Анализ требований по Вигерсу (2004). Этапы сбора требований. Цель анализа требований в проектах

Цель анализа требований в проектах — получить максимум информации о заказчике и специфике его задач, уточнить рамки проекта, оценить возможные риски, а также сформировать проектную группу, на которую будет возложена значительная часть предстоящих работ.
На этом этапе происходит идентификация принципиальных требований методологического и технологического характера, формулируются цели и задачи проекта, а также определяются критические факторы успеха, которые впоследствии будут использоваться для оценки результатов внедрения. Анализ требований выполняется на основе совещаний и собеседований с руководителями и специалистами заказчика, а продолжительность этого этапа, в зависимости от сложности задач и масштаба внедрения, может составлять от нескольких дней до нескольких недель.
Определение и описание требований (методологических и технических) — шаги, которые во многом определяют успех всего проекта, поскольку именно они влияют на все остальные этапы. Практика показывает, что недостаточная проработка требований зачастую проявляется лишь тогда, когда проект почти завершен, а значительная часть ресурсов, выделенных на его реализацию, уже затрачена. К сожалению, устранение проблем на этапе разработки обходится гораздо дороже, чем тщательная проработка на стадии анализа.

Особенности интерпретации требований.

IEEE Standard Glossary of Software Engineering Terminology (1990) определяет требования как:
1. Условия или возможности, необходимые пользователю для решения проблем или достижения целей;
2. Условия или возможности, которыми должна обладать система или системные компоненты, чтобы выполнить контракт или удовлетворять стандартам, спецификациям или другим формальным документам;
3. Документированное представление условий или возможностей для пунктов 1 и 2.
Это определение охватывает требования как пользователей *(внешнее поведение системы)*, так и разработчиков *(некоторые скрытые параметры)*. Термин пользователи следует распространить на всех заинтересованных лиц, так как не все, кто заинтересован в проекте — пользователи.
Требования — это спецификация того, что должно быть реализовано. В них описано поведение системы, свойства системы или ее атрибуты. Они могут быть ограничены процессом разработки системы.

Уровни требований.

Три уровня требований к ПО:

- Бизнес-требования

- Требования пользователей
- Функциональные требования

Типы требований.

Каждая система имеет свои **функциональные и нефункциональные требования.**

Бизнес-требования (business requirements) содержат высокоуровневые цели организации или заказчиков системы. Как правило, их высказывают те, кто финансируют проект, покупатели системы, менеджер реальных пользователей, отдел маркетинга. Бизнес-требования формулируют в документе об образе и границах проекта — уставе проекта (project charter), или документом рыночных требований (market requirements document). Определение границ проекта представляет собой первый этап управление общими проблемами расползания границ.

Требования пользователей (user requirements) описывают цели и задачи, которые пользователям позволит решить система. К отличным способам представления этого вида требований относятся варианты использования, сценарии и таблицы «событие — отклик». Таким образом, в этом документе указано, что клиенты смогут делать с помощью системы.

Функциональные требования (functional requirements) определяют функциональность ПО, которую разработчики должны построить, чтобы пользователи смогли выполнить свои задачи в рамках бизнес-требований. Иногда именуемые требованиями поведения (behavioral requirements), они содержат положения с традиционным «должен» или «должна»: «Система должна по электронной почте отправлять пользователю подтверждение о заказе».

Термином системные требования (system requirements) обозначают высокоуровневые требования к продукту, которые содержат многие подсистемы, то есть система (IEEE, 1998c). Говоря о системе, мы подразумеваем программное обеспечение или подсистемы ПО и оборудования. Люди — часть системы, поэтому определенные функции системы могут распространяться и на людей.

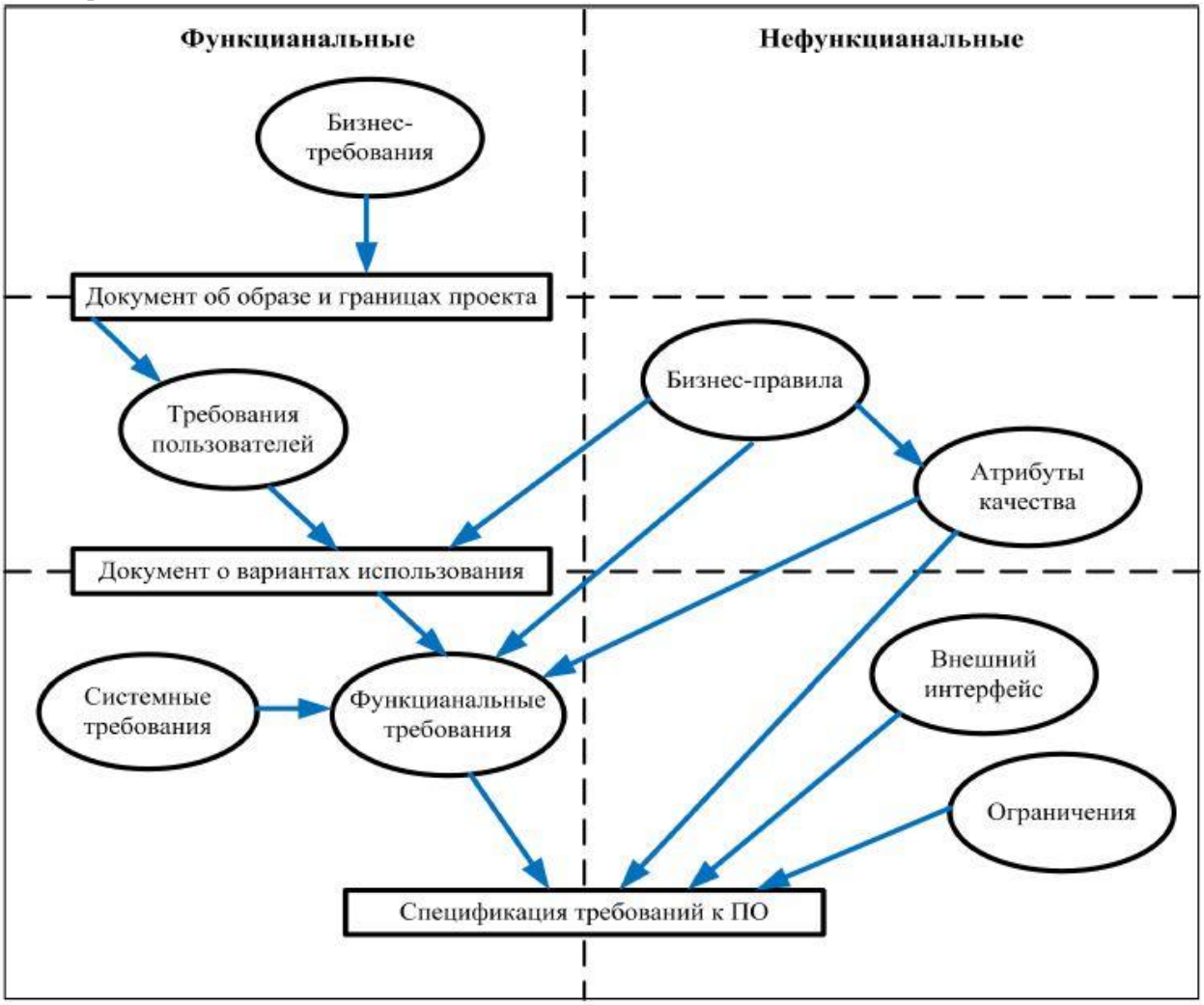
Бизнес-правила (business rules) включают корпоративные политики,правительственные постановления, промышленные стандарты и вычислительные алгоритмы. Бизнес-правила не являются требованиями к ПО, потому что они находятся снаружи границ любой системы ПО. Однако они часто налагают ограничения, определяя, кто может выполнять конкретные варианты использования, или диктовать, какими функциями должна обладать система, подчиняющаяся соответствующим правилам. Иногда бизнес-правила становятся источником атрибутов качества, которые реализуются в функциональности. Следовательно, вы можете отследить происхождение конкретных функциональных требований вплоть до соответсвующих им бизнес-правил.

Нефункциональные требования содержат цели и атрибуты качества. **Атрибуты качества (quality attributes)** представляют собой дополнительное описание функций продукта, выраженное через описание его характеристик, важных для пользователей или разработчиков. К таким характеристикам относятся легкость и простота использования, легкость перемещения, целостность, эффективность и устойчивость к сбоям. Другие нефункциональные требования описывают внешние

взаимодействия между системой и внешним миром, а также ограничения дизайна и реализации. **Ограничения (constraints)** касаются выбора возможности разработки внешнего вида и структуры продукта.

Спецификация требований не содержит деталей дизайна или реализации (кроме известных ограничений), данных о планировании проекта или сведений о тестировании.

Типы требований:



Приемы создания требований

Приемы формулирования требований

Обучение

— Обучите аналитиков требований

— Ознакомьте представителей пользователей и менеджеров с требованиями

— Обучите разработчиков основам предметной области

— Создайте словарь бизнес-терминов

Управление требованиями

— Определите процесс управления изменениями

— Установите границы для контроля изменений

— Проанализируйте, какое влияние оказывают изменения

— Определите основную версию и управляйте версиями требований

— Отслеживайте хронологию изменений

— Отслеживайте состояние требований

— Определите изменяемость требований

— Используйте утилиту управления требованиями

— Создайте матрицу связей требований

Управление проектом

— Выберите соответствующий цикл разработки проекта

— Планируйте на основании требований

— Своевременно пересматривайте обязательства

— Управляйте рисками, касающимися требований

— Отслеживайте объем работ по реализации требований

— Делайте выводы из полученного опыта

Сбор информации

— Определите процесс формулирования требований

— Определите образ и границы проекта

— Определите классы пользователей

— Выделите из пользователей ярого сторонника продукта

— Создайте фокус-группы

— Определите назначение продукта

— Определите системные события и реакцию на них

— Проводите совместные семинары, упрощающие сбор требований

— Наблюдайте за пользователями на рабочих местах

— Изучите отчеты о проблемах

— Используйте требования многократно

Анализ

— Нарисуйте контекстную диаграмму

— Создайте прототипы

— Проанализируйте осуществимость

— Расставьте приоритеты для требований

— Смоделируйте требования

— Создайте словарь терминов

— Распределите требования по подсистемам

— Воспользуйтесь технологией развертывания функций качества (Quality Function Deployment)

Спецификации

— Используйте шаблон спецификации требований к ПО

— Определите источники требований

— Задайте каждому требованию уникальный идентификатор

— Задокументируйте бизнес-правила

— Укажите атрибуты качества

Проверка

— Изучите документы с требованиями

— Протестируйте требования

— Определите критерии приемлемости

Обучение

Обучение аналитиков требований. Всем членам команды, которые будут исполнять функции аналитиков, необходимо научиться приемам формулирования требований — это может занять несколько дней. Квалифицированный аналитик требований терпелив и методичен, обладает навыками межличностного общения и коммуникативными навыками, сведущ в предметной области и знает множество способов формулирования требований к ПО.

Ознакомление пользователей и менеджеров с требованиями. Пользователи, которые будут принимать участие в разработке ПО, должны пройти непродолжительный тренинг (один-два дня), чтоб научиться формулировать требования. Он полезен и для менеджеров по разработке и по работе с клиентами. Обучение поможет понять особое значение выделения требований, суть процесса их разработки, а также опасность пренебрежения ими. Посетив мои семинары по требованиям, некоторые пользователи замечали, что стали теплее относиться к разработчикам ПО.

Ознакомление разработчиков с концепциями предметной области. Чтобы помочь разработчикам в общих чертах понять предметную область, проведите семинар, на котором познакомьте их с бизнесом клиента, терминологией и назначением создаваемого продукта. Это уменьшит вероятность путаницы, непонимания и доработок. Можно также на время проекта назначить каждому разработчику «личного пользователя», который будет разъяснять профессиональные термины и бизнес-концепции. Лучше, если это будет настоящий фанат продукта.

Создание бизнес-словаря. Словарь со специализированными терминами из предметной области снизит вероятность непонимания, Включите в него синонимы, термины, имеющие несколько значений, и термины, имеющие в предметной области и повседневной жизни разные значения.

Выявление требований

Определение процесса формулирования требований. Задокументируйте этапы выявления, анализа, определения и проверки требований. Наличие инструкций по выполнению ключевых операций по может аналитикам качественно и согласованно выполнить их работу Кроме того, вам

будет проще поставить задачи по созданию требований и графики, а также продумать необходимые ресурсы.

Определение образа и границы проекта. Документ об образе и границах проекта содержит бизнес-требования к продукту. Описание образа проекта позволит всем заинтересованным лицам в общих чертах понять назначение продукта. Границы проекта определяют, что следует реализовать в этой версии, а что — в следующих. Образ и границы проекта — хорошая база для оценки предлагаемых требований, Образ продукта должен оставаться от версии к версии относительно стабильным, но для каждого выпуска необходимо составлять отдельный документ о границах.

Определение классов пользователей и их характеристик. Чтобы не упустить из виду потребности отдельных пользователей, выделите их в группы. Например, по частоте работе с ПО, используемым функциям, уровню привилегий и навыкам работы. Опишите их обязанности, местоположение и личные характеристики, способные повлиять на архитектуру продукта.

Выбор сторонника продукта (product champion) в каждом классе пользователей. Это человек, который сможет точно передавать настроения и нужды клиентов. Он представляет потребности определенного класса пользователей и принимает решения от их лица. В случае разработки внутрикорпоративных информационных систем, когда все пользователи — ваши коллеги, такого человека выбрать проще. При коммерческой разработке пораспросите клиентов или используйте площадки бета-тестирования. Выбранные вами люди должны принимать постоянное участие в проекте и обладать полномочиями для принятия решений, касающихся пользовательских требований.

Создание фокус-групп типичных пользователей. Определите группы типичных пользователей предыдущих версий вашего продукта или похожих. Выясните у них подробности о функциональности и качественных характеристиках разрабатываемого продукта. Фокус-группы особенно значимы при разработке коммерческих продуктов, когда приходится иметь дело с большой и разнородной клиентской базой. В отличие от сторонников продукта, у фокус-групп обычно нет полномочий на принятие решений.

Работа с пользователями для выяснения назначения продукта. Выясните у пользователей, какие задачи им требуется выполнять средствами ПО. Обсудите, как должен клиент взаимодействовать с системой для выполнения каждой такой задачи. Воспользуйтесь стандартным шаблоном для документирования всех задач и для каждой сформулируйте функциональные требования. Похожий способ, часто применяемый в правительственных проектах, — создать документ с концепциями операций (ConOps), где указаны характеристики новой системы сточки зрения пользователя (IEEE, 1998a).

Определение системных событий и реакции на них. Определите возможные внешние события и ожидаемую реакцию системы на них. Это могут быть сигналы и данные, получаемые от внешнего оборудования, а также временные события, вызывающие ответную реакцию, например ежевечерняя передача данных, генерируемых системой, внешнему объекту. В бизнес-приложениях бизнес-

события напрямую связаны с задачами.

Проведение совместных семинаров. Совместные семинары по выявлению требований, где тесно сотрудничают аналитики и клиенты — отличный способ выявить нужды пользователей и составить наброски документов с требованиями (Gottesdiener, 2002). Конкретные примеры таких семинаров — Joint Requirements Planning (JRP — совместное планирование требований) (Martin, 1991) и Joint Application Development (JAD — совместная разработка приложений) (Wood и Silver, 1995).

Наблюдение за пользователями на рабочих местах. Наблюдая за работой пользователей, выявляют контекст потенциального применения нового продукта (Beyer и Holtzblatt, 1998). Простые диаграммы рабочих потоков, а также диаграммы потоков данных позволяют выяснить, где, как и какие данные задействовал пользователь. Документируя ход бизнес-процесса, удается определить требования к системе предназначенной для поддержки этого процесса. Иногда даже выясняется, что для выполнения деловых задач клиентам вовсе и не требуется новое ПО (McGraw и Harbison, 1997).

Изучение отчетов о проблемах работающих систем с целью по иска новых идей. Поступающие от клиентов отчеты о проблемах и предложения о расширении функциональности — отличный источник: идей о возможностях, которые можно реализовать в следующей версии или новом продукте. За подобной информацией стоит обратиться и к персоналу службы поддержки.

Повторное использование требований в разных проектах. Если необходимая клиенту функциональность аналогична уже реализованной в другом продукте, подумайте, готовы ли клиенты гибко пересмот реть свои требования для использования существующих компонентов Требования, соответствующие бизнес-правилам компании, можно применить в нескольких проектах. Это требования к безопасности, определяющие порядок доступа к приложениям, и требования, соответствующие постановлениям правительства, например Закон о гражданах США с ограниченными возможностями (Americans with Disabililies Act).

Анализ требований

Создание контекстной диаграммы. Контекстная диаграмма — простая модель анализа, отображающая место новой системы в соответствующей среде. Она определяет границы и интерфейсы между разрабатываемой системой и сущностями, внешними для этой системы, например пользователями, устройствами и прочими информационными системами.

Создание пользовательского интерфейса и технических прототипов. Если разработчики или пользователи не совсем уверены насчет требований, создайте прототип — частичную, возможную или предварительную версию продукта, которая сделает концепции и возможности более осязаемыми. Оценка прототипа поможет всем заинтересованным лицам достичь взаимопонимания по решаемой проблеме.

Анализ осуществимости требований. Проанализируйте, насколько реально реализовать каждое требование при разумных затратах и с приемлемой производительностью в предполагаемой среде. Рассмотрите риски, связанные с реализацией каждого требования, включая конфликты с другими

требованиями, зависимость от внешних факторов и препятствия технического характера.

Определение приоритетов требований. Воспользуйтесь аналитическим подходом и определите относительные приоритеты реализации функций продукта, решаемых задач или отдельных требований. На основании приоритетов установите, в какой версии будет реализована та или иная функция или набор требований. Подтверждая изменения, распределите все их по конкретным версиям и включите в план выпуска этих версий затраты, необходимые на внесение изменений. В ходе работы над проектом периодически корректируйте приоритеты в соответствии с потребностями клиента, условиями рынка и бизнес-целями.

Моделирование требований. В отличие от подробной информации, представленной в спецификации требований к ПО или пользовательского интерфейса прототипа, графическая модель анализа отображает требования на высоком уровне абстракции. Модели позволяют выявить некорректные, несогласованные, отсутствующие и избыточные требования. К таким моделям относятся диаграммы потоков данных, диаграммы «сущность — связь», диаграммы перехода состояний, называемые также автоматами (statecharts), карты диалогов, диаграммы классов, диаграммы последовательностей, диаграммы взаимодействий, таблицы решений и деревья решений.

Создание словаря терминов. В нем соберите определения всех элементов и структур данных, связанных с системой, что позволяет всем участникам проекта использовать согласованные определения данных. На стадии работы над требованиями словарь должен содержать определения элементов данных, относящихся к предметной области, чтобы клиентам и разработчикам было проще общаться.

Распределение требований по подсистемам. Требования к сложному продукту, включающему несколько подсистем, следует соразмерно распределять между программными, аппаратными и операторскими подсистемами и компонентами (Nelsen, 1990). Как правило, это осуществляет системный инженер или разработчик.

Применение технологий развертывания функций качества. Технология развертывания функций качества {Quality Function Deployment, QFD) — точная методика, соотносящая возможности и атрибуты продукта с их значимостью для клиента (Zultner, 1993; Pardee, 1996). Она позволяет аналитически выявить функции, которые максимально удовлетворят потребности клиента. Технология развертывания функций качества рассчитана на три класса требований: ожидаемые, о которых клиент может не упомянуть, но будет расстроен, если их не окажется в продукте, обычные требования и отдельные, специальные требования, которые обеспечивают удобство работы клиентам, но отсутствие которых не влечет санкций со стороны клиента.

Спецификации требований

Использование шаблона спецификации требований к ПО. Создайте стандартный шаблон для документирования требований к ПО в вашей организации. Шаблон предоставляет согласованную структуру, позволяющую фиксировать описания нужной функциональности, а также прочую

информацию, касающуюся требований. Вместо того чтобы изобретать новый шаблон, модифицируйте один из существующих в соответствии со спецификой проекта. Многие компании начинают с использования шаблона спецификации требований к ПО, описанного в стандарте IEEE 830-1998 (IEEE, 1998b). Если ваша компания занимается разными проектами, например проектирует новое крупное приложение и параллельно дорабатывает версии старых программ, создайте соответствующие шаблоны для всех типов проектов. Шаблоны и процессы должны быть масштабируемыми.

Определение источников требований. Чтобы гарантировать, что все заинтересованные лица понимают, почему то или иное требование зафиксировано в спецификации требований к ПО, и упростить последующее прояснение требований, выявите источники всех требований. Это может быть вариант использования или другая информация от пользователей, системное требование высокого уровня, бизнес-правило или иной внешний фактор. Указав всех лиц, заинтересованных в каждом требовании, вы будете знать, к кому обратиться при поступлении запроса на изменение. Источники требований устанавливают на основе связей или определяют для этой цели атрибут требования.

Присвоение уникальных идентификаторов всем требованиям. Выработайте соглашение о присвоении уникальных идентификаторов требованиям, зафиксированным в спецификации требований к ПО. Соглашение должно быть устойчивым к дополнению, удалению элементов и изменениям, вносимым в требования. Присвоение идентификаторов позволяет отслеживать требования и фиксировать вносимые изменения.

Указание атрибутов качества. Выявляя качественные характеристики, удовлетворяющие потребности клиента, не ограничивайтесь только обсуждением функциональности. Выясните ожидаемые производительность, эффективность, надежность, удобство использованичи др. Информация от клиентов об относительной важности тех или иных качественных характеристиках позволит разработчику принять правильные решения, касающиеся архитектуры приложения.

Документирование бизнес-правил. К бизнес-правилам относятся корпоративные политики, правительственные распоряжения и алгоритмы вычислений. Ведите список бизнес-правил отдельно от спецификации требований к ПО, поскольку правила обычно существуют вне рамок конкретного проекта. Для выполнения некоторых приходится создавать реализующие их функциональные требования, и поэтом необходимо определить связь между этими требованиями и соответствующими правилами.

Проверка требований

Изучение документов с требованиями. Официальная проверка документирования требований — один из наиболее ценных способов проверки качества ПО. Соберите небольшую команду, члены которой представляют различные направления (например, аналитик, клиент, разработчик и специалист по тестированию}, и тщательно изучите спецификацию требований к ПО, модель анализа

и соответствующую информацию на предмет недостатков. Также полезно провести в ходе формулирования требований их неофициальный предварительный просмотр. И хотя реализовать это на практике непросто, данный прием — один из самых ценных, так что начинайте внедрять проверку требований в вашей организации прямо сейчас.

