Глава 3

Установление требований

В сравнении с другими этапами разработки системы *установление требований* в наименьшей степени касается технических сторон проекта. В действительности речь идет об успешном решении социальных, коммуникативных и управленческих вопросов. Недостаточно тщательное выполнение этого этапа может привести к более серьезным последствиям, чем в случае с другими этапами. Лавинообразный рост расходов, вызванных не зафиксированными, упущенными или неверно понятыми требованиями заказчиков, может впоследствии оказаться просто невыносимым для процесса разработки.

Эта глава охватывает широкий спектр вопросов, касающихся установления требований. Первая часть главы посвящена методам выявления, согласования и утверждения требований, а также принципам управления требованиями. Остаток первой части касается вопросов прослеживаемости и управления изменениями — темы, которая более подробно обсуждается в главе 10.

Во второй части главы вводятся базовые графические методы моделирования для описания бизнес-модели, относящейся к организации и прикладной области. Кроме того, здесь рассматривается структура документа, описывающего требования.

3.1. Принципы установления требований

Установление требований —первый этап жизненного цикла разработки системы. Система, подлежащая разработке, определяется с помощью вида деятельности, который получил название *системного планирования* (разд.1.2). Цель установления требований состоит в том, чтобы дать развернутое определение функциональных — а также нефункциональных — требований, которые участники проекта ожидают утвердить в реализуемой и разворачиваемой системе.

Требования определяют услуги, ожидаемые от системы (*формулировки сервисов*) и ограничения, которым система должна подчиняться (*формулировки ограничений*). Эти формулировки сервисов можно объединить в несколько групп: одна из групп описывает границы системы, вторая — необходимые бизнес-функции (*функциональные требования*), а третья — требуемые структуры данных (*требования к данным*). Формулировки

ограничений можно классифицировать в соответствии с различными категориями ограничений, накладываемых на систему, такие как требуемое "впечатление и ощущение от использования программы", производительность, безопасность и т.д.

Требования необходимо получить от заказчиков (пользователей и владельцев системы). Этот вид деятельности, называемый *выявлением требований* (*requirements elicitation*), осуществляется аналитиком бизнес-процессов (или системным аналитиком). Для этого подходят различные методы, начиная с традиционных интервью с заказчиком и заканчивая (при необходимости) построением программного прототипа, с помощью которого раскрываются дополнительные требования.

Собранные требования должны быть подвергнуты тщательному анализу для выявления в них повторов и противоречий. Это неизменно приводит к *пересмотру требований и их повторному согласованию* с заказчиками.

Требования, удовлетворительные с точки зрения заказчика, документируются. При этом требованиям дают определение, классифицируют, нумеруют и присваивают им приоритеты. Структура документа, описывающего требования, соответствует *шаблону* (*template*), выбранному в организации для этой цели.

Хотя документ, фиксирующий требования, носит в значительной мере описательный характер, вполне возможно включить в него высокоуровневую схематичную бизнес-модель. Как правило, бизнес-модель состоит из *модели границ системы*, *модели бизнес-прецедентов* и *модели бизнес-классов*.

Требования пользователя подобны движущейся цели. Чтобы справиться с изменчивыми требованиями, необходимо уметь управлять изменениями. *Управление требованиями* включает такие виды деятельности как оценка влияния изменений одних требований на другие, а также на неизменяемую часть системы.

3.2. Выявление требований

Бизнес-аналитик выявляет требования к системе посредством консультаций. К участию в консультациях привлекаются заказчики и эксперты в проблемной области. В некоторых случаях бизнес-аналитик обладает достаточным опытом в проблемной области, и помощь эксперта может не потребоваться. В этом случае `Бизнес-аналитик` представляет собой разновидность `Эксперта проблемной области`, что отражено в модели, показанной на рис. 3.1, с помощью отношения обобщения. (Следует, однако, иметь ввиду, что рис. 3.1 — это не модель прецедентов: нотация для прецедентов используется здесь только для удобства.)

Требования, выявленные с помощью эксперта проблемной области, составляют основу знаний о проблемой области. Они фиксируют широко признанные, не зависящие от времени бизнес-правила, применимые к большинству организаций и систем. Требования, выявленные в ходе консультаций с заказчиками, выражаются в сценариях прецедентов. Они выходят за рамки базовых знаний о проблемной области и фиксируют уникальные черты организации — способ ведения бизнеса "здесь и сейчас" либо пожелания в отношении того, как следует вести бизнес.

Задача бизнес-аналитика состоит в том, чтобы объединить два набора требований в бизнес-модели. Как показано на рис. 3.1, `Бизнес-модель` содержит `Модель бизнес-классов` и `Модель бизнес-прецедентов`. `Модель бизнес-классов` представляет собой диаграмму классов верхнего уровня, которая идентифицирует и связывает между собой *бизнес-объекты*. `Модель бизнес-прецедентов` — это диаграмма прецедентов верхнего уровня, которая идентифицирует основные функциональные строительные блоки системы.

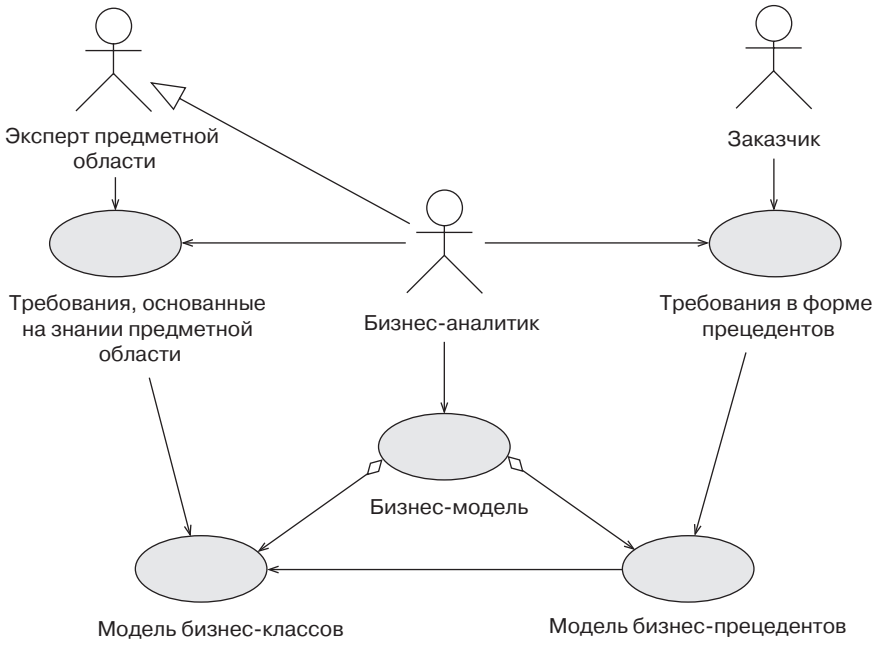

Рис. 3.1. Взаимные влияния моделей, характерные для процесса определения требований

В общем случае, классы проблемной области (бизнес-объекты) не должны выводиться из прецедентов [75]. Однако, на практике правильность модели бизнес-классов должна подтверждаться посредством сравнения с `Моделью бизнес-прецедентов`. Это сравнение, как правило, приводит к некоторой настройке или расширению `Модели бизнес-классов`.

Следуя Хофферу (Hoffer) [37] мы различаем традиционные и современные методы выявления фактов и сбора информации.

3.2.1. Традиционные методы выявления требований

К традиционным методам выявления требований относятся использование интервью и анкет, наблюдение и изучение деловых документов. Это простые и экономичные методы.

Эффективность традиционных методов обратно пропорциональна риску проекта. Высокий риск означает, что систему трудно реализовать — не вполне ясны даже обобщенные требования. Для таких проектов традиционных методов вряд ли будет достаточно.

3.2.1.1. Интервьюирование заказчиков и экспертов в проблемной области

Использование интервью представляет собой основной метод выявления фактов и сбора информации. Большинство интервью проводятся с заказчиками. Интервью с заказчиками позволяют выявить по большей части “прецедентные” требования, т.е. требования вытекающие из “прецедентов” (рис. 3.1). Если бизнес-аналитик не облада-

ет достаточным опытом в проблемной области, можно также проинтервьюировать соответствующих экспертов.

Интервью с экспертами в прикладной области зачастую представляет собой просто процесс передачи знаний — занятие по обучению бизнес-аналитика. Интервью с заказчиками отличает большая сложность [46], [81].

Заказчики могут иметь весьма смутное представление о своих требованиях. Они могут не желать сотрудничать или не умеют выражать свои требования в понятной форме. Они также могут запрашивать реализацию требований, которые превосходят бюджет проекта или нереализуемы. И наконец, весьма вероятно, что требования, исходящие от различных групп пользователей, могут оказаться противоречивыми.

Существуют два основных типа интервью: структурированное (формальное) и неструктурированное (неформальное). *Структурированное интервью* готовится заранее, отличается ясностью постановки вопросов, а многие вопросы могут быть заданы заранее. Заранее сформулированные вопросы можно разделить на две категории: *вопросы с открытым множеством ответов* (*open-ended question*) (ответы на эту категорию вопросов заранее не готовятся) и *вопросы с замкнутым множеством ответов* (*closed-ended question*) (ответы на эту категорию вопросов можно выбрать из списка подготовленных ответов).

