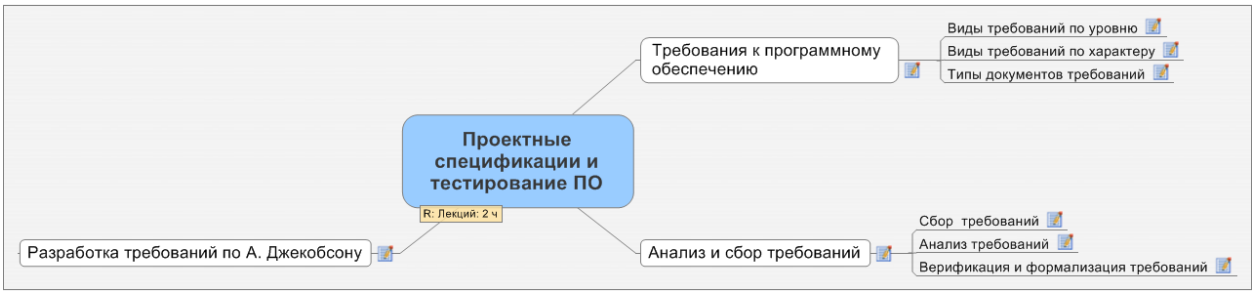
Лекция 12. Проектные спецификации и тестирование ПО



1 Требования к программному обеспечению

Требования к программному обеспечению — совокупность утверждений относительно свойств программной системы, подлежащая реализации при создании ПО. Создаются в процессе анализа требований к программному обеспечению.

Требования могут выражаться в виде текстовых утверждений и графических моделей.

1.1 Виды требований по уровню

- Бизнес-требования — определяют назначение ПО
- Бизнес-правила — определяют ограничения, проистекающие из предметной области и свойств автоматизируемого объекта (предприятия)
- Пользовательские требования — определяют набор пользовательских задач, которые будет решать программа, а также способы (сценарии) их решения в системе
- Системные требования и ограничения — определения набора элементарных операций, которые должна выполнять система, а также различных условий, которым она может удовлетворять

Пользовательские требования могут выражаться в виде фраз утверждений, в виде способов применения (use case), пользовательских историй (user story), сценариев взаимодействия (scenario).

К системным ограничениям относятся ограничения на программные интерфейсы, требования к атрибутам качества, требования к применяемому оборудованию и ПО.

1.2 Виды требований по характеру

- Функциональные требования — требования к поведению системы («ЧТО» она делает)
- Нефункциональные требования — требования к характеру поведения системы («КАК» она это делает)

1.3 Типы документов требований

В зарубежной и российской практике встречаются следующие типы документов требований:

- Концепция программы
- Требования к ПО

Требования к ПО часто (ошибочно) называют техническим заданием, частью которого они являются в автоматизированных информационных системах.

За создание требований к ПО чаще всего в российской практике отвечает Системный аналитик, иногда — Бизнес- аналитик.

Для графических моделей требований исторически использовались диаграммы: ER (IDEF1FX), IDEF0, IDEF3, DFD, UML, OCL, SysML, ARIS (eEPC, VAD).

2 Анализ и сбор требований

В современных информационных технологиях этап жизненного цикла, на котором фиксируются требования на разработку ПО системы, является определяющим для задания функций, сроков и стоимости работ, а также показателей качества и др.

Анализ и сбор требований является довольно нетривиальной задачей, поскольку в реальных проектах приходится сталкиваться с такими общими трудностями:

- требования не всегда очевидны и могут исходить из разных источников, их не всегда удается ясно выразить словами;
- имеется много различных типов требований на различных уровнях детализации и их
- число может стать большим, требующим ими управлять;
- требования связаны друг с другом, а также с процессами разработки ПО и постоянно меняются.

Требования имеют уникальные свойства или значения свойств. Например, они не являются одинаково важными и простыми для согласования.

2.1 Сбор требований

Источниками сведений о требованиях могут быть:

- цели и задачи системы, которые формулирует заказчик. Для однозначного их понимания разработчику системы необходимо их тщательно осмыслить и согласовать с заказчиком;
- действующая система или коллектив, выполняющий ее функции. Система, которую заказывают, может заменять собою старую систему, переставшую удовлетворять заказчика или действующий персонал. Изучение и фиксация ее функциональных возможностей дает основу для учета имеющегося опыта и формулирования новых требований к системе.

