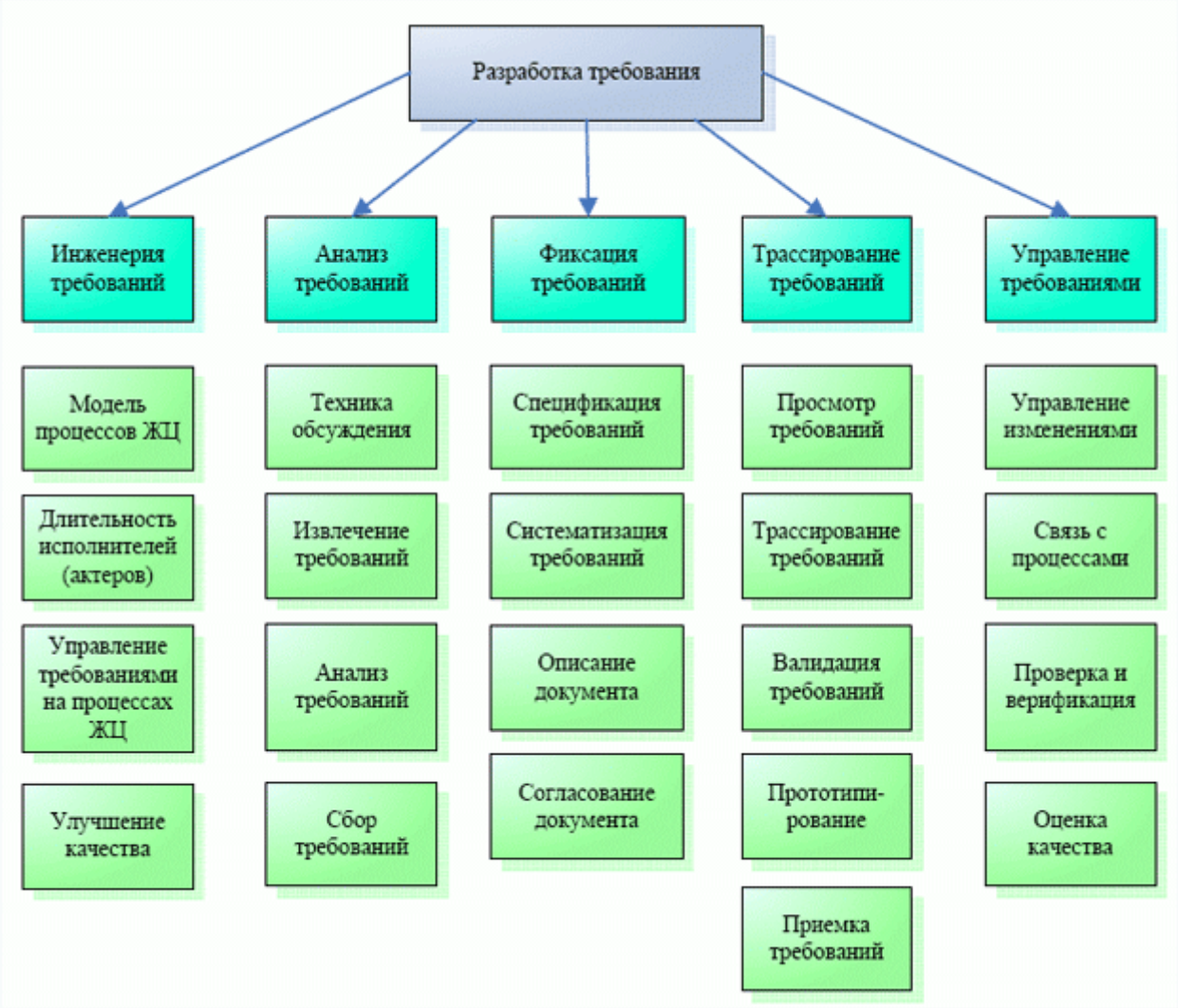
ИНТУИТ
НАЦИОНАЛЬНЫЙ ОТКРЫТЫЙ УНИВЕРСИТЕТ

Национальный Открытый Университет "ИНТУИТ": www.intuit.ru
Екатерина Лаврищева, Владимир Петрухин
Лекция 4. Методы определения требований в программной инженерии

Каждая программная система представляет собой определенный преобразователь данных и *вывод* результатов этого преобразования. Для построения ПС к ней формируются требования, которые определяют функции и разные виды обработки данных при выполнении функций. Эти требования являются предметом практического соглашения между заказчиком и разработчиком системы [3.1].



увеличить изображение

Рис. 3.1. Основные разделы разработки требований

В общем случае под *требованиями* к ПС понимают свойства, которыми должна обладать система для адекватного выполнения предписанных ей функций. Примерами таких функций могут быть бизнес-функции, документооборот, *управление данными* и структурой информации, необходимой для принятия системных решений и др. В ядре знаний SWEBOK изложены основные концепции и особенности инженерии требований и приведены на рис. 3.1.

Основные инструменты инженерии требований – *UML* и *RUP*, согласно которым требования определяются и уточняются с помощью вариантов использования use *case*, которые преобразуются к проектным решениям и к архитектуре системы.

Далее рассмотрим общие, практические и объектно-ориентированные подходы к разработке требований к системе, учитывая приведенные *разделы* на рис. 3.1.

3.1. Общие подходы к определению требований

Определение требований – это нетривиальная задача и проводится, как правило, путем обсуждения взглядов заказчика на систему с будущими ее разработчиками. Заказчик предъявляет свои потребности к автоматизации функций и задач системы и формулирует их в виде разных видов требований, классификация которых приводится ниже.