Структурированному интервью должно сопутствовать *неструктурированное интервью*. Неструктурированное интервью больше напоминает неформальную встречу, которой не свойственны заготовленные впрок вопросы или заранее поставленные цели. Цель неструктурированного интервью — подвигнуть заказчика к тому, чтобы он поделился своими мыслями и в процессе беседы подошел к требованиям, которых бизнес-аналитик мог и не ожидать и, следовательно, не мог задать нужные вопросы.

Как структурированное, так и неструктурированное интервью нуждаются в некоторой отправной точке и контексте для обсуждения. Это может быть небольшой документ или записка, отправленная по электронной почте интервьюируемому перед встречей, цель которых — объяснить цели интервьюера или поставить некоторые вопросы.

Существуют три категории вопросов, которых, в общем случае, необходимо избегать [91].

- *Небеспристрастные вопросы*, в которых интервьюер выражает (прямо или косвенно) свое мнение по вопросу ("Должны ли мы работать так, как мы работаем?").
- *Предвзятые вопросы*, аналогичные небеспристрастным, но отличаются от последних тем, что мнение интервьюера является, очевидно, тенденциозным ("Вы ведь не станете этого делать, не так ли?").
- *Наводящие вопросы*, которые предполагают ответ в самом вопросе ("Вы ведь сделаете именно так, не правда ли?").

Успех интервью зависит от многих факторов, но едва ли не главными среди них являются навыки интервьюера в области *коммуникации и межличностного общения*. Хотя основная задача интервьюера — задавать вопросы и владеть ситуацией, не менее важно в ходе беседы внимательно слушать и проявлять терпение к интервьюируемому так, чтобы он чувствовал себя непринужденно. Для сохранения хороших личных отношений и в расчете на положительный отклик со стороны интервьюируемого необходимо отправить ему в течение одного-двух дней после интервью записку, содержащую краткие итоги интервью.

3.2.1.2. Анкетирование

Использование анкет или *анкетирование* (*questionnaires*) — эффективный способ сбора информации от многих заказчиков. Обычно анкеты используются в дополнение к интервью, а не вместо них. Исключение могут составлять проекты с низким риском, цели которых ясно очерчены. Для таких проектов обычно достаточно использовать анкеты с вопросами, которые носят пассивный характер и не отличаются большой глубиной.

В общем случае, анкетирование менее продуктивно, чем использование интервью, поскольку в вопросы или возможные ответы нельзя внести дополнительную ясность. Анкетирование пассивно — это можно расценивать и как достоинство, и как недостаток. Это достоинство, поскольку у респондента есть время подумать над ответом, а анкета может остаться анонимной. Это недостаток, поскольку респонденту нелегко прояснить для себя вопросы.

Анкета (или вопросник) должна быть разработана таким образом, чтобы, по возможности, облегчить ответы на вопросы. В частности, следует избегать вопросов с неопределенным множеством ответов — большинство вопросов должны относиться к *вопросам с замкнутым списком ответов*. Подобные вопросы могут принимать три формы [91].

- *Многоальтернативные вопросы* (*multiple-choice questions*). При ответе на эти вопросы респондент должен указать один или более ответов, выбрав их из прилагаемого списка. Кроме того, иногда допускаются дополнительные комментарии к вопросам со стороны респондентов.
- *Рейтинговые вопросы* (*rating questions*). При ответе на этот тип вопросов респондент должен выразить свое мнение в отношении высказанного утверждения. Для этого могут использоваться такие рейтинговые значения, как “абсолютно согласен”, “согласен”, “отношусь нейтрально”, “не согласен”, “абсолютно не согласен” и “не знаю”.
- *Вопросы с ранжированием* (*ranking questions*). Этот тип вопросов предусматривает ранжирование ответов с помощью присваивания им последовательных номеров, процентных значений и использования других средств упорядочения.

Тщательно продуманная, облегчающая ответы анкета поощряет респондента к тому, чтобы сразу вернуть заполненный документ. Однако, при оценке результатов анкетирования бизнес-аналитик должен предусмотреть возможность искажения информации из-за того, что те люди, которые не отвечали на вопросы, могли бы ответить на них иначе, чет принимавшие участие в опросе [37].

3.2.1.3. Наблюдение

В некоторых ситуациях бизнес-аналитик находит трудным получить полную информацию с использованием интервью и анкет. У заказчика по тем или иным причинам может отсутствовать возможность (или умение) предоставить необходимую информацию, либо он может не иметь полного представления о бизнес-процессе в целом. В подобных случаях в качестве эффективного метода выявления фактов может выступать *наблюдение* (*observation*). Как-никак, лучший способ научиться завязывать галстук — понаблюдать за процессом.

Наблюдение может выступать в двух формах.

- *Пассивное наблюдение* (*passive observation*), в ходе которого бизнес-аналитик наблюдает за различными видами деловой деятельности без вмешательства или

прямого участия в них. В некоторых случаях для того, чтобы наблюдение было как можно менее навязчивым, можно даже использовать видеокамеры.

- *Активное наблюдение* (*active observation*), в ходе которого бизнес-аналитик участвует в деятельности и становится фактически частью команды.

Чтобы результаты наблюдений были представительными, наблюдения необходимо проводить в течение продолжительного периода времени, в разные временные интервалы и при разной рабочей нагрузке (выборочные периоды). Основная трудность, связанная с наблюдением, состоит в том, что люди, за которыми наблюдают, склонны вести себя иначе, чем в обычной обстановке. В частности, они стремятся работать в соответствии с формальными правилами и процедурами. Это приводит к искажению реального положения дел за счет утаивания "рациональных" приемов работы — как положительных, так и отрицательных. Следует помнить, что "работа по правилам" — это одна из эффективных форм забастовочного движения.

3.2.1.4. Изучение документов и программных систем

Изучение документов и программных систем является неоценимым методом выявления как требований типа прецедентов, так и требований, связанных со знанием проблемной области (см. разд. 3.2). Этот метод используется всегда, хотя он может касаться только отдельных сторон системы.

Требования, формулируемые в виде прецедентов, или *прецедентные требования* (*use case requirements*) выявляются посредством изучения существующих организационных документов и системных форм и отчетов (если, конечно, для текущей системы существует автоматизированное решение, что типично для больших организаций). Одним из наиболее полезных способов постижения сути требований на основе прецедентов является фиксация дефектов (естественно, при их наличии) и формирование запросов на изменение для существующей системы.

К *организационным документам*, подлежащим изучению, относятся: формы деловых документов (по возможности — заполненные), описания рабочих процедур, должностные обязанности, методические руководства, бизнес-планы, схемы организационных структур, внутриофисная корреспонденция, протоколы совещаний, учетные записи, внешняя корреспонденция, жалобы клиентов и т.д.

Системные формы и отчеты, подлежащие изучению, включают: копии экранов, отчеты вместе с соответствующей документацией — системные руководства по эксплуатации, пользовательская документация, техническая документация, системные модели анализа и проектирования и т.д.

Требования, основанные на знании проблемной области, выясняются посредством изучения журналов и учебников, которые относятся к данной сфере. Изучение патентованных программных пакетов наподобие ERP-систем (Enterprise Resource Planning Systems — системы планирования ресурсов предприятия) также может стать богатым источником знаний о проблемной области. Следовательно, посещения библиотек и поставщиков ПО являются частью процесса выявления требований (конечно, Internet позволяет осуществить многие такие "визиты", не покидая офиса).

3.2.2. Современные методы выявления требований

К современным методам выявления требований относится использование программных прототипов, а также такие методы, как JAD (Joint Application Development — совместная разработка приложений) и RAD (Rapid Application Development —

быстрая разработка приложений). Эти подходы предлагают более глубокое проникновение в суть требований, но за счет большей цены и усилий. Однако, долговременные вложения в эти методы могут окупиться с лихвой.

Применение современных методов обычно связано с высоким проектным риском. Существует множество факторов, обусловливающих высокий риск проекта. К таким факторам относятся неясные цели, недокументированные процедуры, нестабильные требования, слабое знание дела пользователями, неопытные разработчики, недостаточная приверженность пользователей разработке и т.д.

3.2.2.1. Прототипирование

Прототипирование (*prototyping*)— это наиболее часто используемый современный метод выявления требований. Программные прототипы конструируются для визуализации системы или ее части для заказчиков с целью получения их отзывов.

Прототип представляет собой демонстрационную систему — "наскоро и грубо" сделанную рабочую модель решения, которая представляет пользовательский GUI-интерфейс и моделирует поведение системы при инициировании пользователем различных событий. Информационное наполнение экранов чаще жестко запрограммировано в программе прототипа, чем получается автоматически из базы данных.

Сложность (и растущие "аппетиты" заказчиков) современных GUI-интерфейсов делают прототипирование обязательным элементом разработки ПО. Прототипы позволяют довольно неплохо оценить реализуемость и полезность системы до начала ее реализации.

В общем случае, прототип — это весьма эффективный способ выявления требований, которые трудно получить от заказчика с помощью других средств. Чаще всего такая ситуация встречается для систем, которые должны предоставить в распоряжение пользователей новые бизнес-функции. Подобная ситуация также характерна для случаев противоречивых требований и наличия проблем в кооперации между заказчиками и разработчиками.

Существуют две основные разновидности прототипов [46].

