение непосредственной цели пользователя.

Шаг варианта использования уровня моря развертывается в подводный вариант использования (цвета индиго или подфункцию), если подчиненный вариант использования составной или применяется в нескольких местах. Сопровождение вариантов использования уровня подфункции обходится недешево, поэтому употребляйте их только при необходимости. Как правило, варианты использования уровня подфункции вам придется создавать для таких шагов, как *Найти клиента, Найти товар* и т.д.

Иногда шаг одного варианта использования цвета индиго развертывается в вариант использования более темного цвета (индиго).

Набор вариантов использования следует рассматривать как бесконечно развертывающийся сюжет, так как перемещения сложных разделов документа в собственные варианты использования или упаковка простого подчиненного варианта использования в вызывающий его вариант использования становятся рядовыми операциями. Каждый шаг действия может в принципе быть развернут в самостоятельный вариант использования (см. раздел 10.1).

Памятка 13. Область действия корпорации и область действия системы

Область действия проектирования может стать причиной недоразумения. Все по-разному понимают точные границы системы. В частности, четко уясните, какой вариант использования вы пишете: для бизнеса или системный.

В варианте использования для бизнеса областью действия проектирования являются деловые операции. Этот вариант описывает, как действующее лицо вне организации достигает цель в отношении этой организации. Вариант использования для бизнеса редко содержит упоминание о технологии, поскольку касается деловых операций.

В системном варианте использования область действия проектирования — компьютерная система, подлежащая разработке. В нем описывается, как действующее лицо достигает цель с помощью компьютерной системы. Он как раз о технологии.

Вариант использования для бизнеса часто создается в виде "прозрачного ящика", описывая взаимодействия между людьми и отделами компании, в то время как системный вариант использования почти всегда имеет тип "черного ящика". Это, как правило, обоснованно, поскольку назначение большинства вариантов использования для бизнеса состоит в описании настоящего или будущего проекта компании, тогда как системный вариант использования создает требования для нового проекта. Вариант использования для бизнеса касается внутреннего механизма бизнеса, а системный вариант использования — внешних проявлений компьютерной системы.

Если вы разрабатываете компьютерную систему, вам следует иметь варианты использования и для бизнеса, и системные. В варианте использования для бизнеса содержится контекст функционирования системы, определяется ее место в бизнесе.

Чтобы уменьшить путаницу, всегда указывайте область действия варианта использования. Попробуйте применить пиктограммы для иллюстрации типа варианта использования (системного или для бизнеса). (См. раздел 3.2, подраздел об использовании графических пиктограмм для выделения области действия проектирования.) Можно поместить картинку системы внутри содержащей ее системы в самом варианте использования (см. вариант использования 8).

Памятка 14. Основные достоинства вариантов использования и вариации

Хотя люди изобретают новые форматы варианта использования, опытные авторы приходят к общему мнению по основным значениям формата. В докладах на конфе-

ренциях TOOLS USA в 1999 г. был приведен первый список ошибок, сделанных при создании вариантов использования. Способы исправления ошибок, описанные в этих работах, отражают основные достоинства вариантов использования.

Основные достоинства

БАЗИРУЕТСЯ НА ЦЕЛИ. При достижении цели варианты использования концентрируются вокруг целей основных действующих лиц и подцелей различных действующих лиц, включая разрабатываемую систему. Каждое предложение описывает достигаемую подцель.

ВЗГЛЯД С ВЫСОТЫ ПТИЧЬЕГО ПОЛЕТА. Вариант использования должен описывать действия, рассматриваемые как бы с высоты птичьего полета, или создаваться как пьеса, в которой фигурируют действующие лица. Не следует писать с точки зрения "изнутри системы".

ЧИТАБЕЛЬНОСТЬ. Вариант использования или любую спецификацию кто-то будет читать. Если документ нечитабелен, он не выполняет своего назначения. Можно сделать документ легким для чтения, принеся в жертву долю точности и даже правильности, компенсировав это удобной формой. В противном случае вы рискуете тем, что руководители не станут читать ваши варианты использования.

ИМЕЕТ НЕСКОЛЬКО НАЗНАЧЕНИЙ. Варианты использования — форма описания поведения, которая может служить различным целям на различных этапах проекта. Например, они нужны для:

- Обеспечения функциональных требований к "черному ящику"
- Обеспечения требований для перепроектирования бизнес-процесса организации
- Документирования бизнес-процесса организации ("прозрачный ящик")
- Выяснения требований у пользователей или участников (от вариантов использования откажутся, если группа напишет окончательные требования в несколько иной форме)
- Определения тестовых примеров, которые нужно создавать и прогонять
- Документирования внутреннего содержания системы ("прозрачный ящик")
- Документирования поведения проекта или структуры проекта

ТРЕБОВАНИЯ К "ЧЕРНОМУ ЯЩИКУ". При использовании функциональной спецификации разрабатываемая система всегда трактуется как "черный ящик". В проектной группе, которая пыталась написать требования к "прозрачному ящику" (гадая, как будет выглядеть внутреннее устройство системы), получившиеся в результате варианты использования трудно читались, не вполне отвечали нормам и были неустойчивы, так как изменялись по мере продвижения проекта.

АЛЬТЕРНАТИВНЫЕ ПУТИ ПОСЛЕ ОСНОВНОГО СЦЕНАРИЯ. Оригинальная идея помещать альтернативные направления после основного сценария открывает путь к

созданию самых легких для чтения вариантов использования. Ветвление внутри основного тела текста слишком затрудняет чтение документа.

ЭКОНОМИЯ УСИЛИЙ. Большие затраты времени на варианты использования не увеличивают их ценность. Первый набросок варианта использования создает, возможно, половину его ценности, добавление расширений повышает ее, но со временем изменения формулировок предложений перестают улучшать взаимодействие. В этот момент энергию следует направить на решение других проблем, например, внешних интерфейсов, бизнес-правил и т.д. Это является частью оставшихся требований.

Подходящие вариации

При сохранении основных достоинств открывается ряд приемлемых вариаций.

НУМЕРОВАННЫЕ ШАГИ И ПРОСТЫЕ АБЗАЦЫ. Некоторые авторы нумеруют шаги, чтобы обращаться к ним в разделах расширения. Другие пишут альтернативы в виде простых абзацев. Оба способа возможны.

СВОБОДНАЯ И ПОЛНАЯ ФОРМЫ. Иногда стоит тратить силы на создание детальных функциональных требований, а в других случаях этого делать не стоит (см. разделы 1.2 и 1.5). Это до такой степени верно, что я даже не рекомендую больше ни одного шаблона, но всегда показываю как произвольную, так и полную версию. Разные авторы предпочитают разные версии. Каждый работает по-своему. Сравните варианты использования 25 и 5.

ПРЕДВАРИТЕЛЬНОЕ МОДЕЛИРОВАНИЕ БИЗНЕС-ПРОЦЕССОВ С ПОМОЩЬЮ ВАРИАНТОВ ИСПОЛЬЗОВАНИЯ ИЛИ БЕЗ НИХ. Некоторые любят документировать или анализировать бизнес-процессы, прежде чем писать функциональные требования к системе. Одни выбирают варианты использования для описания бизнес-процессов, другие отдают предпочтение иным формам их моделирования. С точки зрения функциональных требований к системе, похоже, нет большой разницы в нотациях для моделирования бизнес-процесса.

ДИАГРАММЫ ВАРИАНТОВ ИСПОЛЬЗОВАНИЯ И СПИСКИ ДЕЙСТВУЮЩЕЕ ЛИЦО/ЦЕЛЬ. Некоторые специалисты с помощью списков *Действующее лицо/Цель* показывают разрабатываемый набор вариантов использования; другие предпочитают диаграммы вариантов использования. Последние, отображая основных действующих лиц и соответствующие варианты использования уровня цели пользователя, могут иметь то же назначение, что и список *Действующее лицо/Цель*.

ВАРИАНТЫ ИСПОЛЬЗОВАНИЯ ТИПА "ПРОЗРАЧНОГО ЯЩИКА" И КООПЕРАТИВНЫЕ ДИАГРАММЫ. Существует почти полное соответствие между вариантами использования типа "прозрачного ящика" и кооперативными диаграммами UML. Варианты использования можно рассматривать как текстовую форму кооперативных диаграмм. Различие в том, что кооперативные диаграммы не описывают внутренних действий компонентов, что доступно варианту использования.

Неподходящие варианты

ПРЕДЛОЖЕНИЯ С ЕСЛИ ВНУТРИ ОСНОВНОГО СЦЕНАРИЯ. Если бы в варианте использования было только одно ветвление поведения, проще было бы его оставить в основном тексте. Однако в вариантах использования ветвлений много, и читатели теряют нить сюжета. Те, кто употребляют предложения с если, признаются, что вскоре приходят к форме основного сценария, за которым следуют расширения.

ДИАГРАММЫ ПОСЛЕДОВАТЕЛЬНОСТИ КАК ЗАМЕНА ТЕКСТА ВАРИАНТА ИСПОЛЬЗОВАНИЯ. Некоторые инструменты разработки программного обеспечения претендуют на поддержку вариантов использования на том основании, что создают диаграммы последовательности. Последние тоже показывают взаимодействия действующих лиц, но имеют следующие недостатки:

- Они не фиксируют внутренние действия (необходимые, чтобы продемонстрировать, как система защищает интересы участников).
- Их намного труднее читать (используется специальная нотация, которая к тому же занимает намного больше места).
- Практически невозможно втиснуть необходимый объем текста над стрелками между действующими лицами.
- Большая часть инструментов вынуждает автора прятать текст за всплывающим окном диалога, что мешает следовать за сюжетом.
- Большая часть инструментов вынуждает автора описывать каждый альтернативный путь независимо, возвращаясь каждый раз к началу варианта использования. Это утомительно, вызывает ошибки, а читателю трудно разобраться, чем отличается поведение в каждой вариации.

Диаграммы последовательности — не слишком хорошая форма представления. Ее предпочитают в связи с возможностями инструмента автоматически создавать перекрестные ссылки, гиперссылки в прямом и обратном направлении и глобально изменять имена. Несмотря на то что эти функции выполняются хорошо (и существует недостаток текстовых инструментов), большинство авторов и читателей согласны, что этого недостаточно, чтобы жертвовать удобством написания и читабельностью.

ГРАФИЧЕСКИЙ ИНТЕРФЕЙС ПОЛЬЗОВАТЕЛЯ В ФУНКЦИОНАЛЬНЫХ СПЕЦИФИКАЦИЯХ. Не нужно большого мастерства, чтобы писать требования, не определяя деталей интерфейса пользователя наряду с необходимой функцией. Этому нетрудно обучиться. Большинство заинтересованных лиц выступает против описания интерфейса пользователя в вариантах использования (см. раздел 19.6 и книгу *Designing Software for Use*, Costantine и Lockwood, 1999).

Памятка 15. Качество набора вариантов использования

У меня есть только три вопроса относительно качества полного набора вариантов использования:

- Формулируют ли варианты использования сюжет, который развертывается от цели самого высокого уровня до цели самого низкого уровня?
- Существует ли на границе области действия проектирования вариант использования самого высокого уровня, устанавливающий контекст (возможно, для каждого основного действующего лица)?
- Это все, что необходимо разработать? (Вопрос предназначен заказчикам и пользователям.)

Глава 22

Памятки для работы над вариантами использования

Памятка 16. Это только часть требований

Варианты использования — это лишь небольшая часть общей работы по сбору требований, возможно, одна из глав описания требований. Они находятся в центре этой работы, связывая определения данных, бизнес-правила, проект интерфейса пользователя, модель предметной области и т.д. Однако варианты использования представляют не все требования, а только те, что описывают поведение.

Это необходимо иметь в виду, так как некоторые группы пытаются втиснуть в варианты использования все множество требований.

Памятка 17. Сначала работа должна быть направлена в ширину

Первоначальный охват должен быть широким, а не глубоким, от более низкой точности к более высокой. Круг работы расширяется с повышением точности (см. рис. 22.1), так что это поможет вам дозировать энергию (см. раздел 1.5). Порядок должен быть следующим:

1. *Основные действующие лица.* Сначала перечислите всех действующих лиц, чтобы охватить на какое-то время всю систему. Чаще всего системы настолько необъятны, что вы вскоре потеряете все ориентиры, поэтому стоит взглянуть на систему в общем в начале работы. Организация "мозгового штурма" для выявления этих действующих лиц поможет вам получить большинство целей уже в первом цикле.