3.1.1. Классификация требований

До настоящего времени отсутствуют общепринятые определения терминов, которыми пользуются для описания: требований пользователя, требований к ПО, функциональных требований, системных требований, технологических требований и требований к продукту. В разных источниках понятие требований определяются исходя из разных взглядов на то, что по ним создается.

Согласно ряду публикаций формирование требований к ПО рассматривается как некоторая деловая игра, во время которой выявляются интересы заинтересованных в разработке ПО лиц, правила реализации этих интересов в конкретном ПО. При этом высказываются разного рода претензии и ограничения на свойства и способы обеспечения требований для получения конечного результата – программного продукта. Кроме того, нет формализованных методов сбора и описания требований, а также отсутствует общепринятое определение самого понятия – требование. Приведем некоторые из них [3.1–3.3].

Требования – это утверждения о функциях и ограничениях системы.

Требования – это свойства, которыми должен обладать продукт, чтобы представлять какую-то ценность для пользователей.

Требования – это спецификация того, что и как должно быть реализовано.

Согласно международному глоссарию по терминологии требования включают описание:

1. условий или возможностей, необходимых пользователю для решения поставленных проблем или достижения целей;
2. функций и ограничений, которыми должна обладать система или системные компоненты, чтобы выполнить договор заказчика на разработку системы;
3. положений и регламента используемых стандартов, отображаемых в спецификациях или других формальных документах на систему;
4. документированное представление условий, возможностей ограничений среды на проектирование системы.

Требования к продукту охватывают условия пользователей на внешнее поведение системы и разработчиков на некоторые параметры системы. Термин *пользователи* относится ко всем лицам, заинтересованным в создании системы.

Требования к ПО состоят из требований пользователей, функциональных, системных и нефункциональных требований.

Требования пользователей (*user requirements*) основываются на целях и задачах, которые пользователям позволит решать будущая система. К способам представления этого вида требований относятся варианты использования, сценарии, прецеденты, таблицы "событие-отклик" и т.п.

Системные требования (*system requirements*) определяют внешние условия для выполнения системных функций и ограничений на создаваемый продукт, а также требования к описанию подсистем (функциональных, программно-аппаратных). Системные требования накладывают ограничения на архитектуру системы, средства ее представления и функционирования. Для описания системных требований используются специальные шаблоны и формы, помогающие описывать входные и выходные данные и автоматизировать *трассировку требований*.

Требования к атрибутам качества (*quality attributes*) представляют собой некоторые ограничения к свойствам функций или к системе, важных для пользователей или разработчиков. Например, переносимость, целостность и устойчивость к сбоям системы.

Функциональные требования – это перечень функций или сервисов, которые должна выполнять система, а также ограничений на

данные и поведение системы. Спецификация функциональных требований (software requirements specification) включает в себя описание функций, которые не должны быть противоречивыми и взаимоисключающими.

Нефункциональные требования определяют условия и среду выполнения функций (например, защита и доступ к БД, секретность и др.), они непосредственно не связаны с функциями, а отражают пользовательские потребности к выполнению функций. Они характеризуют принципы взаимодействия со средами или другими системами, а также учитывают время работы, защиту данных, а также стандарты качества для достижения отдельных показателей или атрибутов качества. Эти требования отражают потребности заказчиков системы, т.е. тех, кто финансирует проект, покупает систему или проводит ее маркетинг.

Нефункциональные требования могут иметь числовое представление (время ожидания ответа, количество обслуживаемых клиентов, объем БД и др.), а также содержать значения *показателей надежности* системы, период смены версий системы и др. Для большинства современных многопользовательских ПС требования включают условия и ограничения типа:

конфиденциальность, безопасность и защиту данных;

отказоустойчивость;

одновременность доступа к системе пользователей;

время ожидания ответа при обращении к системе (производительность);

состав выполняемых функций системы (запуск, скорость реакции и др.);

стандартные положения к формулировке требований.

Для каждой системы формулируются нефункциональные требования, относящиеся к защите данных, несанкционированному доступу, к регистрации событий системы (резервное копирование, восстановление БД, аудит отказов и др.). Эти требования на этапе анализа и проектирования структуры системы определяются и формализуются аналитиками. В случае обеспечения безопасности системы определяются категории пользователей системы, которые имеют право доступа к тем или иным данным и компонентам.

При этом используется система мер по регистрации пользователей, их идентификация, аутентификация и др. Защита данных реализуется в соответствии с регламентированным доступом к данным (например, к таблицам, файлам, БД). Если же требуется ограничить

доступа к данным (например, к отдельным записям, полям в таблице), то в системе предусматриваются специальные средства (например, мандатная защита).

Для восстановления и хранения резервных копий БД, архивов баз данных анализируются возможности СУБД и способы обеспечения требуемого уровня бесперебойной работы системы, а также правил доступа неавторизованных пользователей и мер борьбы с разными угрозами, которые поступают извне от пользователей, не имеющих прав доступа к некоторым данным или ко всем данным системы.

К выходному продукту предъявляются нефункциональные требования:

к применению (качество пользовательского интерфейса, учебные курсы и др.);

к производительности (пропускная способность, время реакции и др.);

к надежности выполнения (ошибки, *интенсивность отказов* и др.);

к интерфейсным внешним атрибутам, с которыми взаимодействует система.