При этом имеется определенная опасность перенесения недостатков старой системы в новую, поэтому нужно уметь отделить новые требования к реализуемой проблеме от заложенных неудачных решений в старой системе.

Таким образом, требования к системе формулируются исходя из:

- знаний заказчика относительно проблемной области, формулирующего свои проблемы в терминах понятий этой области;
- ведомственных стандартов заказчика и требований к среде функционирования будущей системы, ее исполнительских и ресурсных возможностей.

Методами сбора требований являются:

- интервью с носителями интересов заказчика и операторами;
- наблюдение за работой действующей системы с целью отделения ее проблемных свойств от тех, которые обусловлены структурою кадров;
- сценарии (примеры) возможных случаев выполнении функций системы, ролей лиц, запускающих эти сценарии или взаимодействующих с системой во время ее функционирования.

Продукт процесса сбора требований – неформализованное их описание – основа контракта на разработку между заказчиком и исполнителем системы. Такое описание является входом для следующего процесса инженерии требований - анализа требований. Исполнитель этого процесса выполняет дальнейшее уточнение и формализацию требований, а также их документирование в нотации, понятной коллективу разработчиков системы.

2.2 Анализ требований

Это процесс изучения потребностей и целей пользователей, классификация и их преобразование к требованиям системы, к ПО, установление и разрешение конфликтов между требованиями, определение приоритетов, границсистемы и принципов взаимодействия со средой функционирования. Требования классифицируются по функциональному и нефункциональному принципу, а также по отношению их в внешней или внутренней стороне системы.

Функциональные требования относятся к заданию функций системы или ее ПО, к способам поведения ПО в процессе выполнения функций и методам преобразования входных данных в результаты.

Нефункциональные требования определяют условия и среду выполнения функций (например, защита и доступ к БД, секретность, взаимодействие компонентов функций и др.).

Разработанные требования специфицируются и отражается в специальном документе, по которому проводится согласование требований для достижения взаимопонимания между заказчиком и разработчиком.

Функциональные требования отвечает на вопрос "что делает система", а нефункциональные требования определяют характеристики ее работы (вероятность сбоя системы, защита данных и др.). К основным нефункциональным требованиям, существенным для большинства ПС и выражающих ограничения, актуальные для многих проблемных областей относятся:

- конфиденциальность;
- отказоустойчивость;
- несанкционированный доступ к системе;
- безопасность и защита данных;

- время ожидания ответа на обращение к системе;
- свойства системы (ограничение на память, скорость реакции при обращении к системе и т. п.).

Для большинства этих ограничений может быть зафиксирован спектр характерных понятий – дескрипторов, которые используются для наименования и раскрытия смыслового названия. Состав дескрипторов для ряда нефункциональных требований зафиксирован в соответствующих международных и ведомственных стандартах, которые позволяют избежать неоднозначности в их толковании.

Функциональные требования отражают семантические особенности проблемной области, а терминологические расхождения для них являются достаточно существенными. Имеется тенденция к созданию стандартизации понятийного базиса большинства проблемных областей, которые имеют опыт компьютеризации. Следующий шаг анализа требований - установление их приоритетов и избежание конфликтов между ними.

Продукт процесса анализа – построенная модель проблемы, которая ориентирована на понимание этой модели исполнителем до начала проектирования системы.

2.3 Верификация и формализация требований

Спецификация требований к ПО – процесс формализованного описания функциональных и нефункциональных требований, требований к характеристикам качества в соответствии со стандартом качества ISO/IEC 12119-94, которые будут учитываться при создании программного продукта на этапах ЖЦ ПО. В спецификации требований отражается структура ПО, требования к функциям, показателям качеству, которых необходимо достигнуть, и к документации. В спецификации задается в общих чертах архитектура системы и ПО, алгоритмы, логика управления и структура данных. Специфицируются также системные требования, нефункциональные требования и требования к взаимодействию с другими компонентами (БД, СУБД, протоколы сети и др.).