2. *Цели.* Перечисление всех целей всех основных действующих лиц — это, может быть, ваш последний шанс охватить всю систему. Сделайте этот список максимально полным и правильным. Следующие шаги вовлекут бо-

льше людей и значительно увеличат объем работы. Проанализируйте список с пользователями, организаторами и разработчиками, чтобы они согласились с приоритетами и поняли разрабатываемую систему.

<table>
<tr><td rowspan="18">Действующее лицо</td><td rowspan="9">Цель</td><td rowspan="6">Действие из успешного сценария</td><td rowspan="3">Условие неудачи</td><td>Действие по восстановлению</td></tr>
<tr><td>Действие по восстановлению</td></tr>
<tr><td>Действие по восстановлению</td></tr>
<tr><td rowspan="3">Условие неудачи</td><td>Действие по восстановлению</td></tr>
<tr><td>Действие по восстановлению</td></tr>
<tr><td>Действие по восстановлению</td></tr>
<tr><td rowspan="6">Действие из успешного сценария</td><td rowspan="3">Условие неудачи</td><td>Действие по восстановлению</td></tr>
<tr><td>Действие по восстановлению</td></tr>
<tr><td>Действие по восстановлению</td></tr>
<tr><td rowspan="3">Условие неудачи</td><td>Действие по восстановлению</td></tr>
<tr><td>Действие по восстановлению</td></tr>
<tr><td>Действие по восстановлению</td></tr>
<tr><td rowspan="9">Цель</td><td rowspan="6">Действие из успешного сценария</td><td rowspan="3">Условие неудачи</td><td>Действие по восстановлению</td></tr>
<tr><td>Действие по восстановлению</td></tr>
<tr><td>Действие по восстановлению</td></tr>
<tr><td rowspan="3">Условие неудачи</td><td>Действие по восстановлению</td></tr>
<tr><td>Действие по восстановлению</td></tr>
<tr><td>Действие по восстановлению</td></tr>
<tr><td rowspan="6">Действие из успешного сценария</td><td rowspan="3">Условие неудачи</td><td>Действие по восстановлению</td></tr>
<tr><td>Действие по восстановлению</td></tr>
<tr><td>Действие по восстановлению</td></tr>
<tr><td rowspan="3">Условие неудачи</td><td>Действие по восстановлению</td></tr>
<tr><td>Действие по восстановлению</td></tr>
<tr><td>Действие по восстановлению</td></tr>
</table>

Рис. 22.1. *С повышением точности увеличивается объем работы*

3. *Основной сценарий.* Основной сценарий обычно бывает коротким и довольно простым. Он повествует об услугах, предоставляемых системой. Обеспечьте, чтобы писатели один раз показали, как работает система в случае успеха, прежде чем исследовать пути, которые могут привести к неудаче.

4. *Условия неудачи (расширения).* Зафиксируйте все условия расширения, прежде чем думать, как их обрабатывать. Это достигается путем "мозгового штурма", который отличается от исследования и описания шагов обра-

ботки расширений. Построенный список служит рабочей таблицей для авторов, которые потом могут делать перерывы, не боясь потерять место, где остановились. Те, кто хочет обработать каждое условие по мере его выявления, обычно никогда не заканчивают список отказов. Они выдыхаются после нескольких сценариев неудачи.

5. *Шаги восстановления.* На шагах восстановления часто обнаруживаются новые цели пользователя, новые действующие лица и новые условия неудачи. Запись этой информации — самое трудное в создании варианта использования, поскольку автор сталкивается с вопросами деловой политики, которые часто оставляют без внимания. Когда я обнаруживаю малопонятную политику, новое действующее лицо или новый вариант использования во время работы над шагами восстановления, я чувствую, что все мои усилия были не напрасны.

6. *Поля данных.* Хотя формально описание данных далеко от процесса создания вариантов использования, разработчики часто получают задание развертывать имена данных (например, "данные о клиенте") в списки полей данных (см. раздел 16.1).

7. *Атрибуты и проверки полей данных.* В некоторых случаях, пока писатели анализируют варианты использования, другие разрабатывают атрибуты и проверки полей данных. Описание атрибутов представляет собой форматы данных конечного уровня точности. Эти атрибуты и проверки неуместны в варианте использования, но они должны быть в конце концов разработаны.

Памятка 18. Рецепт из 12 шагов

1. Найдите границы системы (контекстная диаграмма, список ввода/вывода).
2. Методом "мозгового штурма" составьте список основных действующих лиц (таблица профилей действующих лиц).
3. Методом "мозгового штурма" составьте список целей основных действующих лиц в отношении системы (список Действующее лицо/Цель).
4. Напишите предельные варианты использования обобщенного уровня, охватывающие все вышеназванные составляющие.
5. Пересмотрите и модифицируйте стратегические варианты использования. Добавьте, удалите и объедините цели.
6. Выберите вариант использования для расширения или напишите повествование, чтобы познакомиться с материалом.
7. Запишите участников, интересы, предусловия и гарантии. Дважды проверьте.

8. Напишите основной сценарий; проверьте, соблюдаются ли в нем интересы и гарантии.
9. Методом "мозгового штурма" составьте список возможных условий неудачи и альтернативных условий успеха.
10. Опишите поведение действующих лиц и системы в каждом расширении.
11. Выделите каждый подчиненный вариант использования, которому необходимо собственное пространство.
12. Начните с вершины и уточните созданные варианты использования. Произведите надлежащие добавления, удаления и объединения. Дважды проверьте завершенность, удобочитаемость и условия неудачи.

Памятка 19. Не забывайте, во что обходятся ошибки

Чего вам будет стоить низкое качество варианта использования зависит от вашей системы и проекта. Некоторые проекты почти не нуждаются в качественных формулировках требований вследствие превосходно налаженного взаимодействия пользователей и разработчиков.

> Проектная группа компании Chrysler Comprehensive Compensation, разрабатывая программное обеспечение для расчета зарплаты с помощью методов eXtreme Programming (Beck, 1999), никогда не выходила за пределы кратких описаний вариантов использования. Они писали так мало, что называли свои произведения "рассказами", а не вариантами использования, и помещали их на учетных карточках. При этом был необходим разговор с экспертом по требованиям и разработчиком. четырнадцать членов группы работали в двух соседних комнатах и имели возможность общения внутри группы.

Чем лучше внутреннее взаимодействие экспертов по использованию и разработчиков, тем меньше значат пропущенные части шаблона варианта использования. Люди решают вопросы, просто беседуя друг с другом. Если вы работаете с распределенной группой, с несколькими группами-подрядчиками или с очень большой группой, а также при работе над жизненно важными системами, низкое качество обходится дороже. Если правильное описание функций системы жизненно важно, необходимо очень серьезно отнестись к участникам и интересам, предусловиям и минимальным гарантиям.

Определите важность вашего проекта. Не стоит слишком тщательно выискивать мелкие ошибки небольшого и несложного проекта, но если последствия ошибок значительны, к проектированию следует отнестись со всей строгостью.

Памятка 20. Лучше меньше, но понятней

Как ни странно это звучит, вы, скорее всего, понесете меньший ущерб, если напишете слишком мало, а не слишком много. Когда есть сомнения, пишите меньше, используя

цели более высокого уровня, с меньшей точностью и в простой повествовательной форме. Тогда у вас будет короткий и ясный документ, который люди быстро прочитают и зададут по нему вопросы. Это поможет выяснить, что вы пропустили.

Противоположная стратегия приводит к неудаче. Если вы написали около сотни очень подробных вариантов использования низкого уровня, их мало кто прочтет, и обсуждения в группе не будет. Написание на слишком низком уровне цели — распространенный недостаток программистов.

▪ Невымышленная история

Работая в успешном проекте на $15 млн, в котором участвовало 50 человек, мы написали только основной сценарий и обработку нескольких ошибок просто в виде текста с абзацами. Этого было достаточно, поскольку у нас были отличные условия взаимодействия. Каждый автор требований работал в группе с двумя-тремя разработчиками-программистами. Они сидели по соседству либо посещали друг друга несколько раз в день.

На самом деле повышение качества взаимодействия внутри группы помогает в каждом проекте. Стратегия организации описанной выше группы — это образец Целостного многообразия (Holistic Diversity) из книги *Surviving Object-Oriented Projects* (Cockburn, 1998).

Памятка 21. Обработка ошибок

Одним из достоинств вариантов использования является то, что они называют все условия расширения. Во многих проектах программисты пишут

```
If <условие>
    then <сделать это>
else ..?
```

Здесь он останавливается, размышляя над else, так как в требованиях ничего не говорится об этом странном условии. Затем он быстренько записывает в программу нечто вроде:

```
else <сделать то>
```

Обработку предложения с если, конечно, следовало бы внести в документ о требованиях. Очень часто она включает важные бизнес-правила. Эксперты по использованию часто собираются специально, чтобы выяснить, что должна делать система в этих ситуациях.

Выявление условий отказа и написание обработки ошибок часто открывают новых действующих лиц, новые цели и новые бизнес-правила. Порой это требует некоторого исследования либо изменяет сложность системы.

Если вы создаете только основной сценарий, попробуйте охватить неудачи и обработку ошибок в следующих вариантах использования. Вы, вероятно, будете удивлены и воодушевлены тем, что вам откроется.

Памятка 22. Первоначальные и конечные названия работ

Названия важны в начальной и в конечной стадиях работы над проектом, но не в середине деятельности.

В начале работы вам необходимо собрать все цели, которые должна удовлетворять система, и представить их в удобочитаемой форме. Фокусирование на названиях работ или социальных ролях, на которые будет воздействовать система, позволяет эффективно провести "мозговой штурм" и выполнить наилучшим образом первый проход выявления целей. Когда длинный список целей будет готов, названия работ обеспечат также схему группирования, облегчающую анализ и расстановку приоритетов обозначенных целей.

Названия работ дают представление о навыках и стилях, необходимых для различных работ. Эта информация полезна для проектирования интерфейса пользователя.

Когда начинается разработка вариантов использования, в процессе обсуждений обнаруживается, как перекрываются роли. Названия ролей, употребляемые для основных действующих лиц, становятся более обобщенными (например, "заказчик") или более отдаленными от людей, которые будут на самом деле пользоваться системой. Пока они только "заглушки", напоминающие писателям, что на самом деле есть действующее лицо, имеющее цель.

Когда система вводится в эксплуатацию, названия работ опять приобретают важность. Группа разработчиков должна:

- Назначить уровни доступа определенным пользователям для обновления или, возможно, только чтения каждого вида данных.
- Подготовить материалы по обучению работе с новой системой на основе набора приемов для этих названий и определить, какие варианты использования будет применять каждая группа.
- Упаковать систему для установки, разложив реализации вариантов использования по кластерам.

Памятка 23. Действующие лица играют роли

Термин *"действующее лицо"* означает либо название работы индивидуума, применяющего систему, либо роль, которую этот индивидуум играет при использовании системы. Не столь важно, как мы употребляем термины, так что не тратьте слишком много сил на определение различий.

Важнейшим компонентом является цель, которая указывает, что собирается делать система. Тот, кто обращается к этой цели, будет объектом обсуждений и изменений на весь период жизни системы. Обнаружив, что директор магазина может действовать и как продавец, вы сможете:

- Написать в варианте использования, что основным действующим лицом является "продавец либо директор магазина". (Приверженцам UML: нарисуйте стрелки к эллипсу от обоих действующих лиц.)

- Написать, что при выполнении этого варианта использования директор магазина может выступать в роли продавца. (Приверженцам UML: нарисуйте стрелку обобщения от директора магазина к продавцу.)
- Сделайте "заказчика" основным действующим лицом. Напишите, что при выполнении этого варианта использования продавец или директор магазина выступает в роли заказчика. (Приверженцам UML: нарисуйте стрелку обобщения от продавца и директора магазина к заказчику.)

Все формулировки верны, так что можете выбрать любую.

Индивидуум исполняет много ролей, индивидуум в действии — это просто лицо, исполняющее роль. Лицо, имя которого — это название работы, выступает во многих ролях, даже если действует в роли, соответствующей названию этой работы. Для варианта использования важно не имя основного действующего лица, а цель. В проектной группе полезно установить соглашение о поддержке непротиворечивости в употреблении названий работ.

Как имена действующих лиц меняются на названия ролей и отображаются обратно в имена действующих лиц, см. в разделе 4.2, подраздел о несущественных и существенных действующих лицах, а также в памятке 22.