Тестирование требований. На основе пользовательских требований создайте сценарии функционального тестирования и задокументируйте ожидаемое поведение продукта в конкретных условиях. Совместно с клиентами изучите сценарии тестирования и убедитесь, что они отражают нужное поведение системы. Проследите связь сценариев тестирования с функциональными требованиями и удостоверьтесь, что ни одно требование не пропущено и что для всех требований есть соответствующие сценарии тестирования. Запустите сценарии, чтобы удостовериться в правильности моделей анализа и прототипов.

Определение критериев приемлемости. Предложите пользователям описать, как они собираются определять соответствие продукта их потребностям и его пригодность к работе. Тесты на приемлемость следует основывать на сценариях использования (Hsia, Kung и Sell, 1997).

Управление требованиями

Определение процесса управления изменениями. Определите процесс представления, анализа и утверждения или отклонения изменений. Применяйте его для управления всеми предлагаемыми изменениями. В контексте процесса управления изменениями полезно использовать коммерческие средства отслеживания недостатков.

Создание совета по управлению изменениями. Из представителей заинтересованных в проекте лиц организуйте совет по управлению изменениями, который будет получать информацию о предполагаемых изменениях требований, оценивать ее, решать, какие изменения принять, а какие отклонить, и определять, в какой версии продукта будет внедрена та или иная функция.

Анализ влияния изменений требований. Анализ влияния изменений помогает совету по управлению изменениями принимать обоснованные решения. Оцените, как каждое предлагаемое изменение требований повлияет на проект. На основе матрицы связей выявите другие требования, элементы архитектуры, исходный код и сценарии тестирования, которые, возможно, придется изменить. Определите, что необходимо для реализации изменений, и оцените затраты на реализацию.

Создание базовой версии и управление версиями требований. Базовая версия содержит требования, утвержденные для реализации в конкретной версии продукта. После определения базовых требований изменения можно вносить только в соответствии с процессом управления изменениями. Присвойте всем версиям спецификации требований уникальные идентификаторы, чтобы избежать путаницы между черновыми вариантами и базовыми версиями, а также между предыдущей и текущей версиями требований. Более надежное решение — управлять версиями документов с требованиями при помощи соответствующих средств управления конфигурацией.

Ведение журнала изменений требований. Фиксируйте даты изменения спецификаций требований, сами коррективы, их причины, а также лиц, вносивших изменения. Автоматизировать эти задачи позволяет утилита управления версиями или коммерческая утилита управления требованиями.
Контроль за состоянием всех требований. Создайте БД, включаю щую по одной записи для каждого дискретного функционального требования. Занесите в БД ключевые атрибуты каждого требования, включая его состояние {например «предложено», «одобрено», «реализовано» или «проверено»), чтобы в любой момент вы могли узнать количество требований в каждом состоянии,
Оценка изменяемости требований. Еженедельно фиксируйте количество требований, внесенных в базовую версию, а также число предложенных и одобренных изменений (добавлений, модификаций и удалений). Если требования формируются не самим клиентом, а от его лица, может оказаться, что проблема понята плохо, границы проекта определены нечетко, бизнес стремительно меняется, при сборе информации многие требования были упущены или внутрикорпоративные политики меняются в худшую сторону.
Использование средств управления требованиями. Коммерческие утилиты управления требованиями позволяют хранить различные типы требований в БД. Для каждого требования можно определить атрибуты, отслеживать его состояние, а также выявить связи между требованиями и другими рабочими продуктами. Данный прием поможет вам автоматизировать прочие задачи по управлению требованиями, описанные ниже.
Создание матрицы связей требований. Создайте таблицу, сопоставляющую все функциональные требования с элементами архитектуры и кода, которые реализуют данное требование, и с тестами, проверяющими его. Матрица связей требований позволяет также сопоставить функциональные требования с требованиями более высоких уровней, на основе которых они созданы, и с другими родственными требованиями. Заполняйте эту таблицу входе, а не в конце работы над проектом.

Управление проектом

Выбор цикла разработки ПО. Вашей компании следует определить несколько жизненных циклов разработки для проектов различного типа и различных степеней неопределенности требований (McCormel, 1996). Каждый менеджер проекта должен выбрать и использовать цикл, оптимальным образом подходящий для его проекта. Включите цикл операции по созданию требований. Если на ранних этапах работы над проектом требования или границы проекта определены нечетко, разрабатывайте продукт постепенно (небольшими этапами), начиная с наиболее понятных требований и устойчивых элементов архитектуры. По возможности реализуйте наборы функций, чтобы периодически выпускать промежуточные версии продукта и как можно раньше предоставлять клиенту работоспособные образцы приложение (Gilb, 1988; Cockburn, 2002).
Планы реализации проекта должны быть основаны на требованиях. Разрабатывайте планы и графики работы над проектом постепенно, по мере прояснения границ и подробных требований. Начните с оценки затрат, необходимых на реализацию функциональных требований, определенных

на основе первоначальных образа и границ продукт. Графики и оценка затрат, построенные на основе нечетких требований, окажутся крайне неточными, однако по мере детализации требований их следует уточнить.

Пересмотр обязательств по проекту при изменении требований. Добавляя в проект новые требования, оцените, удается ли соблюдать обязательства, касающиеся графика и требований к качеству, при доступном объеме ресурсов. Если нет, обсудите реалии проекта с менеджерами и согласуйте новые, достижимые обязательства (Humphrey, 1997; Fisher, Ury, и Patton, 1991; Wiegers, 2002). Если переговоры не увенчаются успехом, сообщите менеджерам и клиентам о их результатах, чтобы нарушение планов в реализации проекта не стало для них неожиданностью.

Документирование и управление рисками, связанными с требованиями. Одна из составляющих управления рисками проекта — выявление и документирование рисков, связанных с требованиями. Уменьшайте или предотвращайте их посредством мозговых штурмов, реализуйте корректирующие действия и отслеживайте их эффективность.

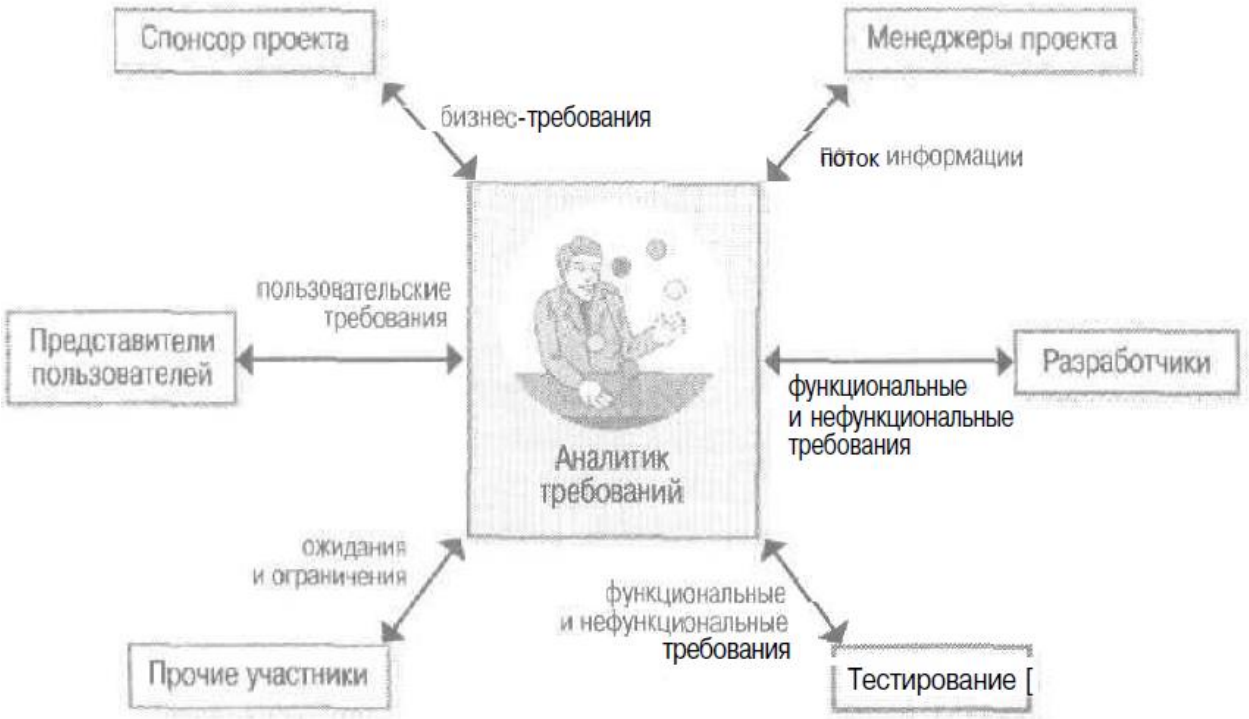
Контроль объема работ по созданию требований. Фиксируйте усилия, прилагаемые вашей командой на разработку требований и управление проектом. Эти данные позволят оценить соответствие планам и эффективнее спланировать необходимые ресурсы для будущих проектов. Также отслеживайте, как ваши действия по регламентации требований влияют на проект в целом. Это позволит оценить отдачу от этой работы.

Извлечение уроков из полученного опыта. Для этого в организации следует провести ретроспективу проектов, называемую также изучением законченных проектов (Robertson и Robertson, 1999; Kerth, 2001; Wiegers и Rothman, 2001). Ознакомление с опытом в области проблем и способов создания требований, накопленным в ходе работы над предыдущими проектами, помогает менеджерам и аналитикам требований более эффективно работать в будущем.

Роль аналитика требований.

Аналитик требований — это основное лицо, отвечающее за сбор, анализ, документирование и проверку требование к проекту. Это основной коммуникативный канал между группой клиентов и командой разработчиков.

Аналитик отвечает за сбор и распространение информации о продукте, а менеджер проекта — за обмен информацией о проекте.



Аналитик требований — это одна из ролей участников проекта, а не обязательно название должности. На эту роль можно назначить одного или нескольких специалистов. Кроме того, функции аналитика могут выполнять другие члены команды параллельно со своими обязанностями, например менеджер проекта, менеджер по продукту, профильный специалист (subject matter expert), разработчик и даже пользователь. В любом случае аналитик должен обладать всеми навыками, знаниями и личными качествами, необходимыми для эффективной работы.

В модели Сосото II, широко применяемой для оценки проектов, указано, что опыт и способности аналитика требований сильно влияют на материальные и трудовые затраты, связанные с реализацией проекта (Boemn et al., 2000). Привлекая опытных специалистов, можно на треть снизить связанные с проектом трудозатраты по сравнению с аналогичными проектами, где заняты неопытные аналитики Способности аналитика оказывают даже большее влияние, чем его опыт: трудозатраты удается сократить вдвое.

Задачи аналитика.

Аналитик — это посредник в общении, проясняющий смутные представления пользователей и обращающий их в четкие спецификации, которыми руководствуется команда разработчиков продукта. Задача аналитика — прежде всего выяснить, для чего нужна пользователям новая система, и затем определить функциональные и качественные требования, на основе которых менеджеры проекта смогут оценить, разработчики — спроектировать и создать, а специалисты по тестированию — проверить продукт. Далее описаны некоторые стандартные обязанности аналитика.

Определить бизнес-требования. Ваша работа в качестве аналитика начинается, когда вы помогаете спонсору, менеджеру продукта или менеджеру по маркетингу определить бизнес-требования к проекту. Возможно, первое, что следует спросить: «Зачем мы начинаем этот проект?» Бизнес-

требования включают бизнес-цели организации и представление о внешнем виде и функциональности системы. Можно разработать шаблон документа об образе и границах и, расспросив людей, имеющих представление о внешнем виде системы, получить у них необходимую информацию.

Определить заинтересованных лиц и классы пользователей. Документ об образе и границах поможет вам выявить важные классы пользователей и прочих заинтересованных в продукте лиц. Затем совместно с заказчиками следует выбрать соответствующих представителей каждого класса, заручиться их поддержкой и согласовать обязанности. Пользователи могут сомневаться, стоит ли участвовать в создании требований, пока не будут точно знать, чего именно вы от них хотите. Запишите, какого именно сотрудничества вы хотите, и ознакомьте их с этим документом.

Выявить требования. Требования к программному продукту не лежат на виду и не ждут, когда какой-нибудь аналитик придет и соберет их. Профессиональный аналитик помогает пользователям четко обрисовать функции системы, необходимые им для достижения бизнес-целей. Для этого у него в арсенале припасено множество способов сбора информации:

- интервью;
- семинары;
- анализ документов;
- опросы;
- посещение рабочих мест клиентов;
- анализ бизнес-процессов;
- анализ документооборота и задач;
- списки событий;
- анализ конкурирующих продуктов;
- исследование существующих систем;
- ретроспективы развития предыдущего проекта.

Обычно пользователи уделяют особое внимание функциональным требованиям к системе, поэтому на дискуссии следует обсудить качественные атрибуты, задачи производительности, бизнес-правила, внешние интерфейсы и ограничения. Нормально, если аналитик спорит с пользователями, однако не стоит навязывать им свое мнении. Некоторые пользовательские требования могут показаться абсурдными, но если клиент утверждает, что они верны, лучше уступить ему.

Анализировать требования. Ищите производные требования, логически проистекающие из запросов клиентов, а также невысказанные ожидания, которые, как считают клиенты, и так будут реализованы. Сразу же проясните неясные и неубедительные слова, порождающее двусмысленность. Выявите конфликтующие требования и области, требующие подробной детализации. Определите функциональные требования с такой степенью подробности, которая необходима разработчикам. От проекта к проекту степень подробности различается. Для Web-узла, постепенно создаваемого небольшой и дружной командой, достаточно документации с ограниченным перечнем требований.

Напротив, для разработки сложной встроенной системы, которую будет строить внешний поставщик, потребуется точная и подробная спецификация требований к ПО.

Создавать спецификации с требованиями. В результате формулировки требований формируется коллективный взгляд на систему. Аналитик отвечает за хорошую организацию спецификаций, в которых этот взгляд четко отражен. В результате применения стандартных шаблонов для вариантов использования продукта и спецификации требований к ПО создание документации ускоряется, поскольку у аналитика перед глазами постоянно находится перечень тем, которые нужно обсудить с представителями пользователей.

Моделировать требования. Аналитик должен определить, полезно ли представлять требования нетекстовыми средствами, например с помощью разнообразных моделей графического анализа, таблиц, математических уравнений, раскадровок и прототипов. Модели анализа отражают информацию на более высоком уровне абстракции, чем подробный текст. Для максимальной прозрачности и эффективного общения рисуйте модели анализа, используя правила стандартного языка моделирования.

Управлять проверкой требований. Аналитик должен гарантировать, что формулировки требований отвечают всем характеристикам и что система на основе этих требований устроит пользователей. Аналитики участвуют в обзорах документов с требованиями. Им также следует изучить архитектуру, код и варианты тестирования, спроектированные на основе спецификаций требований, и убедиться, что требования интерпретированы правильно.

Обеспечить расстановку приоритетов требований. Аналитик обеспечивает общение и взаимодействие различных классов пользователей с разработчиками, что позволяет расставить приоритеты. Возможно, вам будет полезна электронная таблица для назначения приоритетов требованиям.

Управлять требованиями. Аналитик требований вовлечен во все этапы разработки ПО, его задача — помочь создать, обсудить и осуществить план управления требованиями к проекту. Создав документацию, аналитик управляет требованиями и проверяет, как они реализуются в продукте. Хранение требований с помощью специальных коммерческих утилит упрощает управление ими. Управление требованиями подразумевает контроль за состоянием отдельных функциональных требований по мере степени готовности продукта. Расспрашивая коллег, аналитик собирает информацию о связях требований, сопоставляет их с прочими элементами системы. Аналитик — ключевая фигура в управлении изменениями базовых требований, для этого он применяет средства контроля изменений.

Навыки и знания, необходимые аналитику.

Навыки, необходимые аналитику

Аналитик должен уметь применять разные средства сбора информации и представлять эту информацию различными способами на нормальном и понятном языке. Профессионал в этой области

обладает одновременно развитыми коммуникационными навыками, знанием психологии межличностного общения, техническими знаниями, знаниями предметной области бизнеса и личными качествами, подходящими для этой работы. Основные факторы успеха — терпение и искреннее желание работать с людьми.

Умение слушать. Чтобы стать специалистом, научитесь эффективно слушать. Активное слушание подразумевает устранение помех, сохранение вежливой позы и зрительный контакт, а также повторение основных моментов для закрепления их понимания. Вам нужно моментально схватывать, что говорят люди, и уметь читать между строк, чтобы обнаружить вещи, о которых они стесняются говорить. Изучите, как ваши коллеги предпочитают общаться и избегайте налагать собственный фильтр понимания на высказывания клиентов. Ищите допущения которые подчеркивали бы мысли других или их интерпретацию.

Умение опрашивать и задавать вопросы. Большая часть информации о требованиях извлекается в ходе беседе людьми, и поэтому аналитик должен уметь общаться с разными людьми и группами — только так ему удастся выявить их потребности. Возможно, работать со старшими менеджерами или чрезмерно самоуверенными или агрессивными людьми будет трудно. Для выяснения существенных требовании пользователей необходимо задавать правильные вопросы. Например, пользователи обычно делают акцент на ожидаемом поведении системы. Тем не менее значительная часть кода будет обрабатывать исключения, поэтому вам следует определить возможные условия ошибок и реакцию системы на них. По мере приобретения опыта вы научитесь задавать вопросы, которые раскрывают и проясняют неопределенности, расхождения во мнениях, предположения и невысказанные ожидания.

Навыки анализа. Эффективный аналитик способен думать на нескольких уровнях абстракции. Иногда требуется перейти от сведений высшего порядка к подробностям. В некоторых случаях на основе потребности одного из пользователей сформулировать набор требований, которые удовлетворят большинство пользователей данного класса. Критически оценивайте информацию, полученную на основе разных источников, чтобы урегулировать конфликты, отделить мимолетные желания пользователей от их реальных потребностей и отличать варианты решений от требований.

Навыки создания комфортных условий общения. Умение организовать дружескую атмосферу на семинарах для уточнения требований — один из необходимых навыков аналитика. Нейтральный наблюдатель, имеющий опыт опроса, наблюдения и создания комфортных условий общения, создаст доверительные отношения в группе и уменьшит напряженность в отношениях между бизнесменами и ИТ-сотрудниками.

Умение наблюдать. Внимательный аналитик запомнит высказанные мимоходом комментарии, которые могут оказаться важными. Наблюдая за тем, как пользователь выполняет свои обязанности или работает с имеющимся приложением, опытный аналитик выявит моменты, о которых пользователь даже не упомянул. Наблюдательность иногда помогает направить дискуссию в новое русло, чтобы выявить дополнительные требования, о которых никто ничего не сказал.

Навыки написания документации. Основной итог процесса создания требований — письменная спецификация с информацией для клиентов, отдела маркетинга и технического персонала. Аналитик должен отлично владеть языком и ясно выражать сложные идеи.

Я знаком с одной организацией, где аналитиками назначили нескольких специалистов, для которых английский был вторым языком. Написать отличные требования трудно даже на родном языке. Еще сложнее, когда попутно приходится разбираться в нюансах оборотов речи, двусмысленности слов и местных идиомах. Аналитик же должен уметь читать критично и эффективно, поскольку ему приходится просматривать множество материалов и необходимо быстро уяснять их суть.

Организационные навыки. Аналитик имеет дело с большим объемом беспорядочной информации, собранной на первом этапе. Чтобы справиться с данными и выстроить согласованное целое, вам потребуются исключительные организационные навыки, а также терпение и упорство для вычленения основных идей из хаоса.

Навыки моделирования. Аналитик должен уметь работать с разнообразными средствами, начиная с древних блок-схем и структурированных моделей анализа (диаграммы потоков информации, диаграммы «сущность —связь» и т.д.) и заканчивая современным языком UML (Unified Modeling Language, унифицированный язык моделирования). Некоторые из этих средств полезны при общении с пользователями, другие — с разработчиками. Аналитику следует объяснить другим участникам проекта ценность использования этих методов и то, как работать с их данными.

Навыки межличностного общения. Аналитик должен уметь организовать людей с разными интересами для совместной работы, и уверенно чувствовать себя в разговорах с сотрудниками, занимающими разные должности в организации. Подумайте, как сложно иметь дело с сотрудниками из виртуальных групп, различающихся по географическому, временному, культурному или языковому признаку. Опытным аналитикам зачастую приходится наставлять своих коллег-новичков и объяснять клиентам суть процессов создания требований и разработки ПО.

Творческий подход. Аналитик — не просто клерк, записывающий все высказывания клиентов. Лучшие аналитики изобретают требования. Они предлагают инновационные функции продуктов, новые рыночные возможности и возможности для бизнеса и думают, как удивить и удовлетворить своих клиентов. Отличный аналитик творчески подходит к делу: рассказывая о системе, ему удается удивить клиента — тот даже не всегда подозревает, что такая функциональность возможна.

Знания, необходимые аналитику

Аналитику требований требуются обширные знания, большая часть которых приобретается с опытом. Попробуйте понять современные способы создания требований и научитесь применять их на каждом этапе разработки ПО. Эффективный аналитик владеет широким спектром средств и знает когда их стоит использовать, а когда — нет.

Аналитик контролирует требования и управляет проектом на протяжении всего цикла его разработки. Аналитик, знающий тонкости управления проектом и рисками, а также понимающий

методы обеспечения качества продукции, не допустит провала проекта из-за проблем, возникающих на этапе формулирования требований. В области коммерческой разработки ПО аналитику полезно знать концепции управления продуктом и основы позиционирования и разработки корпоративного ПО.

Определение образа продукта вплоть до бизнес-требований

Конфликтующие бизнес-требования

Бизнес-требования, собранные из разных источников, могут **конфликтовать**. Представьте себе компьютер со встроенным ПО, подсоединенный к Интернету и предназначенный для покупателей магазина (киоск). При его разработке следует удовлетворить следующие **бизнес-цели:**
— получение дохода путем сдачи в аренду или продажи киоска продавцу;
— продажу потребительских товаров покупателям;
— привлечение внимания покупателей к торговой марке;
— предоставление широкого ассортимента товаров.
Бизнес-интересы розничного продавца таковы:
— максимальное увеличение дохода с доступного пространства;
— привлечение большего количества покупателей в магазин;
— увеличение объема продаж и уровня прибыли, если киоск заменит операции, выполняемые вручную.