Формализация включает в себя определение компонентов системы и их состояний; правил взаимодействия компонентов и определения условий, которые должны выполняться при изменении состояний компонентов, в формальном виде.

Для формального описания поведения системы используются языки инженерных спецификаций, например, UML.

В качестве формальной модели для описания требований используются базовые протоколы, которые позволяют использовать дедуктивные средства верификации в сочетании с моделированием поведения систем путем трассирования.

Валидация (аттестация) требований - это проверка требований, изложенных в спецификации для того, чтобы убедиться, что они определяют данную систему и отслеживание источников требований.

Заказчик и разработчик ПО проводят экспертизу сформированного варианта требований с тем, чтобы разработчик мог далее проводить разработку ПО.

Одним из методов аттестации является прототипирование, т.е. быстрая отработка отдельных требований на конкретном инструменте и исследование масштабов изменения требований, измерение объема функциональности и стоимости, а также создание моделей оценки зрелости требований.

Верификация требований – это процесс проверки правильности спецификаций 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 продолжения проектирование ПО.

3 Разработка требований по А. Джекобсону

Данный метод является методом с последовательным выявлением объектов, существенных для домена проблемной области. Все методы анализа предметной области в качестве первого шага выполняют выявление объектов. Правильный выбор объектов обуславливает понятность и точность требований.

Рассматриваемый метод Джекобсона дает рекомендации относительно того, с чего начинать путь к пониманию проблемы и поиску существенных для нее объектов.

Автор назвал свой метод как базирующийся на вариантах (примерах или сценариях –use case driven) системы, которую требуется построить. В дальнейшем термин сценарий используется для обозначения варианта представления системы.

Сложность проблемы обычно преодолевается путем декомпозиции ее на отдельные компоненты с меньшей сложностью. Большая система может быть сложной, чтобы представить ее компоненты программными модулями. Поэтому разработка системы с помощью данного метода начинается с осмысления того, для кого и для чего создается система.

Сложность общей цели выражается через отдельные подцели, которым отвечают функциональные или нефункциональные требования и проектные решения. Цели являются источниками требований к системе и способом оценки этих требований,путем выявления противоречий между требованиями и установления зависимостей между целями.

Цели можно рассматривать, как точку зрения отдельного пользователя или заказчика или среды функционирования системы. Цели могут находиться между собою в определенных отношениях, например, конфликтовать, кооперироваться, зависеть одна от другой или быть независимыми.

Следующим шагом после выявления целей является определение носителей интересов, которым отвечает каждая цель, и возможные варианты удовлетворения составных целей в виде сценариев работы системы. Последние помогают пользователю получить представление о назначении и функциях системы. Эти действия соответствуют первой итерации определения требований к разработке.

Таким образом, производится последовательная декомпозиция сложности проблемы в виде совокупности целей, каждая из которых трансформируется в совокупность возможных вариантов использования системы, которые трансформируются в процессе их анализа в совокупность взаимодействующих объектов.

Определенная цепочка трансформации (проблема – цели – сценарии – объекты) отражает степень концептуализации в понимании проблемы, последовательного снижения сложности ее частей и повышения уровня формализации моделей ее представления.

Трансформация обычно выражается в терминах базовых понятий проблемной области, активно используется для представления актеров и сценариев, или создается в процессе построения такого представления. Каждый сценарий инициируется определенным пользователем, являющимся носителем интересов. Абстракция определенной роли личности пользователя– инициатора запуска определенной работы в системе, представленной сценарием, и обмена информацией с системой - называется *актером*. Это абстрактное понятие обобщает понятие действующего лица системы. Фиксация актеров можно рассматривать, как определенный шаг выявления целей системы через роли, которые являются носителями целей и постановщиками задач, для решения которых создается система.

Актер считается внешним фактором системы, действия которого носятнедетерминированный характер. Этим подчеркивается разница между пользователем как конкретным лицом и актером– ролью, которую может играть любое лицо в системе.

В роли актера может выступать и программная система, если она инициирует выполнение определенных работ данной системы. Таким образом, актер – это абстракция внешнего объекта, экземпляр которого может быть человеком или внешней системой. В модели актер представляется классом, а пользователь – экземпляром класса. Одно лицо может быть экземпляром нескольких актеров.