Процесс описания функциональных и нефункциональных требований, а также требований к характеристикам качества с учетом стандарта ISO/IEC 9126-94 уточняется при разработке архитектуры системы. При спецификации требований важной является проблема стандартизации терминологии для нескольких классов ПрО (например, информационные технологии, системы реального времени и др.). Имея стандартизированный понятийный базис для большинства ПрО, можно достигнуть единого с заказчиком понимания терминов, которые используются при описании концептуальной модели и спецификации требований к системе.

В спецификациях требований используются терминологические дескрипторы и общепринятые термины, принятые для структуры ПО и функций, названий показателей качества, документации, алгоритмов и структур данных.

С помощью спецификаций системных и нефункциональных требований задаются принципы взаимодействия проектируемой системы с другими средами, платформами и общесистемным обеспечением (БД, СУБД, сеть и др.).

Документ со спецификациями требований завершается на этапе проектирования архитектуры и согласуется с заказчиком и разработчиком. В дальнейшем этот документ используется в качестве руководства к действиям при выполнении задач на этапах ЖЦ разработки программного продукта. При выявлении на этих этапах несогласованных требований, проводится их уточнение и соответственно изменение процесса разработки системы.

3.1.2. Анализ и сбор требований

В современных информационных технологиях процесс ЖЦ, на котором фиксируются требования на разработку системы, является определяющим для задания функций, сроков и стоимости работ, а также показателей качества, которых необходимо достигнуть в процессе разработки. Выдвижение требований проводится путем обсуждения проекта, анализа предметной области и определения подходов к проектированию промежуточных продуктов на этапах ЖЦ.

Требования отражают потребности людей (заказчиков, пользователей, разработчиков), заинтересованных в создании ПС. Заказчик и разработчик совместно проводят обсуждение проблем проекта, сбор требований, их анализ, пересмотр, определение необходимых ограничений и документирование.

Обсуждение проекта системы проводится в целях выработки первых впечатлений и выводов относительно целесообразности выполнения проекта и прогнозирования реальности его выполнения в заданные сроки и бюджет, которые определяет заказчик.

Современные ПС предоставляют набор услуг для выполнения функций ПрО, которые ориентированны на определенную профессиональную деятельность пользователей (например, веб-сервисы). Лицо, которое

заказало проект системы, желает получить от разработчика набор необходимых услуг. К участникам системы относятся операторы, менеджеры разных уровней, бухгалтеры и т.п. Именно они будут обращаться к системе за услугами, получать от нее сообщения, реагировать на них в соответствии со своими профессиональными обязанностями.

Оценить возможность реализации услуг в проекте заказываемой системы в заданный срок и бюджет, могут разработчики системы. Среди них назначается главный аналитик, ответственный за требования к системе и главный программист, ответственный за их реализацию. Они проводят согласование требований и определение области действия проекта на совместных переговорах с заказчиком для уточнения следующих вопросов:

- спектра проблем ПрО, при решении которых будут реализованы услуги системы;
- функциональное содержание услуг;
- операционную среду работы системы.

В обсуждении требований на систему принимают участие:

- представители заказчика из нескольких профессиональных групп;
- операторы, обслуживающие систему;
- аналитики и разработчики будущей системы.

Согласованная область действий по проекту дает возможность оценить требуемые инвестиции в проект, заранее определить возможные риски и способности разработчиков выполнить проект. Итогом обсуждения проекта может быть решение о развертывании реализационных работ на проекте или отказ от него.

Анализ требований начинается после обсуждения проблематики проекта. При рассмотрении требований среди них могут оказаться

- неочевидные, не одинаково важные, которые брались из устаревших источников и документов заказчика;
- разные типы, которые соответствуют разным уровням детализации проекта и требующие применения методов управления ими;
- постоянно изменяемые, развиваемые и уточняемые;
- с уникальными свойствами или значениями;
- сложные по форме и содержанию, трудные для согласования их с заказчиком.

Разработчики требований должны обладать определенными знаниями в данной предметной области и уметь провести:

- анализ проблем предметной области, потребностей заказчика и пользователей системы,
- выявление функций системы, которые должны быть реализованы в проекте,
- внесение изменений в отдельные элементы требований.

В требованиях к ПС, кроме проблем системы, фиксируются реальные потребности заказчика, касающиеся функциональных, операционных и сервисных возможностей разрабатываемой системы. Результаты обследования и анализа предметной области фиксируются в документе описания требований и в договоре между заказчиком и исполнителем проекта.

Ошибки по причине нечетких или неоднозначных формулировок требований, которые могут привести к тому, что будет изготовлена система, не удовлетворяющая заказчика. Поэтому на этапах разработки требования должны постоянно уточняться и переутверждаться заказчиком. В отдельных случаях внесенные изменения в требования могут привести к необходимости перепроектировать отдельные части или всю систему в целом. Согласно статистике, доля ошибок в постановке требований и в определении задач системы превышает долю ошибок, допускаемых во время кодирования системы. Это объясняется субъективным характером процесса формулирования требований и отсутствием способов их формализации. В США, например, ежегодно расходуется до $ 82 млрд. на проекты, признанные после реализации не соответствующими требованиям заказчиков.

Существующие стандарты (ГОСТ 34.601-90 и ГОСТ 34.201-89) на разработку требований к системе и документам фиксируют результаты создания программного, технического, организационного и др. видов обеспечения автоматизированных систем на этапах ЖЦ.