Разработчику иногда хочется сделать киоск высокотехнологичным и увлекательным для покупателей. Продавцу же нужна простая и полностью готовая к эксплуатации система, а для покупателя важны удобство и функциональные возможности. Напряженность между этими тремя сторонами с их различными целями, ограничениями и бюджетом может привести к несогласованности бизнес-требований. Тот, кто финансирует проект, должен разрешить эти конфликты до того, как аналитик детализирует системные требования и требования к ПО киоска. Основное внимание следует уделять фундаментальным задачам, сулящим наибольшую коммерческую выгоду («увеличатся продажи, будут привлечены новые покупатели»)- При этом можно легко отвлечься не внешние характеристики продукта («инновационный пользовательский интерфейс, привлекающий клиентов»), которые не отражают реальные бизнес-цели.
Кроме того, тот (или те), кто финансирует проект, проекта должен разрешать конфликты различных заинтересованных в проекте лиц — не стоит ждать, что разработчики разберутся сами. По мере того как определяется все больше заинтересованных лиц и появляется все больше групп спонсоров с соперничающими интересами, увеличивается риск увеличения объема проекта. Неконтролируемый рост объема из-за того, что заинтересованные лица пытаются удовлетворить каждое пожелание, может привести к тому, что проект рухнет под собственным весом, так и не дав каких-либо ценных

результатов. Разрешение подобных конфликтов часто удается лишь в результате политической и жестокой борьбы.

Бизнес-требования и варианты использования

Бизнес-требования определяют и набор бизнес-задач **(вариантов использования)**, которые позволяет выполнять приложение (ширина приложения), и глубину уровня, до которого реализуется каждый вариант использования. Если бизнес-требования помогают вам определить, что некий вариант использования выходит за границы проекта, значит, что вы принимаете решение о ширине проекта. Глубина простирается от простой реализации до полной автоматизации с множеством вспомогательных средств. Бизнес-требования позволяют понять, для каких вариантов использования необходима надежная и полная функциональность, а для каких достаточно поверхностной реализации, по крайней мере на первое время.

Бизнес-требования влияют на приоритеты реализации вариантов использования и связанные с ними функциональные требования. Например, такая бизнес-цель, как получение максимальной дохода от киоска, подразумевает реализацию на ранней стадии функций, отвечающих за продажу большего количества продуктов или предоставляющих услуги покупателям. Экзотичные, эффектные функции, привлекающие лишь немногих, жадных до технологичных новинок клиентов и не способствующие выполнению основной бизнес-задачи, не должны получать высокий приоритет.

Бизнес-требования также существенно влияют на способ реализации требований. Например, одна из причин создания Chemical Tracking System — уменьшить закупку новых бутылей с химикатами, используя те химикаты, которые уже есть на складе или в другой лаборатории. Опросы и наблюдения должны помочь выяснить, почему в данный момент это не делается. В свою очередь, на основе этой информации создаются функциональные требования и элементы дизайна, которые облегчают отслеживание химикатов в каждой лаборатории и помогают сотруднику, запрашивающему химикат, отыскивать искомое как можно ближе.

Документ об образе и границах проекта

Документ об образе и границах (vision and scope document) собирает бизнес-требования в единый документ, который подготавливает основу для последующей разработки продукта. В некоторых организациях с этой же целью создают устав проекта или положение о бизнес-задачах. Очень полезен и документ основных рыночных требований (market requirements document, MRD). В нем более детально, чем в документе об образе и границах, рассматриваются целевые сегменты рынка и проблемы, касающиеся коммерческого успеха продукта.

Владельцем документа об образе и границах считается тот, кто финансирует проект или несет аналогичную ответственность. Аналитик требований может вместе с этим человеком разрабатывать документ об образе и границах проекта. Информация, касающаяся бизнес-требований, должны поступать от лиц, четко понимающих, почему они взялись за проект. Это может быть клиент или топ-

менеджер организации, разрабатывающей продукт, тот, кого привлекают технические новинки, менеджер по продукту, эксперт в данной предметной области или специалисты отдела маркетинга. Далее приведен пример шаблона документа об образе и границах пректа.
Шаблоны стандартизируют структуру документов, создаваемых проектными командами каждой организации. Как и в случае с любым шаблоном, измените его в соответствии со спецификой вашего проекта. Может показаться, что части документа об образе и границах повторяются, однако они должны смыкаться.

Пример шаблона документа об образе и границах проекта:

1. Бизнес-требования

1.1. Исходные данные

1.2. Возможности бизнеса

1.3. Бизнес-цели и критерии успеха

1.4. Потребности клиента или рынка

1.5. Бизнес-рынки

2. Образ решения

2.1. Положение об образе проекта

2.2. Основные функции

2.3. Предположения и зависимости

3. Масштабы и ограничения проекта

3.1. Объем первоначально запланированной версии

3.2. Объем последующих версий

3.3. Ограничения и исключения

4. Бизнес-контекст

4.1. Профили заинтересованных лиц

4.2. Приоритеты проекта

4.3. Операционная среда

Бизнес-шансы. Использовать слабо защищенные документы о конкурирующем продукте.

Бизнес-цель. Получить признание в качестве наиболее защищенного продукта на рынке, используя обзоры в отраслевых журналах и опросы потребителей, и захватить 80% рынка.

Пожелания потребителей. Более защищенный продукт.

Функция. Новый надежный механизм защиты.

Бизнес-требования

1. Бизнес-требования

Проекты выпускаются с полным убеждением, что новый продукт для кого-то сделает мир лучше. Бизнес-требования описывают основные преимущества, которые новая система даст ее заказчикам, покупателям и пользователям. Для различных типов продуктов — информационных систем,

коммерческих пакетов ПО и систем контроля, работающих в режиме реального времени, — выделяются различные преимущества.

1.1 Исходные данные

Суммирует обоснование и содержание нового продукта. Здесь помещают общее описание предыстории или ситуации, в результате которых было принято решение о создании продукта.

1.2 Возможности бизнеса

Для коммерческого продукта описывают существующие рыночные возможности и рынок, на котором продукту придется конкурировать с другими продуктами. Для корпоративной информационной системы описывают бизнес-проблему, которая разрешается посредством этого продукта, или бизнес-процессы, для улучшения которых требуется продукт, а также среду, в которой система будет использоваться. Кроме того, приведите здесь сравнительную оценку существующих продуктов и возможных решений, указав, в чем заключается привлекательность продукта и его преимущества. Опишите проблемы, которые не удается разрешить без продукта. Покажите, насколько он соответствует тенденциям рынка, развитию технологий или корпоративной стратегии. Кратко опишите другие технологии, процессы или ресурсы, необходимые для удовлетворения клиента.

1.3 Бизнес-цели и критерии успеха

Суммирует важные преимущества бизнеса, предоставляемые продуктом, в количественном и измеряемом виде. Далее приведены примеры и финансовых и нефинансовых целей (Wiegers, 2002c). Если подобная информация приведена где-то еще, например в документе о бизнес-задачах, сошлитесь на этот документ, а не копируйте информацию. Определите, как заинтересованные лица будут определять и измерять успех проекта (Wiegers, 2002c). Установите факторы, которые максимально влияют на успех проекта — и те, которые организация может контролировать, и те, которые находятся вне сферы ее влияния. Определите меру для оценки того, были ли достигнуты бизнес-цели.

Примеры финансовых и нефинансовых бизнес-целей

1. Финансовые бизнес-цели:

- Освоить X% рынка за Y месяцев
- Увеличить сектор рынка в стране X на Y% за Z месяцев
- Достигнуть объема продаж X единиц или дохода, равного $Y, за Z месяцев
- Получить X% прибыли или дохода по инвестициям в течение Y месяцев
- Достигнуть положительного баланса по этому продукту в течение Y месяцев
- Сэкономить $X в год, которые в настоящий момент расходуются на обслуживание системы
- Уменьшить затраты на поддержку на X% за Z месяцев
- Получить не более X звонков в службу обслуживания по каждой единице товара и Y звонков по гарантии каждой единицы товара в течение Z месяцев после выпуска товара
- Увеличить валовую прибыль для существующего бизнеса с Xдо Y%

2. Нефинансовые бизнес-цели:

- Достигнуть показателя удовлетворения покупателей, равного по крайней мере X, в течение Y месяцев со времени выпуска товара
- Увеличить производительность обработки транзакций на X% и снизить уровень ошибок данных до величины не более Y%
- Достигнуть определенного времени для достижения доминирующего положения на рынке
- Разработать надежную платформу для семьи связанных продуктов
- Разработать специальную базовую технологическую основу для организации
- Получить X положительных отзывов в отраслевых журналов к определенной дате
- Добиться признания продукта лучшим по надежности в опубликованных обзорах продуктов к определенной дате
- Соответствовать определенным федеральным и государственным постановлениям
- Уменьшить время оборота до X часов на Y% звонков покупателей в службу поддержки

1.4 Потребности клиентов или рынка

Опишите потребности типичных покупателей или целевых сегментов рынка, включая потребности, которые не удовлетворяют настоящие продукты или информационные системы. Представьте проблемы, с которыми в настоящее время сталкиваются клиенты и которые решит новая система, и предоставьте примеры того, как покупатели будут использовать этот продукт. Определите на высоком уровне все известные важные требования к интерфейсу или производительности, но не касайтесь деталей дизайна или реализации.

1.5 Бизнес-риски

Обобщает важнейшие бизнес-риски, связанные с разработкой — или не с разработкой — этого продукта. В категории рисков входят рыночная конкуренция, временные факторы, приемлемость для пользователей, проблемы, связанные с реализацией, и возможные негативные факторы, влияющие на бизнес. Оцените возможные потери от каждого фактора риска, вероятность его возникновения и вашу способность контролировать его. Определите все возможные действия по смягчению ситуации. Если вы уже подготовили эту информацию для анализа бизнес-задач или похожего документа, ссылайтесь на этот источник, а не копируйте эту информацию здесь.

Образ решения

В этом разделе документа определяется стратегический образ системы, позволяющей выполнять бизнес-задачи. Этот образ обеспечивает основу для принятия решений в течение жизненного цикла продукта. В него не надо включать детали функциональных требований или информацию, связанную с планированием проекта.

2.1 Положение об образе проекта

Составьте сжатое положение об образе проекта, обобщающее долгосрочные цели и назначение нового продукта. В этом документе следует отразить сбалансированный образ, удовлетворяющий различные заинтересованные лица. Он может быть несколько идеалистичным, но должен быть

основан на существующих или предполагаемых рыночных факторах, архитектуре предприятия, стратегическом направлении развития корпорации или ограничениях ресурсов. Далее показан шаблон, состоящий из ключевых слов, который прекрасно подходят для документа об образе продукта (Moore, 1991);

— для [целевая аудитория покупателей];

— который [положение о потребностях или возможностях];

— эта (этот) [имя продукта]

— является [категория продукта];

— который(-ая) [ключевое преимущество, основная причина для покупки или использования];

— в отличие от [основной конкурирующий продукт, текущая система или текущий бизнес-процесс];

— наш продукт [положение об основном отличии и преимуществе нового продукта].

Вот как выглядит положение об образе для Chemical Tracking System в Contoso Pharmaceuticals:

Для ученых, которым нужно запрашивать контейнеры с химикатом, данная Chemical Tracking System является информационной системой, которая обеспечит единую точку доступа к складу химикатов и к поставщикам. Система будет знать местоположение каждого контейнера с химикатом в компании, количество химиката в контейнерах и полную историю перемещения и использования каждого контейнера.

Эта система сэкономит компании 25% затрат на химикаты в первый год работы, позволив полностью использовать уже полученные химикаты, ликвидировать меньшее количество частично истраченных или просроченных химикатов и применять единую стандартную систему приобретения химикатов. В отличие от действующих сейчас механизмов заказов химикатов, которые выполняются вручную, наш продукт будет генерировать все отчеты, необходимые для соответствие федеральным и государственным постановлениям, в которых требуются сведения об использовании, хранении и ликвидации химикатов.

2.2 Основные функции

Назовите или пронумеруйте каждую основную функцию нового продукта или возможность, предоставляемую пользователям, уникальным последовательным способом, подчеркивая те из них, которые отличают его от предыдущих или конкурирующих продуктов. Дав каждой функции уникальное имя (в отличие от маркера абзаца), вы сможете отследить каждую функцию до отдельных требований пользователей, функциональных требований и других элементов систем.

2.3 Предположения и зависимости

Задокументируйте все предположения, сделанные заинтересованными лицами, когда они обдумывали проект и создавали данный документ об образе и границах. Часто предположения одних лиц не разделяют другие стороны. Если вы запишите их и просмотрите позже, TCI получите возможность обговорить основные положения проекта. Так вы избежите путаницы и ухудшения ситуации в будущем. Например, спонсор Chemical Tracking System предположил, что новая система заменит существующую систему складского учета и будет взаимодействовать с системой закупок

компании Contoso. Также задокументируйте важнейшие зависимости проекта от внешних факторов — изменения индустриальных стандартов или правительственных положений, других проектов, поставщиков со стороны или партнеров по разработке.

Масштабы и ограничения проекта

Когда химик изобретает новую химическую реакцию, которая преобразует один тип химиката в другой, он пишет документ, в который входит раздел «Рамки и ограничения», где описывает, что получится и не получится в результате этой реакции. Точно так же для проекта по разработке ПО следует определить его рамки и ограничения. Вам необходимо указать, что может делать система, а что не может.

Границы проекта определяют концепцию и круг действия предложенного решения. В ограничениях указываются определенные возможности, которые не будут включены в продукт. Рамки и ограничения помогают установить реалистичные ожидания заинтересованных лиц. Иногда клиенты запрашивают функции, слишком дорогостоящие или выходящие за предполагаемые границы продукта. Требования, выходящие за границы продукта, следует отклонять, если только они не настолько ценны, чтобы специально под них расширить проект, естественно, соответствующим образом изменив в бюджет, график и кадровый состав. Документируйте отклоненные требования и причины отказа от них, поскольку они имеют свойство появляться снова.

3.1 Объем первоначальной версии

Обобщает основные запланированные функции, включенные в первоначальную версию продукта. Опишите характеристики качества, которые позволят продукту предоставлять предполагаемые выгоды различным классам пользователей. Если ваша задача — сосредоточиться на разработке и уложиться в график, вам следует избегать искушения включить в версию 1.0 каждую функцию, которая когда-нибудь в будущем может понадобиться какому-то потенциальному покупателю. Увеличение сроков и сдвиг графика— типичный исход такого коварного расползания объема. Сосредоточьтесь на наиболее ценных функциях, имеющих максимально приемлемую стоимость, годных для самой широкой целевой аудитории, которые удастся создать как можно раньше. Команда с которой мой коллега Скотт делал последний проект, решила, что пользователи должны иметь возможность запускать собственную службу доставки вместе с первой версии ПО. Версия 1 не обязательно должна быть быстрой, красиво оформленной или легкой в использовании, но она должна быть надежной; это основа работы команды. Первая версия системы выполняет лишь базовые задачи. В будущие выпуски будут включены дополнительные функции, возможности и средства, обеспечивающие легкость и простоту использования,

3.2 Объем последующих версий

Если вы представляете поэтапную эволюцию продукта, укажите, какие функции будут отложены и желательные сроки последующих выпусков. В последующих версиях вы сможете реализовать дополнительные варианты использования и функции и расширить возможности первоначальных

вариантов использования и функций (Nejmeh и Thomas, 2002). Вы также сможете повысить производительность, надежность и другие характеристики качества по мере совершенствования продукта. Чем дальше вы заглядываете, тем более расплывчатыми будут границы проекта. Вам наверняка придется передвинуть функциональность с одного запланированного выпуска до другого и, возможно, добавлять незапланированные функции. Короткие циклы выпусков часто удобны для сбора отзывов клиентов.

3.3 Ограничения и исключения

Определение границы между тем, что входит и выходит за границы проекта, — отличный способ управления расползанием объема и ожиданиями клиентов. Перечислите все возможности или характеристика которых могут ожидать заинтересованные в проекте лица, но включение которых в продукт или в определенную версию не запланировано.

Бизнес-контекст

В этом разделе обобщаются некоторые бизнес-проблемы проекте, включая профили основных категорий заинтересованных лиц и приоритеты управления.

4.1 Профили заинтересованных лиц

Заинтересованными в проекте лицами (stakeholders) называются отдельные лица, группы или организации, которые активно вовлечены в проект, на которых влияет результат проекта и которые сами могут влиять на этот результат (Project Management Institute, 2000; Smitr,2000). Профили заинтересованных лиц описывают различные категории клиентов и других ключевых лиц, заинтересованных в этом проекте. Вам не нужно описывать каждую группу заинтересованных лиц, например юристов, которые проверяют соответствие надлежащим законам. Сферой вашего интереса должны стать различные группы клиентов, целевые рыночные сегменты и различные классы пользователей, входящих в эти сегменты. **В профиль каждого заинтересованного в проекте лица включается следующая информация:**

Основная ценность или преимущество, которое продукт принесет заинтересованным лицам и то, как продукт удовлетворит покупателей. **Ценность для заинтересованных лиц представляют:**

— улучшенная производительность;

— меньшее количество переделок;

— снижение себестоимости;

— ускорение бизнес-процессов;

— автоматизация задач, ранее выполнявшихся вручную;

— возможность выполнять совершенно новые задачи;

— соответствие соответствующим стандартам и правилам;

— лучшая, по сравнению с текущими продуктами, легкость и простота использования;

— их вероятное отношение к продукту;

— наиболее интересные функции и характеристики;

— все известные ограничения, которые должны быть соблюдены.

4.2 Приоритеты проекта

Чтобы принимать эффективные решения, заинтересованные лица должны договориться о приоритетах проекта. Один из подходов к этому заключается в рассмотрении пяти измеряемых параметров проекта: функции (или объем), качество, график, затраты и кадры (Wiegers. 1996a). В любом проекте каждый из этих параметров относится к одной из трех категорий:

— ограничение — лимитирующий фактор, в рамках которого должен оперировать менеджер проекта;

— ключевой фактор — важный фактор успеха, ограниченно гибкий при изменениях;

— степень свободы — фактор, который менеджер проекта может до определенной степени изменять и балансировать относительно других параметров.

Задача менеджера проекта — настроить те факторы, которые представляют собой степени свободы для достижения ключевых факторов успеха проекта е рамках, налагаемых ограничениями. Не все факторы могут быть ключевыми, как и не все — ограничениями. Менеджеру проекта необходима определенная степень свободы для того, чтобы он мог реагировать должным образом на изменение требований к проекту или внешних обстоятельств. Представьте себе, что отдел маркетинга неожиданно требует создать продукт на месяц раньше сроке.

Какова будет ваша реакция?

— Вы отложите реализацию определенных требований до более поздней версии?

— Сократите запланированный цикл тестирования системы?

— Оплатите сверхурочную работу вашим специалистам или пригласите специалистов по контракту для ускорения разработки?

— Привлечете ресурсы других проектов для разрешения ситуации?

Именно от приоритетов проекта зависят ваши действия в подобных ситуациях.

4.3 Операционная среда

Опишите среду, в которой будет использоваться система, и определите важнейшие требования к доступности, надежности, производительности и целостности. Эта информация существенно влияет на определение архитектуры системы, что является первым — и часто самым важным — этапом дизайна. Архитектура системы, предназначенной для поддержки пользователей, которые находятся далеко друг от друга и которым необходим круглосуточный доступ, сильно отличается от той, что предназначена для доступа пользователей, находящихся рядом, только в рабочие часы. На нефункциональные требования, такие. как отказоустойчивость и способность обслуживать систему во время ее работы, требуется значительное количество средств, отпущенных на дизайн и реализацию. Чтоб прояснить ситуацию, задайте заинтересованным лицам **уточняющие вопросы.**

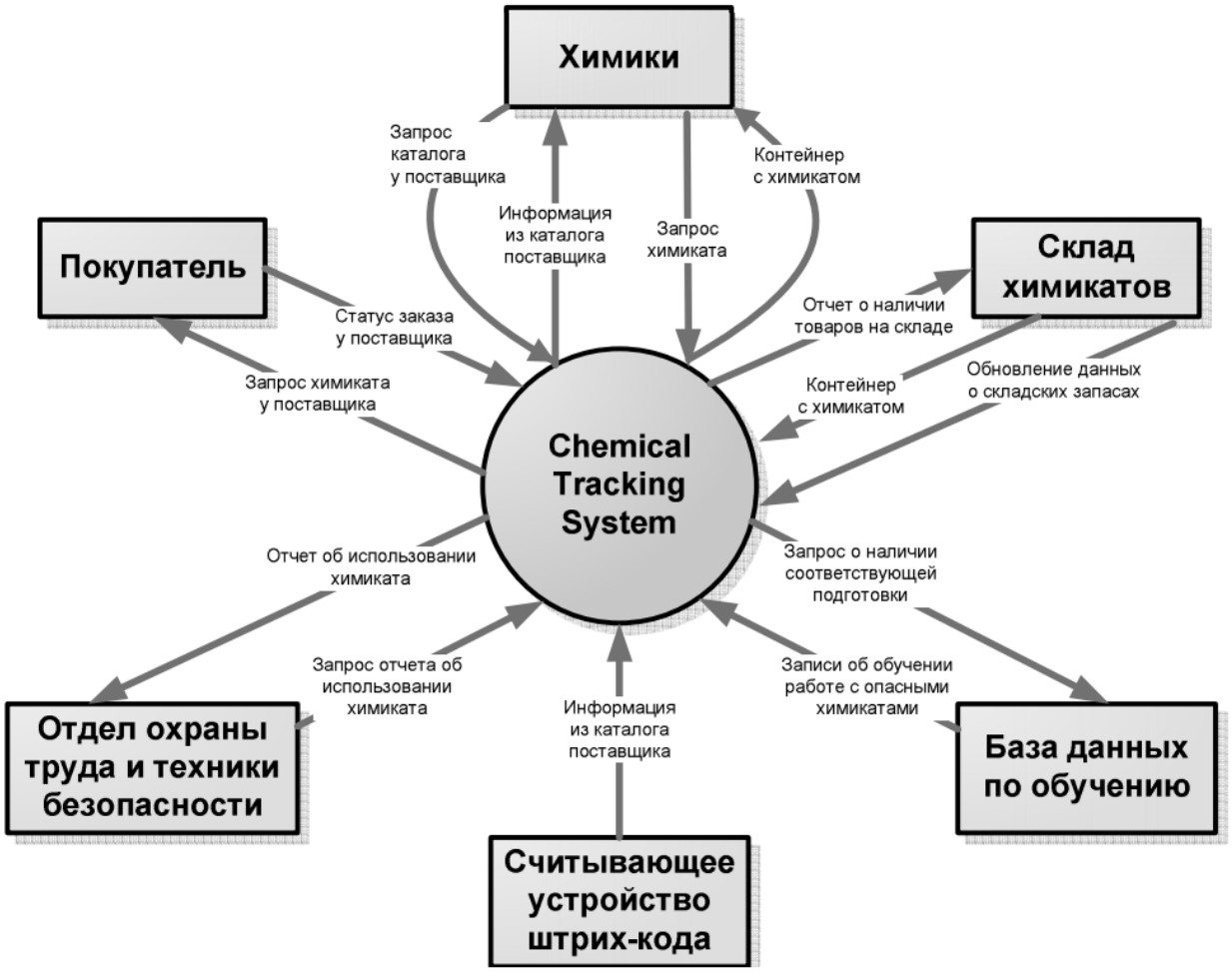
— Пользователи расположены далеко (географически) или близко друг от друга? В скольких часовых поясах работают ваши пользователи?