Лицо в роли (экземпляр актера) запускает операции в системе, соотнесенные с поведением последовательности транзакций системы и называется *сценарием*. Когда пользователь, как экземпляр актера, вводит определенный символ для старта экземпляра соответствующего сценария, то это приводит к выполнению ряда действий с помощью транзакций, которые заканчиваются тогда, когда экземпляр сценария находится в состоянии ожидания входного символа. Экземпляр сценария существует, пока он выполняется и его можно считать экземпляром класса, в роли которого выступает описание транзакции.

Сценарий определяет протекание событий в системе и обладает состоянием и поведением. Каждое взаимодействие между актером и системой это новый сценарий или объект. Если несколько сценариев системы имеют одинаковое поведение, то их можно рассматривать как класс сценариев. Вызов сценария – это порождение экземпляра класса. Таким образом, сценарии – это транзакции с внутренним состоянием.

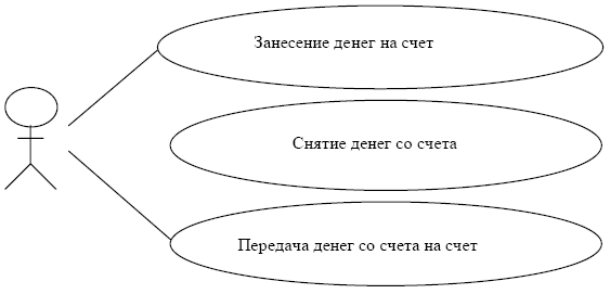
Модель системы по данному методу основывается на сценариях и внесение в нее изменений должно осуществляться повторным моделированием актеров и запускаемых ими сценариев. Такая модель отражает требования пользователей и легко изменяется по их предложению. Сценарий – это полная цепочка событий, инициированных актером, и спецификация реакции на них.

Совокупность сценариев определяет все возможные варианты использования системы. Каждого актера обслуживает своя совокупность сценариев. Можно выделить базовую цель событий, существенную для сценария, а также альтернативные, при ошибках пользователя и др.

Для задания модели сценариев используется специальная графическая нотация, со следующими основными правилами:

- актер обозначается иконой человека, под которой указывается название;
- сценарий изображается овалом, в середине которого указывается название иконы;
- актер связывается стрелкой с каждым запускаемым им сценарием.

На рисунке представлен пример диаграммы сценариев для клиента банка, как актера, который запускает заданный сценарий. Для достижения цели актер обращается не к кассиру или клерку банка, а непосредственно к компьютеру системы.



Пример диаграммы сценариев для клиента банка

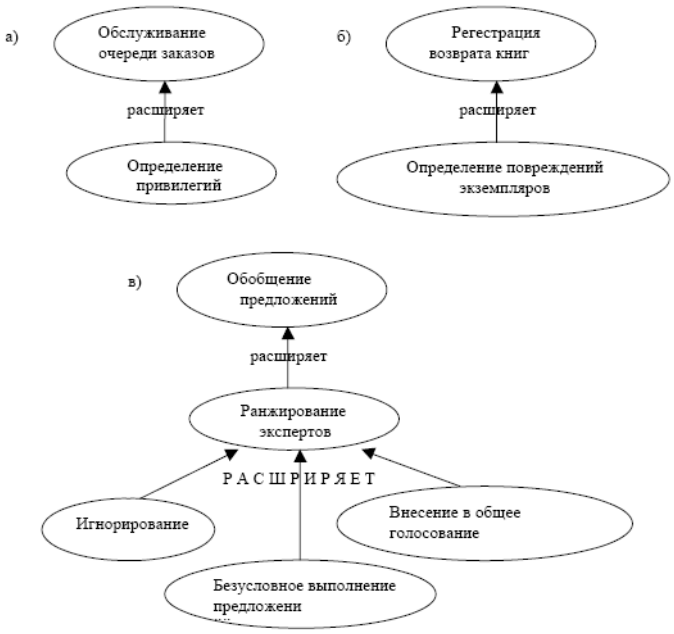
Все сценарии, которые включены в систему, обведены рамкой, определяющей границы системы, а актер находится вне рамки, являясь внешним фактором по отношению к системе.