Сбор требований. Источниками сведений для формирования требований могут быть:

- цели и задачи проекта, которые формулирует заказчик разработчиком будущей системы, должны осмысливаться ими;
- коллектив, выполняющий реализацию функций системы, не должен использовать старую систему, переставшую удовлетворять заказчика или персонал.

Изучение и фиксация реализованных функциональных возможностей в действующей системе дает основу для учета имеющегося опыта и формулирования новых требований к ней. При этом необходимо отделить новые требования от требований к старой системе, чтобы не повторить неудачные решения старой системы в новом ее варианте.

Требования к системе формулируются заказчиком в терминах понятий проблемной области с учетом терминологического словаря, ведомственных стандартов, условий среды функционирования будущей системы, а также трудовых и финансовых ресурсов, выделенных на разработку системы.

Методы сбора требований следующие:

- интервью с представителями интересов заказчика системы;
- наблюдение за работой действующей системы для отделения проблемных свойств, которые обусловлены кадровыми ресурсами;
- примеры возможных вариантов выполнения функций, ролей ответственных лиц, запускающих эти варианты или взаимодействующих с системой при ее развертывании и функционировании.

Внешние и внутренние аспекты требований соответствуют характеристикам качества и касаются свойств создаваемого продукта, а именно функциональности системы, ее назначения и выполнения в заданной среде. Конечный пользователь ожидает достижения максимального эффекта от применения выходного продукта и ориентируется на его конечное эксплуатационное качество.

При определении требований, относящихся к внешним и внутренним характеристикам качества, выбираются методы их достижения на процессах ЖЦ. Внутренние характеристики предназначены для достижения необходимых внешних показателей качества и применяются при оценке промежуточных (*рабочих) продуктов* ПС на процессах ЖЦ.

Разработанные требования представляются в специальном документе неформально, который является основой заключения контракта на разработку системы между заказчиком и разработчиком.

3.1.3. Инженерия требований

Инженерная дисциплина анализа и документирования требований заключается в преобразовании предложенных заказчиком требований к системе в описание требований к ПО, их спецификацию и валидацию. Она базируется на *моделях* процессов разработки требований, действиях актеров и управлении постепенным преобразованием требований к проектным решениям и спецификациям компонентов с обеспечением их качества.

Модель процесса определения требований – это схема процессов ЖЦ, которые выполняются от начала проекта и до тех пор, пока не будут определены и согласованы требования.

Управление требованиями к ПО заключается в планировании и контроле формирования требований, задании на их основе проектных решений, в преобразовании их в спецификации компонентов системы на других процессах.

Качество и процесс улучшения требований – это процесс проверки характеристик и атрибутов качества (надежность, реактивность и др.), которыми должна обладать система и ПО, методы их достижения на процессах ЖЦ.

Управление требованиями к системе – это руководство процессами формирования требований на всех этапах ЖЦ, которое включает *управление изменениями требований*, отражающих свойства программного продукта, а также восстановление источника требований. Неотъемлемой составляющей процесса управления является трассирование требований, состоящее в отслеживании правильности задания и *реализации требований* к системе и ПО на этапах ЖЦ и обратный процесс сверки ПС с заданными требованиями.

Основные задачи управления требованиями это:

- разработка атрибутов требований,
- управление вариантами требований,
- управление рисками, возникающими при неточном определении требований,
- контроль статуса требований, измерение усилий при формировании требований;
- реализация требований на этапах ЖЦ.

Разработка и управление требованиями связана с другими областями знаний (рис. 3.2.): проектирование, интеграция, управление качеством, версиями, рисками и др. Кроме того, приведены основные задачи разработки требований: спецификация и утверждение, формирование проектных решений.

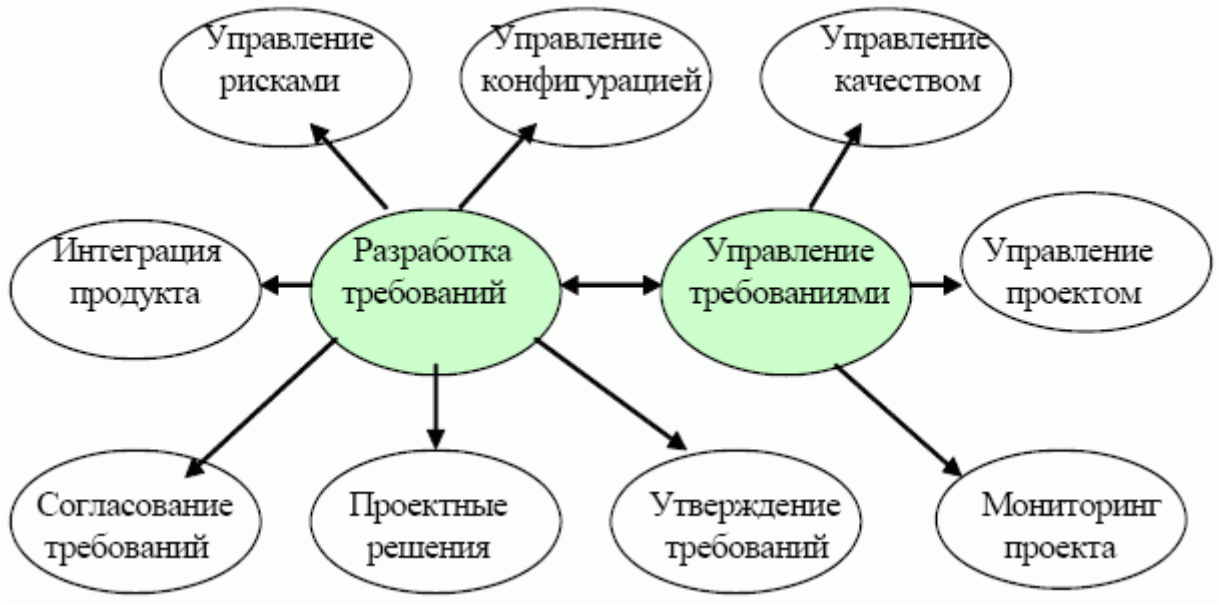


Рис. 3.2. Разработка, управление требованиями и связь с задачами SWEBOK

Планирование работ на проекте касается вопросов организации интеграции компонентов, управления рисками, версиями системы, на которые влияют заданные требования и их изменения.