— Когда пользователям, находящимся в различных географических местоположениях, требуется доступ к системе?
— Где данные генерируются и используются? Насколько далеко друг от друга расположены эти местоположения? Нужно ли объединять данные из разных местоположений?
— Известно ли максимальное время отклика для получения доступа к данным, которые могут храниться удаленно?
— Готовы ли пользователи смириться с прерыванием работы службы или непрерывный доступ к системе крайне важен для работы их компании?
— Какие элементы управления безопасностью и требования к защите данных необходимы?

Контекстная диаграмма

Уточнение рамок определяет границу и связи системы, которую мы разрабатываем, со всем остальным миром. **Контекстная диаграмма (context diagram)** графически иллюстрирует эту границу. Она определяет оконечные элементы (terminators), расположенные вне системы, которые определенным образом взаимодействуют с ней, а также данные, элементы управления и материальные потоки, протекающие между оконечными элементами и системой. Контекстная диаграмма представляет собой высший уровень абстракции в диаграмме потока данных, разработанной по принципам структурного анализа (Robertson и Robertson, 1994), но эта модель полезна и в случае применения какой-либо другой методики разработки. Вы можете включить контекстную диаграмму в документ об образе и границах, или определить ее как приложение к спецификации требований, или как часть модели потоков данных системы.
Ниже, на рисунке, показана **часть контекстной диаграммы для Chemical Tracking System**. Вся система изображена кружком; на контекстной диаграмме намеренно не показывают внутренние объекты системы, процессы и данные. «Система» внутри кружка может иметь любую комбинацию ПО, оборудования или людских ресурсов. Оконечные элементы в прямоугольниках представляют классы пользователей («Химик» или «Покупатель»), отделы («Отдел охраны труда и техники безопасности»), другие системы («База данных по обучению») или аппаратные устройства («Считывающее устройство штрих-кода»). Стрелками показаны потоки данных («запрос химиката») или физические элементы («контейнер с химикатом») между системой и оконечными элементами. Вы можете ожидать, что поставщики химикатов должны быть показаны на диаграмме в виде оконечных элементов. Ведь компания направляет заказы для выполнения поставщикам, а те отправляют контейнеры с химикатами и счета в Contoso Pharmaceuticals, отдел же закупок пересылает чеки продавцам. Однако эти процессы происходят вне Chemical Tracking System, как часть операций отделов закупок и приобретений. Глядя на контекстную диаграмму становится совершенно ясно, что система не участвует напрямую в размещении заказов у поставщиков, в получении продуктов или оплате счетов.
Назначение таких средств, как контекстная диаграмма, заключается в стимулировании ясного и

точного взаимодействия между заинтересованными в проекте лицами. Эта ясность гораздо важнее слепого следования правилам создания «правильной» контекстной диаграммы. Однако я горячо рекомендую использование схему, показанную на рисунке, в качестве стандарта, когда вы возьметесь рисовать контекстные диаграммы. Предположим, вы решите использовать треугольник вместо кружка для изображения системы и эллипсы вместо прямоугольников для изображения оконечных элементов. Вашим коллегам будет трудно читать диаграмму, нарисованную в соответствии с вашими личным предпочтениями, а не с общепринятым стандартом.



Основные источники получения информации о потребностях клиентов

Способы и источники получения информации от клиентов зависят от специфики продукта и среды разработки. Необходимо выслушивать разные точки зрения и выбирать различные источники информации, а это лишний раз подтверждает, что работа по сбору требований тесно связана с общением. Вот несколько типичных источников получения информации при сборе требований к ПО.

Опросы потенциальных пользователей и дискуссии с ними. Самый очевидный способ выяснить потребности потенциальных пользователей нового программного продукта — опросить их. В этой главе рассказывается, как выбрать толковых представителей пользователей, а в главе 7 — как выяснить, что же им действительно необходимо.

Документы, где описан уже работающий или конкурирующий продукт. Эти документы могут также содержать корпоративные или отраслевые стандарты, которых необходимо придерживаться, а также постановления и законы, которым должен соответствовать продукт. Пригодятся и описания уже реализованных и будущих бизнес-процессов. Опубликованные сравнительные обзоры позволяют выявить недостатки других аналогичных продуктов, на которые стоит вовремя обратить внимание, чтобы получить конкурентное преимущество.

Спецификации требований к системе. Для продукта, включающего программные и аппаратные компоненты, создается спецификация требований ксистеме, описывающая продукт в целом. Отдельные требования к системе в целом касаются каждой подсистемы (Nelsen, 1990). Аналитик может вычленить дополнительные, более мелкие функциональные требования из требований к конкретной подсистеме.

Отчеты об ошибках и претензии к возможностям работающей системы. Персонал внутрикорпоративной и выездной службы поддержки — ценный источник информации. Они в курсе проблем, с которыми сталкиваются пользователи работающей системы, и постоянно выслушивают идеи клиентов по совершенствованию ее следующей версии.

Маркетинговые исследования и опросы пользователей. Опросив массу потенциальных пользователей продукта, вы получите кучу информации. Проконсультируйтесь с экспертом по планированию и управлению опросами, дабы убедиться, что задаете нужные вопросы нужным людям (Fowler, 1995). Опрос позволяет проверить, насколько четко вы понимаете требования, которые собираете или, как вы думаете, уже выявили, однако это не лучший способ стимулировать творческое мышление. Прежде чем начать опрос, необходимо критически изучить предполагаемые вопросы. Велико же будет ваше разочарование, когда вы поздно обнаружите, что один из вопросов сформулирован неоднозначно или что важного вопроса в списке не оказалось.

Наблюдение за пользователями на рабочих местах. Наблюдая «один рабочий день из жизни пользователя», аналитик выявляет особенности работы действующей системы, а также потребности потенциальных пользователей будущей системы. Такое исследование позволяет проверить информацию, собранную в ходе интервью, определить темы новых опросов, выявить проблемы действующей системы и понять, какие функции новой системы необходимы для автоматизации операций (McGraw и Harbison, 1997; Beyer и Holtzblatt, 1998). Наблюдая за работой пользователей, вы лучше и полнее поймете суть рабочего процесса, чем если просто попросите их описать все этапы их работы. Излагая и обобщая данные, аналитик должен абстрагироваться от ситуации и рассматривать ее несколько «сверху; это гарантирует,что собранные требования будут представлять интересы класса пользователей в целом, а не отдельных лиц. Кстати, опытные аналитики зачастую способны предложить что-то весьма дельное для совершенствования текущих бизнес-процессов.

Сценарий анализа задач пользователей. Определив, какие задачи пользователю требуется выполнять средствами системы, аналитик должен выработать необходимые функциональные требования к системе. Это — суть подхода, основанного на применении вариантов использования,

описанного в главе 8. Не забудьте указать, какие данные необходимы и генерируются входе выполнения задачи, а также источники этих данных.

События и реакция на них. Перечислите внешние события и соответствующую реакцию системы на них. Данный способ особенно хорош для систем реального времени, которые считывают и обрабатывают потоки данных, коды ошибок, управляющие сигналы и сигналы прерывания от внешних устройств.

Классы пользователей

Помимо всего прочего, пользователей продукта можно подразделять по таким признакам:

— по частоте использования продукта;

— по опыту в предметной области и опыту работы с компьютерными системами;

— по требуемой им функциональности;

— по задачам, которые им приходится выполнять;

— по правам доступа к системе (например, обычный пользователь, гость или администратор).

На их основе формируются классы пользователей. Некоторых сотрудников можно отнести к нескольким классам. Так, администратор иногда работает с системой, как рядовой пользователь. Конечные пользователи ПО вне системы, показанные на контекстной диаграмме (предыдущий рисунок), представляют собой потенциальных кандидатов на отдельные классы пользователей. Это часть непосредственных пользователей продукта из большой группы клиентов, которых можно выделить из всех заинтересованных в проекте лиц.

Иерархия клиентов, пользователей и заинтересованных лиц:

Очень заманчиво поделить пользователей на классы по их географическому положению, виду бизнеса, которым занимается компания, или занимаемой должности, а не на основании того, как они

взаимодействуют с системой. Например, компания — производитель банковского ПО первоначально предполагала разделить пользователей по типам финансовых учреждений, в которых те работают: крупный коммерческий банк, небольшой коммерческий банк, ссудо-сберегательная общество или кредитный союз. В действительности же таким образом выделяются потенциальные сегменты рынка, а не различные классы пользователей. Во всех этих финансовых учреждениях у людей, занимающихся ссудами, формируются более или менее аналогичные функциональные требования к системе, поэтому логично выделить их в отдельный класс пользователей, назвав его, скажем, сотрудники, принимающие заявки. Это может быть специалист по кредитованию, вице-президент, менеджер по работе с клиентами и даже банковский кассир — при условии, что все они используют систему, чтобы помочь кому-либо подать заявку на получение займа.

Одни классы пользователей для вас важнее других. Когда вы принимаете решения о приоритетах или пытаетесь найти компромисс требований, выдвигаемых различными классами пользователей, мнение привилегированных классов имеет первостепенное значение. К последним относятся группы пользователей, работа которых с продуктом определяет, способствует ли он достижению заявленных бизнес-целей или нет. Это не означает, что заинтересованных лиц, оплачивающих разработку системы (они, вполне вероятно, вообще не являются ее пользователями), или тех, кто имеет большое политическое влияние, следует обязательно включать в привилегированные классы.

Непривилегированные классы составляют те пользователи, которые по причинам безопасности, конфиденциальности или правовым причинам не работают с продуктом (Cause и Lawrence, 1999).

Остальные классы пользователей можно проигнорировать. Они получат то, что получится, то есть при разработке системы вам не надо учитывать их интересы. Мнение прочих классов пользователей при определении требований к продукту имеют примерно одинаковое значение.

У каждого класса пользователей есть свой набор требований, сформировавшийся в ходе выполнения задач. Кроме того, появляются различные нефункциональные требования, например удобство применения, которые влияют на проектирование пользовательского интерфейса. Новичков волнует, насколько легко научиться работать (или вспомнить принципы работы) с продуктом. Такие клиенты предпочитают графические интерфейсы, упорядоченное представление информации на экране, подробные подсказки, мастеров и согласованность с другими приложениями, с которыми они уже знакомы. Тех, кто поопытнее, волнует легкость использования и эффективность работы. Они ценят клавиатурные сокращения, макросы, возможности настройки, панели инструментов, возможности создания сценариев и в некоторых случаях даже предпочитают интерфейс командной строки графическому интерфейсу пользователя.

Ловушка Не упустите из виду классы косвенных или вторичных пользователей, Они могут обращаться к вашему приложению не напрямую, а работать с его данными и сервисами через другие приложения или отчеты. Однако даже опосредованный клиент все равно остается вашим клиентом. Это может показаться странным, однако классы пользователей не обязательно состоят из людей, В качестве дополнительных классов пользователей можно рассматривать сторонние приложения и

аппаратные компоненты, с которыми взаимодействует ваша система. Например, система впрыска топлива в автомобиле представляет собой пользовательский класс ПО, встроенного в систему управления двигателем. Система впрыска топлива не может говорить сама за себя, поэтому аналитику придется выяснить у инженера, разработавшего систему впрыска, требования к ПО, которое управляет процессом.

В самом начале работы над проектом необходимо определить и охарактеризовать различные классы пользователей, чтобы узнать у представителей всех важных классов их требования. Один из полезных способов определения классов называется «от расширения — к сжатию» («Expand Then Contract») (Gottesdiener, 2002). Для начала придумайте как можно больше классов пользователей: столько, сколько сможете. Не бойтесь, если их окажется несколько дюжин — позже вы объедините их и классифицируете. Важно не пропустить какой-либо класс, иначе это аукнется вам позже.

Следующий этап —выявить группы с похожими потребностями: их можно объединить в один класс или рассматривать как несколько подклассов одного крупного класса пользователей. Постарайтесь, чтобы список отдельных классов не превышал пятнадцати.

Одна компания, разрабатывавшая ПО для примерно 65 корпоративных клиентов, рассматривала каждого из них как отдельного пользователя со своими потребностями. Классификация клиентов по шести классам значительно упростило работу с требованиями для будущих версий продукта.

Помните: не все заинтересованные в проекте лица на самом деле будут работать с продуктом, поэтому класс пользователей — это лишь группа людей, которым следует предоставить возможность выразить свое мнение о создаваемом продукте.

Задокументируйте классы пользователей и их отличительные черты, меру ответственности и физическое расположение в спецификации требований к ПО. Менеджер проекта по разработке системы контроля химикатов, о которой шла речь в предыдущих главах, выявил следующие классы пользователей и их характеристики:

Наименование класса пользователей	**Описание класса пользователей**
Химики (привилегированный класс)	Примерно 1000 химиков, работающие в шести зданиях, посредством с химикаты у пхтавщиков и со склада. Каждый химик использует систе преимущественно для запроса химикатов и контроля за контейнерам лабораторию и отправляемыми из нее. Химикам необходима возможн поставщиков специальные химические структуры, импортированные для рисования таких структур.

Отдел закупок	Около пяти сотрудников отдела закупок обрабатывают запросы на хи поступающие от других специалистов. Они размещают и отслеживают поставщиками. Они не очень-то разбираются в химии, главное, что им каталогах был максимально простым. Специалистам отдела закупок н возможности отслеживания контейнеров, предоставляемые системой обращается к системе примерно 20 раз в день.
Склад химикатов	Персонал склада химикатов — это шесть техников и один контролер, более чем 500 000 химических контейнеров. Они обрабатывают запро поставку контейнеров с трех складов, запросы поставщикам на новые контролируют поступление контейнеров на склады и их отгрузку. Со необходима только функция отчета о складских запасах. Из-за больш эта функция должны быть эффективной и автоматизированной.
Отдел охраны труда и техники безопасности (привилегированный класс)	Специалисты отдела охраны труда и техники безопасности с помощь ежеквартальные отчеты в соответствии с местным и федеральным по использовании и утилизации химических веществ. Скорее всего мене придется несколько раз в год запрашивать новую форму отчетов, кот последних правительственных постановлений. Эти запросы об измен приоритет и должны быть реализованы в самые короткие сроки.

Включите в спецификацию требований к ПО всю существенную информацию о каждом классе пользователей, например относительный или абсолютный размер и привилегированность класса. Это поможет команде разработчиков определить, какие запросы об изменениях наиболее важны, и в дальнейшем оценить ценность изменений. Примерная оценка объема и типа системных транзакций позволит специалисту по тестированию разработать алгоритм использования системы и в последующем планировать действия по проверке системы.
Чтобы было легче внедрить идею с классами пользователей в жизнь, стоит описать типичного представителя каждого класса (Cooper, 1999), например, вот так:

Фред, 41 год, работает химиком в Contoso Pharmaceuticals уже 14 лет, с тех пор как получил степень кандидата наук. Нетерпелив с компьютерами. Обычно Фред одновременно работает над двумя проектами в смежных областях химии. В его лаборатории хранится около 400 контейнеров и баллонов с химикатами. Ежедневно ему требуется в среднем 4 новых химиката со склада. Два из них

— промышленные химикаты из имеющихся на складе, один необходимо заказать и еще один придется взять из химических образцов компании. Иногда Фреду требуются опасные химикаты, для работы с которыми необходима специальная подготовки. Приобретая какой-либо химикат впервые, Фред хочет, чтобы ему по электронной почте автоматически поступали соответствующие материалы по технике безопасности. Ежегодно Фред создает около 10 новых химикатов, которые отправляет на склад. Он хочет получать по электронной почте автоматически сгенерированный отчет о заказанных им в прошлом месяце химикатах; это позволит ему контролировать расход химических веществ.

Изучая требования химиков, рассматривайте Фреда как архетип данного класса пользователей и задайте себе вопрос: «Что Фреду нужнее всего?»

Разработка требований. Подход с применением вариантов использования

Вариант использования (use case) продукта описывает последовательность взаимодействия системы и внешнего действующего лица.

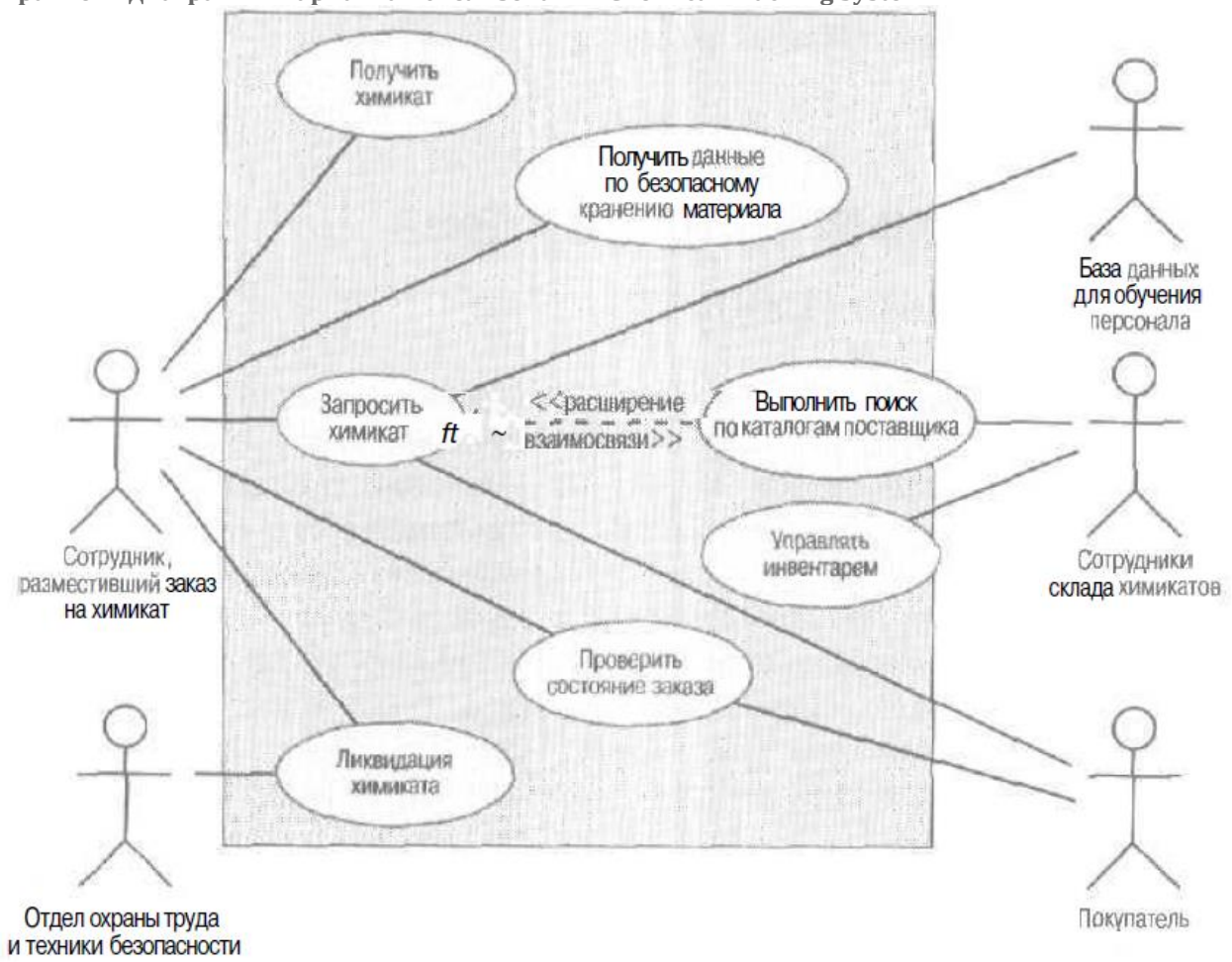
Действующим лицом (actor) может быть человек, другая система ПО или аппаратное устройство, взаимодействующее с системой для доптижения некой цели (Cockburn, 2001). Еще одно название действующего лица — роль пользователя, поскольку это роль, которую члены одного или нескольких классов пользователей могут выполнять по отношению к системе (Constantine и Lockwood, 1999). Например, в варианте использования Chemical Tracking System «Запрос химиката» участвует действующее лицо — Сотрудник, разместивший заказ на химикат. Класса пользователей Chemical Tracking System с таким именем не существует. И химики, и специалисты, работающие на складе химикатов, могут запрашивать химикаты, поэтому члены любого из этих классов могут играть роль Сотрудника, разместившего заказ на химикат.

Понятие «вариант использования» пришло из мира объектно-ориентированного программирования. Тем не менее они годятся и для проектов, где применяются любые приемы разработки, поскольку пользователям безразлично, как именно создается ПО. Варианты использования лежат в основе широко применяемого Унифицированного процесса разработки ПО (Jacobson, Booch и Rumbaugh, 1999).

Варианты использования меняют традиционный подход к сбору информации; пользователей не спрашивают, как прежде, что, с их точки зрения, должна делать система, а выясняют, какие задачи собирается с ее помощью решать пользователь. Цель такого подхода — описать все подобные задачи. До включения каждого варианта использования в утвержденную версию требований, заинтересованные в проекте лица проверяют, не выходит ли он за границы проекта. Теоретически в конечный набор вариантов использования должна входить вся желаемая функциональность системы. На практике же вам вряд ли удастся добиться стопроцентного результата, однако варианты

использования помогут вам выполнить эту задачу полнее, чем какой-либо другой прием сбора информации, которым я когда-либо пользовался.

Фрагмент диаграммы варианта использования Chemical Tracking System:



Диаграммы вариантов использования (use-case diagrams) позволяют получить отличное визуальное представление о требованиях пользователей. На рис. 8-1 показан фрагмент диаграммы варианта использования Chemical Tracking System, где применяются нотации UML (Unitied Modeling Language — унифицированный язык моделирования) (Booch, Rumbaugh и Jacobson, 1999; Armour и Miller, 2001). Прямоугольник показывает границы системы. Линии соединяют каждое действующее лицо (нарисованный человечек) с вариантами использования (эллипсами), с которыми это лицо взаимодействует. Обратите внимание на сходство этой диаграммы варианта использования с контекстной диаграммой на рис. 5-3. Здесь прямоугольник отделяет некоторые внутренние элементы высокого уровня системы — варианты использования — от внешних действующих лиц. Контекстная диаграмма также отображает объекты, расположенные вне системы, но на ней не видны внутренние части системы.