- *"Одноразовый" прототип* (*"throw-away" prototype*), который после того, как выявление требований завершено, просто отбрасывается. Разработка "одноразового" прототипа нацелена только на этап установления требований ЖЦ ПО. Как правило, этот прототип концентрируется на наименее понятных требованиях.
- *Эволюционный прототип* (*evolutionary prototype*), который сохраняется после выявления требований и используется для создания конечного программного продукта. Эволюционный прототип нацелен на ускорение поставки продукта. Как правило, он концентрируется на ясно изложенных требованиях, так что первую версию продукта можно предоставить заказчику довольно быстро (хотя ее функциональные возможности, как правило, неполны).

Дополнительным доводом в пользу использования именно "одноразового" прототипа может служить стремление избежать риска "консервации" скорых и грубых и, как следствие, неэффективных решений в конечном продукте. Однако мощь и гибкость современных средств создания ПО ослабляют этот довод. Если управление проектом осуществляется надлежащим образом, то причины, по которой нельзя было бы избавиться от неэффективных предложенных для прототипа решений, не существует.

3.2.2.2. Совместная разработка приложений (JAD-метод)

JAD-метод полностью сответствует своему названию — это *совместная разработка приложений* (*Joint Application Development*), осуществляемая в ходе одного или нескольких совещаний с привлечением всех участников проекта (заказчиков и разработчиков) [95]. Хотя мы относим JAD-подход к современным методам выявления требований, этот метод был впервые введен в конце 1970-х годов компанией IBM.

Существует много разновидностей JAD-метода и много фирм, предлагающих услуги по организации и проведению JAD-совещаний. Проведение JAD-совещаний может занимать несколько часов, несколько дней или даже пару недель. Количество участников не должно превышать 25-30 человек. В совещании принимает участие следующий круг лиц [37], [91].

- *Ведущий* — человек, который проводит и модерирует встречу (поэтому иногда его называют *модератором*). Этот человек должен обладать исключительными способностями в области коммуникации, не должен относиться к числу участников проекта (помимо того, что он является лидером JAD-сессии), обладает основательным знанием проблемной области (однако не обязательно хорошо владеет проблемами разработки ПО).
- *Секретарь* — человек, который фиксирует ход JAD-сессии с использованием компьютера. Этот человек должен уметь быстро вводить текст в компьютер и хорошо владеть вопросами разработки ПО. Для документальной фиксации хода сессии и разработки первоначальных моделей решений секретарь может использовать CASE-средства.
- *Заказчики* (пользователи или руководители) — это основные участники, которые излагают и обсуждают требования, принимают решения, утверждают проектные цели и т.д.
- *Разработчики* — бизнес-аналитики и другие члены проектной бригады. Эти участники больше слушают, чем говорят — они присутствуют на совещании для того, чтобы уяснить фактическую сторону дела и собрать информацию, а не занимать всецело внимание других участников в процессе совещания.

JAD-метод зиждется на групповой динамике. Групповые усилия более перспективны с точки зрения получения лучших решений проблем. Группы способствуют повышению продуктивности, быстрее обучаются, склонны к более квалифицированным заключениям, позволяют исключить многие ошибки, принимают рискованные решения (иногда это может носить негативный характер!), концентрируют внимание участников на наиболее важных вопросах, объединяют людей и т.д.

Если JAD-сессия проводится в соответствии с правилами, можно добиться удивительно хороших результатов. Однако, не следует забывать о предупреждении: "... компания Ford Motor в 1950-х годах потерпела неудачу, пытаясь вывести на рынок модель Edsel — автомобиль, разработанный комитетом" [95].

3.2.2.3. Быстрая разработка приложений (RAD-метод)

Метод *быстрой разработки приложений* (*Rapid Application Development — RAD*) — это нечто большее, чем метод выявления требований — это целостный подход к разработке ПО [37]. Как ясно из названия метода, он предполагает быструю поставку системных решений. Техническое превосходство отступает на второе место в сравнении со скоростью поставки.

Согласно Вуду (Wood) и Сильверу (Silver) [95] технология RAD сочетает в себе пять методов, перечисленных ниже.

- *Эволюционное прототипирование* (разд. 3.2.2.1).
- *CASE-средства* с возможностями генерации программ и циклической разработкой с переходом от проектных моделей к программе и обратно.
- *Специалисты, владеющие развитыми инструментальными средствами* (*Specialists with Advanced Tools — SWAT*) — RAD-бригада разработчиков. Лучшие аналитики, проектировщики и программисты, которых только может привлечь организация. Бригада работает в рамках строгого временного режима и размещается вместе с пользователями.
- *Интерактивный JAD-метод* — JAD-сессия (разд. 3.2.2.2), во время которой секретарь заменяется бригадой SWAT, оснащенной CASE-средствами.
- *Жесткие временные рамки* (*timeboxing*) — метод управления проектом, который отводит бригаде SWAT фиксированный период времени (*timebox*) для завершения проекта. Этот метод препятствует "расползанию рамок проекта"; если проект затягивается, то рамки решения сужаются, чтобы дать возможность завершить проект своевременно.

Использование RAD-подхода может оказаться привлекательным вариантом для многих проектов, в особенности, для небольших проектов, которые не затрагивают сферу ключевых бизнес-процессов организации, и которые, таким образом, не задают план решения для других проектов по разработке ПО. Маловероятно, чтобы быстрые решения были оптимальными или долговременными для ключевых сфер деятельности организации. С использованием RAD-метода связан ряд проблем; некоторые из них перечислены ниже.

1. Несовместимый проект GUI-интерфейса.
2. Вместо общих решений, способствующих многократному использованию ПО, специализированные решения.
3. Неполная документация.
4. Трудное для поддержки и масштабирования ПО и т.д.

3.3. Согласование и проверка обоснованности требований

Требования, полученные от пользователей, могут дублироваться или противоречить друг другу. Некоторые требования могут быть неясными или нереальными Другие требования могут остаться невыясненными. По этой причине прежде, чем требования попадут в документ описания требований, их необходимо согласовать.

В действительности *согласование и проверка обоснованности требований* осуществляется параллельно с *выявлением требований*. После того как требования выявлены, они подвергаются определенному уровню проверки. Для всех современных методов выявления требований, которые связаны с так называемой "групповой динамикой", это вполне естественно. Как бы там ни было, после того как выявленные требования соб-

раны вместе, они в любом случае должны быть подвергнуты тщательному обсуждению и проверке.

Согласование и проверка требований не могут быть отделены от процесса подготовки документа описания требований. Обычно *согласование требований* основано на черновом варианте документа. Требования, перечисленные в черновом варианте документа, обсуждаются и при необходимости модифицируются. Побочные требования удаляются. Вновь выявленные требования добавляются.

Для *проверки обоснованности требований* (*requirements validation*) необходима более полная версия документа, в котором все требования четко идентифицированы и классифицированы. Участники проекта изучают документ и проводят совещания по их формальному пересмотру. *Пересмотры* (*reviews*) часто структурированы в виде так называемых процедур *сквозного контроля* (*walkthroughs*) или *инспекций* (*inspections*). *Пересмотры* являются разновидностью *тестирования* (*testing*) (разд. 10.1.1).

3.3.1. Требования, выходящие за рамки проекта

Выбор проекта в сфере ИТ и, следовательно, подлежащей реализации системы (и их четких границ) определяется в рамках вида деятельности, получившего название *системного планирования* (разд. 1.2). Однако, детализированные взаимозависимости между системами могут быть раскрыты только на этапе анализа требований. Установление *границ системы* (*системных рамок*) — задача этапа анализа требований, так что проблема "растягивания рамок" может решаться на ранних этапах процесса разработки как составная часть этой задачи.

Чтобы иметь возможность решить, лежит ли конкретное требование в рамках системы или выходит за рамки, необходима эталонная модель, по отношению к которой и должно приниматься решение. Исторически роль подобной эталонной модели сыграла *контекстная диаграмма* (*context diagram*) — высокоуровневая диаграмма популярного метода структурного моделирования под названием диаграмма потоков данных (Data Flow Diagrams — DFD). Хотя в языке UML место этой диаграммы заняла диаграмма прецедентов, контекстная диаграмма по-прежнему остается превосходным методом установления границ системы (разд. 3.5.1).

Существуют, впрочем, и другие причины, по которым требование может быть расценено, как выходящее за рамки проекта [81]. Например, требование может представлять большую трудность для реализации в компьютеризованной системе и остается прерогативой человека, участвующего в процессе. Либо требование может иметь низкий приоритет и исключается из первой версии системы. Требование может быть также реализовано с помощью аппаратного обеспечения или внешних устройств и, таким образом, может оказаться вне контроля со стороны программного обеспечения.

3.3.2. Матрица зависимости требований

Полагая, что все требования четко идентифицированы и пронумерованы (разд. 3.4.1), можно сконструировать *матрицу зависимостей требований* (*requirements dependency matrix*) (или *матрицу взаимодействия* (*interaction matrix требований*)) [81], [46]. В столбце и строке заголовка перечислены упорядоченные идентификаторы требований, как показано на рис. 3.2.