Памятка 24. Большая графическая мистификация

С тех пор как вышла книга *Object-Oriented Software Engineering* (1993) многие авторы при создании вариантов использования сфокусировали внимание на фигурах из палочек и эллипсах, упуская из виду, что варианты использования — это текстовая форма представления. Развитая индустрия CASE-средств, которая уже располагала графическими, а не текстовыми средствами моделирования, ухватилась за этот факт и представила инструменты, способные довести количество рисунков в вариантах использования до максимума. Эта ситуация не была исправлена в стандарте Unified Modeling Language, принятом OMG. Как мне объяснили, UML — это лишь стандарт обмена между различными инструментами. Следовательно, текст, прячущийся каким-то образом за каждым эллипсом, не входит в стандарт и является частным делом каждого автора.

Сейчас очень многие думают, что эллипсы и есть варианты использования, даже учитывая, что они несут очень мало информации. Опытные разработчики могут отнестись к этому довольно саркастически. Выражаю благодарность Энди Ханту и Дэйву Томасу за их удачную шутку по поводу взгляда на варианты использования как на "легко полученные требования" (см. рис. 22.2; из книги *The Pragmatic Programmer*, 1999).

Важно сознавать, что эллипсы никак не могут заменить текст. Диаграмма вариантов использования (намеренно) опускает упорядочение, данные и скрывает действующее лицо. Она применяется как:

- Оглавление вариантов использования
- Контекстная диаграмма системы, показывающая, как действующие лица добиваются различных и перекрывающихся целей.

- "Большая картина", показывающая отношение между вариантами использования более высокого уровня и более низкого уровня.

Варианты использования — это в основном текстовая форма, поэтому применяйте эллипсы для наглядности, а не в качестве замены. На рис. 22.3 и в таблице 22.1 показаны два способа представления контекстной диаграммы. В таблице содержатся те же действующие лица и цели, причем общие варианты использования для ясности повторяются.

Рис. 22.2. *"Мама, я хочу домой"*

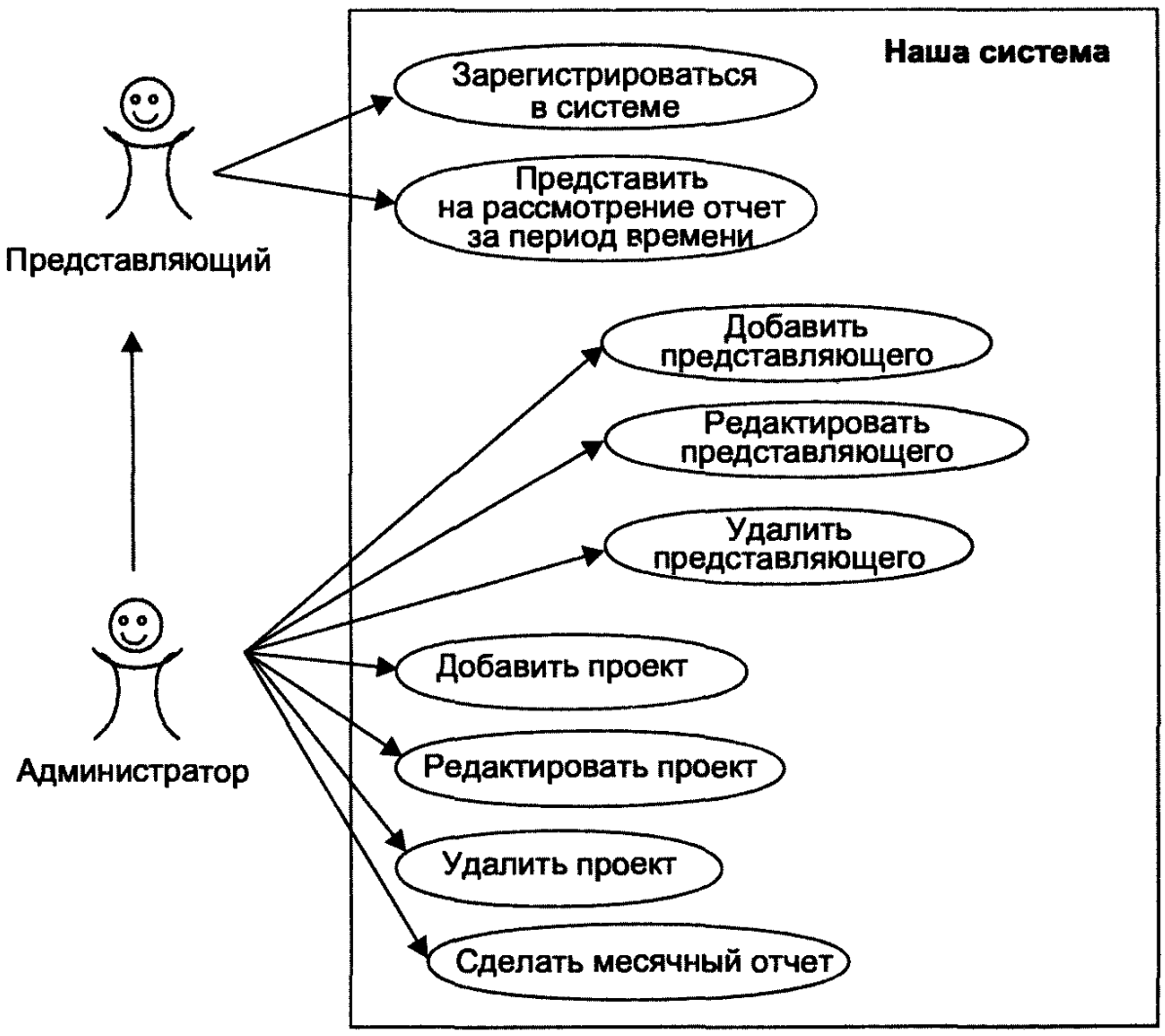


Рис. 22.3. *Контекстная диаграмма с использованием эллипсов (адаптирована из работы Гради Буча)*

Таблица 22.1. Список Действующее лицо/Цель для контекстной диаграммы

Действующее лицо	Цель
Представляющий	Зарегистрироваться в системе
	Представить на рассмотрение отчет за период времени
Администратор	Зарегистрироваться в системе
	Представить на рассмотрение отчет за период времени
	Добавить представляющего
	Редактировать представляющего
	Удалить представляющего
	Добавить проект
	Редактировать проект
	Удалить проект
	Сделать месячный отчет

Памятка 25. Дебаты об инструментах

К сожалению, на сегодняшнем рынке инструментов ни один не поддерживает варианты использования в надлежащем виде. Многие компании заявляют, что поддерживают их в текстовом либо в графическом формате. Однако ни один из их инструментов не содержит метамодели, близкой к описанной в разделе 2.3, и с большинством довольно трудно работать. В результате создателю вариантов использования выбирать особенно не из чего.

LOTUS NOTES. Он является моим любимым инструментом, однако не имеет метамодели вариантов использования, зато поддерживает совместную работу, гиперссылки, общий шаблон, историю документа, быстрый просмотр с редактированием всего набора вариантов использования и легко конструируемые упорядоченные представления. Все это важные достоинства. Lotus Notes также позволяет держать расширяющиеся описания данных в одной базе данных, но в различных представлениях. Когда вы обновляете шаблон, изменяются все варианты использования в базе данных. Шаблон довольно легко настраивать и чрезвычайно просто применять. С помощью Lotus Notes я просмотрел свыше 200 вариантов использования с заказчиками в связи с заявкой на проект с фиксированной стоимостью.

Недостатком Lotus Notes, как и всех инструментов для текстов в свободной форме, является то, что перенумерация шагов и расширений вскоре становится помехой. Вставляемые вручную обратные связи быстро устаревают. Не автоматизирована работа с обратными связями и гиперссылками, поэтому нельзя сказать, какой вариант использования вызывает тот вариант, на который вы смотрите.

Мне больше всего нравится в Lotus Notes легкость применения в сочетании с тем, как аннотированный список Действующее лицо/Цель обеспечивает динамическое генерирование представления набора вариантов использования. Просто создайте новый вариант использования, и он моментально появится в представлении.

Представление — это одновременно оглавление с гиперссылками, список Действующее лицо/Цель и таблица отслеживания хода работ. Мне нравятся представления вариантов использования по приоритетам, версиям, состоянию завершенности, названию, основному действующему лицу, предметной области, уровню и названию.

ТЕКСТОВЫЕ ПРОЦЕССОРЫ С ГИПЕРССЫЛКАМИ. С появлением гиперссылок текстовые процессоры стали, наконец, приспособленными для работы с вариантами использования. Поместите шаблон варианта использования в файл шаблонов. Каждый вариант использования запишите с помощью шаблона в отдельный текстовый файл, и без проблем устанавливайте связи между вариантами использования. Только не изменяйте имена файлов! Авторы обычно знакомы с текстовыми процессорами, и им удобно создавать описания с их помощью.

Текстовым процессорам присущи все недостатки Lotus Notes. Кроме того, у них нет возможности создавать список всех вариантов использования, отсортированный по версии или статусу, и открывать их щелчком мыши. Это означает, что нужно генерировать и поддерживать отдельный список для просмотра, который быстро устаревает. Не существует глобального механизма обновления шаблона, и поэтому со временем версии шаблона накапливаются.

РЕЛЯЦИОННЫЕ БАЗЫ ДАННЫХ. Я знаю о нескольких попытках создать модель действующих лиц, целей и шагов в реляционной базе данных, такой как Microsoft Access. Хотя это желание естественно, инструменты неудобны в применении, и разработчики вариантов использования возвращаются к своим текстовым процессорам.

СРЕДСТВА УПРАВЛЕНИЯ ТРЕБОВАНИЯМИ. Специализированные средства управления требованиями, такие как DOORS и Requisite Pro, получают все большее распространение. Они обеспечивают поддержку прямых и обратных гиперссылок и предназначены для текстовых описаний требований. Недостатком является отсутствие (насколько мне известно) поддержки модели основного сценария и расширений, т.е. ядра вариантов использования. Несколько вариантов использования, которые я видел и которые были получены с помощью таких средств, были очень длинными, с многочисленными отступами, обилием нумерации и строк, что затрудняло их чтение (см. памятки 2 и 20). Если вы применяете подобный инструмент, сделайте так, чтобы сюжет был ясно виден.

CASE-СРЕДСТВА. CASE-средства поддерживают глобальные изменения любой составляющей метамодели и обратные связи. Однако, как описано выше, они склонны к использованию блоков и стрелок в ущерб тексту. Диаграммы последовательности не являются приемлемой заменой текстовым вариантам использования, и большая часть CASE-средств предлагает лишь диалоговое окно для ввода текста. Я был свидетелем того, как группа разработчиков вариантов использования взбунтовалась, оставила CASE-средства и вернулась к текстовому редактору.

Это оставляет вам, мягко говоря, небольшой выбор. Удачи.

Памятка 26. Планирование проекта с помощью названий и кратких описаний

Плюсы и минусы применения вариантов использования для отслеживания продвижения проекта приведены в разделе 17.1. Там же вы найдете пример списка Действующее лицо/Цель в качестве основы для планирования проекта. Вот две памятки.

ТАБЛИЦА ПЛАНИРОВАНИЯ ВАРИАНТА ИСПОЛЬЗОВАНИЯ. В первых двух столбцах таблицы запишите действующих лиц и цели, а в следующих столбцах — необходимые вам атрибуты из следующего списка: ценность для бизнеса, сложность, версию, группу, завершенность, требование к производительности, внешние интерфейсы и т.д.

С помощью этой таблицы ваша группа может договариваться о реальном приоритете разработки для каждого варианта использования, обсуждать деловые потребности и технические трудности, а также определять очередность разработки.

ВЫПУСК НЕПОЛНЫХ ВАРИАНТОВ ИСПОЛЬЗОВАНИЯ. Как описано в подразделе об управлении пересекающимися версиями вариантов использования (см. раздел 17.1), вам придется довольно часто принимать решение о выпуске только части варианта использования в определенной версии. В большинстве групп просто применяют выделение желтым цветом или жирным шрифтом, чтобы обозначать фрагменты варианта использования. Отметьте в таблице планирования первую версию выпущенного варианта использования и его окончательную версию, которая представит этот вариант использования в полном виде.

Приложения

Приложение А

Варианты использования на языке UML

Унифицированный язык моделирования UML определяет набор используемых графических нотаций. Он не касается содержания варианта использования и стиля изложения, но создает значительные сложности при их обсуждениях. Лучше научиться писать ясные тексты. Если вам нравятся диаграммы, освойте основы отношений и затем введите несколько простых стандартов, чтобы диаграммы были понятны.

А.1. Эллипсы и фигуры из палочек

Если вы хотите графически изобразить, как люди применяют систему, вполне естественно нарисовать пользователя в виде фигурки из палочек, а варианты использования, которые он вызывает, в виде эллипсов или прямоугольников. Пометьте фигурку из палочек именем действующего лица, а эллипсы названиями вариантов использования. Информация та же, что в списке Действующее лицо/Цель, но представлена по-другому. Эта диаграмма подойдет в качестве оглавления.