Варианты использования и сценарии использования

Вариант использования (use case) — это отдельное, независимое действие, которое действующее лицо может выполнить для получения определенного значимого результата. Один вариант

использования может охватывать несколько схожих задач с одинаковыми целями. Следовательно, он представляет собой набор связанных между собой сценариев использования, где сценарий — это отдельный пример варианта использования. Вы можете начать разработку требований с абстрактных вариантов использования, а затем на их основе создать конкретные сценарии использования или, наоборот, перейти от некоего сценария к более широкому варианту использования. Далее в этой главе показан подробный шаблон для документирования вариантов использования. К важным элементам описания варианта использования относятся:

— уникальный идентификатор;

— имя, кратко описывающее задачи пользователи в формате «глагол + объект», например «разместить заказ»;

— краткое текстовое описание на естественном языке;

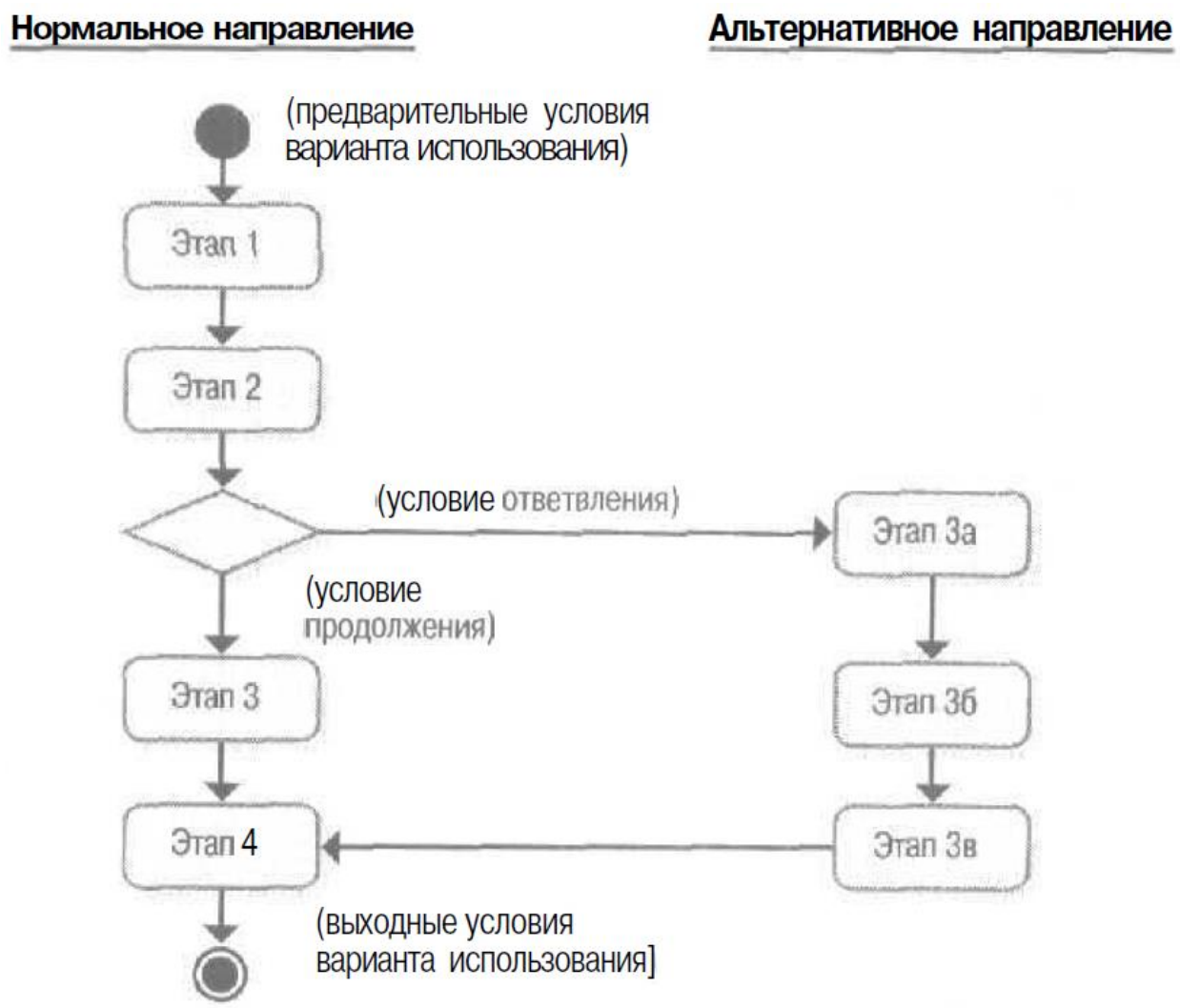
— список предварительных условий, которые должны быть удовлетворены до начала разработки варианта использования;

— выходные условия, описывающие состояние системы после успешного завершения разработки варианта использования;

— пронумерованный список действий, иллюстрирующий последовательность этапов взаимодействия лица и системы от предварительных условий до выходных условий.

Один сценарий считается нормальным направлением развития (normal course) варианта использования, его также называют основным направлением, базовым направлением, нормальным потоком, основным сценарием, главным успешным сценарием и благоприятным путем. Нормальное направление для варианта использования «Запрос химиката» — запрос химиката, который есть на складе.

UML-диаграмма, иллюстрирующая диалоговый поток при нормальном и альтернативном развитии варианта использования:



Другие допустимые сценарии из варианта использования, называются альтернативными направлениями (alternative courses) или вторичными сценариями (secondary scenarios) (Schneider и Winters, 1998). Они также могут привести к успешному выполнению задания и удовлетворяют выходным условиям варианта использования. Однако они представляют вариации решения задачи или диалоговой последовательности, необходимой для выполнения задачи. В определенной точке принятия решений в диалоговой последовательности нормальное направление может перейти в альтернативное, а затем вернуться обратно в нормальное. Хотя большинство вариантов использования можно описать простым языком, блок-схема или UML-диаграмма позволяют визуально представить логическое развитие сложного варианта использования, как показано на рис. 8-2. Блок-схема и диаграммы взаимодействия показывают точку принятия решений и условия при которых основное направление развития событий переходит в альтернативное.
Альтернативное направление для варианта использования «Заказать химикат» — это «Заказать химикат у поставщика». В обеих ситуациях конечная цель действующего лица одна и та же — запрос химиката, поэтому оба сценария входят в один вариант использования. Некоторые этапы альтернативного направления аналогичны этапам нормального направления, но для завершения альтернативного направления необходимо выполнить особые действия. В нашем случае пользователь может искать необходимый химикат по каталогам поставщиков. Если альтернативное

направление само по себе является автономным вариантом использования, вы можете расширить нормальное направление, включив этот отдельный вариант использования в нормальный поток (Armour и Miller, 2001). Диаграмма на рис. 8-1 иллюстрирует подобную расширенную взаимосвязь. Вариант использования «Запросить химикат» был расширен вариантом использования «Выполнить поиск по каталогам поставщика». Кроме того, сотрудники склада химикатов используют «Поиск по каталогам поставщиков» как отдельный вариант.

Иногда несколько вариантов использования имеют общие наборы этапов. Чтобы избежать повторения этих этапов в каждом варианте использования, определите отдельный варианте общей функциональностью и укажите, что он включен в другие варианты использования как подвариант. Эта процедура аналогична вызову общей подпрограммы в компьютерной программе.

В качестве примера рассмотрим работу ПО для бухгалтерии. Возможны два варианта использования — «Оплатить счет» и «Подтвердить кредитную карту», причем в обоих, чтобы выполнить платеж, пользователь должен подписать чек. Вы можете создать отдельный вариант использования «Подписать чек», который содержит набор действий, необходимых для подписи чека. Этот вариант будет включен в обе транзакции, как показано на рис. 8-3.

Ловушка Не затягивайте обсуждение того, когда, как и стоит ли вообще использовать взаимоотношения типа расширить и включить. Чаще всего применяется следующий прием — указать общие действия во включенном варианте использования.

Пример того, как вариант использования связан с бухгалтерским приложением:

Условия, препятствующие успешному завершению задания, называются исключениями (exceptions). Для варианта использования «Запросить химикат» существует одно исключение — «Химиката нет в продаже». Если в процессе сбора информации вы не укажете, как обрабатывать исключение, то возможны два пути:

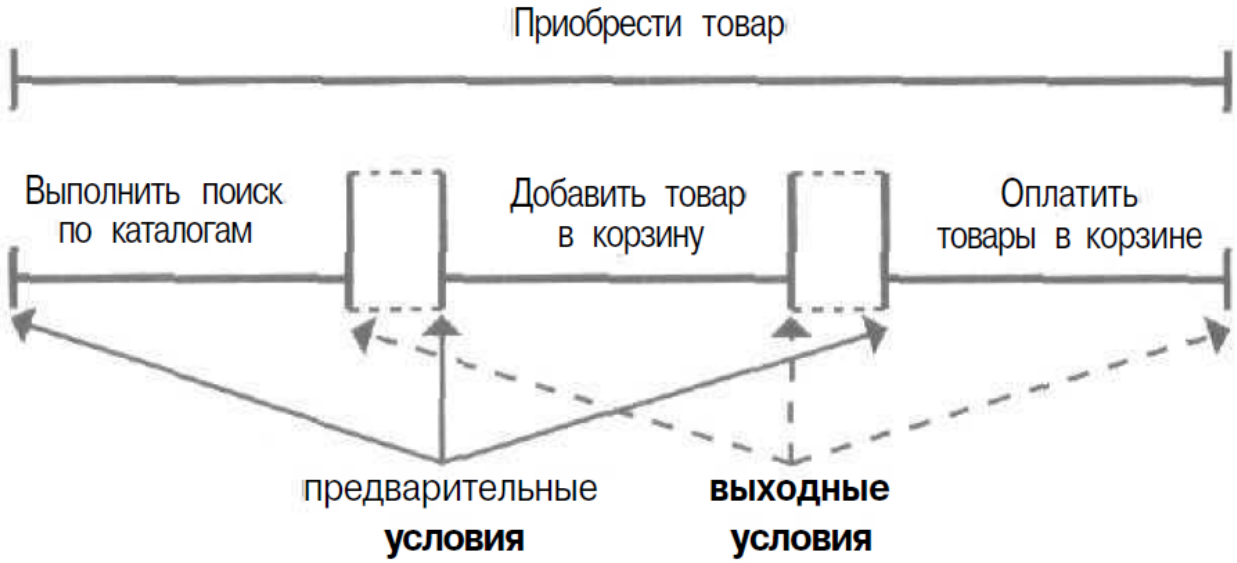
1. разработчики предложат лучший по их мнению способ обработки исключений;
2. при генерации пользователем неверного условия произойдет сбой системы, так как никто не предусмотрел такой ситуации. Бьюсь об заклад, что сбои системы не входят в список требований пользователей.

Иногда исключения рассматриваются как тип альтернативного направления (Cockburn, 2001), однако эти понятия следует разделять. Не обязательно реализовывать каждое альтернативное направление, которое вы определили для варианта использования; кроме того, вы можете отложить

его реализацию до следующего выпуска. Однако вы обязаны реализовать исключения, из-за которых завершение сценариев может оказаться не успешным. Любой, кто когда-либо занимался программированием, знает, что часто именно на работу над исключениями приходится большая часть программирования. Многие недостатки конечного продукта присущи именно обработчикам исключений (или происходят из-за их отсутствия). Указать условия исключений в ходе сбора информации по требованиям — верный способ создать устойчивые к сбоям продукты.

Для многих систем пользователь может связать последовательность вариантов использование в «макровариант использования», описывающий объемную задачу. Для коммерческого Web-сайта предлагаются, например, такие варианты использования; «Поиск по каталогу», «Добавить товар в корзину» и «Оплатить товары в корзине». Если вы можете выполнить все эти действия по отдельности, то все они считаются независимыми вариантами использования. Кроме того, вы вправе выполнить все три указанные операции подряд, назвав этот один большой вариант использования «Приобрести товар», как показано на рис. 8-4. Причем после каждого варианта использования система должна быть в таком состоянии, чтобы пользователь мог приступить к следующему варианту использования немедленно. То есть выходные условия одного варианта использования должны удовлетворять предварительным условиям следующего за ним варианта. Подобным же образом в приложении обработки транзакций, например АТМ, каждый вариант использования должен оставлять систему в состоянии готовности к следующей транзакции. Предварительные условия и выходные условия каждой транзакции варианта использования должны вставать в один ряд.

Предварительные условия и выходные условия определяют границы отдельных вариантов использования, которые можно связать в цепочку для выполнения объемной задачи:



Определение вариантов использования

Вы можете определить варианты использования несколькими способами (Ham, 1998; Larman, 1998):

— сначала определить действующие лица, а затем бизнес-процессы, в которых каждое лицо участвует;
— выразить бизнес-процессы в терминах определенных сценариев, обобщить сценарии в варианты использования и определить действующие лица для каждого варианта;определить внешние события, на которые система должна реагировать, а затем соотнести эти события с участвующими лицами и определенными вариантами использования;
— определить вероятные варианты использования на основе функциональных требований, Если какие-либо требования невозможна проследить до какого-либо варианта использования, подумайте нужны ли они.
В Chemical Tracking System применен первый подход. Аналитик провел несколько семинаров по вариантам использования, примерно два раза в неделю, по два-три часа каждый. Члены различных классов пользователей участвовали в параллельных семинарах, так как лишь несколько вариантов использования были общими для нескольких классов пользователей. На каждом семинаре присутствовали сторонник продукта от класса пользователей, другие выбранные представители пользователей, а также разработчик. Участие в семинарах позволило разработчикам еще на ранней стадии получить представление о создаваемом продукте. Кроме того, они возвращали собравшихся к реальности, когда те предлагали невыполнимые требования.
До начала семинаров аналитики попросили пользователей продумать, какие задачи те собираются выполнять с помощью новой системы. Каждая такая задача становилась потенциальным вариантом использования. Несколько предложенных вариантов, как выяснилось, выходят за границы проекта, поэтому далее они не обсуждались. По мере изучения вариантов использования стало ясно, что некоторые варианты связаны между собой сценариями, которые можно объединить в один абстрактный вариант использования. Также удалось выявить дополнительные варианты использования, не входившие в первоначальный набор.
Некоторые участники предложили варианты использования, не сформулированные как задачи, например «Данные по безопасному хранению материала». Название варианта использования должно указывать на решаемую задачу, поэтому в названии необходим глагол. Что клиент хочет: просмотреть, напечатать, заказать, отправить по электронной почте, исправить, удалить или создать данные по безопасному хранению материала? Иногда предложенный вариант использования — это всего лишь одно действие, которое выполняется как часть варианта использования, например «сканировать штрих-код». Аналитик должен выяснить, какую цель подразумевал пользователь, когда предлагал это. Он может задать такой вопрос: «Сканируя штрихкод на контейнере с химикатами, что вы пытаетесь сделать?» Предположим, в ответ он услышит: «Это необходимо для того, чтобы я мог отправить этот химикат в свою лабораторию». Следовательно, настоящий вариант использования выглядит как-то так — «Отправить химикат в лабораторию». Сканирование штрих-кода — только один из этапов взаимодействия действующего лица и системы, передающей химикат в лабораторию.

Как правило, пользователи сначала определяют самые важные варианты использования, поэтому порядок предлагаемых тем позволит получить представление о приоритетах. Другой способ расстановки приоритетов — кратко описать каждый потенциальный вариант использования втом виде, какой был предложен. Расставьте приоритеты и предварительно распределите их по выпускам продукта. В первую очередь укажите детали для вариантов использования с наивысшим приоритетом, чтобы разработчики смогли приступить к их реализации как можно скорее.

Документирование вариантов использования

На этой стадии обсуждения участники должны обдумать важнейшие варианты использования. Constantine и Lockwood (1999) дают следующее определение важнейших вариантов использования (essential use cases): «… упрощенное, обобщенное, абстрактное, не зависящее от технологии и реализации описание одной задачи или взаимодействия… в котором воплощена цель или намерения, лежащие в основе взаимодействия». То есть следует сосредоточиться на задаче, которую пользователю необходимо выполнить, и возможностях системы для выполнения этой задачи. Важнейшие варианты использования характеризуются более высоким уровнем абстракции, чем конкретные варианты использования (concrete use cases), которые описывают определенные действия, предпринимаемые пользователем для взаимодействия с системой. Чтобы проиллюстрировать различие, рассмотрим два способа начала реализации пользователем варианта использования «Запросить химикат».

Конкретный вариант использования. Введите номер ID химиката.

Важнейший вариант использования. Укажите необходимый химикат.

Формулировка на важнейшем уровне позволяет пользователю выполнить задачу многими способов: ввести номер ID химиката, импортировать химическую структуру из файла, нарисовать структуру на экране с помощью мыши, выбрать химикат из списка и многое другое. Слишком быстрое углубление в детали определенного взаимодействия ограничивает широту мышления участников семинара. Независимость важнейших вариантов использования от реализации, делает их более пригодными для повторных применений, чем конкретные варианты использования.

Участники семинара, посвященного Chemical Tracking System, начинали каждое обсуждение с определения действующего лица, которое получит преимущество отданного варианта использования, и краткого описания этого варианта. Затем они указывали предварительные условия и выходные условия, ограничивающие вариант использования, а также все этапы внутри этих границ. Выяснив частоту использования, вы на ранних стадиях получите представление о необходимости и важности требования. Далее аналитики спрашивали участников, как те себе представляют взаимодействие с системой для выполнения задачи, Установленная последовательность действий лиц и реакции системы определялась, как нормальное направление. Нумерация этапов последовательности окончательно проясняла ситуацию. Хотя у каждого участника

было свое представление об интерфейсе и специальных механизмах взаимодействия, группа смогла выработать общее понимание важнейших этапов диалога системы и пользователя.

Придерживаясь границ

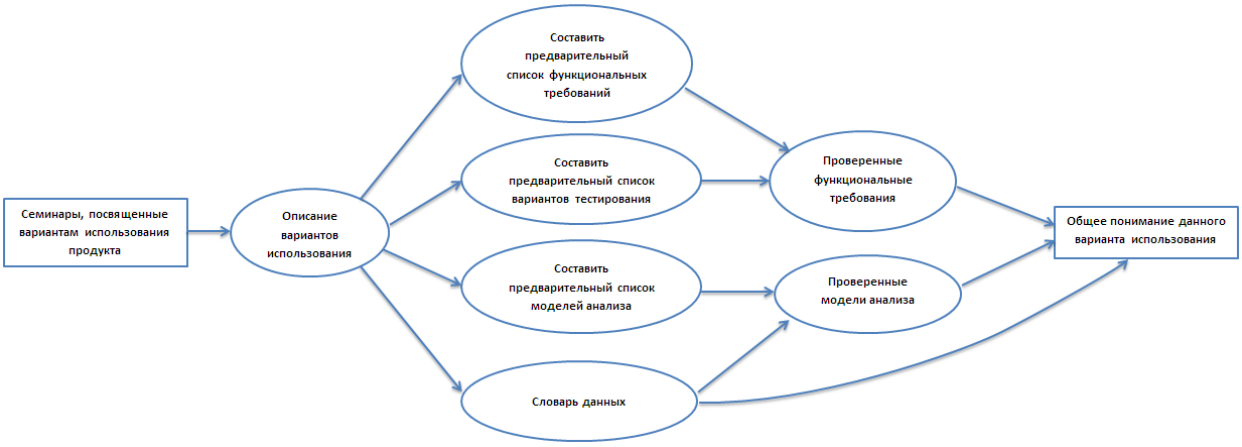
Изучая вариант использования из восьми этапов, я понял, что выходные условий удовлетворены уже после пятого этапа. Следовательно, этапы 6, 7 и 8 были излишними, так как выходили за границы варианта использования. Точно так же предварительные условия варианта использования должны быть удовлетворены до начала первого этапа. Изучая описание варианта использования, убедитесь, что предварительные и выходные условия ограничивают его соответствующим образом.

Аналитик фиксировал действия отдельного лица и реакцию системы на отдельных клейких листочках, которые затем прикреплял на схему. Возможен и другой способ проведения семинара — в ходе обсуждения спроецировать с помощью компьютера шаблон варианта использования на большой экран и отредактировать его, хотя иногда это замедляет обсуждение.

Команда, занимающаяся сбором информации, разработала аналогичные диалоги для определения альтернативных направлений и исключений. Если пользователь говорит: «По умолчание это должно быть…», значит, он описывает нормальное направление развития варианта использования. Такая фраза, как: «Необходимо, чтобы пользователь также смог..» предполагает обсуждение альтернативное направление. Многие условия исключений удавалось выяснились, когда аналитик задавал примерно такие вопросы: «Что должно произойти, если в текущий момент БД отключена?» или «Что, если этого химиката нет в продаже?». На семинаре также удобно обсудить ожидания пользователей, касающиеся качества продукта, например времени реакции, доступности и надежности системы, ограничений дизайна пользовательского интерфейса и требований по безопасности.

После того как участники семинара описали каждый вариант использования, и никто не предлагал уже никаких дополнительных вариантов, исключений или специальных требований, приступали к следующему варианту использования. Участники не пытались рассмотреть все варианты использования за одно марафонское обсуждение или точно определить все детали каждого варианта использования. Они запланировали изучение вариантов использования по возрастающей и их последующее многократное рассмотрение и уточнение.

Рисунок «Последовательность этапов разработки вариантов использования»



Существует два способа представить этапы взаимодействия пользователя и системы, которые и составляют основу варианта использования.

1) Составление пронумерованного списка этапов с указанием того, какой элемент (система или определенное действующее лицо) выполняет каждый шаг. Так же описываются и альтернативные направления и исключения, для которых показан этап нормального направления развития, на котором появляется ответвление, или этап, где возможно исключение.