Требование	Т1	Т2	Т3	Т4
Т1	X	X	X	X
Т2	Конфликт	X	X	X
Т3			X	X
Т4		Перекрытие	Перекрытие	X

Рис. 3.2. Матрица зависимости требований

Верхняя правая часть матрицы (над диагональю включительно) не используется. Оставшиеся ячейки указывают на то, перекрываются ли два любых требования, противоречат друг другу или независимы друг от друга (пустые ячейки). *Противоречивые требования* необходимо обсудить с заказчиками и при возможности переформулировать для смягчения противоречий (фиксацию противоречия, видимую для последующей разработки, необходимо сохранить). *Перекрывающиеся требования* также должны быть сформулированы заново, чтобы исключить совпадения.

Матрица зависимости требований представляет собой простой, но эффективный метод обнаружения противоречий перекрытий, когда количество требований относительно невелико. В противном случае описанный метод все же можно применить, если удается сгруппировать требования по категориям (разд. 3.4.1), а затем сравнить их отдельно в пределах каждой категории.

3.3.3. Риски и приоритеты требований

После того, как в результате снятия противоречий и устранения повторов в требованиях выработан пересмотренный набор требований, их необходимо подвергнуть анализу рисков и назначить им приоритеты. *Анализ рисков* (*risk analysis*) направлен на идентификацию требований, которые являются потенциальными источниками трудностей в разработке. *Назначение приоритетов* (*prioritization*) необходимо для того, чтобы обеспечить возможность без труда изменить рамки проекта в случае возникновения непредвиденных задержек.

Требования могут быть "рискованными" вследствие влияния различных факторов. Требованиям свойственны следующие типичные виды *рисков* [81].

- *Технический риск*, когда требование технически трудно реализовать.
- *Риск, связанный со снижением производительности*, когда требование — будучи реализовано — может неблагоприятно сказаться на времени реакции системы.
- *Риск, связанный с нарушением безопасности*, когда требование — будучи реализовано — может создать бреши в защите системы.
- *Риск, связанный с процессом разработки*, когда для реализации требования необходимо использование необычных методов разработки, незнакомых разработчикам (например, методов формальной спецификации).
- *Риск, связанный с нарушением целостности баз данных*, когда требование не может быть легко проверено и может привести к противоречивости данных.
- *Политический риск*, когда требование может оказаться трудным выполнить по внутриполитическим причинам.
- *Риск, связанный с нарушением законности*, когда требование может привести к нарушению действующих законов или ожидаемого изменения закона.

- *Риск, связанный с изменчивостью*, когда требование может потенциально изменяться или эволюционировать в течение процесса разработки.

В идеале *приоритеты* требованиям назначают отдельные заказчики в процессе выявления требований. Затем они согласовываются на совещаниях и снова изменяются после приложения к ним факторов риска.

Для исключения неопределенности и облегчения назначения приоритетов количество групп приоритетов не должно быть слишком велико. Обычно используется от трех до пяти приоритетов. Их можно обозначить как "высокий", "средний", "низкий" и "неопределенный". Альтернативный перечень приоритетов может выглядеть, например, так: "существенное", "полезное", "трудно определимое" и "отложенное".

3.4. Управление требованиями

Требованиями необходимо управлять. Управление требованиями в действительности представляет собой часть общего управления проектом. Оно связано с тремя основными вопросами.

1. Идентификация, классификация, организация и документирование требований.
2. Изменение требований (с помощью процессов, которые устанавливают способы выдвижения, согласования, проверки достоверности и документирования неизбежных изменений к требованиям).
3. Прослеживаемость требований (с помощью процессов, которые поддерживают отношения взаимозависимости между требованиями и другими системными артефактами, а также, собственно, между требованиями) (обратитесь к главе 10).

3.4.1. Идентификация и классификация требований

Требования описываются на естественном языке, например, следующим образом.

- "Система должна запланировать следующий телефонный звонок клиенту по запросу телемаркетера".
- "Система должна автоматически набирать запланированный телефонный номер и одновременно отображать на экране телемаркетера информацию, включающую телефонный номер, номер клиента, имя клиента".
- "В случае успешного соединения система должна отобразить вводный текст, с которым телемаркетер может обратиться к клиенту для того, чтобы завязать разговор".

Типичная система может состоять из сотен или тысяч формулировок требований, наподобие тех, что приведены выше. Для надлежащего управления таким огромным количеством требований их необходимо пронумеровать с помощью некоторой *схемы идентификации* (*identification scheme*) Схема может включать *классификацию* требований в виде групп, которые легче поддаются управлению.

Существует несколько методов идентификации и классификации требований [46].

- *Уникальный идентификатор* — обычно последовательный номер, присвоенный вручную или сгенерированный с использованием базы данных CASE-средства (т.е. базы данных (или репозитория), в которой CASE-средства хранят артефакты, выработанные в результате анализа или проектирования).

- *Последовательный номер внутри иерархии документа* — присваивается с учетом положения требований в пределах документа описания требований (например, седьмое требование в третьем разделе второй главы может быть пронумеровано как 2.3.7).
- *Последовательный номер в пределах категории требований* — присваивается в дополнение к мнемоническому имени, которое обозначает категорию требований (где под категорией требований подразумевают функциональное требование, требование к данным, требования к производительности, требования к безопасности и т.д.)

Каждый из методов идентификации обладает своими за и против. Наибольшей гибкостью и защищенностью от ошибок отличаются уникальные *идентификаторы, генерируемые с помощью базы данных*. Базы данных обладают встроенными возможностями генерации уникальных идентификаторов для каждой новой записи данных в условиях одновременного доступа к данным со стороны многих пользователей.

Некоторые базы способны дополнительно поддерживать сопровождение нескольких версий одной и той же записи (за счет расширения значения уникального идентификатора с помощью номера версии). Наконец, базы данных могут обладать возможностями поддержки ссылочной целостности связей между артефактами моделирования, включая требования, и могут, таким образом, обеспечить необходимую поддержку управления и прослеживаемости требований.

3.4.2. Иерархии требований

Требования можно упорядочить в виде иерархически упорядоченной структуры, представляющей *отношение родитель–потомок*. Отношение родитель-потомок подобно *отношению композиции* (разд. 2.1.4). Родительское требование составлено из дочерних требований. Дочернее требование — это фактически “под-требование” родительского требования.

Иерархические отношения позволяют ввести дополнительный уровень классификации требований. Это может непосредственно отражаться (или не отражаться) в идентификационном номере (с помощью использования точки в записи составного имени). Поэтому требование, пронумерованное как 4.9, может быть девятым “потомком” “родителя” с идентификационным номером, равным 4.

Приведенный ниже набор формулировок представляет собой иерархически упорядоченные требования.

1. “Система должна запланировать следующий телефонный звонок клиенту по запросу телемаркетера”.
   1.1. “Система должна активизировать клавишу `Next Call` (Следующий звонок) при открытии формы `Telemarketing Control` (Управление телемаркетингом) или при завершении предыдущего вызова”.
   1.2. “Система должна удалить звонок из начала очереди запланированных звонков и придать ему статус текущего звонка”.
   1.3. Другие требования...

Иерархии требований позволяют определять требования на различных *уровнях абстракции*. Это согласуется с общим принципом моделирования, в соответствии с которым при переходе к более низкому уровню абстракции модели систематически обогащаются все новыми деталями. В результате для родительских требований можно сконструировать обобщенные требования, а детализированные модели можно связать с дочерними требованиями.

3.4.3. Управление изменениями

Требования изменяются. Требования могут изменяться, они могут быть удалены, а новые требования могут быть добавлены на любом этапе разработки. Изменения нельзя рассматривать как "удар", а вот неуправляемые изменения — это настоящий удар по проекту.

Чем дальше продвинулась разработка, тем дороже обходится внесение изменений. Фактически, минимальные затраты, необходимые для выправления проекта после внесения изменений, всегда растут и зачастую растут экспоненциально. Изменение только что сформулированных требований, которые не связаны с другими требованиями, — это простое упражнение в редактировании. Изменение тех же требований, после того как они реализованы, может обойтись чрезвычайно дорого.

Изменения могут быть связаны с субъективными ошибками, но зачастую они вызваны изменениями внутренней политики или внешними факторами, такими как конкурентные силы, глобальные рынки или технологические достижения. Вне зависимости от причин, для документирования *запросов на изменения* (*change request*) оценки *влияния, оказываемого изменениями* (*change impact*), и осуществления изменений необходима сильная стратегия управления изменениями.

Поскольку изменение требований обходится дорого, для каждого запроса на изменение необходимо создавать формальный *бизнес-прецедент*. Обоснованное изменение, которое не встречалось раньше, оценивается с точки зрения его технической осуществимости, влияния на остальной проект и затрат. После согласования требование включается в соответствующие модели и реализуется в программном обеспечении.

Управление изменениями, помимо прочего, включает отслеживание больших объемов взаимосвязанной информации в течение длительного периода времени. Без инструментов поддержки управление изменениями обречено. В идеале, изменения требований должны храниться и отслеживаться с помощью *средств конфигурационного управления ПО*, которое используется разработчиками для контроля версий моделей и программ на протяжении ЖЦ разработки. Подходящие CASE-средства должны либо обладать собственными возможностями конфигурационного управления, либо возможностями взаимодействия с автономными средствами конфигурационного управления.

3.4.4. Прослеживаемость требований

Прослеживаемость требований (*requirements traceability*) — это всего лишь часть, хотя и крайне важная часть, *управления изменениями* (*change management*). Блок прослеживаемости требований технологии управления изменениями поддерживает отношения прослеживаемости, чтобы фиксировать изменения, *исходящие от* или *вносимые в* требования на протяжении ЖЦ разработки.