Управление рисками состоит в контроле появления и обнаружения неадекватных ситуаций при *реализации требований*, оценке их влияния на другие процессы и в предупреждении *рисковых ситуаций*. Управления версиями системы включает формирование конфигурации системы в принятых для системы терминах и обозначениях.

3.1.4. Фиксация требований

Сбор требований является начальным этапом процесса разработки ПС, завершается формированием списка требований к системе, именуемого в отечественной практике техническим заданием. Фиксация требований (Requirement Capturing) в техническом задании определяется желаниями заказчика получить при реализации заданные им свойства системы. При этом предусматривается спецификация, верификация и валидация требований на правильность, соответствие и полноту.

Спецификация требований к ПО – это формализованное описание функциональных, нефункциональных и системных требований, требований к характеристикам качества, а также к структуре ПО, принципам взаимодействия с другими компонентами, алгоритмам и структуре данных системы.

Валидация требований – это проверка требований, для того чтобы убедиться, что они определяют именно данную систему. Заказчик и разработчик ПО проводят экспертизу сформированного варианта требований с тем, чтобы разработчик мог далее проводить его проектирование. Одним из методов валидации является прототипирование, т.е. быстрая отработка отдельных требований на конкретном инструменте, анализ масштаба изменения требований, измерение функциональности и стоимости системы, а также определение зрелости процессов определения требований.

Верификация требований – это процесс проверки правильности спецификации требований на их соответствие, непротиворечивость, полноту и выполнимость, а также на соответствие стандартам. В результате проверки требований создается согласованный выходной документ, устанавливающий полноту и корректность требований к ПО, а также возможность продолжить проектирование ПО.

3.1.5. Трассировка требований

Одной из главных проблем сбора требований является проблема их изменения. Требования создаются итерационно путем постоянного общения представителей заказчиков с аналитиками и разработчиками будущей системы в целях выявления потребностей. Требования изменяются в зависимости от задач и условий их определения, а также постоянного уточнения на этапе заключения договора на создание системы. На момент заключения договора состав требований, их виды и свойства становятся более полными, т.е. соответствуют взглядам заказчика на создаваемую систему [3.4].

Одним из инструментов установления зависимости между сформулированными требованиями и их изменениями является трассировка, т.е. поддерживается развитие и обработка требований с прослеживанием идентифицированных связей, которые должны быть зафиксированы по двум направлениям – от источника требований к реализации и, наоборот (рис. 3.3.). Выявляются причины появления разнообразных неточностей, добавлений и определяется необходимость внесения изменений в требования в одном из приведенных направлений.

Рис. 3.3. Типы трассируемости требований

Если после разработки некоторого *рабочего продукта* возникает потребность в изменении отдельных требований или необходимость проследить за происхождением внесенных требований в одном из направлений данной схемы трассирования, то уточняются связи между отдельными требованиями и элементами рабочих продуктов В случае трассирования требований от продукта, движение в обратном направлении, т.е. – к требованиям, можно выяснить, как написана каждая строка этого продукта и соответствует ли она отдельным атрибутам требований. Связи трассируемости требований помогают найти незапланированные и реализованные некоторые функции или фрагменты программ, не соответствующие заданным требованиям. И, наоборот, выявить нереализованные требования к функциональности. Взаимосвязи и зависимости между отдельными требованиями сохраняются, например, в таблице трассируемости, удаляются или модифицируются при различных изменениях.

Методы трассировки базируются на формальных спецификациях связей между элементами требований либо ограничиваются описаниями функций, ситуаций, контекста и возможных решений. Основу трассировки составляют:

- требования, которые изменяются при их формировании;
- некоторые детали выполнения функций в рабочем продукте системы, которые не предусматривались, но появились в связи с возникшей практической ситуацией;
- связи между различными моделями процесса проектирования системы на ЖЦ программного продукта и принятые решения о необходимости изменения требований в связи с появившимися недостатками в промежуточном продукте;
- информация о согласованных атрибутах требований на разных уровнях рассмотренной схемы трассирования и сохранение ее матрице трассирования;
- специальные системные требования, касающиеся повторного использования готовых компонентов или частей системы;
- результаты тестирования, по которым можно определить наиболее вероятные части кода, требующие проверки на наличие в них дефектов.

В матрице требований в строках указываются *пользовательские требования*, а столбцах – функциональные требования, элемент проектирования, вариант версии и др. В этих столбцах заполняются данные о степени выполнимости системных требований на каждом элементе создаваемого продукта. Механизм ссылок в таблице позволяет проверять связанные с каждым элементом продукта *диаграммы вариантов использования*, потоки данных, классы и др.