Таблица — Фрагмент описание варианта использования «Запросить химикат»

Идентификатор варианта использования	*Вариант использования-1*
Автор	*Сергей*
Дата создания	*01.08.2014*
Название варианта использования	*Запросить химикат*
Автор последнего обновления	*Иван*
Дата последнего обновления	*06.08.2014*
Действующее лицо	Сотрудник, разместивший заказ на химикат
Описание	Сотрудник, разместивший заказ на химикат, указывает в запросе необходимый химикат, вводя его название или номер идентификатора химиката или импортируя его структуру из соответствующего графического средства. Система выполняет запрос, предлагая новый или использованный контейнер с химикатом со склада или позволяя создать запрос на заказ у стороннего поставщика.
Предварительные условия	1. Личность пользователя аутентифицирована. 2. Пользователь имеет право запрашивать химикаты. 3. База данных по запасам химикатов в данный момент подключена.
Выходные условия	1. Запрос сохраняется в Chemical Tracking System. 2. Запрос был отправлен по электронной почте на склад химикатов или поставщику.
Нормальное направление развития варианта использования	**1.0. Запросить химикат со склада** 1) Сотрудник указывает требуемый химикат. 2) Система подтверждает, что такой химикат доступен, 3) Система перечисляет контейнеры с необходимым химикатом, имеющиеся на складе. 4) Сотрудник может просмотреть историю любого контейнера. 5) Сотрудник выбирает определенный контейнер или просит отправить запрос поставщику (альтернативное направление 1.1.). 6) Сотрудник вводит остальную информацию, чтобы завершить запрос. 7) Система сохраняет запрос и отправляет его по электронной почте на склад химикатов.
Альтернативное направление развития варианта использования	**1.1. Запросить химикат у поставщика** 1) Сотрудник ищет химикат по каталогам поставщика. 2) Система отображает список поставщиков, где также указаны размеры, класс и цена контейнеров. 3) Сотрудник выбирает поставщика, размер, класс и количество контейнеров. 4) Сотрудник вводит остальную информацию, чтобы завершить запрос. 5) Система сохраняет запрос и отправляет его по электронной почте поставщику.
Исключения	**Химикат не доступен** 1) Система отображает сообщение "Химикат не существует". 2) Система спрашивает сотрудника, хочет ли он запросить другой химикат или выйти из программы. 3) Сотрудник просит запросить другой химикат. 4) Система заново начинает нормальное направление развития варианта использования. 3) Сотрудник решает выйти из системы. 4) Система завершает вариант использования. **Химиката нет в продаже** 1) Система отображает сообщение "Нет поставщика этого химиката". 2) Система спрашивает сотрудника, хочет ли он запросить другой химикат или выйти из программы. 3) Сотрудник запрашивает другой химикат. 4) Система заново начинает нормальное направление развития варианта использования. 3) Сотрудник решает выйти из системы. 4) Система завершает вариант использования.
Включение	**Вариант использования-22** Просмотреть хронологию контейнера
Приоритет	Высокий
Частота использования	Используется примерно пять раз в неделю каждым химиком, 100 раз в неделю каждым работником склада.
Бизнес-правила	**Бизнес-правило-28** Только сотрудник, получивший разрешение заведующего лаборатории, может запрашивать химикаты.
Специальные требования	Система должна импортировать химические структуры в стандартной зашифрованной форме из любых средств, поддерживающих рисование химических структур.
Предположения	Импортированные химические структуры должны быть достоверными.
Замечания и вопросы	Сергей выяснит, нужно ли одобрение руководства для запроса химиката, относящегося к уровню 1 списка опасных химикатов. Дата выполнения: 20.08.2014

2) Существует еще один прием — описать процесс средствами таблицы из двух столбцов. Действия лица показаны в левом столбце, а реакция системы — в правом. Цифры указывают на последовательность этапов диалога. Эта схема отлично работает, когда с системой взаимодействует только один пользователь.

Таблица — Описание этапов варианта использования в двухстолбцовом формате

Действия лица	**Реакция системы**
1. Укажите необходимый химикат. 4. Если нужно, запросите историю любого контейнера. 5. Выберите определенный контейнер (выполнено) или попросите отправить заказ поставщику (альтернативное направление 1.1.).	2. Убедится, что такой химикат 3. Отобразит список контейнер химикатом, в данный момент ч

Чтобы сделать эту таблицу более понятной, вы можете записать каждое действие или реакцию системы в отдельной строке, чтобы альтернативная последовательность стала очевидной.

Таблица — Альтернативный макет для описания этапов варианта использования в двухстолбцовом формате

Действия лица	**Реакция системы**
1. Укажите необходимый химикат. 4. Если нужно, ззпросите историю любого контейнера. 5. Выберите опредепенньй контейнер (выполнено) или попросите отправить заказ поставщику (альтернативное направление 1.1.).	1. Убедиться, что такой химика 3. Отобразить список контейне химикатом, в данный момент ч

Часто варианты использования содержат дополнительную информацию или требования, которые не годятся ни для одного из разделов шаблона. Для таких данных предусмотрен раздел «Специальные требования»: сюда записывают соответствующие атрибуты качества, требования к производительности и др. Также зафиксируйте любую информацию, которая может быть неочевидна пользователям, например, необходимое для завершения варианта использования неявное взаимодействие одной системы с другой.

Вам не всегда необходимо полное описание варианта использования. Alistair Cockburn (2001) описывает рабочий (casual) и полный (fully dressed) шаблоны вариантов использования. Рабочий вариант использования — это просто текстовое изложение целей пользователя и взаимодействий с системой. Рассказы пользователей, которые рассматриваются как требования в экстремальном программировании, представляют собой, по существу, рабочие версии варианта использования, как правило, написанные на учетных карточках (Jeffries, Anderson и Hendrickson, 2001). **Полные описания вариантов использования необходимы, если:**

— в работе над проектом представители пользователей действуют не в тесной связи с

разработчиками;
— приложение сложное, и высок риск сбоев системы;
— описания вариантов использования — это самый низкий уровень детализации требований, предназначенный для разработчиков;
— вы планируете разработать подробные варианты тестирования на основании пользовательских требований;
— для совместной работы территориально удаленных друг от друга команд необходимо детальные общие данные.
Ловушка. *Не пытайтесь догматически определять количество деталей, которые необходимо включить в вариант использования; помните, что ваша задача — достаточно хорошо разобраться, для чего система нужна пользователям, чтобы разработчики могли заняться приложением, не опасаясь необходимости будущих переделок.*

На рисунке «Последовательность этапов разработки вариантов использования» показано, что после каждого семинара аналитики Chemical Tracking System выявляли функциональные требования к ПО на основе описаний вариантов использования. Они также создали модели анализа для некоторых сложных вариантов использования, например диаграмму перехода состояний, показывающую все возможные состояния запроса химиката и все допустимые изменения состояний. Спустя день или два после семинара аналитик раздал описания вариантов использования и функциональных требования участникам, чтобы те просмотрели их до начала следующего семинара. Эти неофициальные просмотры выявили множество ошибок: ранее не выявленные альтернативные направления, новые исключения, некорректные функциональные требования и пропущенные этапы диалога. Оставьте между семинарами хотя бы один день. Умственное расслабление, которое наступает, когда минует день или два после мозгового штурма, позволяет людям взглянуть на проделанную работу с новой точки зрения. Один аналитик, проводивший семинары каждый день, заметил, что участникам стало трудно находить ошибки в просматриваемых документах, потому что информация была еще слишком свежа в памяти. Они прокручивали в уме дискуссии, которые только что завершились, и не замечали ошибок.
Ловушка. *Не ждите окончания сбора информации по требованиям, чтобы просить пользователей заняться просмотром собранного материала.*

На ранних стадиях разработки Chemical Tracking System руководитель тестирования создал концептуальные варианты тестирования, не зависящие от специфики реализации, на основе вариантов использования (Collard, 1999). Эти варианты тестирования помогли прийти команде к общему четкому пониманию того, как система должна функционировать при реализации определенных сценариев использования. Эти варианты тестирования позволили аналитикам проверить, действительно ли выявлены все функциональные требования, необходимые для выполнения пользователями каждого варианта использования. На заключительном семинаре,

участники вместе провели критический анализ вариантов тестирования, чтобы убедиться, что одинаково понимают, как должны работать варианты использования. Как и при любом контроле качества, они нашли ошибки и в требованиях, и в вариантах тестирования.

Варианты использования продукта и функциональные требования

Разработчики ПО не реализуют бизнес-требования или варианты использования. Они реализуют функциональные требования, определенные фрагменты поведения системы, позволяющие клиентам выполнять варианты использования и выполнять свои задачи. Варианты использования описывают поведение системы с точки зрения действующего лица, при этом упускается множество деталей. Чтобы разработать и реализовать систему должным образом разработчику необходимо ознакомиться с множеством точек зрения.

Некоторые практики считают, что варианты использования это и есть функциональные требования. Однако иногда возникают проблемы из-за того, что варианты использования просто передавались разработчикам для реализации. Варианты использования описывают точку зрения пользователя, его взгляд на внешнее, видимое поведение системы. В них не содержится всей информации, которая необходима разработчику для написания ПО. Так, человеку, использующему банковский автомат, не важно, какие неочевидные действия этот автомат должен выполнить, например взаимодействовать с компьютером банка. Эти детали невидимы для пользователя, однако разработчик должен о них знать. Естественно, в описания вариантов использования могут входить такие детали подобных неочевидных процессов, однако, как правило, в ходе дискуссий с пользователями такие вопросы не поднимаются. Даже у разработчиков, которым передали полные варианты использования, часто возникает много вопросов.

Совет. *Чтобы уменьшить эту неопределенность, аналитикам требований следует ясно расписывать детали функциональных требований, необходимых для реализации каждого варианта использования (Arlow, 1998).*

Многие функциональные требования не записаны в последовательности диалогов действующего лица и системы. Некоторые из них очевидны, например: «Система назначит уникальный порядковый номер каждому запросу». Нет смысла повторять эти детали в спецификации требований к ПО, если они совершенно ясно указаны в варианте использования. Другие функциональные требования не входят в описание варианта использования. Их выявляет аналитик на основании общего представления о варианте использования и операционной среды системы. Преобразование требований, как их видит пользователь, в форму, полезную разработчикам, — задача аналитика, один из многих способов, позволяющих ему обогатить проект.

В Chemical Tracking System варианты использования в основном применялись в качестве механизма для определения необходимых функциональных требований. Аналитики составляли только рабочие описания наименее сложных вариантов использования. Затем они выявляли все функциональные

требования, которые после реализации позволяли выполнять все варианты использования, в том числе альтернативные направления и обработчики исключений. И далее документировали эти функциональные требования в спецификации требований к ПО, сформированной с учетом особенностей продукта.

Задокументировать функциональные требования, связанные с вариантом использованием, можно несколькими способами. Выбор способа зависит от того, как ваша команда будет выполнять дизайн, сборку и тестирование — на основе документации о вариантах использования, спецификации требований к ПО или применяя оба этих документа. Ни один из этих методов нельзя назвать идеальным, поэтому выберите тот, что лучше позволит вам документировать и управлять требования к ПО вашего проекта.

Только варианты использования

Вы можете включить функциональные требования непосредственно в описание каждого варианта использования. Независимо от этого вам понадобится отдельная дополнительная спецификация для фиксирования нефункциональных и всех функциональных требований, которые не связаны с конкретными вариантами использования. Несколько вариантов использования могут нуждаться в одном и том же функциональном требовании, Если для пяти вариантов использования потребуется аутентификация пользователя, вам вряд ли захочется писать пять разных блоков кода. Вместо их повторения, установите перекрестные ссылки на функциональные требования, присутствующие в нескольких вариантах использования.

Варианты использования и спецификация требований к ПО

Другая возможность — в довольно простой форме описать варианты использования и задокументировать функциональные требования, выявленные из каждого варианта использования в спецификации требований к ПО. В этом случае вам придется установить трассируемость между вариантами использования и связанными с ними функциональными требованиями. Лучше всего управлять трассируемостью, если вы сохраните все варианты использования и функциональные требования в средстве управления требованиями.

Только спецификация требований к ПО

Третий способ — упорядочить спецификацию требований к ПО по вариантам использования или функциям и включить последние в спецификацию. Именно его и применяли специалисты, работавшие над Chemical Tracking System. В этой схеме нет отдельных документов для вариантов использования. Вы определяете повторяющиеся функциональные требования или указываете каждое функциональное требование только один раз и ссылаетесь на него при каждом появлении его в другом варианте использования.

Преимущества способа с применением вариантов использования

Преимущества применения вариантов использования в том, что каждый вариант сосредоточен на поставленной задаче и пользователе. Пользователи будут более четко представлять, что же им даст новая система, если вы выберете способ, который фокусируется на функциях системы. При разработке нескольких Интернет-проектов представители клиентов заявили, что варианты использования позволили им более четко представить, какие возможности должны получить посетители их Web-сайтов. А аналитики и разработчики смогли разобраться и в бизнесе-процессах пользователей, и в предметной области. Тщательное изучение этапов взаимодействия лица и системы помогает еще на ранних стадиях разработки выявить неясности и неточности, а также позволяет составить варианты тестирования на основе вариантов использования.
Очень расточительно и болезненно для разработчиков писать код, которым никогда не удастся воспользоваться. Если вы будете заблаговременно определять требования и включать в них все мыслимые функции, то рискуете создать избыточные требования. Способ с применением вариантов использования позволяет выявить функциональные требования, с помощью которых пользователи будут выполнять конкретные задачи. Так вы предотвратите появление «функций-сирот», тех, что в ходе сбора информации казались весьма полезными, однако которыми никто че будет пользоваться из-за того, что они не связаны напрямую с решением рабочих задач.
Способ с применением вариантов использования облегчает расстановку приоритетов требований. Высшим приоритетом обладают те функциональные требования, которые созданы на основе вариантов использования с высшим приоритетом. **Высший приоритет назначается по следующим причинам:**
— варианты импользования описывают один из основных бизнес-процессов, активизируемых системой;
— многие пользователи часто обращаются к ним;
— их запросил привилегированный класс пользователей;
— они предоставляют возможности, необходимые для соответствия требованиям;
— функции других систем зависят от их наличия.
Ловушка. *Не тратьте много времени на обсуждение деталей вариантов использования, которые не будут реализованы в ближайшие месяцы или годы. Вероятнее всего, они изменятся еще до начала сборки.*

Существуют также и преимущества технического характера. С помощью варианта использования можно выявить некоторые важные объекты предметной области и их взаимоотношения. Разработчики, использующие объектно-ориентированные методы дизайна, могут преобразовать варианты использования в объектные модели, такие, как диаграммы классов и диаграммы последовательностей. (Однако следует помнить, что варианты использования ни в коем случае не

ограничиваются разработкой объектно-ориентированных проектов). По мере того как с течением времени изменяются бизнес-процессы, задачи, реализуемые посредством определенных вариантов использования, также изменяются. Если вы проследите функциональные требования, дизайн, код и тесты вплоть до их истоков — пожеланий клиента — вам будет легче внести эти изменения бизнес-процессов во всю систему.

Каких ловушек следует опасаться при способе с применением вариантов использования

Как и любой другой способ разработки ПО, применение вариантов использования чревато возникновением множества проблем:

— Слишком много вариантов использования. Если вас захлестнул вал вариантов использования, вам, возможно, не удастся записать каждый из них на соответствующем уровне абстракции. Не создавайте отдельный вариант использования для каждого возможного сценария. Лучше включите нормальное направление развития, альтернативные направления и исключения в виде сценариев в один вариант использования, Как правило, количество вариантов использования превышает количество бизнес-требований и функций, однако функциональных требований обычно бывает намного больше, чем вариантов использования.

— Очень сложные варианты использования. Однажды я изучал вариант использования объемом в четыре страницы, плотно заполненных описанием этапов взаимодействия, встроенной логики и условий ответвления. Документ казался весьма невразумительным. Вам не дано контролировать сложность бизнес-задач, но вы можете выбрать способ их представления в вариантах использования. Выберите один успешный способ выполнения варианта использования, с одной комбинацией значений правильных и ложных значении для различных логических решений и назовите его нормальным на правлением. Другие успешные логические ответвления определите как альтернативные направления и назначьте исключения для обработки неудачных ответвлений. Альтернативных направлений может быть множество, однако каждое должно быть кратким и понятным, Чтобы не усложнять эту схему, записывайте варианты использования в терминах, основных для взаимодействий пользователя и системы, без особой детализации.

— Включение пользовательского интерфейса в варианты использования. Варианты использования должны описывать то, что пользователям необходимо выполнить с помощью системы, а не на то, как это будет выглядеть на экране. Сосредоточьтесь на концептуальном взаимодействии пользователя и системы, отложите работу над пользовательским интерфейсом до стадии дизайна. Например, правильная формулировка на этой стадии — «система предоставляет выбор», а не «система отображает раскрывающийся список». Не допускайте, чтобы дизайн пользовательского интерфейса диктовал направление разработки требований. Применяйте наброски экрана и карты диалогов (диаграммы архитектуры интерфейса), чтобы в визуальной форме представить взаимодействия исполнителя и системы, а не утвердить дизайн.

— Включения определения данных в варианты использования. Мне приходилось видеть варианты использования, которые включали в себя определения элементов данных и структур, которыми манипулируют в варианте использования. Участникам проекта было трудно отыскать в них необходимые определения, поскольку неясно, в каком именно варианте использования оно содержится. В таких случаях возможен повтор определений, а значит, и их рассинхронизация, когда один экземпляр изменяется, а остальные нет. Вместо того чтобы «разбрызгивать» определения по вариантам использования, соберите их данных в словарь данных, созданный для всего проекта.
— Варианты использования, которые непонятны пользователям. Если пользователи не видят связи описания вариантов использования со своими бизнес-процессами или задачами, то вариант использования следует подкорректировать. Составляйте варианты использования с точки зрения клиентов, а не системы и попросите клиентов проверить их. Возможно, вам не нужны излишне сложнее полные варианты использования.
— Новые бизнес-процессы. Пользователям будет трудно придумывать новые варианты использования, если ПО создается для поддержки процесса, которого еще нет. В этом случае сбор информации по требованиям может оказаться не моделированием бизнес-процесса, а изобретением такового. Рискованно ожидать, что с помощью новой информационной системы будет создан эффективный бизнес-процесс. Выполните модернизацию бизнес-процесса, прежде чем браться за полное описание новой информационной системы.
— Слишком частое упоминание слов расширить и включить. Начиная работу с вариантами использования, необходимо изменить отношение аналитиков, пользователей и разработчиков к требованиям. Тонкости взаимоотношений, определяемых словами расширить и включить, могут запутать кого угодно. Лучше избегать их, пока вы не освоите способ с применением вариантов использования.

Таблицы «событие — реакция»

Есть еще один способ систематизации и документации пользовательских требований: определить внешние события, на которые система должна реагировать. **Событием (event)** называется какое-либо изменение или действие в среде пользователя, вызывающее реакцию системы ПО (McMenamin и Palmer, 1984; Wiley, 2000). В таблице «событие-реакция» (event-response table) [ее также называют таблицей событий (event table) или списком событий (event list)] перечислены все такие события и ожидаемое поведение системы, которое должно последовать, как реакция на каждое событие. **Существует несколько типов системных событий:**
— взаимодействие пользователя с ПО, как когда он, например, вызывает вариант использования [иногда называется бизнес-событием (business event)]. Последовательность событий и реакций соответствует этапам взаимодействия для этого варианта использования, В отличие от вариантов использования, в таблице «событие — реакция» не описываются цели пользователя при работе с системой и не приводятся причины, по которым эта последовательность событий и реакций имеет

значение для пользователя;

— контрольный сигнал, чтение данных или прерывание, полученное от внешнего аппаратного устройства, например при изменении положения выключателя, изменении напряжения или перемещении мыши пользователем;

— событие инициируется в определенный момент времени (скажем, автоматический запуск экспорта данных в полночь) или когда истекает предварительно заданный период времени с момента определенного события (например, система фиксирует температуру, считываемую сенсором каждые 10 секунд).

Рисунок «Примеры системных событий и реакций»

Таблицы «событие — реакция» особенно хороши для управления системами, работающими в режиме реального времени. Таблица, приведенная ниже, представляет собой простую таблицу «событие — реакция», в которой частично описано поведение очистителей ветрового стекла автомобиля. Обратите внимание, что ожидаемая реакция зависит не только от события, но и от состояния, в котором находится система во время события. Например, результаты событий 4 и 5.1 различаются из-за того, были ли включены «дворники» в момент, когда пользователь настроил управление «дворниками» на периодический режим. Реакция может просто изменить некоторую внутреннюю системную информацию (события 4 и 7.1 в таблице) или привести к результату, заметному извне (большинство других событий).

Таблица «событие — реакция» для автомобильных дворников

ID	Событие	Состояние системы	Реакция системы
1.1	установка системы управления "дворниками" в режим медленной очистки	«дворники» отключены	установка механизма «дворников» в режим медленной очистки
1.2	установка системы управления "дворниками" в режим медленной очистки	быстрая очистка	установка механизма «дворников» в режим медленной очистки
1.3	установка системы управления "дворниками" в режим медленной очистки	периодическая очистка	установка механизма «дворников» в режим медленной очистки
2.1	установка системы управления "дворниками" в режим быстрой очистки	«дворники» отключены	установка механизма «дворников» в режим быстрой очистки
2.2	установка системы управления "дворниками" в режим быстрой очистки	медленная очистка	установка механизма «дворников» в режим медленной очистки
2.3	установка системы управления "дворниками" в режим быстрой очистки	периодическая очистка	установка механизма «дворников» в режим медленной очистки
3.1	отключение системы управления "дворниками"	быстрая очистка	1. завершение текущего цикла очистки 2. выключение механизма «дворников»
3.2	отключение системы управления "дворниками"	медленная очистка	1. завершение текущего цикла очистки 2. выключение механизма «дворников»
3.3	отключение системы управления "дворниками"	периодическая очистка	1. завершение текущего цикла очистки 2. выключение механизма «дворников»
4	установка системы управления "дворниками" в режим периодической очистки	«дворники» отключены	1. считывание временного интервала очистки 2. инициализация таймера очистки
5.1	установка системы управления "дворниками" в режим периодической очистки	быстрая очистка	1. считывание временного интервала очистки 2. завершение текущего цикла очистки 3. инициализация таймера очистки
5.2	установка системы управления "дворниками" в режим периодической очистки	медленная очистка	1. считывание временного интервала очистки 2. завершение текущего цикла очистки 3. инициализация таймера очистки
6	со времени предыдущей очистки прошел установленный интервал времени	периодическая очистка	выполнение одного цикла медленной очистки
7.1	изменение интервала периодической очистки	периодическая очистка	1. считывание временного интервала очистки 2. инициализация таймера очистки
7.2	изменение интервала периодической очистки	«дворники» отключены	реакция отсутствует
7.3	изменение интервала периодической очистки	быстрая очистка	реакция отсутствует
7.4	изменение интервала периодической очистки	медленная очистка	реакция отсутствует
8	получение сигнала на выполнение периодической очистки	«дворники» отключены	выполнение одного цикла медленной очистки

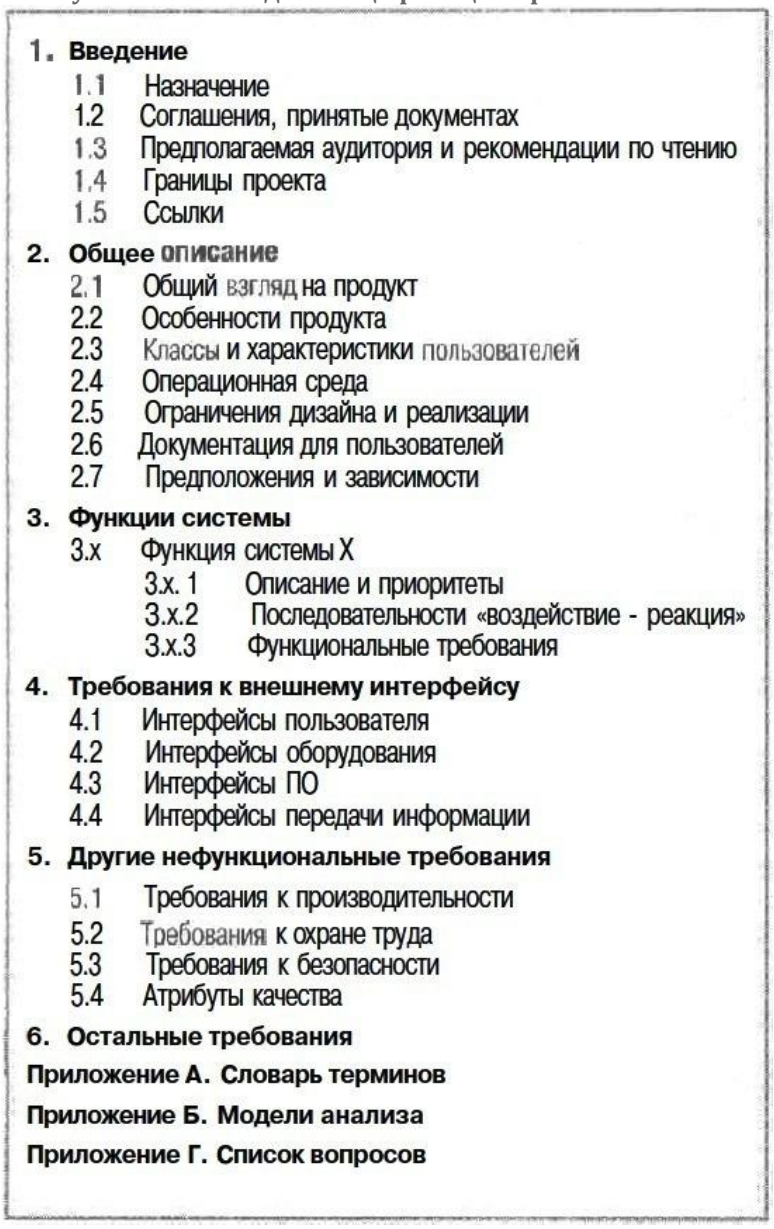
Информация в таблице «событие — реакция» фиксируется на уровне пользовательских требований. Если в таблице определены и названы все возможные комбинации событий, состояний и реакций (включая условия исключений), таблица также может служить частью функциональных требований для этого фрагмента системы. Однако аналитик должен добавить дополнительные функциональные и нефункциональные требования в спецификацию требований к ПО. Например, сколько «взмахов» в минуту делают «дворники» в быстром и медленном режиме очищения? В быстром или медленном режиме выполняется периодическая очистка? Является ли периодический режим непрерывно изменяющимся или он работает дискретно? Каковы максимальный и минимальный интервал времени при работе в периодическом режиме? Если вы закончите работу на уровне пользовательских требований, разработчику самому придется выяснять эту информацию. Помните, ваша цель — сформулировать требования настолько точно, чтобы разработчик знал, что строить.