Рассмотрим требование: "Система должна запланировать следующий телефонный звонок клиенту по запросу телемаркетера". Это требование можно впоследствии смоделировать с помощью диаграммы последовательностей, которая активизируется из GUI-интерфейса с помощью кнопки запуска действия, помеченной как Next Call, и триггера, запрограммированного в базе данных. Если между всеми этими элементами существует отношение прослеживаемости, изменение любого элемента вновь открывает отношение для обсуждения — траектория системы "подпадает под подозрение" (говоря языком одного из инструментальных средств).

Отношение прослеживаемости может охватывать многие модели последовательных этапов ЖЦ. Непосредственной модификации подлежат только смежные связи по прослеживаемости. Например, если элемент А восходит к элементу В, а элемент В

восходит к элементу C, изменение, внесенное в любой из конечных точек отношения, должно осуществляться в два этапа: сначала посредством модификации связи A–B, а затем B–C. (Более подробно прослеживаемость и управление требованиями рассматриваются в главе 10).

3.5. Бизнес-модель требований

На этапе *установления требований* осуществляется выявление требований и их определение, преимущественно, в виде формулировок на естественном языке. Формальное моделирование требований с использованием языка UML проводится позже на этапе *спецификации требований*. Тем не менее, во время установления требований постоянно ведется деятельность по обобщенному визуальному представлению собранных требований, называемая *бизнес-моделированием требований*.

Для установления рамок системы требуется, по меньшей мере, высокоуровневая визуальная модель, позволяющая обозначить ключевые прецеденты и ввести наиболее существенные бизнес-классы. На рис. 3.3 показаны зависимости между этими тремя моделями этапа установления требований и модели остальных этапов ЖЦ разработки.

Ведущая роль диаграмм прецедентов в ЖЦ разработки нашла свое отражение на рис. 3.3, из которого ясно, что источником тестовых прецедентов, пользовательской документации и проектных планов выступают модели прецедентов. Более того, диаграммы прецедентов и модели классов используются параллельно и поочередно играют роль "лидера гонки" в рамках последовательных итераций разработки. Проектирование и реализация также тесно переплетены и могут инициировать обратную связь с моделями спецификации требований.

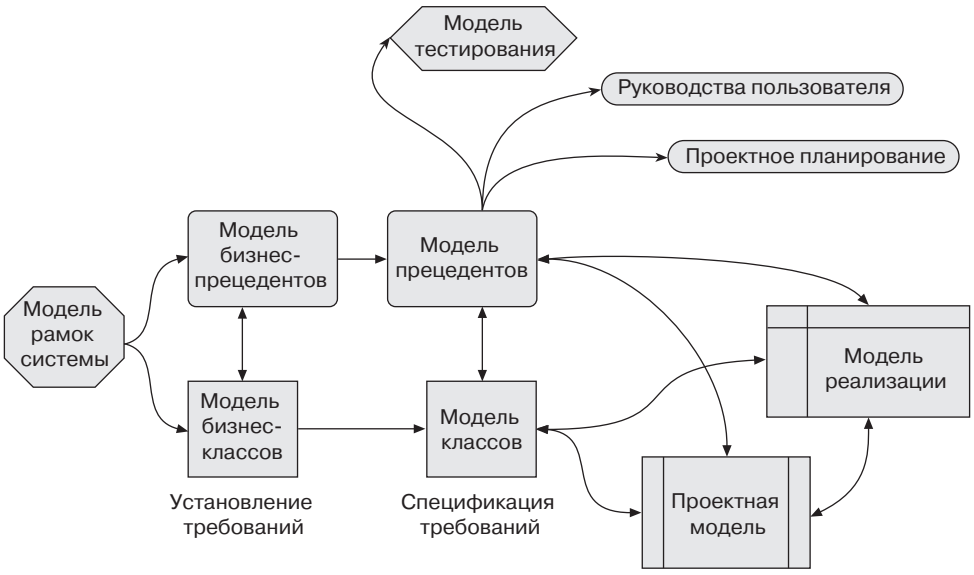

Рис. 3.3. Использование бизнес-моделей на различных этапах выработки требований

3.5.1. Модель рамок системы

Вследствие того, что требования подвержены постоянным изменениям, наверное, наибольшее беспокойство при разработке доставляет так называемое "*расползание рамок*" системы. Хотя некоторые изменения требований неизбежны, необходимо строго следить за тем, чтобы заявленные изменения не выходили за пределы принятых рамок проекта.

На самом деле, попытка ответить на вопрос: "Каким образом мы определяем рамки системы?" показывает, что ответ не лежит на поверхности. Это связано, в первую очередь, с тем, что любая система является всего лишь частью большей по своим масштабам среды — частью совокупности систем, которые вместе образуют эту среду. Системы взаимодействуют, обмениваясь информацией и обращаясь к услугам друг друга. Следовательно, поставленный выше вопрос можно переформулировать следующим образом: "Должны ли мы реализовать требования или же выполнение требуемых функций возлагается на другую систему?".

Чтобы ответить на вопрос о рамках системы, необходимо знать, в каком контексте функционирует наша система. Необходимо знать, какие *внешние сущности* (*external entities*) — другие системы, организации, люди, машины и т.д.— рассчитывают на получение услуг от нас или готовы предоставить услуги нам. В бизнес-системах подобные услуги приобретают форму информации и представляются *потоками данных* (*data flow*).

Поэтому рамки системы можно определить, обозначив внешние сущности и входные/выходные потоки данных между внешними сущностями и нашей системой. Наша система получает входную информацию и выполняет ее необходимую *обработку* с целью выработки выходной информации. Всякое требование, которое не может быть поддержано за счет *внутрисистемных* возможностей обработки, выходит за рамки системы.

Язык UML не обладает средствами построения визуальной модели, позволяющей определить границы системы. Поэтому зачастую для решения этой задачи прибегают к помощи давно известных *контекстных диаграмм* потоков данных — DFD — (разд. 3.3.1). На рис. 3.4 показана контекстная диаграмма для приложения *Телемаркетинг*.

3.5.1.1. Пример моделирования рамок системы

Пример 3.1. Телемаркетинг

Рассмотрите постановку задачи для приложения *Телемаркетинг* (раздел 2.3.4) и постройте для нее контекстную диаграмму. Кроме того, примите к сведению следующие замечания.

1. Кампании планируются на основе рекомендации доверенных лиц общества, которые решают, насколько являются своевременными и подобающими те или иные благотворительные мероприятия. Кампании должны быть согласованы с местными органами исполнительной власти. Разработка проекта и планирование кампании поддерживается отдельной прикладной системой `Campaign Database` (База данных кампаний).
2. Существует также отдельная база данных `Supporter Database` (База данных благотворителей), предназначенная для хранения и ведения информации обо всех благотворителях — прошлых и настоящих. Эта база данных используется для отбора благотворителей, с которыми необходимо установить контакт в ходе конкретной кампании. Отобранный сегмент благотворителей передается для работы с ним во время кампании с использованием телемаркетинга.
3. Заявки от благотворителей на лотерейные билеты регистрируются во время сеансов телемаркетинга для их анализа системой обработки заказов (`Order Processing`). Система обработки заказов отслеживает состояние заявок в базе данных благотворителей.

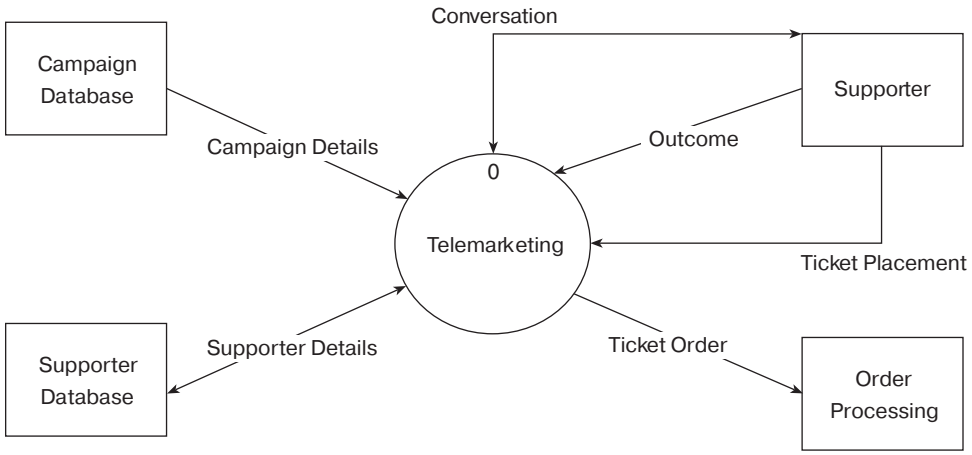

Рис. 3.4. Модель рамок системы – контекстная диаграмма (Телемаркетинг)

Контекстная диаграмма, к которой вы, возможно, также пришли в результате работы над данным упражнением, показана на рис. 3.4. "Шарик" в центре диаграммы представляет нашу систему. Прямоугольники вокруг него обозначают внешние сущности. Стрелки изображают потоки данных. Подробное содержание информации, передаваемой потоками данных, на рисунке не показано — содержимое потоков данных определяется отдельно и хранится в репозитории CASE-системы.