Пока все очевидно.

Проблема возникает, когда авторы или читатели решают, что диаграммы определяют функциональные требования к системе. Некоторые становятся фанатиками диаграмм, думая, что смогут легко выполнить трудную работу (как на рис 22.1). Они стараются втиснуть в диаграмму как можно больше информации, возможно, надеясь, что писать текст уже не придется. Ниже описано, что происходит в этой ситуации.

> Не так давно студент моего курса развернул связанную шнуром диаграмму длиной больше метра с эллипсами, указывающими во все стороны стрелками и с перемешанными повсюду связями *включения*, *расширения* и *обобщения* (различаемыми, конечно, только по мелким текстовым меткам над каждой стрелкой). Он хотел выяснить, все ли связи употреблены в его проекте

правильно. Он не подозревал, что в результате невозможно было понять, что должна делать его система.

Другой с гордостью демонстрировал, как он "исправил" очевидный дефект диаграммы, которая не показывала порядок вызова вариантов использования. Он добавил еще стрелок, направленных на предшествующий вариант использования, с помощью связи предшествования UML. В результате, конечно, получилась чрезвычайно запутанная диаграмма, которая заняла больше места, чем эквивалентный текст, и в которой труднее было разобраться. В данном случае графика не сделала вариант использования ни лаконичным, ни наглядным, скорее наоборот.

Рисунок — это двумерная мнемоническая схема, которая служит выделению связей. Применяйте его для данной задачи, а не заменяйте им текст.

Имея в виду это предназначение, рассмотрим отдельные связи в UML.

A.2. Связь включения

Основной вариант использования "*включает*" "*включенный*" вариант использования, если шаг действия основного варианта использования вызывает по имени включенный. Это обычные и очевидные связи между вариантами использования более высокого и более низкого уровней. Включенный вариант использования описывает цель уровнем ниже по сравнению с основным.

Глагольная группа в шаге действия может служить названием подчиненного варианта использования. Если вы никогда не выделите эту цель, это будет просто шаг. А если вы выделяете эту цель в самостоятельный вариант использования, этот шаг вызывает подчиненный вариант использования (в моей терминологии) или он включает поведение включенного варианта использования (в терминах текущей версии UML 1.3). До UML 1.3 это называлось "применять" (use) вариант использования более низкого уровня (ныне это выражение устарело).

Штриховая линия со стрелкой идет от основного (более высокого уровня) варианта использования к включенному варианту использования, обозначая, что основной вариант использования "знает" о включенном, как показано на рис. A.1.

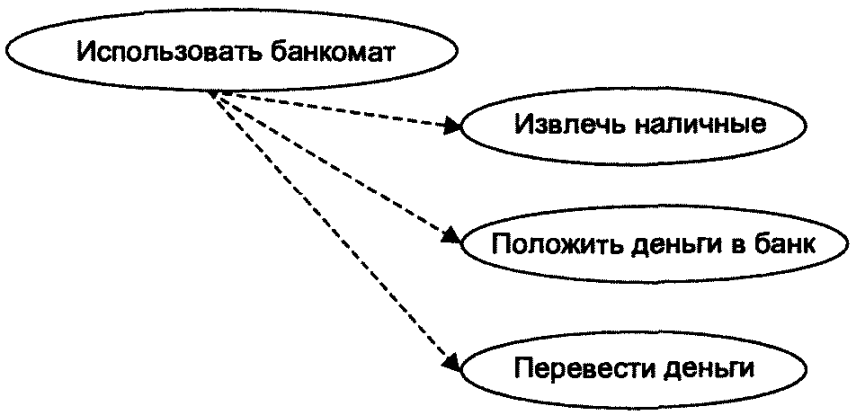


Рис. A.1. *Использование связей включения*

Правило 13. Более высокие цели изображайте выше

Всегда изображайте на диаграмме цели более высокого уровня над целями более низкого уровня. Это уменьшает путаницу с уровнями цели и интуитивно понятно читателям. При этом стрелка от основного варианта использования к включенному всегда будет направлена вниз.

UML позволяет изменять графическое представление каждого своего элемента. Я считаю, что большинство людей, рисуя от руки, изображают стрелку *сплошной* линией от основного варианта использования к включенному (штриховые стрелки требуют больше времени). В графическом редакторе вы будете применять стрелки того стиля, который предоставляет программа.

Для большинства программистов должно быть очевидно, что связь *включения* — это традиционный вызов подпрограммы (процедуры) в переводе на язык программирования. Это естественное применение механизма, который подходит и для программирования, и для повседневной жизни. Иногда это соответствует параметризованным вариантам использования, передаче им значений аргументов функции и даже получению от них возвращаемых значений (см. главу 14). Несмотря на это, имейте в виду, что вариант использования предназначен для взаимодействия с другим лицом, а не с CASE-средством или компилятором.

A.3. Связь расширения

Расширяющий вариант использования или вариант использования *расширения* расширяет основной вариант использования, именуя его и определяя условия, при которых он прерывает основной вариант использования. Последний не называет расширяющий вариант использования. Это полезно, если вы хотите, чтобы основной вариант использования прерывало несколько вариантов использования без обновления основного варианта использования при добавлении каждого нового прерывающего варианта использования (см. раздел 10.2).

С точки зрения поведения, расширяющий вариант использования определяет некоторое внутреннее условие в основном варианте использования и условие запуска. Поведение определяет основной вариант использования, пока не выполнится условие. С этого момента его определяет расширяющий вариант использования. Когда расширяющий вариант использования заканчивается, управление получает основной вариант использования с той точки, где его выполнение прекратилось.

Ребекка Вирс-Брок удачно назвала *расширяющий вариант использования* "заплатой" (patch) на основном варианте использования (для программистов это должно ассоциироваться с программными "заплатами"). Другие программисты понимают это как текстовую версию фиктивной программной команды, оператора come-from).

Мы обычно используем форму *расширения*, когда пишем условия расширения в варианте использования. Вариант использования расширения — это просто условие расширения, обработка которого выделена в самостоятельный вариант использования (см. раздел 10.2). Считайте это расширением сценария, который перерос свой вариант использования и получил собственное место.

По умолчанию связь расширения обозначается в UML штриховой линией со стрелкой (так же, как и связь включения) от расширяющего варианта использования к основному со словом "<extends>" над стрелкой. Я изображаю это с помощью крючка от расширяющего варианта использования назад к основному (см. рис. А.2), чтобы подчеркнуть различие между связями включения и расширения.

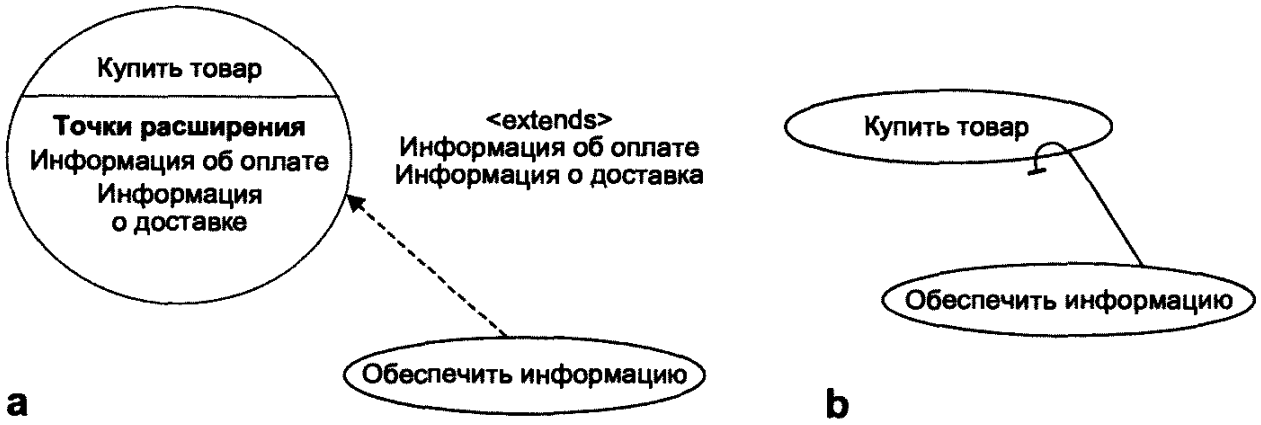


Рис. А.2. *Изображение связи расширения*

На рис. А.2(а) вы видите стандартный способ изображения в UML связи *расширения* (пример взят из книги *UML Distilled* (Fowler, 1999)). На рис. А.2(b) показан коннектор в виде крючка.

Правило 14. Изображайте расширяющие варианты использования ниже

Вариант использования расширения обычно находится уровнем ниже, чем вариант использования, который он расширяет, поэтому на диаграмме он должен быть расположен ниже. Однако в связи расширения именно вариант использования более низкого уровня знает о варианте использования более высокого уровня. Поэтому стрелка или крючок должны быть направлены вверх от расширяющего к основному варианту использования.

Правило 15. Используйте различные формы стрелок

В UML намеренно не решен вопрос о форме стрелок, соединяющих варианты использования. Любую связь можно изобразить с помощью стрелки с открытым острием и небольшим текстом, объясняющим, что это за связь. Идея заключается в том, что если различные поставщики инструментов или проектные группы захотят переделать форму стрелки, стандарт UML не должен этому препятствовать.

Плохо то, что одинаковые стрелки используются для всех связей — это затрудняет просмотр диаграммы. Читатель вынужден изучать мелкий текст, чтобы определить, какие же связи обозначены. Кроме того, их сложно запомнить из-за отсутствия простых визуальных подсказок. Все это, а также отсутствие других соглашений об изображении делает многие диаграммы вариантов использования малопонятными.

По этой причине определите разные формы стрелок, учитывая три вида связи:

- *Включение.* Используйте форму стрелки по умолчанию, с открытым острием, так как она будет употребляться чаще других.
- *Обобщение.* Стандарт стрелки обобщения в UML — стрелка с острием в виде треугольника. Употребляйте именно ее.
- *Расширение.* Создайте совершенно другую форму. Я предпочитаю крючок, направленный от расширяющего к основному варианту использования. Читатели понимают его назначение сразу, он не конфликтует с другими символами UML и привносит собственную метафору расширяющего варианта использования, прицепившегося к основному. Что бы вы ни выбрали, сделайте так, чтобы на диаграмме коннектор расширения отличался от других.

Корректное использование расширения

Создание вариантов использования расширения (см. раздел 10.2, подраздел о вариантах использования расширения) связано в основном с наличием большого количества асинхронных услуг, которые может активизировать пользователь, прерывая основной вариант использования. Часто они разрабатываются разными группами. Эти случаи реализуются в отдельных пакетах программного обеспечения (см. рис. А.3).

Это также происходит, когда вы пишете дополнения к утвержденному описанию требований. При итерационном проектировании вы можете фиксировать требования после каждой сдачи, а затем расширять зафиксированный вариант использования другим, который добавляет функцию.

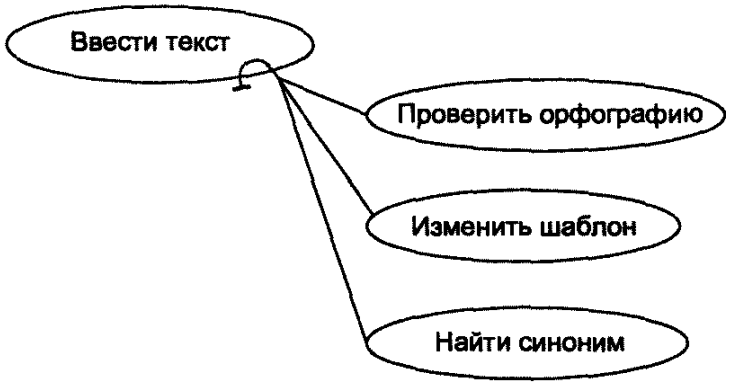


Рис. А.3. *Три прерывающих варианта использования, которые расширяют основной*

Точки расширения

Связь *расширения* была введена прежде всего благодаря практике никогда не трогать документ с требованиями предыдущей системы. В первых телефонных системах, для которых разрабатывались варианты использования, бизнес часто требовал добавления асинхронных услуг, поэтому связь расширения была удобной.

Новая группа могла проектировать на основе надежно зафиксированного документа о требованиях, добавляя требования для новой асинхронной услуги в основном варианте использования в любом месте, где это возможно, не затрагивая набор первоначальных требований к системе.

Однако обращение к поведению в другом варианте использования проблематично. Если нет нумерации строк, как следует обращаться к точке, с которой управление поведением переходит к расширению? А что происходит при наличии нумерации строк, когда основной вариант использования редактируется и номера строк изменяются?