Обратите внимание, что события, перечисленные в таблице «событие — реакция» записаны на важнейшем уровне (описывается суть события), а не на уровне реализации (описываются особенности реализации). В таблице ничего не говорится о том, как выглядит элемент управления «дворниками» или о том, как пользователь управляет ими. Дизайнер может реализовать эти требования как угодно — от традиционных элементов управления «дворниками», как в современных автомобилях, до системы распознавания голоса, реагирующей на произносимые команды: «включить дворники», «выключить дворники», «быстрая очистка», «очистить один раз» и т.д. Записывая пользовательские требования на важнейшем уровне, вы избежите введения ненужных ограничений дизайна. Однако следует фиксировать все известные ограничения по дизайну, чтобы заставить разработчика думать.

Шаблон спецификации требований к ПО

Каждая организация, специализирующаяся на разработке ПО, должна принять один или несколько стандартных шаблонов спецификации требований к ПО для использования в проектах. Доступны различные шаблоны спецификации (Davis, 1993; Robertson и Robertson, 1999; Leifingwell и Widrig, 2000). Многие применяют шаблоны, созданные на основе того, что описан в IEEE Standard 830-1998, «IEEE Recommended Practice for Software Requirements Specifications». Он годится для самых разных проектов, однако в нем встречаются ограничения и неясные места. Если вы беретесь за проекты различных типов и размеров, от конструирования новой объемной системы до небольших улучшений уже работающих систем, заведите для проектов каждого крупного класса отдельный шаблон спецификации.

Рисунок «Шаблон для спецификации требований к ПО»



1. **Введение**
   - 1.1 Назначение
   - 1.2 Соглашения, принятые документах
   - 1.3 Предполагаемая аудитория и рекомендации по чтению
   - 1.4 Границы проекта
   - 1.5 Ссылки
2. **Общее описание**
   - 2.1 Общий взгляд на продукт
   - 2.2 Особенности продукта
   - 2.3 Классы и характеристики пользователей
   - 2.4 Операционная среда
   - 2.5 Ограничения дизайна и реализации
   - 2.6 Документация для пользователей
   - 2.7 Предположения и зависимости
3. **Функции системы**
   - 3.x Функция системы X
     - 3.x. 1 Описание и приоритеты
     - 3.x.2 Последовательности «воздействие - реакция»
     - 3.x.3 Функциональные требования
4. **Требования к внешнему интерфейсу**
   - 4.1 Интерфейсы пользователя
   - 4.2 Интерфейсы оборудования
   - 4.3 Интерфейсы ПО
   - 4.4 Интерфейсы передачи информации
5. **Другие нефункциональные требования**
   - 5.1 Требования к производительности
   - 5.2 Требования к охране труда
   - 5.3 Требования к безопасности
   - 5.4 Атрибуты качества
6. **Остальные требования**

Приложение А. Словарь терминов

Приложение Б. Модели анализа

Приложение Г. Список вопросов

На рисунке «Шаблон для спецификации требований к ПО» показан шаблон спецификации требований к ПО, созданный на основе стандарта IEEE 830. В нем предлагается множество примеров дополнительных требований к продукту, которые вы можете включить в свою спецификацию. Если какой-то раздел вашего шаблона не годится для конкретного проекта, не удаляйте его заголовок, но укажите, что он неприменим. В этом случае у пользователя не возникнет подозрения, что что-то важное было пропущено по невнимательности. Если вам постоянно приходится пропускать одни и те же разделы, это означает, что шаблон следует настроить. Создайте оглавление и журнал изменений спецификации требований к ПО, где указаны дата изменения, сотрудник, внесший изменение, и ее причина. Иногда фрагмент информации логически подходит для нескольких разделов шаблона. Гораздо важнее аккуратно и последовательно фиксировать информацию, чем горячо обсуждать, где следует хранить каждый элемент.

Шаблон спецификации требований к ПО и шаблон документа об образе и границах в некоторых местах перекрываются (например, общие элементы есть в границах проекта, в описании особенностей продукта и в разделах операционной среды). Причина кроется в том, что вы создали только один документ, касающийся требований. Если вы используете оба шаблона, измените их таким образом, чтобы ликвидировать избыточность, при необходимости объединив разделы. Возможно, соответствующие разделы спецификации требований к ПО пригодятся для детализации определенного прогноза или высокоуровневой информации, содержащихся в документе об образе и границах проекта. Если же вы решите вырезать и вставить фрагменты из документа в другой, то возможна опасность того, что в обоих появятся ненужные повторы, В следующих разделах этой главы рассказано, какую информацию следует включать в каждый раздел спецификации требований к ПО. Вы можете объединить материал с помощью ссылки на другие документы (например, об образе и границах проекта или спецификацию интерфейса), а не дублировать его в спецификации требований к ПО.

Пользовательские интерфейсы и спецификация требований к ПО

Включение элементов пользовательского интерфейса в спецификацию имеет как преимущества, так и недостатки. Отрицательным моментом можно считать то, что изображения и архитектура пользовательского интерфейса отображают решения (дизайн), а не требования. Откладывая согласование решений в спецификации требований к ПО до завершения разработки пользовательского интерфейса, вы можете заставить нервничать людей, которые и так обеспокоены тем, что на работу над требованиями затрачено слишком много времени. Включение элементов пользовательского интерфейса в требования может сместить приоритеты, и описание функциональности окажется неполной.
Макет экрана не заменит пользовательских и функциональных требований. Не следует ожидать, что

разработчики смогут сделать вывод о базовой функциональности и взаимосвязи данных по моментальным снимкам экрана. У одной компании, создающей ПО для Интернета, постоянно возникали одни и те же проблемы из-за того, что разработчики непосредственно переходили к работе над визуальным дизайном сразу же после подписания контракта. У них не было ясного представления о том, как пользователи будут работать с Web-сайтом, поэтому им приходилось тратить массу времени на последующие исправления.

Из положительных сторон следует отметить, что изучение возможных пользовательских интерфейсов (таких, как рабочий прототип) делает требования более осязаемыми и для пользователей, и для разработчиков. Изображения пользовательского интерфейса помогают при планировании и оценке проекта. Подсчет элементов графического интерфейса пользователя (graphical user interface, GUI) или числа ***функциональных точек**** (function points), связанных с каждым экраном, позволяет оценить размер проекта и, следовательно, затраты на реализацию.

«Золотая середина» подразумевает включение концептуальных изображений — набросков — в спецификацию требований к ПО без обязательного точного соблюдения этих моделей при реализации. При этом улучшается взаимодействие специалистов без ненужных ограничений для разработчиков. Например, предварительный набросок сложного диалогового окна может проиллюстрировать назначение части требований, однако опытный визуальный дизайнер сумеет превратить его в диалоговое окно с вкладками для удобства работы пользователя.

*** *Функциональной точкой** называется количество обнаруженных пользователем функций приложения независимо от того, как они сконструированы. Вы можете оценить функциональные точки, исходя из требований пользователя, по числу внешних логических файлов и элементов вводимых данных, выводимых данных и запросов.*

Моделирование требований

Опыт показывает, что **модели анализа должны дополнять**, а не заменять спецификации требований на естественном языке (т.е. требования в текстовом представлении).

К моделям визуального представления относятся:

— диаграммы потоков данных (data flow diagrams, DFD);

— диаграммы «сущность — связь» (entity-relationship diagrams, ERD);

— диаграммы перехода состояний (state-transition diagrams, STD), называемые также диаграммами состояний;

— карты диалогов (dialog maps);

— диаграммы вариантов использования;

— диаграммы классов;

— диаграммы взаимодействия.

Системы обозначений (нотации) обеспечивают общий, стандартный для всей индустрии язык, которые пользуются участники проекта. Конечно, вы можете специально разработать диаграммы,

чтобы дополнить возможности устного и письменного общения в рамках проекта, однако не факт, что пользователи интерпретируют их одинаково. В то же время нельзя не признать ценность нестандартных приемов моделирования. Одна команда применяла средство для составления графика моделирования временных требований для встроенного ПО, причем за единицу измерения были приняты миллисекунды, а не дни или недели.

Эти модели годятся для разработки и изучения требований, а также для построения ПО. Будете ли вы использовать их для анализа или для дизайна, зависит от временных рамок и целей моделирования. Применение этих диаграмм для анализа требований позволит вам смоделировать предметную область или создать концептуальные представления новой системы. Они отображают логические аспекты компонентов данных предметной области, транзакции и преобразования, объекты реального мира и изменения состояния системы. Вы можете создавать модели на основании текстовых требований, чтобы отобразить их с различных точек зрения, или вывести детализированные функциональные требования из моделей высокого уровня, созданных на основе предоставленной пользователями информации. В процессе разработки модели демонстрируют, как вы намерены реализовать систему: ту базу данных, которую вы планируете создать, классы объектов, которые вы будете иллюстрировать примерами, и модули кода, которые вы собираетесь разрабатывать.

Ловушка. *Не следует полагать, что разработчики смогут просто преобразовать модели анализа в код, не разобрав подробно весь процесс. Поскольку в обоих типах диаграмм используются одни и те же системы обозначений, ясно назовите каждый из них моделью анализа (концепция) или моделью дизайна (то, что вы планируете создать).*

Приемы моделирования анализа, описанные в этой главе, поддерживаются различными коммерческими инструментами автоматизированного проектирования ПО (computer-aided software engineering, CASE). Использование этих инструментов обеспечивает некоторое преимущество перед обычными средствами рисования. Во-первых, они легко позволяют улучшить качество диаграмм при повторных просмотрах требований. Вам не удастся создать отличную модель с первого раза, поэтому итерацию можно назвать ключом к успеху при моделировании систем (Wiegers, 1996a). Кроме того, инструменты автоматизированного проектирования ПО «знают» правила для каждого метода моделирования, который они поддерживают. Они способны определить синтаксические ошибки и несоответствия, которые специалистам, проверяющим диаграммы, не всегда удается обнаружить. Эти инструменты связывают различные диаграммы друг с другом и с их общими определениями данных в словаре данных, а также позволяют поддерживать модели в согласованном состоянии и в соответствии с функциональным требованиям в спецификации требований к ПО.

Редкий случай, когда команде необходимо создать полный набор моделей анализа для всей системы. При моделировании сосредоточьтесь на наиболее сложных и опасных участках системы, а также на тех, где наиболее вероятны неясности и неопределенности. Элементы системы, от которых зависит ее безопасность и защита, а также элементы, влияющие на выполнение критически важных задач,

можно смело назвать кандидатами на моделирование, так как последствие дефектов в них окажутся особенно тяжелыми.

Варианты прототипов

Создание прототипов ПО делает требования более реальными, приближает варианты использования к «жизни» и закрывает пробелы в вашем понимании требований. Прототипы предоставляют пользователям экспериментальную модель или первоначальный срез новой системы, стимулируя их мышление и катализируя обсуждение требований. Обсуждение прототипов на ранних стадиях процесса разработки помогает заинтересованным в проекте лицам прийти к общему пониманию требований к системе, что уменьшает риск недовольства заказчиков.
Если вы не исправите эти проблемы, разница в ожиданиях между видением продукта пользователями и пониманием разработчиков гарантирована. Трудно точно представить поведение нового ПО при чтении текстовой документации или изучении аналитических моделей. Пользователи более готовы экспериментировать с прототипом (что весело), чем читать спецификацию требований к программному обеспечению (что скучно). Услышав от пользователей «узнаю, когда увижу», подумайте, как вы можете помочь им наглядно представить свои нужды. Однако если ни один из участников не представляет себе, что же должны создать разработчики, проект обречен.
Прототип (prototype) имеет множество значений, поэтому ожидания участников процесса прототипирования могут весьма различаться. Прототип самолета действительно летает — ведь это первый вариант настоящего самолета. Напротив, прототип ПО — это только часть или образец реальной системы — он может в принципе не делать ничего полезного. Прототипы ПО могут быть действующими моделями или статичными образцами; детализированным изображением экрана или его эскизами; наглядной демонстрацией или примерами реальной функциональности; имитацией или подражанием.

Назначение приоритетов требований

Для каждого продукта, на разработку которого выделены ограниченные ресурсы, необходимо определить относительные приоритеты возможностей. Расстановка приоритетов помогает менеджеру проекта разрешать конфликты, планировать выпуски и принимать необходимые компромиссы.

Зачем определять приоритеты требований

Когда ожидания клиентов высоки, а сроки поджимают, вам нужно, чтобы в продукте были реализованы самые ценные функции как можно раньше. Приоритеты — это способ разрешения борьбы между конкурирующими требованиями за ограниченные ресурсы. Определение относительного приоритета каждой возможности позволяет вам так планировать разработку, чтобы обеспечивать наибольшую ценность при наименьших затратах. Определение приоритетов наиболее

критично для работы в очень строгих временных рамках или при применении инкрементальной модели с жесткими, фиксированными сроками выпуска каждой версии продукта. При экстремальном программировании клиенты выбирают, какие рассказы пользователей (user stories) они желают видеть в каждом двух- или трехнедельном выпуске, а разработчики оценивают, сколько из этих рассказов они могут вместить в каждый выпуск.

Менеджер проекта должен сбалансировать желаемый объем проекта и ограничения, определяемые сроком, бюджетом, людскими ресурсами и качеством. Один из способов достижения этого — убрать (или отложить до более поздней версии) требования с низким приоритетом, когда принимаются новые, более важные требования или изменяются другие условия проекта. Если клиенты не классифицируют свои требования по важности и срочности, менеджерам проектов приходится делать это своими силами. Неудивительно, что клиентам не всегда нравится результат; поэтому они должны указывать, какие требования необходимы с самого начала, а какие могут подождать. Определяйте приоритеты на ранних стадиях проекта, когда больше шансов на успешный результат, и периодически возвращайтесь к ним.

Достаточно сложно заставить даже одного клиента решить, какие из его требований приоритетнее. Согласовать мнения нескольких клиентов с различными ожиданиями еще труднее. Люди по природе склонны отстаивать свои интересы и не жаждут жертвовать собственными нуждами ради чужой выгоды. Тем не менее участие в определении приоритетов требований — одна из обязанностей клиента в отношениях «клиент — разработчик».

И клиенты, и разработчики должны внести вклад в определение приоритетов требований. Клиентам больше всего нужны функции, наиболее ценные для бизнеса или удобства работы. Однако, когда разработчики обрисуют затраты, трудоемкость, технический риск или компромиссы, связанные с каждым требованием, клиенты могут передумать и прийти к выводу, что это требование не так важно, как они считали изначально. Разработчики же иногда решают на ранних стадиях реализовать некоторые функции с низким приоритетом из-за их влияния на архитектуру системы.

Шкала приоритетов

Четыре комбинации для определения шкалы приоритетов:

— требования с высоким приоритетом (high priority) — и важные (пользователям нужны функции), и срочные (они необходимы уже в следующем выпуске). Некоторые требования приходится включать в эту категорию согласно контрактным или юридическим обязательствам либо из-за непреодолимых бизнес-причин;

— требования со средним приоритетом (medium priority) — важные (пользователям нужны функции), но не срочные (они могут ждать следующего выпуска);

— требования с низким приоритетом (low priority) — не важные (пользователи при необходимости могут обойтись без этой функций) и не срочные (пользователи могут ждать, причем вечно);

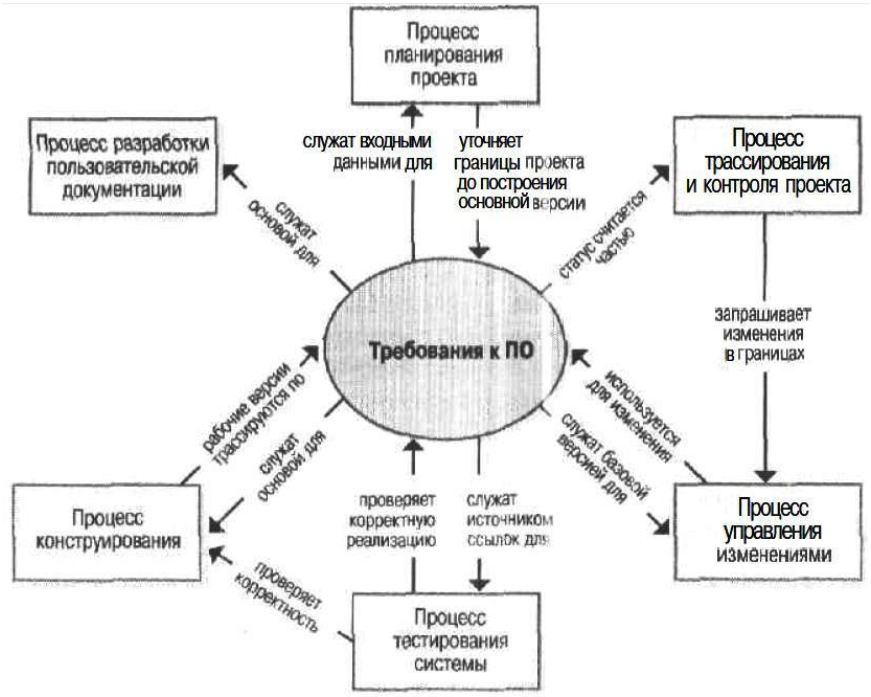
— требования в четвертой клетке кажутся срочными, но в действительности — они не важны. Не тратьте время на работу над ними. Они не сделают продукт более ценным.

Определение приоритетов на основе ценности, стоимости и риска

В малом проекте заинтересованные лица могут согласовать приоритеты требований неформально. Крупные или спорные проекты требуют более структурированного подхода, устраняющего из процесса некоторые эмоции, политику и догадки. Для помощи в определении приоритетов предлагается несколько аналитических и математических методик. Они предполагают оценку относительной ценности и относительной стоимости каждого требования. Требования с самым высоким приоритетом — те, что обеспечивают большую ценность продукта при меньшей стоимости. Субъективная оценка стоимости и ценности посредством попарного сравнения всех требований не годится, если требований более двух десятков.
Другая альтернатива — **Quality Function Deployment (QFD),** всесторонний метод определения относительной ценности для клиента предлагаемых функций продукта.
Третий подход, заимствованный из **Total Quality Management (TQM)**, позволяем оценить каждое требование по нескольким весомым критериям успеха проекта и подсчитать количество баллов для назначения приоритетов
требований.

Связь требований с другими составляющими проекта

Требования составляют основу любого хорошо организованного проекта по разработке ПО, поддерживая технические и организационные задачи. Изменения способов разработки и управления требованиями повлияют на эти задачи, и наоборот.



Планирование проекта. Требования служат основой для процесса планирования проекта. Те, кто отвечает за планирование, выбирают подходящий жизненный цикл разработки ПО и разрабатывают ресурсные сметы и график работы на основе требований. Эта процедура может выявить невозможность реализации всего желаемого набора возможностей за отведенное время при выделенных ресурсах, В результате планирования иногда становится ясно, что следует сузить проект или выбрать инкрементальную модель либо модель по-стадийной реализации необходимой функциональности.

Трассирование и контроль проекта. Трассирование подразумевает мониторинг статуса каждого требования, чтобы менеджеру проекта было ясно, действительно ли конструирование и проверка выполняются согласно принятому по плану. Если нет, он может потребовать сузить границы рассматриваемого объекта через процесс управления изменениями.

Управление изменениями. После того как набор требований внесен в основную версию, все последующие изменения выполняются только через определенный процесс управления изменениями. Этот процесс помогает гарантировать, что:

— все понимают эффект предлагаемого изменения;

— уполномоченные сотрудники смогут обоснованно принимать решения об изменениях;

— все люди, которых затронут изменения, осознают их;

— ресурсы и обязательства выверены;

— документация требований обновляется своевременно и точно.

Тестирование системы. Пользовательские и функциональные требования представляют собой важнейшие входные данные для тестирования системы. Если предполагаемое поведение ПО в различных условиях не определено четко, тестировщикам будет чрезвычайно трудно обнаружить

дефекты и проверить, вся ли запланированная функциональность реализована должным образом.