Система `Telemarketing` получает информацию о текущей кампании от внешней сущности `Campaign Database`. Эта информация включает количество и цены билетов, перечень призов для победителей лотереи, продолжительность кампании и т.д.

Аналогично, приложение сущность `Telemarketing` получает подробную информацию о благотворителях от сущности `Supporter Database`. Во время телемаркетингового звонка может появиться новая информация о благотворителе (например, о том, что благотворитель собирается сменить свой телефонный номер). База `Supporter Database` должна быть соответствующим образом актуализирована — поэтому поток данных `Supporter Details` (Детальная информация о благотворителях) двунаправленный.

Основная деятельность разворачивается между приложениями-сущностями `Telemarketing` и `Supporter` (Благотворитель). Поток данных `Conversation` (Разговор) содержит информацию, которой обмениваются собеседники во время телефонного разговора. Ответ благотворителя на предложение телемаркетера приобрести лотерейные билеты передается с потоком данных `Outcome` (Результат). Для регистрации деталей, касающихся билетов, заказанных благотворителем, используется отдельный поток данных `Ticket Placement` (Размещение билетов).

Дальнейшая обработка заказов на билеты выходит за рамки нашей системы. Поток данных `Ticket Order` (Заказ билетов) перенаправляется внешней сущности `Order Processing` (Обработка заказов). Можно предположить, что после ввода заказов, другие сущности могут обработать платежи от благотворителей, отправку билетов по почте, розыгрыш призов и т.д. Это нас не интересует, поскольку текущее состояние заказов, платежей и т.д. доступны нам через внешние сущности `Campaign Database` и `Supporter Database`.

3.5.2. Модель бизнес-прецедентов

Модель бизнес-прецедентов (*business case model*) [47] представляет собой модель прецедентов на верхнем уровне абстракции (разд. 2.2.2). Модель бизнес-прецедентов определяет обобщенные бизнес-процессы — бизнес-прецеденты. Бизнес-прецедент соответствует тому, что иногда называют *возможностями системы* (*system feature*). (Возможности системы определяются в документе, описывающем *видение системы* (*system vision*). Если при разработке системы представляется документ описания видения системы, он может использоваться взамен модели бизнес-прецедентов).

Диаграмма бизнес-прецедентов концентрируется на *архитектуре* бизнес-процессов. Эта диаграмма дает возможность взглянуть на предполагаемое поведение системы так сказать "с высоты птичьего полета". Неформальное описание каждого из *бизнес-прецедентов* должно быть кратким, ориентированным на деловую сторону системы, и концентрироваться на основных потоках видов деятельности. Модель бизнес-прецедента не вполне подходит для того, чтобы объяснить *разработчику*, что же должна делать система.

На этапе спецификации требований *бизнес-прецеденты* превращаются в *прецеденты*. Именно на этом этапе определяются детальные прецеденты, и неформальное описание расширяется за счет включения в него подпроцессов и альтернативных процессов, некоторых копий экранов, демонстрирующих GUI-интерфейс, а также взаимосвязей между введенными прецедентами.

Субъекты (*actor*) диаграммы бизнес-прецедентов отличаются от *внешних сущностей* на диаграмме контекста. Различие заключается в способе взаимодействия субъектов с системой. Субъекты активны. Они управляют процессом. Они активизируют прецеденты, отправляя им сообщения о *событиях*. Прецеденты управляются событиями. Линии, связывающие субъектов и прецеденты, — это не потоки данных. Эти линии связи представляют поток событий, исходящих от субъектов и поток откликов, исходящих от прецедентов.

Субъекты обладают любопытной двойственной природой. *По отношению к системе субъекты могут быть как внешними, так и внутренними сущностями.* Они являются *внешними сущностями*, поскольку они взаимодействуют с системой, оставаясь вне ее. В то же время они — *внутренние сущности*, поскольку система может поддерживать информацию о субъектах, так что она может "со знанием дела" взаимодействовать с "внутренними" субъектами. Системная спецификация — как модель системы — должна описывать систему и ее среду. Среда содержит субъекты. Система сама может хранить информацию о субъектах. Следовательно, спецификация заключает в себе две модели, связанные с субъектами, — модель субъекта и модель данных о субъектах, которые подлежат регистрации.

3.5.2.1. Пример моделирования бизнес-прецедентов

Пример 3.2. Телемаркетинг

Рассмотрите постановку задачи для приложения *Телемаркетинг* (раздел 2.3.4 и 3.5.1.1) и постройте для нее диаграмму бизнес-прецедентов.

Один из возможных вариантов диаграммы бизнес-прецедентов показан на рис. 3.5.

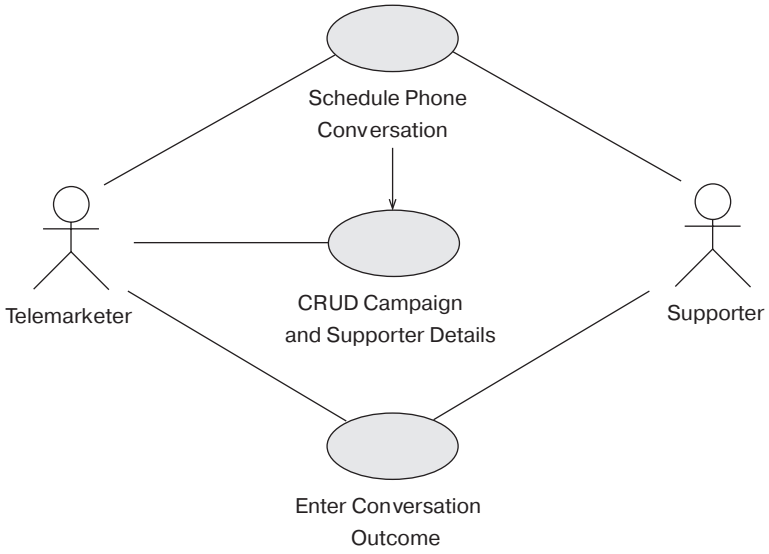

Рис. 3.5. Модель бизнес-прецедентов (Телемаркетинг)

На диаграмме представлены два субъекта: `Telemarketer` (Телемаркетер) и `Supporter` (Благотворитель). Субъект `Telemarketer` просит систему запланировать телефонный звонок благотворителю и осуществить соединение. При успешном соединении `Supporter` участвует как субъект. Бизнес-прецедент `Schedule Phone Conversation` (Планирование телефонного разговора) (который в данном случае включает установление соединения) становится частью функции, видимой извне обоими субъектами.

Во время разговора субъекту `Telemarketer` может потребоваться получить доступ и внести изменения в детализированную информацию о кампании и благотворителе. Эти функции фиксируются в виде бизнес-прецедента (`CRUD Campaign and Supporter Details` (Подробная информация о кампании и благотворителе). (CRUD — популярная аббревиатура, которая обозначает четыре основных операции над данными: Create, Read, Update, Delete (“Создать”, “Читать”, “Обновить” “Удалить”).

Наконец, бизнес-прецедент `Enter Conversation Outcome` (Ввод результатов разговора) предназначен для ввода успешных или неуспешных результатов телемаркетинговой деятельности. Этот прецедент имеет очевидное значения для обоих субъектов.

Указание отношений между прецедентами `CRUD Campaign and Supporter Details` и `Enter Conversation Outcome` не обязательно. В общем случае все отношения между прецедентами можно опустить, чтобы избежать беспорядка и загромождения диаграммы. Как правило, прецеденты некоторым образом связаны с большинством остальных прецедентов, и включение в диаграмму всех отношений противоречит ее предназначению.

3.5.3. Модель бизнес-классов

Модель бизнес-классов (*business class model*) — это модель классов. Точно так же, как в случае модели бизнес-прецедентов, различие заключается в уровне абстракции. Модель

бизнес-классов определяет основные "бизнес-объекты" системы — структуры данных бизнес-процессов, которые составляют основу системы и определяют ее деятельность.

Модель бизнес-классов представляется на высоком уровне абстракции. На этом уровне нас меньше интересует содержание атрибутов классов — достаточно имен классов и их краткого описания.

Довольно любопытно отметить, что нередкой бывает ситуация, при которой субъекты модели бизнес-прецедентов представляются как классы в модели бизнес-классов. Это подтверждает нашу мысль о том, что субъекты зачастую выступают по отношению к системе как внешние и внутренние сущности (разд. 3.5.2).

3.5.3.1. Пример моделирования бизнес-классов

Пример 3.3. Телемаркетинг

Рассмотрите постановку задачи, диаграмму контекста и диаграмму бизнес-прецедентов для приложения *Телемаркетинг* (раздел 2.3.4, 3.5.1.1 и 3.5.2.1) и постройте для нее диаграмму бизнес-классов. Следующие соображения можно использовать как подсказки.

1. Акцент в системе сделан на планирование звонков. Само по себе планирование звонков является вычислительной процедурой, т.е. здесь решение является строго алгоритмическим по природе. Тем не менее, очереди запланированных звонков и результаты звонков должны запоминаться в некоторой структуре данных.
2. Как обсуждалось выше, может потребоваться хранить информацию о субъектах в классах.