Процедура трассирования состоит в следующем:

- выбирается элемент (функция, фрагмент или некоторая часть) из матрицы трассирования требований, за которым проводится прослеживание на этапах ЖЦ;
- составляется список вопросов, по которым на каждом этапе проверяются связи при *реализации требований* в продукте, и если изменяется какое-то звено в цепочке требований (рис. 3.3.), то может модифицироваться процедура разработки этого элемента на последующем этапе ЖЦ;
- проводится мониторинг статуса каждого требования на соответствие выполнения согласно принятому плану;
- уточнение ресурсов выполнения проекта при необходимости проведения изменений в требования и в элементы проекта.

Условием принятия решения о возможных модификациях требований и результатов промежуточного проектирования, является обновленная информация о связях между различными частями системы и первоначально заданными требованиями к ним. Трассировка обеспечивает:

- ввод более сложных отношений вместо простых связей или специфических отношений;
- использование разных путей трассировки (между моделями или иерархическими связями);
- ведение базы данных объектов трассировки и отношений между ними.

Трассировка может быть выборочной для отдельных объектов или связанной с другими объектами, а также с возможными переходами от одной *модели проектирования* к другой путем проверки трансформации одних

объектов в другие.

3.2. Объектно-ориентированная инженерия требований

В объектно-ориентированных подходах и методах разработки программных систем главным является *объект*. *Объектно-ориентированный подход*, в частности *UML моделирование* позволяет с помощью вариантов использования задавать требований. *Основные средства UML* к формированию и *представлению требований* к системе и к *ПО* – это use *case* сценарии или прецеденты.

Представленные сценариями или прецедентами требования к системе в *UML*, последовательно трансформируются к другим сценариям, приближающим к логической структуре системы. Главные элементы сценариев – это объекты, классы объектов и акторы, задающие действия *по* выполнению системы.

3.2.1. Сценарный подход

Одним из методов построения проектной модели системы, логической и физической моделей являются сценарии use case, используемые для визуального представления требований в проектной модели системы. Эта модель уточняется и дополняется новыми сценариями для получения логической и физической моделей. Термин сценарий обозначает некоторый вариант представления модели выполнения системы [3.1, 3.5].

При применении сценарного подхода общая цель системы декомпозируется на отдельные подцели, для которых определяются функциональные или нефункциональные требования и проектные решения. Цели, как источники требований к системе, позволяют выявить противоречия и ограничения на выполнение функций и установить зависимости между ними, устранить конфликты между целевыми функциями, а также объединить некоторые из них между собою в определенные отношения.

После выявления целей определяются носители интересов, которым отвечает каждая цель, и возможные варианты удовлетворения составных целей в виде сценариев работы системы, которые помогают пользователю получить представление о назначении и функциях системы. Это соответствуют первой итерации определения требований к разработке системы.

Далее производится последовательная декомпозиция сложной проблемы к виду совокупности целей, каждая из которых трансформируется в совокупность возможных сценариев использования системы, а затем в совокупность взаимодействующих объектов.

Т.е. имеем цепочку трансформаций:

проблема ⟶ *цель* ⟶ *сценарий* ⟶ *объект*.

Она отражает степень *концептуализации* анализируемой проблемы, и ее декомпозицию с помощью вариантов использования для снижения сложности системы. *Трансформация данной* цепочки выражается в терминах базовых понятий проблемной области и активно используется для представления и развития моделей системы.

Каждый сценарий инициирует актор, выступающий в роли пользователя определенной работы в системе, представленной этим сценарием. Фиксацию ролей акторов можно рассматривать как определенный шаг при выявлении целей системы и постановщиков задач, для решения которых создается система.

Актор – это внешний фактор и его действия, носящие недетерминированный характер. В роли актора может выступать и программная система, если она инициирует выполнение некоторых работ, удовлетворяющих поставленной цели системы. Актор, как абстракция внешнего объекта, может быть человеком или внешней системой. В модели системы актор может быть представлен классом, а пользователь – экземпляром класса. Если актором является другая

ПС, то он будет представлять ее интересы. При этом одно лицо может быть экземпляром нескольких акторов.

Если актор находится вне системы, то он взаимодействует с ней через сценарий, который инициализирует последовательность операций для выполнения системы. Когда пользователь, как экземпляр актора, вводит определенный символ для старта экземпляра соответствующего сценария, то это приводит к выполнению ряда действий в системе, которые заканчиваются тогда, когда экземпляр сценария находится или в состоянии ожидания очередного входного символа или завершения сценария

Экземпляр сценария существует, пока он выполняется и его можно считать экземпляром класса, он имеет состояние и поведение. Взаимодействие между актором и системой порождает новый сценарий или объект, которые изменяют внутреннее состояние системы. Если несколько сценариев системы имеют одинаковое поведение, то их можно рассматривать как класс сценариев.

При внесении изменений осуществляется повторное моделирование акторов и запускаемых ими сценариев. Сценарий – это цепочка событий, инициированных актором, и реакции на них. Каждого актора обслуживает соответствующая совокупность сценариев.

Можно выделить базовую цель событий, существенную для сценария, и альтернативную в случае ошибок пользователя и др. Для задания модели сценариев используется специальная графическая нотация со следующими правилами:

актор обозначается иконой человека, под которой указывается название;

сценарий изображается овалом, в середине которого указывается название иконы;

актор связывается стрелкой с каждым овалом запускаемого им сценария.

Пример *диаграммы сценариев* для читателя библиотеки в роли актора, который запускает заданный сценарий при обращении к автоматизированной системе обслуживания библиотеки, представлен на рис. 3.4.