Процесс конструирования. Несмотря на то что работающая программа и есть конечный продукт проекта по разработке ПО, требования предоставляют основу для работы по конструированию и реализации и связывают воедино различные этапы разработки. Проверяйте построенные компоненты, чтобы удостовериться, что каждый из них точно соответствует требованиям. Тестирование блоков позволяет проверить, удовлетворяет ли их код спецификациям и соответствующим требованиям. Трассирование требований позволяет задокументировать, какие элементы конструкции и кода ПО были получены на основе каждого требования.

Разработка пользовательской документации. Однажды я делил офис с техническими писателями, которые готовили пользовательскую документацию для сложных программных продуктов. Я спросил одного из них, почему они работают так долго. «Мы уже заканчиваем, — ответил он. — Нам приходится учитывать окончательные изменения в интерфейсах пользователей и функции, убранные или добавленные в последний момент». Требования к продукту содержат данные для процесса разработки пользовательской документации, так что плохо сформулированные или запаздывающие требования порождают проблемы в документации. Неудивительно, что страдальцы в конце цепочки разработки требований — технические писатели и тестировщики — часто с энтузиазмом поддерживают улучшенные приемы конструирования требований.

Требования к ПО и управление риском

Риск (risk) — это условие, которое может повлечь какие-либо потери или другим способом поставить под угрозу успех проекта. Это условие еще не породило проблему, и вы хотите, чтобы так оно и оставалось. Эти потенциальные проблемы могут оказать неблагоприятный воздействия на стоимость, сроки, технический успех, качество продукта или эффективность работы команды. Управление риском — лучшая практика индустрии ПО — это процесс выявления, оценки и управления риском до того, как он нанесет ущерб вашему проекту.

Если что-либо нехорошее уже произошло с вашим проектом, то это — проблема, а не риск. Решайте текущие проблемы и вопросы, постоянно контролируя статус проекта и процессы введения поправок. Так как никто не может уверенно предсказывать будущее, управление риском — это способ минимизации вероятности потенциальных проблем или ущерба от них. Управление риском означает работу над потенциальной опасностью до того, как она перейдет в кризисную фазу. Это улучшает вероятность успеха проекта и уменьшает финансовые или другие последствия риска, которого не удастся избежать. Риск, лежащий вне сферы компетенции команды, следует передать руководству соответствующего уровня.

Поскольку требования так важны в проектах по созданию ПО, предусмотрительный менеджер уже на ранних стадиях проекта выявит риск, связанный с требованиями, и будет достаточно агрессивно контролировать его. Распространенные факторы риска, связанные с требованиями, включают неверное понимание требований, недостаточное вовлечение пользователей, неточности или и

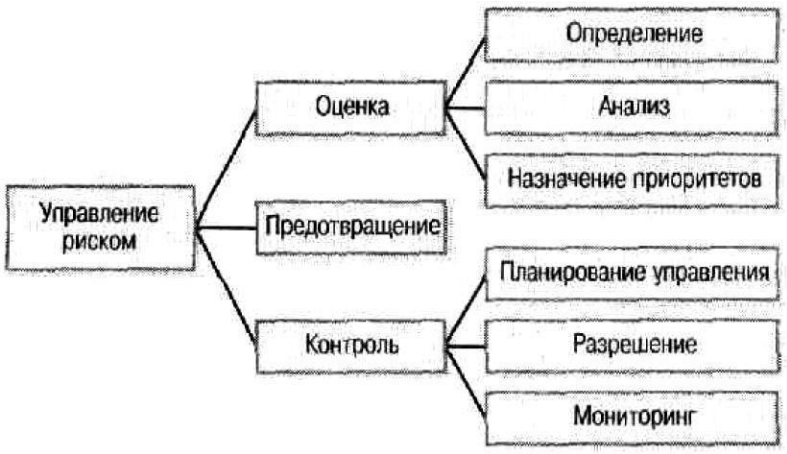
шенения в масштабах и целях проекта, и постоянно изменяющиеся требования. Менеджеры проектов могут управлять риском, связанным с требованиями, только в сотрудничестве с клиентами или их представителями. Совместное документирование факторов риска, связанных с требованиями и планирование действий по уменьшению их последствий укрепляет партнерство клиента и разработчика.

Основы управления риском при создании ПО

Проекты подстерегает масса рисков, помимо тех. что связаны с границами проекта и требованиями. Один из них — зависимость от внешних элементов, таких, как субподрядчик или другой проект, предоставляющий компоненты для повторного использования. При управлении проектом на первое место выходят такие виды риска, как неточные оценки, неприятие менеджерами точных оценок, недостаточная прозрачность статуса проекта и текучка кадров. Технологический риск представляет угрозу для очень сложных или использующих передовые разработки проектов. Недостаток знаний — — тоже очень опасен: исполнители не обладают достаточным опытом работы с используемыми технологиями или областью применения приложения. А постоянно изменяющиеся нормы законодательства могут разрушить и самые лучшие планы проектов.
Как и в случае с другими процессами, соотносите действия по управлению риском с масштабами своего проекта. Для небольших проектов достаточно простого списка источников риска, а ключевым элементом успешного крупномасштабного проекта считается структурированное планирование управления риском.

Составляющие управления риском

Управление риском (risk management) — это применение инструментальных средств и процедур для ограничения факторов риска в проекте приемлемыми рамками. Управление риском предоставляет стандартный подход к выявлению и документированию факторов риска, оценке их потенциального ущерба и предлагает стратегии их смягчения. Управление риском включает в себя следующие действия:



Оценка риска (risk assessment) — это процесс исследования проекта для выявления областей потенциального риска. Облегчите задачу выявления факторов риска, составляя списки факторов риска, типичных для разработки ПО, в том числе связанных с требованиями. При анализе риска вы исследуете потенциальные последствия конкретных факторов риска для вашего проекта. Определение приоритетов факторов риска поможет вам сконцентрироваться на наиболее опасных, оценивая потенциальную подверженность каждому из них. Подверженность — это функция, включающая как вероятность потерпеть урон из-за риска, так и масштабы этого урона.
Предотвращение риска (risk avoidance) — это один из способов решения вопроса: не делайте того, что рискованно. Вы можете избежать риска, не начиная определенные проекты, полагаясь на испытанные, а не самые передовые технологии или исключая функции, которые будет особенно трудно воплотить верно. Однако — разработка ПО рисковое предприятие, так что избежать риска означает потерять возможность.
Большую часть времени вам придется выполнять **контроль риска (risk control)**, что позволит управлять выявленными факторами риска с высоким приоритетом. Планирование управления риском подразумевает создание плана действий для каждого отдельного фактора, включая методы смягчения, планы на случай непредвиденных обстоятельств, ответственных лиц и сроки исполнения. Цель действий по смягчению воздействия риска — либо не позволить риску стать проблемой, либо уменьшить его вредное воздействие. Риск не будет контролировать сам себя, поэтому разрешение риска подразумевает реализацию планов по сдерживанию каждого риска. И, наконец, трассируйте свое продвижение к разрешению каждого риска посредством мониторинга риска, который должен стать частью вашего стандартного трассирования статуса проекта. Отслеживайте, насколько хорошо работают ваши действия по смягчению риска, ищите новые факторы риска, возникшие недавно, убирайте риски, угроза которых миновала, и периодически обновляйте приоритеты в вашем списке факторов риска.

Документирование рисков проекта

Недостаточно просто знать риски, угрожающие вашему проекту. Вам нужно управлять ими так, чтобы иметь возможность сообщать об областях и статусе риска заинтересованным в проекте лицам на протяжении проекта. Шаблон документирования отдельной области риска:

Идентификатор	*<порядковый номер>*
Дата открытия	*<дата, когда элемент риска был обнаружен>*
Дата закрытия	*<дата, когда элемент риска был закрыт>*
Описание	*<описание элемента риска в форме "причина-следствие">*
Вероятность	*<вероятность того, что элемент риска станет проблемой>*
Влияние	*<потенциальный урон, который может быть нанесен, если элемент риска станет проблемой>*
Подверженность	*<вероятность, умноженная на ущерб>*
План смягчения риска	*<один или несколько методов управления, избежания, уменьшения последствий или других способов минимизации риска>*
Ответственное лицо	*<человек, отвечающий за разрешение элемента риска>*
Срок исполнения	*<дата, к которой действия по смягчению риска должны быть выполнены>*

Иногда удобно хранить его в электронной таблице, что позволяет легко сортировать список областей риска. Вместо того чтобы включать его в свой план управления проектом или в спецификацию требований к ПО, ведите список областей риска в качестве отдельного документа — так его будет

легко обновлять на протяжении проекта.

Используйте формат «причина — следствие», документируя факторы риска. То есть сначала формулируйте причину риска, вызывающую вашу озабоченность, а затем ее потенциальный неблагоприятный результат—следствие. Часто люди, говорящие о риске, приводят только условие («клиенты не придут к единому мнению относительно требований к продукту») или только следствие («мы сможем удовлетворить лишь одного из наших основных клиентов»). Одна причина может повлечь несколько следствий, и несколько причин могут привести к одному и тому же следствию.

В шаблоне предусмотрены поля для записи о вероятности материализации риска в проблему, о негативном влиянии на проект как результат этой проблемы и об общей подверженности проекта риску. Я оцениваю вероятность по шкале от 0,1 (практически невозможно) до 1,0 (обязательно произойдет), а ущерб — по относительной шкале от 1 (нет проблем) до 10 (полный кошмар). Еще лучше попытаться оценить потенциальное влияние в единицах потерянного времени и денег. Умножьте вероятность на влияние, чтобы оценить уязвимость для каждого элемента риска.

Не пытайтесь оценить факторы риска слишком точно. Ваша цель — отделить наиболее угрожающие риски от тех, за которые не нужно хвататься немедленно. Проще оценивать и вероятность, и ущерб как высокий, средний или низкий. Факторы, имеющие как минимум одну оценку высокого уровня, требуют вашего внимания на ранних стадиях.

В поле для плана смягчения риска записывайте действия, которые вы намереваетесь предпринять для контроля риска. Некоторые стратегии смягчения снижают вероятность риска, другие — уменьшают его влияние. Учитывайте стоимость этих действий при планировании. Неразумно тратить $20000 на управление риском, который стоил бы вам лишь $10000. Вы также можете составить планы на непредвиденные обстоятельства для самых угрожающих областей риска, заранее решив, какие действия предпринять если несмотря на ваши усилия проект подвергнется воздействию риска. Назначьте для каждого элемента риска, которым намереваетесь управлять, ответственное лицо и определите сроки исполнения действий по смягчению. Долгосрочные или сложные элементы риска могут потребовать ступенчатой стратегии минимизации риска с массой промежуточных целей.

Планирование управления риском

Список факторов риска — не то же самое, что план управления риском. Для малого проекта вы можете включить свои планы управления риском в план управления проектом разработки ПО. Для большого проекта необходимо написать отдельный план управления риском, формулирующий предполагаемые подходы для выявления, оценки, документирования и учета риска. Этот план должен включать распределение ролей и обязанностей в действиях по управлению риском. Для многих проектов назначается менеджер по управлению риском, который отвечает за все, что может

пойти не так. В одной компании такого сотрудника называли «Иа-Иа», в честь печального персонажа из «Винни-Пуха», который постоянно горевал о том, как все может быть плохо.
Ловушка. *Не предполагайте, что фактор риска под контролем просто потому, что вы его обнаружили и выбрали действия по его смягчению. Эти действия нужно еще выполнить. Выделите достаточное время на управление риском в расписании проекта, чтобы не потратить впустую усилия, затраченные на планирование управления риском. Предусмотрите действия для смягчения риска, составление отчетов о статусе риска и обновление списка элементов риска в вашем списке задач по проекту.*

Установите периодичность мониторинга риска. Не упускайте из вида примерно десять факторов риска с наибольшим коэффициентом и регулярно оценивайте эффективность методов по смягчению. Когда действие по минимизации завершено, заново оценивайте вероятность и влияние соответствующего риска, а затем в соответствии с результатами обновляйте список факторов и остальные текущие планы по смягчению. Риск не обязательно взят под контроль просто потому, что выполнены действия по его минимизации. Вам нужно понять, снизили ли эти методы подверженность до приемлемого уровня и не осталась ли вероятность, что отдельный элемент риска перерастет в проблему.

Не под контролем
Менеджер проекта однажды спросил меня, что делать, если одни и те же пункты остаются в его еженедельном списке главных пяти факторов риска из недели в неделю. Это предполагает, что действия по минимизации воздействия этих элементов либо не исполняются, либо не эффективны. Если это так, то подверженность рискам, которые вы активно стараетесь взять под контроль, будет уменьшаться. В этом случае более значимыми и требующими вашего внимания становятся элементы риска, представлявшие меньшую угрозу, чем пять главных. Периодически проводите переоценку вероятности материализации угрозы каждого элемента риска и потенциального ущерба от этого, чтобы понять, насколько результативны ваши действия по смягчению риска.

Риск, связанный с требованиями

Факторы риска группируются по принадлежности к основным этапам конструирования требований: выявлению, анализу, документированию, проверке и управлению.

Выявление требований

Образ и границы проекта. Расползание границ становится более вероятным, если у лиц, заинтересованных в проекте, нет ясного, единого мнения о том, что продукт должен из себя представлять (и не представлять) и делать. Начиная проект, составьте документ об образе и границах, который содержит ваши бизнес-требования, и используйте его при выработке решений о принятии или изменении требований.

Время, затраченное на разработку требований. Жесткие сроки проекта часто заставляют менеджеров и клиентов пренебрегать составлением требований; они считают, что если программисты не начну работать над кодом немедленно, то не закончат вовремя. Проекты широко варьируются в зависимости от размера и класса приложения (например, информационные системы, системное ПО, коммерческие или военные), но по самым грубым прикидкам стоит расходовать 10-15% ресурсов проекта на действия по разработке требований. Записывайте, сколько усилий понадобилось в реальности на разработку требований к каждому проекту, чтобы иметь возможность оценить, достаточно ли этого, и улучшить планирование следующих проектов.

Полнота и корректность спецификации требований. Чтобы требования точно определяли потребности клиента, применяйте метод вариантов использования, чтобы выявить требования, концентрируясь на пользовательских задачах. Составляйте конкретные сценарии использования, пишите варианты тестирования на основе требований и просите клиентов разрабатывать свои критерии принятия. Создавайте прототипы, чтобы требования были понятны пользователям и чтобы получать от них конкретные отзывы. Привлеките представителей клиентов к экспертизе спецификации требований и моделей анализа.

Требования для суперсовременных продуктов. Легко ошибиться в оценке реакции ранка на продукты, первые в своем роде. Уделите большое внимание исследованию рынка, создавайте прототипы и используйте фокус-группы пользователей, чтобы получать раннюю и частую обратную реакцию относительно вашего продукта.

Определение нефункциональных требований. Поскольку основное внимание, естественно, обращено к функциональности продукта, легко упустить нефункциональные требования. Запрашивайте клиентов о качественных характеристиках, таки;:, как производительность, легкость и простота использования, целостность и надежность. Документируйте эти нефункциональные требования и критерии их принятия как можно точнее в спецификации требований к ПО.

Единство мнений клиентов относительно требований к продукту. Если различные клиенты вашего продукта не достигли соглашения относительно того, что именно вы должны разрабатывать, кто-то их них останется недоволен результатом. Определите основных клиентов и используйте сторонников продукта для получения адекватного представительства и вовлечения клиентов. Удостоверьтесь, что вы полагаетесь на правильно выбранных людей, которые имеют полномочия для принятия решений.

Не сформулированные требования. У клиентов часто есть скрытые ожидания, о которых они не сообщают, и поэтому те не документируются. Постарайтесь выявить любые допущения, которые может делать клиент. Используйте вопросы в свободной форме, чтобы поощрить клиентов больше делиться своими мыслями, пожеланиями, идеями, информацией и сомнениями.

Использование существующего продукта в качестве базовой версии. Разработка требований считается не важной в проектах следующего поколения или при переделке предыдущих проектов. Разработчикам иногда рекомендуют использовать существующий продукт в качестве источника

требований, со списком изменений и дополнений. Это вынуждает их собирать требования через инженерный анализ текущего проекта. Однако это не эффективный и не полный способ выявления требований, поэтому не удивляйтесь, если у новой системы проявятся некоторые недостатки исходной системы. Документируйте требования, извлекаемые таким способом, и просите клиентов проверить их, чтобы убедиться, что они верны и до сих пор актуальны.

Решения, предлагаемые в качестве потребностей. Предлагаемые пользователями решения могут скрывать их действительные нужды, вести к повтору неэффективных бизнес-процессов и заставлять разработчиков принимать плохие конструкторские решения. Задача аналитика — докопаться до потребности клиента, скрытой предлагаемым решением.

Анализ требований

Определение приоритетов требований. Удостоверьтесь, что каждое функциональное требование, функция или вариант использования получило приоритет и приписано к конкретной версии или итерации системы. Оцените приоритет каждого нового требования по отношению к объему оставшейся работы, чтобы иметь возможность принимать разумные решения о компромиссах.

Технически сложные функции. Оцените осуществимость каждого требования, чтобы определить те из них, реализация которых может занять больше времени, чем ожидается. Нам всегда кажется, что успех — рядом, за углом, поэтому трассируйте статус проекта, чтобы следить за требованиями, реализация которых отстает от графика. Старайтесь исправлять ситуацию как можно раньше.

Незнакомые технологии, методы, языки, инструменты или аппаратное обеспечение. Не нужно недооценивать кривую обучения работе с новыми методами, необходимыми для удовлетворения некоторых требований. Выявляйте эти рискованные требования в начале проекта и выделяйте достаточное время на фальстарты, обучение, эксперименты и прототипирование.

Спецификация требований

Понимание требований. Различия в интерпретации требований разработчиками и клиентами ведут к расхождениям в ожиданиях, когда произведенный продукт не способен удовлетворить нужды клиентов. Формальная экспертиза документации требований разработчиками, тестировщиками и клиентами уменьшают этот риск. Подготовленные, опытные аналитики требований могут задать клиентам верные вопросы и составить высококачественные спецификации. Модели и прототипы, представляющие требования с разных точек зрения, также позволяют выявить неясные, неопределенные требования.

Спешка, заставляющая пропускать пометки «TBD». Полезно выделять области спецификации требований, которые требуют доработки специальными пометками, например «TBD», но рискованно начинать конструирование, пока они не сняты. Записывайте имя человека, отвечающего за разрешение каждой неясности и планируемые сроки разрешения.

Неоднозначная терминология. Создайте словарь для определение бизнес- или технических

терминов, которые могут быть истолкованы разными читателями по-разному. В частности, определите любые термины, имеющие как общее, так и техническое или узкоспециальное значения. Создайте словарь данных, где определяются элементы и структуры данных. Экспертиза спецификации требований поможет участникам выработать общее понимание ключевых терминов и концепций.

Включение дизайна в требования. Описание дизайна в спецификации требований налагает ненужные ограничения на возможности, доступные разработчикам. А те сдерживают создание оптимальных конструкций. Удостоверьтесь, что требования описывают, что необходимо сделать для решения бизнес-проблемы, а не то, как это должно быть сделано.

Утверждение требований

Неутвержденные требования. Перспектива утверждения длинной спецификации требований пугает, как и мысль о написании вариантов тестирования в самом начале процесса разработки. Тем не менее, если вы подтвердите корректность и качество требований до начала конструирования, то сможете избежать значительной и дорогой переделки на более поздних стадиях проекта. Выделите время и ресурсы на эти действия в план проекта. Обяжите представителей клиентов участвовать в проверке требований, потому что только они могут судить, удовлетворят ли сформулированные требования их нужды. Также проводите пошаговые, неформальные экспертизы, чтобы выявлять проблемы как можно раньше и дешевле.

Качество проверки. Если проверяющие не знают, как проводить экспертизу документации требований и какие действия эффективны при проверке, они могут упустить серьезные дефекты. Обучайте всех членов команды, которые будут участвовать в экспертизе. Пригласите человека с опытом проверки из вашей или сторонней организации для наблюдения и, возможно, координации ваших первых опытов, чтобы тренировать участников.

Управление требованиями

Изменение требований. Вы можете уменьшить расползание границ проекта, используя документ о его образе и границах в качестве контрольных точек для утверждения изменений. Объединение выявления требований с существенным участием пользователей может сократить расползание требований примерно вполовину. Методы контроля качества, определяющие ошибки в требованиях на ранних стадиях, сокращают количество последующих модификаций. Чтобы уменьшить влияние изменения требований, отложите реализацию тех из них, которые, скорее всего, изменятся, пока они не будут определены, а также проектируйте систему, закладывая возможность легкой модернизации.

Процесс изменения требований. Риск, связанный с тем, как происходит изменение требований, зависит и от отсутствия определенного процесса изменений, использования неэффективных механизмов изменений и внесения коррективов без учета этого процесса. Развитие культуры и дисциплины управления изменениями требует времени. Очень важно, чтобы при изменении

требований применялся анализ воздействия предлагаемых изменений, решения принимал совет по управлению изменениями и использовалось инструментальное средство для поддержки определенной процедуры.

Нереализованные требования. Матрица трассируемое требований помогает избежать пропуска каких-либо требований во время проектирования,конструирования или тестирования.

Увеличение объема проекта. Если требования изначально плохо определены, дальнейшее их выявление может увеличить объем проекта. Реализация нечетко определенных областей продукта потребует больше усилий, чем ожидалось. Ресурсы проекта, распределенные в соответствии с начальным неполным требованиям, могут оказаться недостаточными для удовлетворения полного спектра нужд пользователя. Для смягчения этого риска планируйте жизненный цикл, состоящий из поэтапных или разработанных средствами инкрементального моделирования версий. Реализуйте базовую функциональность в первом выпуске и наращивайте способности системы далее.

Управление риском — ваш друг

Менеджер проекта может использовать управление риском, чтобы выяснить условия, которые могут повлечь неблагоприятные последствия для проекта. Представьте себе менеджера нового проекта, озабоченного привлечением нужных пользователей в выявление требований. Если он умен, то сообразит, что это условие представляет собой элемент риска, и внесет его в список, оценив его вероятность и влияние, основываясь на предыдущем опыте. Если по прошествии некоторого времени пользователи все еще не вовлечены в работу над проектом, риск возрастет и, может быть, даже уже угрожает успеху проекта. Мне удавалось убеждать менеджеров отложить проект, когда не удавалось привлечь достаточно представителей пользователей, приводя такой аргумент; мы не должны впустую тратить деньги компании на заранее обреченный проект.

Периодическое трассирование рисков позволяет менеджеру проекта знать масштабы угрозы от выявленных областей риска. Сообщайте о недостаточно контролируемом риске вышестоящему руководству, которое может принять решение исправить его либо продолжать работу, несмотря на риск. Управление риском помогает вам держать глаза открытыми и принимать обоснованные решения, даже если вам не удается контролировать каждую неприятность, с которой может столкнуться ваш проект.