Модель бизнес-классов в первом приближении показана на рис. 3.6. Диаграмма содержит шесть классов, два из которых (`Supporter` и `Telemarketer`) являются производными от субъектов модели бизнес-прецедентов. Алгоритм планирования звонков получает телефонный номер и другую информацию от класса `Supporter` и планирует звонок одному из имеющихся в наличии телемаркетеров (`Telemarketer`). Алгоритм в конечном счете долен быть реализован в базе данных системы в виде *хранимой процедуры* (*stored procedure*) (разд. 9.1.2.1).

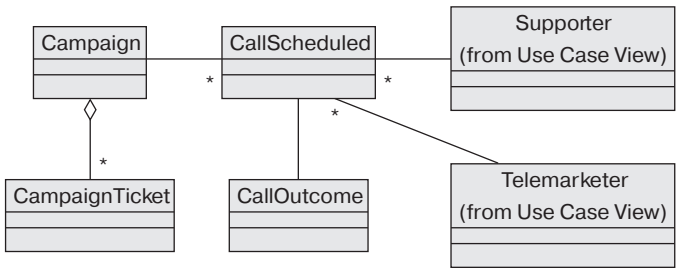

Рис. 3.6. Модель бизнес-классов (Телемаркетинг)

Класс `CallScheduled` (Планируемый звонок) содержит текущую очередь звонков, включая звонки, которые в текущий момент являются активными. Результаты звонков регистрируются в классе `CallOutcome` (Результат звонка), а также распространяются в другие задействованные классы, такие как `CampaignTicket` (Билет кампании) и `Supporter`.

Класс Campaign содержит класс CampaignTickets и связан с классом Scheduled отношением "один ко многим". Аналогично, объекты классов Supporter и Telemarketer могут обладать многими объектами класса CallScheduled. Ассоциация между классами CallScheduled и CallOutcome имеет кратность "один к одному".

3.6. Документ описания требований

Документ, описывающий требования, является осязаемым результатом этапа установления требований. Большинство организаций вырабатывают документ описания требований в соответствии с заранее определенным шаблоном. Шаблон определяет структуру (содержание) и стиль документа.

Ядро документа описания требований состоит из формулировок (изложения) требований. В разделе 3.1 указывалось, что требования могут быть сгруппированы в виде *формулировок сервисов* (зачастую называемых функциональными требованиями) и *формулировок ограничений*. Формулировки сути сервисов могут быть затем разделены на *требования к функциям* (*function requirements*) и *требования к данным* (*data requirements*). (В литературе термин "*функциональные требования*" (*functional requirements*) в широком и в узком смысле используется как взаимозаменяемый. При использовании в узком смысле он соответствует тому, что мы называем требованиями к функциям).

Не говоря уже о самих требованиях, документ описания требований должен обращаться к проектным вопросам. Обычно проектные вопросы рассматриваются в начале документа, а затем в конце документа. Во вводной части документа рассматривается бизнес-контекст проекта, включая цель проекта, участников проекта и основные ограничения. Ближе к заключительной части документа поднимаются другие проектные вопросы, включая план-график выполнения проектных работ, бюджет, риски, документацию и т.д.

3.6.1. Шаблоны документа

Шаблоны для документов описания требований широко доступны. Их можно найти в учебниках, стандартах, выпускаемых такими организациями как ISO, IEEE и т.д., на Web-страницах консалтинговых фирм, программных средствах разработки и т.д. Со временем каждая организация разрабатывает свои собственные стандарты, которые соответствуют принятой в организации практике, корпоративной культуре, кругу читателей, типам разрабатываемых систем и т.д.

Шаблон документа описания требований определяет структуру документа и содержит подробные указания о содержании каждого из разделов документа. Указания могут включать содержание вопросов, мотивацию, примеры и дополнительные соображения [71].

На рис. 3.7 показано типичное оглавление документа описания требований. Последующие разделы включают объяснение к приведенному оглавлению.

3.6.2. Предварительные замечания к проекту

Часть документа описания требований, содержащая предварительные замечания к проекту, преимущественно дает ориентиры тем руководителям и участникам проекта, ответственным за принятие решений, которые, вероятно, не станут подробно изучать документ целиком. В начале документа необходимо ясно обозначить *цели и рамки проекта*, а затем описать деловой контекст системы.

Документ описания требований должен создать *прецедент для системы*. В частности, необходимо упомянуть обо всех усилиях, приложенных для обоснования необходимости системы на этапе *планирования системы* (разд. 1.2). Документ описания требований должен прояснить вопрос о том, каким образом предлагаемая система может способствовать достижению деловых целей и решению задач организацией.

Необходимо обозначить *участников проекта* (разд. 1.1.2) системы. Важно, чтобы *заказчик* выступал не в виде безликого подразделения или офиса — необходимо привести конкретные имена. К концу дня человек должен быть в состоянии решить, приемлемо ли поставляемое ПО для организации.

Документ описания требований

Содержание документа

1. Предварительные замечания к проекту
- 1.1. Цели и рамки проекта
- 1.2. Деловой контекст
- 1.3. Участники проекта
- 1.4. Идеи в отношении решений
- 1.5. Обзор документа

2. Системные сервисы
- 2.1. Рамки системы
- 2.2. Функциональные требования
- 2.3. Требования к данным

3. Системные ограничения
- 3.1. Требования к интерфейсу
- 3.2. Требования к производительности
- 3.3. Требования к безопасности
- 3.4. Эксплуатационные требования
- 3.5. Политические и юридические требования
- 3.6. Другие ограничения

4 Проектные вопросы
- 4.1. Открытые вопросы
- 4.2. Предварительный план-график
- 4.3. Предварительный бюджет

Приложения
- Глоссарий
- Деловые документы и формы
- Ссылки

Рис. 3.7. Содержание документа описания требований

Хотя документ описания требований может быть как угодно далек от технических решений, все же важно *обсудить идеи, касающиеся решения*, на самых ранних этапах ЖЦ разработки. Особый интерес представляют готовые решения. Всегда неплохо рассмотреть вариант приобретения готового продукта вместо его разработки "с нуля".

Документ описания требований должен предоставлять перечень существующих программных пакетов и компонент, которые должны быть в дальнейшем изучены в качестве вариантов возможных решений. Обратите внимание, что приобретение го-

тового решения изменяет процесс разработки, однако это не избавляет от необходимости проведения анализа требований и проектирования системы!

Наконец, неплохо в заключение раздела предварительных замечаний к проекту документа описания требований привести обзор оставшейся части документа. Это может подтолкнуть занятого читателя к тому, чтобы изучить остальные части документа, а также способствует лучшему пониманию содержания документа. Обзор также может содержать пояснения в отношении методологии анализа проектирования, выбранной разработчиками.

3.6.3. Системные сервисы

Основная часть документа описания требований посвящена определению *системных сервисов* (разд. 1.3.1 и 3.1). Эта часть может занимать до половины всего объема документа. Это также, пожалуй, единственная часть документа, которая может содержать обобщенные модели — *модели бизнес-требований* (разд. 3.5).

Рамки системы можно моделировать с помощью *диаграммы контекста* (разд. 3.5.1). В пояснениях к диаграмме контекста должны быть четко определены рамки системы. Без подобного определения проект не может быть застрахован от попыток "растянуть" его рамки.

Функциональные требования можно моделировать с помощью *диаграммы бизнес-прецедентов* (разд. 3.5.2). Однако, диаграмма охватывает перечень функциональных требований только в самом общем виде. Как отмечалось в разд. 3.4, все требования необходимо обозначить, классифицировать и определить.

Требования к данным можно моделировать с помощью диаграммы бизнес-классов (разд. 3.5.3). Так же, как и в случае функциональных требований, диаграмма бизнес-классов не дает полного определения структур данных для бизнес-процессов. Каждый бизнес-класс требует дальнейших пояснений. Необходимо описать атрибутное наполнение классов и определить идентифицирующие атрибуты классов. В противном случае невозможно правильно представить ассоциации.

3.6.4. Системные ограничения

Системные сервисы определяют, *что* должна делать система. *Системные ограничения* (разд. 1.3.1 и 3.1) определяют, *насколько* система ограничена при выполнении обслуживания. Системные ограничения связаны со следующими видами требований.

- Требования к интерфейсу.
- Требования к производительности.
- Требования к безопасности.
- Эксплуатационные требования.
- Политические и юридические требования.

Требования к интерфейсу определяют, как система взаимодействует с пользователями. В документе описания требований определяется только "впечатление и ощущение" от GUI-интерфейса. Начальное проектирование (закрашивание экрана) GUI-интерфейса проводится во время *спецификации требований* и позже во время *системного проектирования*.

В зависимости от области приложения требования к производительности могут играть довольно значительную роль в успехе проекта. В узком смысле они задают скорость (время отклика системы), с которой должны выполняться различные задания. В

широком смысле, требования к производительности включают другие ограничения — в отношении надежности, готовности, пропускной способности и т.д.

Требования к безопасности описывают пользовательские права доступа к информации, контролируемые системой. Пользователям может быть предоставлен ограниченный доступ к данным или ограниченные права на выполнение определенных операций с данными.

Эксплуатационные требования определяют программно-техническую среду, если она известна на этапе проектирования, в которой должна функционировать система. Эти требования могут оказывать влияние на другие стороны проекта, такие как подготовка пользователей и сопровождение системы

Политические требования и *юридические требования* зачастую подразумеваются, чем явно формулируются в документе описания требований. Подобная ошибка может обойтись очень дорого. Пока эти требования не выведены явно, программный продукт может быть трудно развернуть по политическим или юридическим причинам.