Отношения между сценариями. Между сценариями можно задавать отношения, которые изображаются на диаграмме сценариев пунктирными стрелками с указанием названия соответствующих отношений.

Для сценариев определены два типа отношений: *Отношение "расширяет"* означает, что функции одного сценария является дополнением к функциям другого, и используется когда имеется несколько вариантов одного и того же сценария. Инвариантная его часть изображается как основной сценарий, а отдельные варианты как расширения. При этом основной сценарий является устойчивым, не меняется при расширении вариантов функций и не зависит от них. Типовой пример отношения расширения: моделирование необязательных фрагментов сценариев приведен на рисунке. При выполнении сценария расширение прерывает выполнение основного расширяемого сценария, причем последний не знает, будет ли расширение и какое. После завершения выполнения сценария расширение будетпродолжено путем выполнения основного сценария.

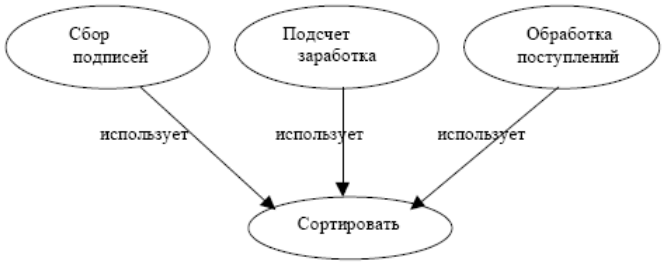
Отношение "использует" означает, что некоторый сценарий может быть использован как расширение несколькими другими сценариями. Оба отношения – инструменты определения наследования, если сценарии считатьобъектами. Отличие между ними состоит в том, что при расширении функция рассматривается как дополнение к основной функции и может быть понятной

только в паре с ней, тогда как в отношении "использует " дополнительная функция имеет самостоятельное определение и может быть использована всюду.



Примеры расширения сценариев

На следующем рисунке показано, что сценарий "сортировать" связан отношением "использует" с несколькими сценариями.



Пример отношения "использует "

Таким образом, продуктом первой стадии инженерии требований (сбора требований) является модель требований, которая состоит с трех частей:

- онтология домена;
- модель сценариев, называемая диаграммой сценариев;
- описание интерфейсов сценариев.

Модель сценариев сопровождается неформальным описанием каждого из сценариев.

Нотация такого описания не регламентируется. Как один из вариантов, описание сценария может быть представлено последовательностью элементов:

- название, которое помечает сценарий на диаграммах модели требований и служит средством ссылки на сценарий;
- аннотация (краткое содержание в неформальном представлении);
- актеры, которые могут запускать сценарий;

- определение всех аспектов взаимодействия системы с актерами (возможные действия актера и их возможные последствия), а также запрещенных действий актера;
- предусловия, которые определяют начальное состояние на момент запуска сценария, необходимое для успешного выполнения;
- функции, которые реализуются при выполнении сценария и определяют порядок,содержание и альтернативу действий, выполняемых в сценарии;
- исключительные или нестандартные ситуации, которые могут появиться при выполнении сценария и потребовать специальных действий для их устранения (например, ошибка в действиях актера, которую способна распознать система);
- реакции на предвиденные нестандартные ситуации;
- условия завершения сценария;
- постусловия, которые определяют конечное состояние сценария при его завершении.

На дальнейших стадиях сценарий трансформируется в сценарий поведения системы, т.е. к приведенным элементам модели добавляются элементы, связанные с конструированием решения проблемы и нефункциональными требованиями:

- механизмы запуска сценария (позиции меню);
- механизмы ввода данных;
- поведение при возникновении чрезвычайных ситуаций.

Следующим шагом процесса проектирования является преобразование сценария в описание компонентов системы и проверка их правильности. Описание интерфейсов делается неформально, они определяют взгляд пользователя на систему, с которым необходимо согласовать требования, собранные на этапе анализа системы и определения требований. Все вопросы взаимодействия отмечаются на панели экрана для каждого из актеров и их элементы (меню, окна ввода, кнопки - индикаторы и т.п.). Построение прототипа системы, моделирующего действия актера, помогает отработать детали и устранить недоразумения между заказчиком и разработчиком.