Помните, что номера строк — это на самом деле их метки. Поэтому они не должны быть числовыми или последовательными. Они облегчают чтение и служат для того, чтобы условия расширения имели точку обращения. Однако обычно они являются последовательными числами, и это значит, что со временем они будут изменяться.

Точки расширения были введены, чтобы решить эти вопросы. Точка расширения — это явно видимая метка в основном варианте использования, которая обозначает момент в поведении варианта использования с помощью краткого имени (технически она может обращаться к множеству мест, но сейчас мы это не обсуждаем).

Явно видимые точки расширения влекут новую проблему. Создатели основного варианта использования отвечают за то, где он должен расширяться. Они должны возвращаться и модифицировать его каждый раз, когда кто-то придумает новое место для его расширения. Помните, что первоначально связи расширения использовались во избежание модифицирования основного варианта использования.

Вам придется столкнуться с одной из вышеназванных проблем. Я считаю, что явно объявляемые точки расширения приносят больше вреда, чем пользы. Я предпочитаю описывать в тексте, где в основном варианте использования управление переходит к расширяющему варианту использования, игнорируя краткие имена (см. пример с банкоматом ниже).

Если вы устанавливаете точки расширения, не показывайте их на диаграмме. Они занимают большую часть площади эллипса, поглощая внимание пользователя и затеняя гораздо более важное название цели (см. рис. А.2). Поведение, к которому они обращаются, на диаграмме не показано. Они только увеличивают путаницу.

И еще кое-что о точках расширения. С помощью имени точки расширения разрешается вызвать не одно только место в основном варианте использования, в котором расширяющим вариантам использования необходимо подключить дополнительное описание поведения, а столько, сколько вы пожелаете. Это может понадобиться, скажем, для банкомата при добавлении варианта использования расширения *Использовать банкомат другого банка.* В расширяющем варианте использования необходимо записать:

Перед тем, как выполнить транзакцию, система получает разрешение клиента заплатить за дополнительную услугу.

...

После завершения запрошенной транзакции система вносит на счет клиента оплату дополнительной услуги.

Конечно, вы могли бы просто сказать об этом.

A.4. Связь обобщения

Вариант использования может *специализировать* более общий вариант использования (общий вариант использования обобщает специализированный). (Специализирующий) потомок должен иметь "подобный вид" в отношении (общего) родителя. Точнее, в UML 1.3 говорится: "Связь обобщения между вариантами использования подразумевает, что дочерний вариант использования содержит все атрибуты, последовательность поведения и точки расширения, определенные в родительском варианте использования, а также участвует во всех связях родительского варианта использования".

Корректное использование связи обобщения

Хорошим проверочным словом является *обобщенный* (т.е. определенного типа). Будьте осторожны, когда говорите, что пользователь выполняет некоторую разновидность этого действия. Говоря так, вы получаете кандидата для обобщения.

Вот фрагмент варианта использования Использовать банкомат.

1. **Клиент вводит карточку и PIN.**
2. **Банкомат удостоверяется в правильности номера счета и PIN клиента.**
3. **Клиент выполняет транзакцию, одну из:**
   - **Извлечь наличные**
   - **Внести наличные на счет**
   - **Перевести деньги**
   - **Проверить баланс**

   Клиент выполняет транзакции, пока не выберет пункт меню "завершить сеанс".
4. **Банкомат возвращает карту.**

Что выполняет клиент на шаге 3? Обобщенный ответ — "транзакцию". Для клиента существуют четыре *типа* транзакции. Слова "*обобщенный*" и "*типа*" предупреждают нас о присутствии обобщенной цели **Выполнить транзакцию**. В версии с обычным текстом мы не замечаем, что применяем связь обобщения между вариантами использования. Мы просто перечисляем типы операций или транзакций, которые может выполнить пользователь, и идем дальше. В UML, однако, это служит сигналом для рисования стрелки обобщения.

На самом деле здесь две возможности. Можно игнорировать *обобщение* и просто включить определенные операции, как показано на рис. A.4(a). Можно также создать общий вариант использования *Выполнить одну транзакцию* и представить определенные операции в качестве его специализаций, как на рис. A.4(b).

Используйте то, что вам нравится. Работая с текстом, я не создаю обобщенных вариантов использования. Редко когда бывает нужно вставить какой-нибудь текст в обобщенную цель, поэтому нет необходимости создавать для него новую страницу варианта использования. Однако графически выразить "выполнить одну из следующих транзакций" нельзя, поэтому вам придется найти и назвать обобщающую цель.

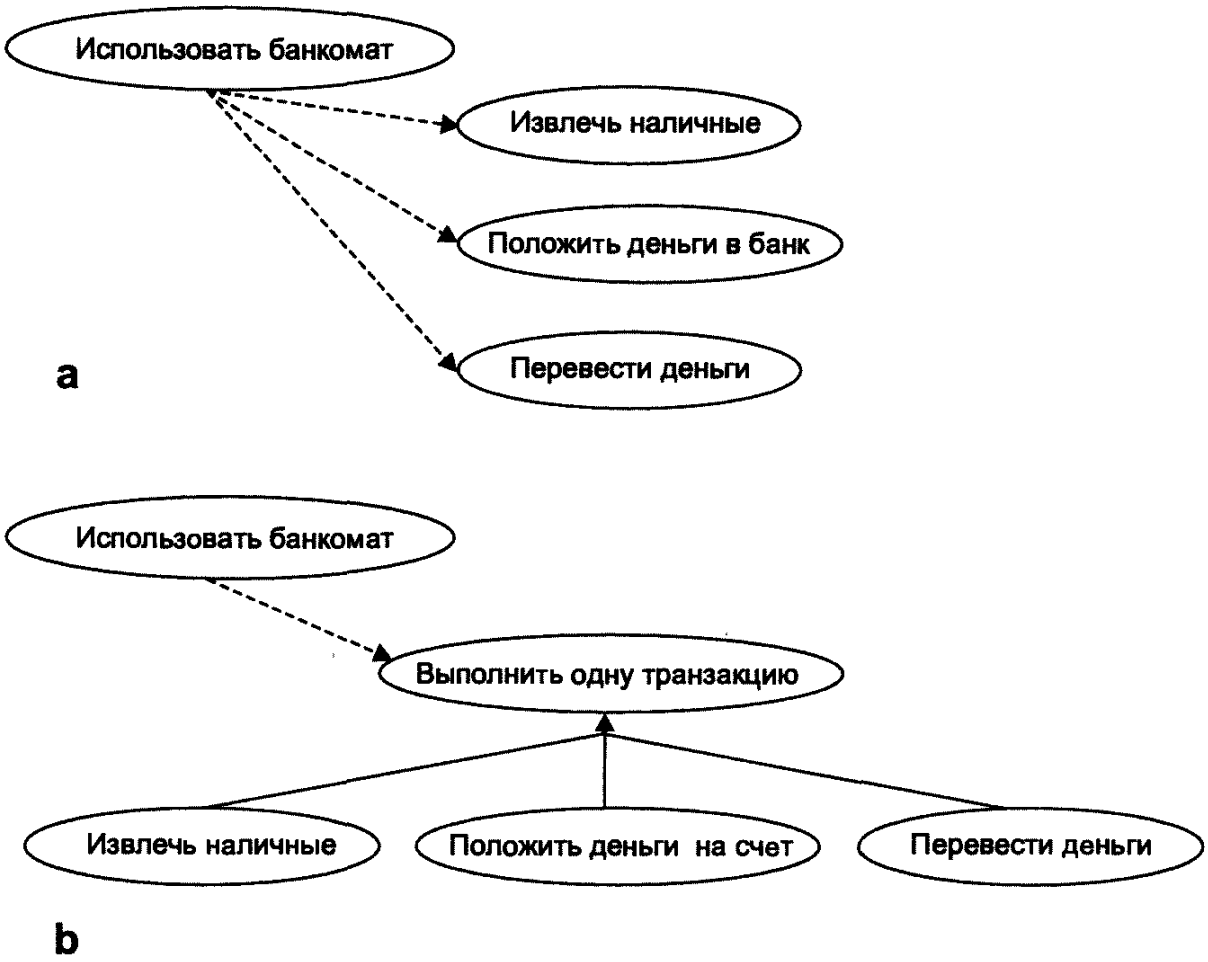


Рис. А.4. *Изображение обобщения; набор включенных вариантов использования превращается в специализации обобщенного действия*

Правило 16. Общие цели изображайте выше

На диаграмме всегда изображайте общие цели выше. Треугольное острие стрелки должно указывать вверх и в дно эллипса общей цели, а не в его бока (см. рис. А.4).

Опасности обобщения

Будьте осторожны при сочетании специализации действующих лиц со специализацией вариантов использования. Идиома во избежание этого — *специализированное действующее лицо, применяющее специализированный вариант использования*. На рис. А.5 отображена ситуация: служащий отдела продаж может заключить любую сделку, но заключить сделку сверх определенного лимита может служащий отдела продаж специального типа, старший агент. Тем не менее в действительности это выражает противоположное.

В разделе 4.2 говорится, что специализированное действующее лицо может выполнять любой вариант использования, который может выполнять главное действующее лицо. Таким образом, служащий отдела продаж — это обобщенный старший агент. Многим людям это кажется трудным для понимания, но это соответствует форме и вполне корректно.

Другая специализация представляется вполне естественной: заключение большой сделки — специальный случай заключения обычной сделки. Однако согласно правилу UML, *специализированный вариант использования можно подставить везде, где упоминается общий вариант использования*. Поэтому на рисунке показано, что обычный служащий отдела продаж может заключить большую сделку!

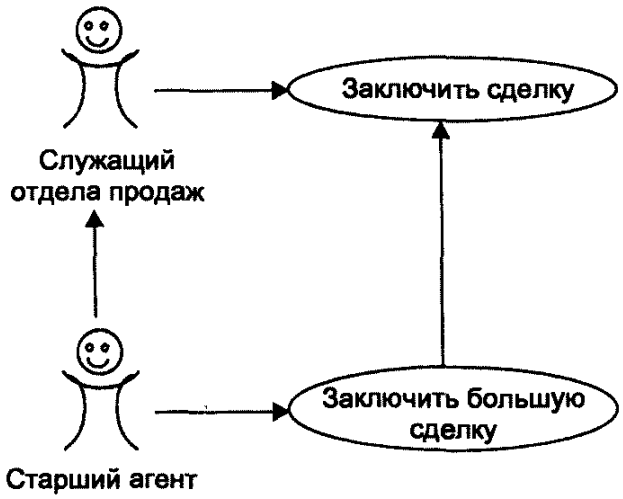


Рис. А.5. *Рискованное обобщение — заключение большой сделки*

Скорректированный вариант показан на рис. А.6. Вы можете поинтересоваться, *специализирует* ли на самом деле заключение небольшой сделки основную сделку, или *расширяет* ее. Поскольку при работе с текстовыми вариантами использования таких загадок не возникает, а искать разгадку расточительно, оставляю этот вопрос в качестве упражнения для заинтересованных читателей.

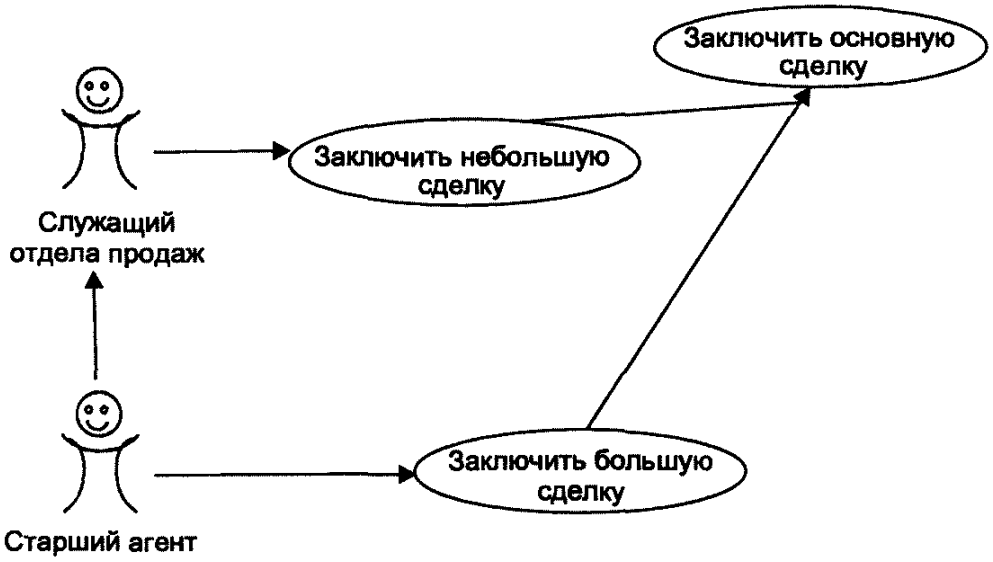


Рис. А.6. *Правильное заключение большой сделки*

Вообще говоря, проблема с обобщением заключается в том, что профессионалы еще не договорились, что означает определить подтип и специализировать поведе-

ние, т.е. какие подразумеваются свойства и возможности. Так как варианты использования описывают поведение, не может быть стандартного определения, что означает их специализировать.