Все сценарии, которые включены в систему, обведены рамкой, определяющей границы системы, а актор находится вне рамки, являясь внешним фактором по отношению к системе.

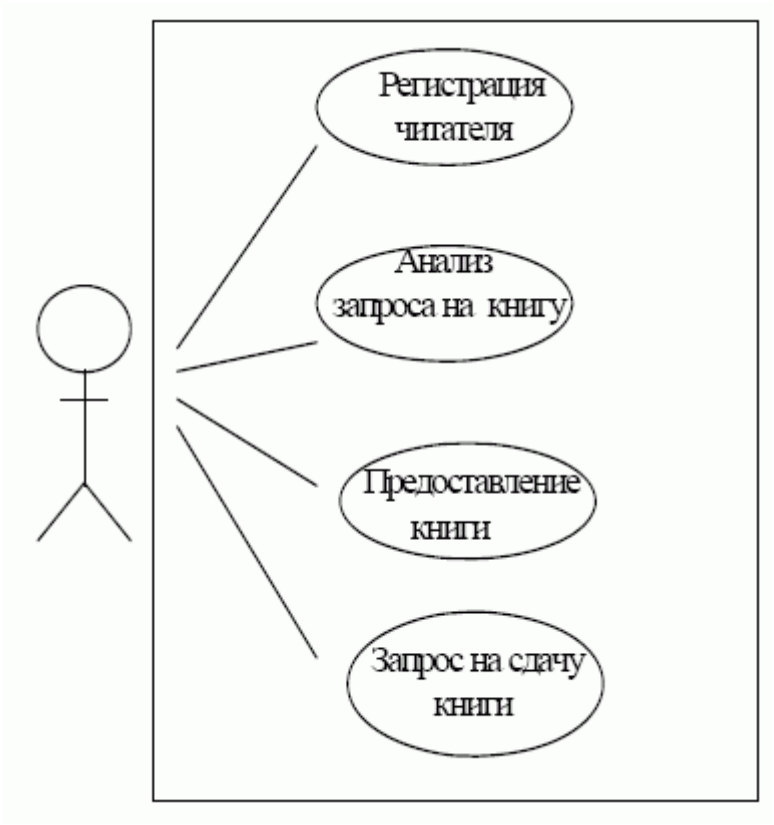


Рис. 3.4. Пример диаграммы сценариев

Отношения между сценариями. Между сценариями задаются отношения пунктирными стрелками с указанием названия типа отношений.

Для сценариев можно задавать два типа отношений:

1. *отношение "расширяет"* означает, что функция одного сценария является дополнением к функции другого и используется при наличии нескольких вариантов одного и того же сценария (рис. 3.5).

Рис. 3.5. Пример отношения "расширяет" Инвариантная часть сценария изображается в виде основного сценария, а отдельные варианты как расширения. При этом основной сценарий является устойчивым, не меняется при расширении вариантов функций и не зависит от них.

2. *отношение "использует"* означает, что некоторый сценарий может быть использован как расширение нескольких других сценариев (рис. 3.6).

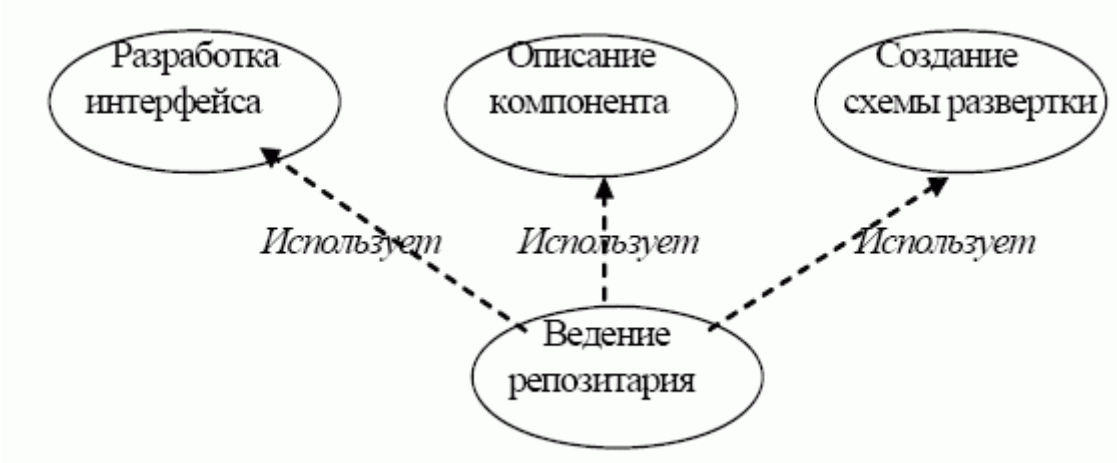


Рис. 3.6. Пример отношения "использует"

На данном рисунке показан сценарий "ведение репозитария", который связан отношением "использует" с несколькими сценариями – разработка интерфейса, описание компонента, создание схемы развертки.

Инженерия требований завершается построением модели требований, которая включает в себя:

1. описание требований и основных понятий ПрО;
2. модель сценариев;
3. интерфейсы сценариев.

Модель сценариев – это неформальное описание каждого из входящих в нее *диаграмм сценариев*.