Возможны и другие виды *ограничений*. Например, в отношении некоторых систем могут предъявляться повышенные требования к легкости их использования (*требования в отношении пригодности к использованию*) или легкости их сопровождения (*требования в отношении пригодности к сопровождению*).

Значение выработки недвусмысленных определений для системных ограничений трудно переоценить. Существует немало примеров проектов, которые провалились из-за упущенных или неверно понятых ограничений. Эта проблема в равной мере относится как к заказчикам, так и к разработчикам. Недобросовестные или нерассудительные разработчики могут разыграть "карту системных ограничений", чтобы получить преимущество в своем стремлении уклониться от ответственности.

3.6.5. Проектные вопросы

Заключительная часть документа описания требований обращается к другим проектным вопросам. Один из важных разделов этой части называется "*Открытые вопросы*". Здесь поднимаются все вопросы, которые могут сказаться на успехе проекта и которые не рассматривались в других разделах документа. Сюда относится ожидаемое возрастание значения некоторых требований, которые в текущий момент выходят за рамки проекта. Сюда можно отнести также любые потенциальные проблемы и отклонения в поведении системы, которые могут начаться в связи с развертыванием системы.

В этой же части необходимо представить предварительный план-график выполнения основных проектных заданий (разд. 1.3.8). Сюда же относится предварительное распределение людских и других ресурсов. Для выработки стандартных плановых графиков можно использовать программные средства управления проектами, например, такие как система PERT (program evaluation-and-review technique — метод оценки и пересмотра планов) или карты Гантта [51].

Прямым результатом составления план-графика может быть разработка предварительного бюджета. Стоимость проекта может быть выражена скорее в виде диапазона значений затрат, а не конкретного значения. При наличии надлежащим образом документированных требований для оценки затрат можно использовать один из подходящих методов (например, *балльную функциональную оценку* (*function point analysis*)).

3.6.6. Приложения

Приложения содержат остальную, полезную для понимания требований, информацию. Основным добавлением здесь служит глоссарий. *Глоссарий* определяет термины, сокращения и аббревиатуры, используемые в документе описания требований. Значение толкового глоссария трудно переоценить. Неверное истолкование терминологии таит в себе большую опасность для проекта.

Одна из особенностей, которую часто упускают из виду при составлении документа описания требований, состоит в том, что в проблемной области, определяемой документом, можно довольно неплохо разобраться с помощью изучения *документов и форм*, используемых в процессах делопроизводства. При возможности следует включать в документ заполненные формы — "пустые" формы не дают такого же уровня понимания бизнес-процессов.

Раздел *ссылок* содержит перечень документов, которые упоминаются или используются при подготовке документа описания требований. К ним могут относиться книги и другие опубликованные источники информации, но — что, пожалуй, даже более важно — необходимо также упомянуть протоколы совещаний, служебные записки и внутренние документы.

Резюме

В этой главе были полностью рассмотрены вопросы, связанные с установлением требований. Установление требований предшествует их спецификации. Установление требований касается их выявления и оформления в виде документа, который носит преимущественно описательный характер. В результате спецификации требований — рассматриваемой в следующей главе — вырабатываются более формализованные модели требований.

Выявление требований связано с их поиском по двум направлениям — в области знаний о проблемной области и в области прецедентов. Эти два направления исследования требований дополняют друг друга и приводят к определению модели деловой деятельности для разрабатываемой системы.

Существуют *различные методы выявления требований*, среди которых можно назвать такие методы как проведение интервью с заказчиками и экспертами в проблемной области, анкетирование, наблюдение, изучение документации и программных систем, создание прототипов, JAD- и RAD-методы.

Полученные от заказчиков требования могут перекрываться и противоречить друг другу. Устранение дублирования и снятие противоречий — задача бизнес-аналитика, который решает ее с помощью *согласования и проверки обоснованности требований*. Для надлежащего решения этой задачи бизнес-аналитик должен построить матрицу зависимости требований, а также приписать требованиям определенные *риски* и *приоритеты*.

Большие проекты связаны с *управлением* большими массивами требований. Для таких проектов весьма существенным моментом является *обозначение и классификация* формулировок требований. Затем можно определить *иерархию требований*. Осуществление подобных шагов обеспечивает необходимый уровень прослеживаемости требований на последующих стадиях проекта, а также надлежащую обработку *запросов на изменение требований*.

Несмотря на то что установление требований не включает формального моделирования систем, уже на этом этапе ЖЦ разработки ПО можно построить *бизнес-модель*

основных требований. Бизнес-модель может быть представлена в виде трех общих диаграмм — диаграммы контекста, диаграммы бизнес-прецедентов и диаграммы бизнес-классов.

Документ, который создается в результате выполнения этапа установления требований — *документ описания требований* — начинается с общего описания замечаний к проекту (которое создается, в основном, в интересах представления проекта руководству). Основные части документа описывают *системные сервисы* и *системные ограничения*. Заключительная часть документа связана с остальными проектными вопросами, включая подробности, касающиеся проектного *план-графика* и *бюджета*.

Вопросы

B1. В процессе установления требований мы стремимся согласовать общие требования, связанные с представлениями о проблемной области, и требования, полученные в результате анализа прецедентов. Объясните, в чем состоит различие между этими двумя типами требований? Может ли один из них превалировать над другим в процессе установления требований? Объясните свои доводы.

B2. При проведении интервью рекомендуется избегать небеспристрастных, предвзятых и наводящих вопросов. Можете ли вы представить себе ситуацию, при которой подобные вопросы могли бы принести выгоду проекту?

B3. Что такое прототипирование? Насколько оно полезно для установления требований?

B4. Что такое JAD-метод? В чем заключаются основные преимущества JAD-метода по сравнению с другими методами установления требований?

B5. Что такое “растягивание рамок проекта”? Как справиться с этим явлением при установлении требований?

B6. Перечислите и дайте определения проектных рисков.

B7. Почему требования необходимо пронумеровать?

B8. Что такое “подозрительная траектория изменений системы”?

B9. В чем отличие субъектов диаграммы бизнес-прецедентов от внешних сущностей диаграммы контекста?

B10. Опишите типичную структуру (оглавление) документа описания требований.

Упражнения

Постановка задачи. Измерение затрат на рекламу

Рассмотрите постановку задачи для коммерческой деятельности по измерению затрат на рекламу (Advertising Expenditure Measurement — AEM).

1. Организация, занимающаяся AEM-деятельностью, собирает данные по рекламе в отношении различных торговых точек, предлагающих средства аудиовизуальной информацию, теле- и радиостанций, газет, журналов, а также кинематографа, наружной и Internet-рекламы.
2. Собранные данные позволяют охватить две области анализа, интересующие AEM-клиентов. Клиенты могут заказать отчет о том, действительно ли реклама, за которую они платят, появляется так, как было предусмотрено (это называется мониторингом рекламной кампании). Клиент может также заказать отчет, который очерчивает их конкурентные позиции в отношении рекламы применительно к отрасли, в которой они работают (это называется отчетом о затратах). Отчеты о затратах

касаются уровня, которого достигли затраты рекламодателя или затраты на рекламный товар с точки зрения различных критериев (время, географические регионы, средства информации и т.д.).

3. Отчеты о затратах составляют основу АЕМ-деятельности. Фактически, любой АЕМ-клиент (не только клиент, занимающийся рекламой) может приобрести отчет в виде программного обеспечения для генерации отчетов, изготовленного под заказ или в виде отпечатанных отчетов. Клиентская база АЕМ-предприятия состоит из индивидуальных рекламодателей, рекламных агентств, средств массовой информации, консультантов, покупающих средства информации, а также руководителей подразделений сбыта и маркетинга, органов, планирующих развитие средств информации, покупателей и т.д.
4. АЕМ-предприятие заключает контракты со многими торговыми точками, предлагающими средства информации, касающиеся получения от них на регулярной основе электронных файлов журналов, которые содержат информацию по содержанию рекламы в этих торговых точках. Информация из регистрационных журналов переносится в базы данных АЕМ, а затем подвергается тщательной проверке — отчасти автоматически, отчасти вручную. Задача проверки состоит в том, чтобы подтвердить, что все зафиксированные детали, касающиеся рекламы, обладают достоверностью и логичностью в контексте информационной среды. Существенную часть АЕМ-деятельности составляет ручной ввод (мониторинг) данных по рекламе, для которой не существует электронных учетных журналов.
5. После ввода и проверки реклама подвергается ревалоризации — процессу оценивания затрат, необходимых на ту или иную рекламу.

У1. *Измерение затрат на рекламу (АЕМ)* — обратитесь к приведенной выше постановке задачи. Изобразите контекстную диаграмму для АЕМ-системы. Сделайте пояснения к модели.

У2. *Измерение затрат на рекламу (АЕМ)* — обратитесь к приведенной выше постановке задачи. Изобразите диаграмму бизнес-прецедентов для АЕМ-системы. Сделайте пояснения к модели.

У3. *Измерение затрат на рекламу (АЕМ)* — обратитесь к приведенной выше постановке задачи. Изобразите диаграмму бизнес-классов для АЕМ-системы. Сделайте пояснения к модели.