Если вы все-таки используете обобщение, советую сделать обобщенный вариант использования пустым, как Выполнить одну транзакцию в вышеприведенном примере. Далее специализирующий вариант использования будет определять все поведение и вам будет угрожать лишь одна только что описанная ловушка.

A.5. Зависимые и подчиненные варианты использования

В расширенном текстовом разделе спецификации UML 1.3 авторы описывают две малоизвестные связи между вариантами использования, которые не имеют двойников на диаграмме и не определены на языке связей объектов, а просто записываются в пояснительном тексте. Это *зависимые варианты использования* и их противоположность — *независимые варианты использования*.

Назначение этих связей — показать, как варианты использования *компонентов* системы работают вместе, чтобы реализовать вариант использования более крупной части системы. Сами компоненты системы не показаны. Вместо соответствующих вариантов использования присутствуют заглушки. Это выглядит так, как будто вам пришлось изобразить анонимную кооперативную диаграмму, особый вид функциональной декомпозиции, который вы позднее планируете пояснить с помощью надлежащей кооперативной диаграммы. Согласно спецификации UML:

> Вариант использования, определяющий один элемент модели, затем детализируется и превращается в ряд более мелких вариантов использования, каждый из которых описывает услугу элемента модели, содержащуюся в нем... Однако обратите внимание, что структура содержащего элемента не показывается в вариантах использования, так как они лишь определяют функции, предоставляемые элементом. Зависимые варианты использования взаимодействуют для выполнения независимого варианта использования. Их взаимодействие определяется кооперацией и может быть представлено в кооперативных диаграммах.

Цель ввода этих специфических связей в пояснительный текст спецификации варианта использования неясна, и я не собираюсь ее объяснять. Я только предоставляю факты, поскольку в этой книге я употребляю термин *"подчиненный вариант использования"*, и кто-нибудь может спросить, чем отличается подчиненный вариант использования Коберна от зависимого варианта использования в UML.

Здесь термин подчиненный вариант использования обозначает цель более низкого уровня. Обычно вариант использования более высокого уровня будет вызывать (включать) подчиненный вариант использования. Я предпочитаю термины *"зависимый"* и *"независимый"* для вариантов использования более низкого и более высокого уровня, но с тех пор как в UML 1.3 появились эти слова, я изменил лексику. Мой опыт показывает, что люди не находят ничего странного в терминах *"вызываю-*

щий вариант использования" и *"подчиненный вариант использования"*. Они понятны даже начинающему автору или читателю.

A.6. Изображение диаграмм вариантов использования

Диаграммы вариантов использования облегчат вам общение с читателями, если вы установите несколько простых соглашений по диаграммам и будете их выполнять. Пожалуйста, не заставляйте своих читателей пробираться через джунгли стрелок, ожидая при этом, что они проследят вашу мысль. Правила 13-16 для различных связей вариантов использования помогут вам в этом. Вот еще два полезных правила:

Правило 17. Цели пользователя в контекстной диаграмме

В основной контекстной диаграмме не показывайте никаких вариантов использования, уровень цели которых ниже уровня цели пользователя. Как-никак назначение диаграммы — обеспечить контекст и оглавление разрабатываемой системы. Если вы производите декомпозицию вариантов использования в виде диаграмм, помещайте их на других страницах.

Правило 18. Вспомогательные действующие лица должны быть справа

Я рекомендую помещать всех основных действующих лиц слева от блока системы, а место справа от него отвести вспомогательным (второстепенным) действующим лицам. Это помогает не смешивать основные и второстепенные действующие лица. Некоторые авторы никогда не изображают на диаграммах второстепенных действующих лиц. Это позволяет им размещать основных действующих лиц с обеих сторон.

A.7. Вместо диаграмм пишите текстовые варианты использования

Не уделяйте слишком много времени изучению графики и связей. Приложите лучше силы к написанию легкого для чтения повествования. В тексте прямо указаны связи между вариантами использования.

Это мнение разделяют многие эксперты по вариантам использования. С моей стороны, наверное, нескромно упоминать о следующем событии, но я хочу подчеркнуть серьезность совета. Приношу благодарность Брюсу Андерсону из European Object Technology Practice (IBM) за комментарий, который он сделал во время встречи специалистов по вариантам использования на OOPSLA '98. В ходе встречи возник ряд вопросов о различиях между связями включения и расширения и о затруднениях в связи с лавиной сценариев и эллипсов. Брюс заявил, что в его группе не произошло катастрофичного размножения сценариев и не было никакой путаницы. Он заявил, что он делает то, что говорит Алистер. Это означает, что следует писать ясные тексты, избегая связей расширения, и оставить в покое диаграммы.

Те, кто создают хорошие текстовые варианты использования, просто не встречают проблем, с которыми сталкиваются любители фигурок из палочек, эллипсов и стрелок UML. Связи появляются естественным образом, когда вы пишете разворачивающуюся историю. Они становятся проблемой, только если вы на них останавливаетесь. Чем больше опыта приобретают консультанты в текстовой и в UML-форме, тем легче они с этим соглашаются.

Приложение B

Ответы к упражнениям

Глава 3

Упражнение 3.1

Можно описать **окрестности** или **ряд объединенных сетью предприятий**. Можно также спроектировать **систему управления зданием и освещением банка** или **новую банковскую компьютерную систему и банкомат** или просто **банкомат**. Можно обсудить проект **новой клавишной панели** или **проект новой клавиши "Enter"**. Из этого фрагмента повествования не видно, какая система обсуждается.

Упражнение 3.2

Нельзя сказать по фрагменту из истории пользователя, какая система обсуждается.

Глава 4

Упражнение 4.2

Вспомните тесты "прошел/не прошел". Поведение действующего лица должно соответствовать предложению с если. Основное действующее лицо имеет цель, вызывающую услугу, которая предоставляется системой.

Банкомат. Разрабатываемая система (SuD).

Клиент. Основное действующее лицо и участник.

Карточка для банкомата. Не является действующим лицом. Не обладает достаточно развитым поведением (это относится к магнитным картам; смарт-карты с встроенными чипами могут оценивать). Карточка для банкомата на самом деле просто конверт для данных, и ее назначение — быстрый ввод постоянных данных клиента.

Банк. Не является действующим лицом для нашей задачи. Это система, в состав которой входит банкомат.

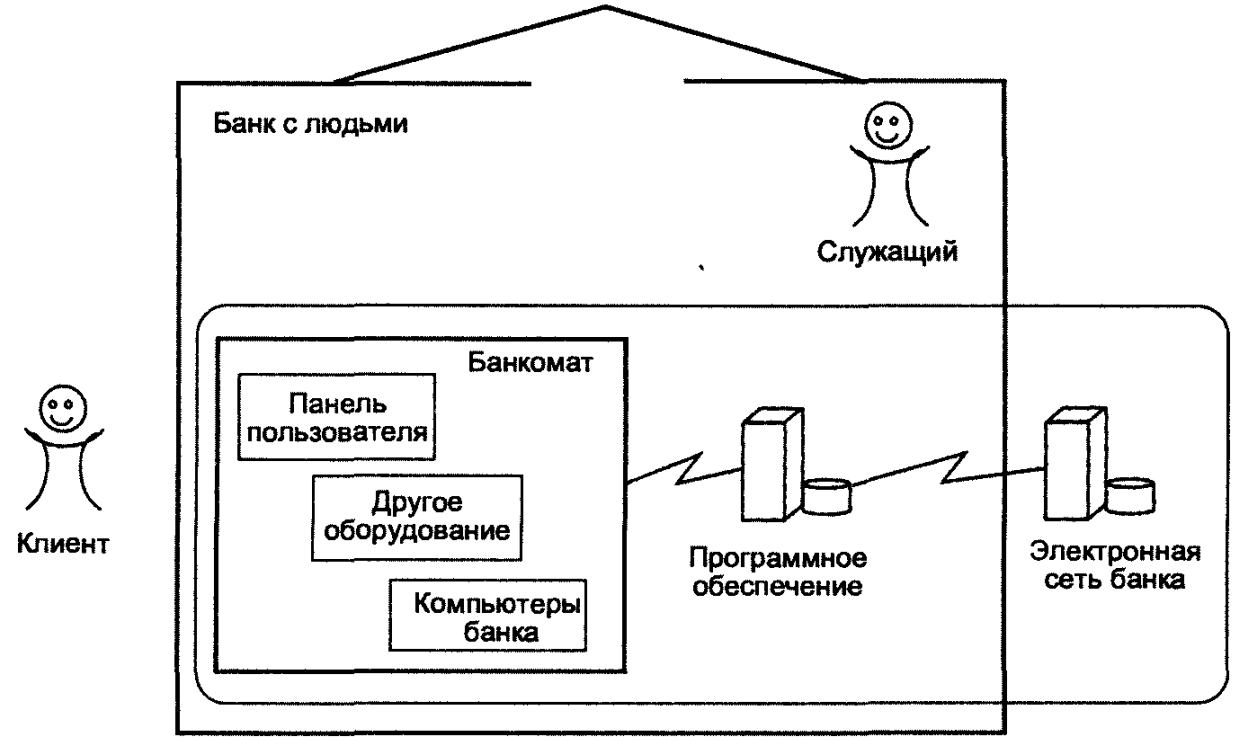


Рис. B.1. *Области действия проектирования для банкомата*

Передняя панель. Не является действующим лицом для нашей задачи. Это компонент SuD.

Владелец банка. Участник, возможно, второстепенное действующее лицо.

Наладчик. Основное действующее лицо.

Принтер. Не является действующим лицом для нашей задачи. Это компонент SuD.

Основная банковская компьютерная система. Второстепенное действующее лицо. Могла бы быть основным действующим лицом в ситуации, в которой она инициирует диалог с банкоматом.

Банковский кассир. Зависит от выполняемой работы. Кто опорожняет банкомат и загружает его наличными? Если вы скажете "загрузчик" или "обслуживающий персонал", вы, возможно, никогда не создадите вариант использования с банковским кассиром в качестве основного действующего лица. Если вы ответите: "Банковский кассир", он будет основным действующим лицом.

Грабитель банков. Зависит от области действия проектирования и вашего творчества. Мне бы не пришел в голову вариант использования для грабителя банков вместо условия расширения варианта использования клиента, если бы кто-то не предложил ограбить банкомат. Это породило идею о детекторе движения. В зависимости от того, как мы формулируем цель, мы можем закончить отдельным вариантом использования для грабителя (чья цель никогда не будет достигнута), либо просто дополнительными условиями расширения в варианте использования клиента.

Упражнение 4.3

Ответы зависят от того, какую охватывающую систему вы выбираете (см. рис. B.1).

Банкомат. Не является действующим лицом для нашей задачи. Сейчас это компонент SuD.

Клиент. По-прежнему основное действующее лицо и участник.

Карточка банкомата. Не является действующим лицом по тем же причинам, что и в упражнении 4.2 (с теми же объяснениями).

Банк. Посмотрите на рис. B.1. Если в качестве охватывающей системы вы выбираете "банк с людьми", то это ваша SuD. Если же вы выбираете "электронную банковскую сеть", это, скорее всего, действующее лицо (зависит от того, можете ли вы обосновать услугу электронной банковской сети, за которой он обращается).

Передняя панель. Не является действующим лицом для нашей задачи. Это компонент.

Владелец банка. Зависит от того, какую охватывающую систему вы выбираете, и каковы ваши цели. Его можно определить как компонент банка и, следовательно, второстепенное действующее лицо, либо как основное действующее лицо банка, которое, возможно, не является основным действующим лицом электронной банковской системы.

Наладчик. Основное действующее лицо, если его наняли в другой организации, и компонент, если является работником банка и в качестве SuD вы выбрали банк.

Принтер. Не является действующим лицом для нашей задачи. Это компонент разрабатываемой системы.

Основная банковская компьютерная система. Теперь это компонент охватывающей системы (любой из вышеназванных).

Банковский кассир. Компонент (банка) или, может быть, основное действующее лицо электронной банковской системы.