Описание сценария задается последовательностью следующих элементов:

название сценария (некоторой цели системы), задаваемое на диаграммах модели требований, и которое является ссылкой на другой сценарий;

краткое содержание сценария в неформальном представлении;

список акторов, которые будут запускать сценарии модели;

параметры взаимодействия системы с акторами, запрещенные действия актора и возможные последствия;

предусловия, определяющие начальное состояние сценария на момент его запуска, и условия успешного выполнения;

функции, которые реализуются при выполнении сценария;

нестандартные ситуации, которые могут появиться при выполнении сценария (например, ошибка в действиях актора или системы).

На дальнейших этапах ЖЦ сценарий актора трансформируется в сценарий поведения системы, т.е. к элементам модели добавляются нефункциональные требования, обеспечивающие запуск сценария, ввод данных и отработку нестандартных ситуаций.

На процессе проектирования ЖЦ выполняется преобразование трансформированного сценария в описание функциональных компонентов системы и проверка их правильности методом верификации.

В описании интерфейсов компонентов отражаются требования пользователей к системе, собранные на этапах анализа, сбора и определения требований. Компоненты и их интерфейсы размещаются в репозитарии. На основе построенных сценариев можно построить прототип системы для моделирования действий акторов при выполнении сценариев и отработке разных их деталей.

3.2.2. Описание требований прецедентами

Альтернативным термином для сценария являются прецеденты. Как и в случае сценариев, задача описания требований прецедентами сводится к анализу *дерева целей* системы и к описанию реакции системы в случае недостижимости той или иной поставленной цели к проектируемой системе. Основным условием описания требований с помощью прецедентов является полнота системных требований, касающихся интерфейса пользователя, протоколов и форматов ввода-вывода [3.2,3.5].

Содержательной стороной системной части требований является описание функций, данных и принципов функционирования. Методология формирования требований с помощью прецедентов реализована в среде Rational Rose (www.rational.com.uml) и включает построение ряда моделей на основе прецедентов.

Функциональные возможности проектируемой системы могут задаваться набором прецедентов, каждый из которых представляет некоторый поток событий в системе, когда прецедент будет выполнен. Каждый прецедент выполняет свою задачу. Набор прецедентов устанавливает все возможные пути применения системы.

Прецеденты играют определенную роль в каждом из основных процессов проектирования: разработка требований, анализ и проектирование, выполнение и испытание системы. Экземпляр прецедента – это последовательность действий, выполняемых системой и наблюдений за получением результата.

В управляемом прецедентами проекте разрабатываются два представления системы – внешнее и внутреннее. Прецедент отражает внешнее представление ПрО, которое определяет, что должно исполниться в системе, чтобы обеспечить заказчику требуемые результаты. Это представление разрабатывается, когда проводится обсуждение целей и задач системы и задание их прецедентами. Во внешнем представлении прецендента определяются принципы взаимодействия системы и ее субъектов.

Реализация прецедента является внутренним его представлением в системе и принципом организации работы системы для достижения планируемых результатов. Она охватывает сущности, которые участвуют в выполнении прецедента, и связи между ними. Иными словами, разрабатывается такое представление прецедентов, чтобы каждый из прецедентов выполнял определенное действие для достижения требуемых результатов.

В процессе анализа проблемы и формирования требований создается модель прецедентов моделируемой цели системы, которая состоит из:

- используемых терминов (глоссария) предметной области;
- основных действующих лиц и их целей;
- используемых технологий и принципов взаимодействия с другими системами;
- требований к форматам и протоколам взаимодействия;
- требований к тестированию и к процедуре развертывания системы у заказчика;
- организационных вопросов управления процессом разработки системы.

На этапе анализа и проектирования модель прецедентов реализуются в модели проекта в терминах взаимодействующих объектов, т.е. дается описание того, как прецедент будет выполняться в системе.

С синтаксической точки зрения эта модель имеет следующий вид.

```
<Модель прецедента> ::=
    <имя прецедента/действующего лица>,
    <имя роли / краткое описание роли действующего лица>,
    <описание границ системы>,
    <список всех заинтересованных лиц в анализе ключевых целей системы>,
    <исходные условия>,
    <результат успешного завершения определения целей системы>,
    <шаги сценария для формирования шаблона достижения целей проекта>,
    <описание информации, необходимой разработчику для реализации системы>.
```

Данный подход к представлению системы с помощью прецедентов задается в форме Vision-шаблонов [3.5]. Он применяется в офисной сфере, где деловой прецедент отражает представление этой сферы с внешней стороны, чтобы обеспечить субъекта требуемыми результатами. При выполнении делового прецедента определяется взаимодействие деловой сферы и субъекта. Совокупность деловых прецедентов устанавливает границы системы.

Внутреннее представление делового прецедента – это реализация, которая охватывает функции деловых работников и соответственно участвующих в их выполнении, а также связи между ними. Такое задание системы разрабатываются для того, чтобы решить, как должна быть организована работа в каждом деловом прецеденте для достижения результатов.

Контрольные вопросы и задания

1. Приведите классификацию требований.
2. Определите назначение функциональных и нефункциональных требований.
3. Назовите источники сведений для задания требований.
4. Приведите задачи обследования, анализа и сбора требований.
5. Определите инженерию требований.
6. Дайте определение спецификации требований.
7. Приведите задачи трассировки требований.
8. Определите сущность объектно-ориентированной инженерии требований.
9. Назовите роли действующих лиц формирования требований.

10. Определите понятие актора.
11. Назовите виды отношений объектов в модели.
12. Определите виды отношений в сценарном подходе.
13. Определите представление модели ПрО прецедентами.

Внимание! Если Вы увидите ошибку на нашем сайте, выделите её и нажмите Ctrl+Enter.