Грабитель банков. То же, что в упражнении 4.2.

Глава 5

Упражнение 5.1

- Обобщенный (белый). *Пригласить кого-нибудь на обед* (возможно, сложно для случая с банкоматом).
- Обобщенный (белый). *Использовать банкомат*.
- Цель пользователя (голубой). *Получить деньги из банкомата*.
- Подфункция (индиго). *Ввести PIN*.
- Подфункция (черный). *Найти кнопку Ввод*.

Упражнение 5.2

Действующее лицо	Цель	Уровень
Наладчик	Привести банкомат в рабочее состояние	Обобщенный
	Запустить самопроверку банкомата	Цель пользователя
Банковский служащий	Пополнить запасы денег	Цель пользователя
	Пополнить ресурсы	Цель пользователя
Клиент	Использовать банкомат	Обобщенный
	Извлечь наличные	Цель пользователя
	Положить деньги на счет	Цель пользователя
	Перевести деньги	Цель пользователя
	Проверить баланс	Цель пользователя

Глава 6

Упражнение 6.1

Самый легкий способ определить минимальные гарантии — это выяснить, что может огорчить участника. Участниками являются клиенты, банк и агентство банковского контроля.

Клиенты огорчатся, если не получат наличных, но это не то, что следует ожидать от минимальной гарантии. Предположим, что они не получили своих наличных. В этом случае их огорчит, если их счет дебетован для транзакции. На самом деле они будут расстраиваться всякий раз, когда им в дебет будут записывать большую сумму, чем они получили наличными. Они хотят также защитить себя от мошенничества, для этого нужно регистрировать все транзакции.

Интересы банка будут ущемлены, если клиент получит больше наличных, чем внесенная в дебет сумма. Банку тоже нужно защитить себя, возможно, с помощью особого вида регистрации, которая позволит узнать, как далеко зашла транзакция в случае катастрофического отказа, чтобы можно было разобраться с любыми ошибками.

Агентство банковского контроля хочет, чтобы правила соблюдались, поэтому оно главным образом заинтересовано в регистрации всех транзакций.

В результате у нас есть минимальные гарантии, что дебетованная сумма равна выданной сумме. При этом производится микрорегистрация транзакции, что в случае катастрофического отказа позволяет определить, как далеко продвинулось ее выполнение. Регистрируются все транзакции.

Упражнение 6.4.

Гарантия успеха — это когда в дебет внесена сумма, выданная наличными (а не запрошенная — проверьте условие неудачи), карточка возвращена, машина приведена в исходное состояние и транзакция зарегистрирована.

Глава 7

Упражнение 7.1

Ниже представлено подробное описание интерфейса для извлечения наличных из банкомата. Выпуск 100 подобных вариантов использования сделает несчастными нескольких читателей. Упражнение 7.2 дает пример описания намерений, а не интерфейса.

1. Клиент пропускает карточку для банкомата через считывающее устройство.
2. Банкомат считывает ID (идентификатор) банка и номер счета.
3. Банкомат спрашивает клиента, на каком языке продолжать: испанском или английском.
4. Клиент выбирает английский.
5. Банкомат просит ввести PIN, а затем нажать на "Ввод" (Enter).
6. Клиент вводит PIN, нажимает на "Ввод".
7. Банкомат представляет список действий для клиента.
8. Клиент выбирает "извлечь наличные".
9. Банкомат просит клиента ввести сумму, кратную $5, и нажать на "Ввод".
10. Клиент вводит сумму, кратную $5, и нажимает на "Ввод".
11. Банкомат уведомляет основную банковскую систему о номере счета клиента и требуемой сумме.
12. Основная банковская система соглашается выдать наличные, сообщает банкомату новый баланс.
13. Банкомат выдает наличные.
14. Банкомат спрашивает, нужна ли клиенту квитанция.
15. Клиент отвечает "да".
16. Банкомат выдает квитанцию, показывая новый баланс.
17. Банкомат регистрирует транзакцию.

Упражнение 7.2

Вот рациональная версия Быстрого получения наличных, показывающая намерения действующих лиц.

1. Клиент пропускает карточку для банкомата через считывающее устройство.
2. Банкомат считывает с карточки ID банка и номер счета, подтверждает их с помощью главного компьютера.
3. Клиент вводит PIN, банкомат подтверждает PIN.
4. Клиент выбирает быстрые наличные (FASTCASH) и указывает сумму, кратную $5.
5. Банкомат уведомляет основную банковскую систему о номере счета клиента, требуемой сумме и получает подтверждение и новый баланс.
6. Банкомат выдает наличные, карточку и квитанцию, показывая новый баланс.
7. Банкомат регистрирует транзакцию.

Упражнение 7.4

Пример содержит ошибки трех видов. Первое, что нужно понять —регистрация в системе не является назначением этого варианта использования, несмотря на его название и характеристику. Он посвящен использованию системы обработки заказов. Правильным в данном случае является обобщенный вариант использования уровня воздушного змея. Первые шесть шагов говорят о регистрации, но они совсем другого уровня и должны быть выделены. Сделав это, мы обнаружим, что пользователь входит в систему, но никогда из нее не выходит!

Пока пользователь не выберет "выйти из цикла", конец предложения с если и **конец цикла** — это структуры программиста, они не имеют смысла для читателей варианта использования. Многочисленные предложения с если загромождают текст. Шаги описывают проектирование интерфейса пользователя. Все это следует исправить.

Вариант использования запускается, когда... и **Вариант использования завершается, когда...** — это стилистические соглашения, предлагаемые некоторыми преподавателями. В них все правильно. Однако я считаю их излишними. Само собой разумеется, что вариант использования начинается с первого шага и заканчивается, когда кончается описание.

Другой стиль — это формулировка **Пользователь..., затем применить Разместить заказ**. Применить в этом предложении указывает на связь включения UML (прежде называлась связью использования). Я считаю, это больше запутывает, чем проясняет чтение, поэтому предпочитаю писать **Пользователь... размещает заказ**. Вы можете следовать любому соглашению, которое установит ваша проектная группа для взаимодействия с другими вариантами использования.

В конечном счете мы приходим к двум отдельным вариантам использования: *Использовать систему обработки заказов* уровня воздушного змея и *Зарегистрироваться в системе уровня подфункции*. Последний вы можете написать сами. Обратите внимание, что вызовы других вариантов использования подчеркнуты.

Вариант использования 38

Использовать систему обработки заказов

Основной сценарий:

1. Пользователь регистрируется в системе.
2. Система представляет доступные функции. Пользователь выбирает и выполняет одну из них:
   - Разместить заказ
   - Отменить заказ
   - Получить статус
   - Послать каталог
   - Зарегистрировать жалобу
   - Выдать отчет по продажам
3. Это повторяется, пока пользователь не решит выйти.

4. Система регистрирует выход пользователя, когда пользователь указывает на выход.

Глава 8

Упражнение 8.1

Это пример условий неудачи. Обычно на моих курсах писали список раза в два длиннее этого. Обратите внимание, что все условия поддаются обнаружению и должны быть обработаны. А как вы написали?

Сломался считыватель карточек или карточка повреждена.
Карточка банка, с которым банкомат не работает.
Неверный PIN.
Клиент вовремя не ввел PIN.
Банкомат не работает.
Главный компьютер не работает или сеть не работает.
Недостаточная сумма на счете.
Клиент вовремя не ввел сумму.
Сумма не кратна $5.
Запрошенная сумма слишком велика.
Сеть или главный компьютер отказывают во время транзакции.
Не хватает денег в кассовом аппарате.
Купюры замялись во время выдачи.
Бумага для квитанции кончилась или замялась.
Клиент не берет деньги из раздаточного устройства.

Упражнение 8.5

Вариант использования 39

Купить товары через Интернет

Основное действующее лицо: покупатель (пользователь)
Область действия: PAF
Уровень: цель пользователя
Предусловие: пользователь уже открыл PAF.
Минимальные гарантии: есть достаточный объем регистрационных данных, чтобы PAF мог обнаружить непорядок и запросить у пользователя детали.
Гарантии успеха: удаленный web-сайт подтвердил покупку; журналы PAF и портфель пользователя обновлены.
Основной сценарий:

1. Пользователь выбирает покупку товаров через Интернет.
2. PAF получает имя web-сайта для покупок (E*Trade, Schwab и т.д.).
3. PAF открывает соединение с сайтом, сохраняя управление.
4. Пользователь просматривает и покупает товары на web-сайте.

5. PAF перехватывает ответы web-сайта и обновляет портфель пользователя.
6. PAF показывает пользователю новое состояние портфеля.

Расширения:

2a. Пользователь назвал web-сайт, не поддерживаемый PAF:
  2a1. Система предлагает пользователю ввести другое имя web-сайта либо закончить вариант использования.

3a. Отказ любого вида во время установки:
  3a1. Система сообщает пользователю об отказе и дает рекомендацию; возвращается к предыдущему шагу.
  3a2. Пользователь выходит из этого варианта использования либо пробует еще раз.

4a. Отказывает или выключается компьютер во время транзакции покупки:
  4a1. (Что мы здесь делаем?)

4b. Web-сайт не подтверждает покупку, а откладывает ее:
  4b1. PAF регистрирует задержку; устанавливает таймер, чтобы спросить пользователя о результате.
  4b2. (см. Обновить сомнительную покупку).

5a. Web-сайт не возвращает необходимую информацию о покупке:
  5a1. PAF регистрирует отсутствие информации, предоставляет пользователю обновить сомнительную покупку.

5b. Диск выходит из строя или на нем отсутствует свободное пространство во время операции обновления портфеля:
  5b1. При повторном запуске PAF обнаруживает несовместимость в журнале и просит у пользователя разрешения обновить сомнительную покупку.

Глава 11

Упражнение 11.1

Вариант использования 40

Оказать услугу по прочистке свечей зажигания

Предусловие: автомобиль стоит в гараже, двигатель работает.

Минимальная гарантия: клиента уведомили о более серьезной проблеме; автомобиль не отремонтирован.

Гарантия успеха: двигатель работает ровно.

Основной сценарий:

1. Откройте капот и накройте крыло защитными материалами.
2. Удалите свечи зажигания.
3. Сотрите со свечей смазку.
4. Прочистите и отрегулируйте зазоры.
5. Проверьте и засвидетельствуйте работу свеч.

6. Замените свечи.
7. Подключите провода зажигания к соответствующим свечам.
8. Проверьте, работает ли двигатель ровно, и засвидетельствуйте это.
9. Промойте инструмент и оборудование.
10. Снимите с крыла защитные материалы; сотрите смазку с автомобиля.

Расширения:

4а. Свеча треснула или изношена: замените ее новой.

8а. Двигатель все еще не работает ровно:

8а1. Провести грубую диагностику двигателя (вариант использования 23).

8а2. Уведомить клиента о более серьезной проблеме с автомобилем (Вариант использования 41).

Приложение C
Глоссарий

Основные термины

ВАРИАНТ ИСПОЛЬЗОВАНИЯ. Выражает поведенческую часть соглашения между участниками системы. Описывает поведение системы и взаимодействие при различных условиях, когда система реагирует на запрос от имени одного из участников, основного действующего лица. Показывает, как достигается (или не достигается) цель основного действующего лица. Вариант использования собирает сценарии, имеющие отношение к цели основного действующего лица.

ВЗАИМОДЕЙСТВИЕ. Сообщение, последовательность взаимодействий или ряд последовательностей взаимодействий.

ДЕЙСТВУЮЩЕЕ ЛИЦО. Некто (или нечто), обладающее поведением (может выполнять условие, выраженное в предложении с если). Это может быть механическая либо компьютерная система, индивидуум, организация или некоторое их сочетание.

Внешнее действующее лицо — это действующее лицо вне разрабатываемой системы.

Внутреннее действующее лицо — это либо SuD, либо подсистема SuD или ее активный компонент.

Вспомогательное или второстепенное действующее лицо — это система, относительно которой разрабатываемая система (SuD) имеет цель.

Закулисное, или третьестепенное действующее лицо — это участник, который не является основным действующим лицом.

Основное действующее лицо — это участник, который обращается к системе, чтобы она обеспечила достижение его цели. Часто, но не всегда, основное действующее лицо инициирует взаимодействие с системой. Основное действующее лицо может инициировать взаимодействие с системой через посредника, а также автоматически, по совершении некоторого события.

Участник — это внешнее действующее лицо, имеющее право на защиту системой своих интересов, для удовлетворения которых система должна выполнять определенные действия. Разные варианты использования могут иметь разных участников.

РАСШИРЕНИЕ СЦЕНАРИЯ. Фрагмент сценария, который запускается при возникновении конкретного условия в другом сценарии.

Вариант использования расширения — это вариант использования, который прерывает другой вариант использования, запускаясь при определенном условии. Вариант использования, который прерывается, называется основным вариантом использования.

Подчиненный вариант использования — это вариант использования, вызываемый на шаге сценария. В UML вызывающий вариант использования включает поведение подчиненного варианта использования.

Точка расширения — это метка или краткое имя точки линии поведения в основном варианте использования, в которой вариант использования расширения может прервать его. Точка расширения на самом деле может давать имя ряду мест в основном варианте использования, с тем чтобы вариант использования расширения мог собрать все родственные линии поведения для расширения, которые прерывают основной вариант использования для одного набора условий.

Условие расширения называет обстоятельства, при которых изменяется линия поведения.

СЦЕНАРИЙ. Последовательность действий и взаимодействий, происходящих при определенных условиях, изложенная без предложений с если и ветвления.

Альтернативное направление — это любой другой сценарий или фрагмент сценария, написанный как расширение основного сценария.

Конкретный сценарий — это сценарий, в котором названы все детали: имена вовлеченных действующих лиц и величин. Эквивалентен повествованию в прошедшем времени с указанием всех подробностей.

Основной сценарий — это один сценарий, написанный полностью, от триггера запуска до завершения, который включает достижение цели и регистрацию этого факта. Это типичный и наглядный сценарий успеха, даже если он имеет другие пути, кроме успешного.

Описание использования, или просто описание — это конкретный сценарий, открывающий мотивы и намерения различных действующих лиц. Нужен для разминки перед чтением или написанием вариантов использования.

При написании требований в сценарии иногда употребляются термины-заполнители, например, *"клиент"* и *"адрес"* для действующих лиц и значений данных. Когда надо их отличить от конкретных сценариев, их можно назвать общими сценариями. Для общего сценария путь через вариант использования и направление варианта использования — синонимы.

Шаг действия — это единица повествования в сценарии. Обычно одно предложение, описывающее поведение только одного действующего лица.

Типы вариантов использования

ОБЛАСТЬ ДЕЙСТВИЯ. Определяет, насколько велика разрабатываемая система.

Область действия подсистемы указывает, что SuD в этом варианте использования является частью приложения, возможно, подсистемой или структурой. Раздел области действия шаблона варианта использования содержит название подсистемы. Вариант использования помечается значком болта с резьбой (...).

Область действия предприятия указывает, что SuD является предприятием или оргаинизацией. Раздел области действия шаблона варианта использования заполняется названием организации, отрасли бизнеса или предприятия. Графическая метка — домик, серый (...) для варианта использования типа "черного ящика" или белый (...) для варианта использования типа "прозрачного ящика".

Область действия системы означает, что SuD — это механическая либо аппаратная или программная система или приложение. Раздел области действия варианта использования содержит название системы. Графическая метка варианта использования — ящик серого (...) или белого (...) цвета для типа "черного" или "прозрачного ящика" соответственно.

ПРОЗРАЧНОСТЬ. Определяет, какие объекты варианта использования видимы.

Вариант использования типа "прозрачного ящика" упоминает о поведении компонентов SuD. Обычно он применяется для моделирования бизнес-процессов.

Вариант использования типа "черного ящика" не упоминает ни о каких внутренних компонентах SuD. Обычно он применяется для документирования требований к системе.

УРОВЕНЬ. Определяет, насколько высока цель.

Вариант использования ***обобщенного уровня*** требует для выполнения несколько сеансов уровня цели пользователя, возможно, недели, месяцы или годы. Подчиненные ему варианты использования могут иметь любой уровень цели. Графическая метка — облако ( ) или воздушный змей ( ). Облако предназначено для вариантов использования, которые содержат шаги уровня облака или воздушного змея. Воздушный змей нужен для вариантов использования, которые содержат шаги уровня цели пользователя.

Вариант использования ***уровня подфункции*** направлен на частичную цель варианта использования уровня цели пользователя или другой подфункции; его шаги — подфункции более низкого уровня. Помечается значком рыбы ( ) или моллюска ( ). Моллюск означает, что вариант использования находится на таком низком уровне, что его вовсе не следует писать.

Вариант использования ***уровня цели*** пользователя описывает достижение основным действующим лицом определенной и непосредственной цели. Он обычно выполняется одним основным действующим лицом за один сеанс продолжительностью от 2 до 20 мин (или меньше, если основным действующим лицом является компьютер). Далее основное действующее лицо может продолжить работу с другими целями. Шаги имеют уровень цели пользователя или более низкий. Графическая метка варианта использования цели пользователя — волны ( ).

ФОКУС. Вариант использования фокусируется либо на бизнесе, либо на разрабатываемой системе.

Выражение ***вариант использования для бизнеса*** — это сокращение, означающее, что этот вариант использования описывает, как протекают бизнес-процессы, а не как работает компьютерная система. Вариант использования для бизнеса можно писать на любом уровне цели, но его областью действия может быть только предприятие или организация.

Выражение ***системный вариант использования*** — это сокращение, означающее, что этот вариант использования описывает работу компьютерной или механической системы, а не бизнеса. Системный вариант использования можно писать на любом уровне цели и для любой области действия, включая предприятие. Системный вариант использования, написанный для предприятия, отводит главное место воздействию SuD на поведение предприятия.

ФОРМАЛИЗМ. Определяет степень усилий, строгости и формальности при создании варианта использования.

Краткое описание варианта использования — это конспект варианта использования в одном абзаце.

Вариант использования в свободном формате пишется в виде простого абзаца текста. Это, скорее всего, пропущенная информация о проекте, связанная с вариантом использования, и в менее строгой форме, чем полный формат варианта использования.

Вариант использования в полном формате пишется с помощью одного из полных шаблонов, указывающих действующих лиц, область действия, уровень, условие запуска, предусловие и другую информацию в соответствии с заголовками шаблона плюс аннотацию проекта.

Диаграммы

ДИАГРАММА ВАРИАНТОВ ИСПОЛЬЗОВАНИЯ. В UML эта диаграмма показывает внешних действующих лиц, границу системы, варианты использования в виде эллипсов и стрелки, соединяющие действующих лиц с эллипсами или эллипсы с другими эллипсами. Полезна на первом этапе проектирования в качестве контекстной диаграммы и оглавления.

КООПЕРАТИВНАЯ ДИАГРАММА. В UML эта диаграмма показывает ту же информацию, что и диаграмма последовательности, но в другой форме. Действующих лиц размещают вокруг диаграммы, а взаимодействия представлены с помощью пронумерованных стрелок между действующими лицами. Последовательность определяют только номера стрелок.

ДИАГРАММА ПОСЛЕДОВАТЕЛЬНОСТИ. В UML эта диаграмма показывает действующих лиц, как названия столбцов, а взаимодействия — в виде стрелок между столбцами, причем время течет сверху вниз страницы. Схема полезна для графического представления одного сценария.

Приложение D

Источники информации

Книги, на которые имеются ссылки в тексте

Beck, Kent. *Extreme Programming Explained*. Reading, MA: Addison-Wesley, 2000.

Cockburn, Alistair. *Surviving Object-Oriented Projects*. Reading, MA: Addison-Wesley, 1998.

Cockburn, Alistair. *Software Development as a Cooperative Game*. Boston: Addison-Wesley, (due 2001).

Constantine, Larry, and Lucy Lockwood. *Software for Use*. Reading, MA: Addison-Wesley, 1999.

Мартин Фаулер, *UML в кратком изложении*, Издательство "Мир", 1999.

Hummer, Michael, and James Champy. *Reengineering the Corporation, Reprint Edition*. New York: HarperBusiness, 1994.

Hohmann, Luke. *Guis width Glue* (in preparation as of July, 2000).

Robertson, Suzanne, and James Robertson. *Mastering the Requirements Process*. Reading, MA: Addison-Wesley, 1999.

Wirfs-Brock, Rebecca, Wilkerson, Brian, and Wiener, Lauren. *Designing Object-Oriented Software*. Upper Saddle River, NJ: Prentice-Hall, 1990.

Статьи, на которые имеются ссылки в тексте

Beck, Kent, and Ward Cunningham. "A Laboratory for Object-Oriented Thinking", ACM SIGPLAN 24(10):1-7, 1989.

Cockburn, Alistair. "VW-Staging," at *http://members.aol.com/acockburn/papers/vwstage.htm*.

Cockburn, Alistair. "CRC Cards," *http://members.aol.com/humansandt/papers/crc.htm*.

Cocrburn, Alistair. "An Open Letter to Newcommers to OO", *http://members.aol.com/humansandt/papers/oonewcomers.htm.*

Cunningham, Ward. "CRC Cards", at *http//c2.com/cqi/wiki?CrcCards.*

Kraus, Andy, and Michael Dillon. "Use Case Blue", Object Magazine, SIGS Publications, May 1996.

Lilly, Susan. "Haw to Avoid Use Pitfalls", Software Development 8(1):40-44, 2000.

McBreen, Peter. "The Cases from Use Cases", at *http://www.mcbreen.ab.ca/papers/TestsFromUseCases.html.*

Полезные ресурсы Интернета

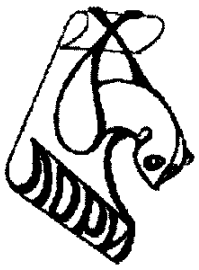
Сеть содержит огромное количество информации. Вот лишь несколько отправных точек.

http://www.usecases.org

http://members.aol.com/acockburn

http://www.foruse.com

http://www.pols.co.uk/usecasezone/



Книги издательства "ЛОРИ" Вы можете приобрести:

В Москве:

"Московский Дом Книги"
ул. Новый Арбат, 6

"Библио-Глобус"
ул. Мясницкая, 6

"Дом Технической Книги"
Ленинский пр-т., 40

"Мир"
Ленинградский пр-т., 78

"Молодая гвардия"
ул. Большая Полянка, 28

"Компьютерная и деловая книга"
Ленинский пр-т., 38

В Санкт-Петербурге:

ЗАО "Диалект"
(812) 247-1483

"Дом Книги"
Невский пр-т., 24

В Киеве:

"Техническая книга"
(044) 419-7061
(044) 464-6895

Виртуальные книжные магазины

В Санкт-Петербурге WWW.BOOKS.RU, WWW.OZON.RU
В Риге WWW.636.LV
В Новосибирске WWW.TOP-KNIGA.RU
В Нью-Йорке WWW.KNIGA.COM
В Москве WWW.BOOK.RU

Книги нашего издательства Вы можете заказать по почте:
г.Москва, ЗАО "Кнорус" тел. (095) 280-0207, 280-7254, 280-9116
г. Санкт-Петербург, "Дом Книги", отдел "Книга-почтой"
тел. (812) 219-6224
г. Новосибирск, "Топ Книга", тел. (3832) 36-10-26, 36-10-27

Практика создания вариантов использования как средств уточнения требований к поведению программных систем и бизнес-процессов быстро завоевывает популярность. Варианты использования обеспечивают эффективное планирование проекта, показывая, как будет применяться будущая система. На первый взгляд идея вариантов использования кажется простой. Однако разработчиков ждет трудная задача: приступая к созданию набора вариантов использования, необходимо выяснить, насколько точными они должны быть.

Данная книга эксперта по объектной технологии Алистера Коберна служит новейшим практическим руководством по написанию вариантов использования. Богатый опыт в этой области помогает автору расширить классическое толкование вариантов использования. В книге представлены начальная, промежуточная и развитая концепции, поэтому она подходит читателям с разным уровнем подготовки. Инструкции подкреплены наглядными примерами и упражнениями.

Книга содержит:

- Подробный анализ основных элементов вариантов использования (действующие лица, участники, области действия проектирования, сценарии и т.д.)
- Руководство по стилю вариантов использования, описание этапов действия и предлагаемых форматов
- Рекомендации по ускорению написания вариантов использования
- Шаблоны вариантов использования с пояснениями по их применению
- Испытанную методологию получения преимуществ от вариантов использования

Эта книга является руководством по изучению важнейших элементов вариантов использования. Научившись их создавать, вы преуспеете в осуществлении вашего следующего проекта.

Алистер Коберн — признанный эксперт по вариантам использования, консультант компании *Humans and Technology*. Более 20 лет руководит проектами разработки оборудования и программного обеспечения в страховых компаниях, компаниях розничной и электронной торговли, а также в таких крупных организациях, как Центральный банк Норвегии и IBM. Автор книги *Surviving Object-Oriented Projects* (Addison-Wesley, 1998).

Издательство "Лори"
www.lory-press.ru

Addison-Wesley
Pearson Education