

## **МДК02.01 Технология разработки программного обеспечения**

### **Введение**

Любая программная система создается для решения одной или нескольких проблем будущих пользователей программной системы. Программа – это ни что иное, как некоторый алгоритм, заложенный в компьютер для решения определенного круга задач, работа которого должна принести пользователю ощутимый результат.

Здесь кроется одна из проблем разработки программного обеспечения (ПО). Программисты и пользователи говорят на разных языках. Программисты знают как писать программы, а пользователи знают, или, по крайней мере, должны знать что должна делать для них программа. Однако пользователи не идеальный источник информации. Большинство пользователей знают, как выполнять свою работу, однако далеки от понятия того, как переложить все это на компьютер и часто не могут изложить свои требования к будущей системе.

Традиционный подход решения этой проблемы – это поручение определения требования аналитикам, которые проводят с пользователями интервью, выявляя их реальные потребности. Но даже аналитикам трудно получить непротиворечивый и в дальнейшем мало изменяемый список требований, если не использовать систематизированный подход к определению требований.

Основная цель рабочего процесса определения требований состоит в том, чтобы направить процесс разработки на получение правильной системы. А правильная система – это система, которая делает то, что необходимо и ничего более. Конечно, нашим программистам трудно делать систему так, чтобы ничего от себя не приложить, и тем более ничего не забыть. Описание требований должно быть достаточно хорошим, для того чтобы между пользователями и разработчиками могло быть достигнуто понимание того, что система должна делать и чего не должна. В противном случае пользователи будут считать, что система может сделать для них все, а программисты не будут понимать, какие функции будущей системы обязательно должны будут включены в первую версию, и без них нельзя обойтись, а какие можно отложить до будущего релиза.

Можно определить следующие шаги рабочего процесса определения требований\*:

- Перечисление возможных требований;
- Осознание контекста системы;
- Определение функциональных требований;
- Определение нефункциональных требований.

### **Перечисление возможных требований**

Первое, что нужно сделать – это начать собирать всевозможные требования и идеи насчет будущей системы. Это будет несистематизированный список, в который должно попасть все, что касается системы и приходит в голову

разработчикам, аналитикам и пользователям. Эти идеи будут кандидатами на реализацию в будущих версиях системы и используются для планирования работ.

Каждое предложение в списке должно иметь короткое название и краткое описание, в чем оно состоит, также для дальнейшей работы необходима дополнительная информация для планирования и последующей реализации требований, в которые могут входить:

- состояние предложения (например, предложено, одобрено, включено в план, утверждено);
- трудоемкость в человеко-часах или стоимость реализации;
- приоритет (например, критический, важный или вспомогательный);
- уровень риска, связанного с реализацией предложения (например, критический, значительный или обычный).

Этот список в ходе работ может уменьшаться, когда требования преобразуются в другие артефакты, например – варианты использования или нефункциональные требования, и увеличиваться, когда выдвигаются новые предложения.

## **Осознание контекста системы**

Для того чтобы верно определить требования разработчики системы должны понимать контекст (часть предметной области) в котором работает система. Существует по крайней мере два подхода к описанию контекста системы:

- Моделирование предметной области;
- Бизнес-моделирование.

### *Модель предметной области*

Модель предметной области описывает важные понятия предметной области и их связи между собой. Нельзя путать модель предметной области с логической и физической моделями системы. Модель предметной области описывает только объекты предметной области, но не показывает, как программная система будет с ними работать. Данная модель также позволяет составить глоссарий системы для лучшего ее понимания пользователями и разработчиками.

### *Бизнес-модель*

Бизнес-модель описывает процессы (существующие или будущие), которые должна поддерживать система. Бизнес-модель можно представить как подмножество модели предметной области. Кроме определения бизнес-объектов, вовлеченных в процесс, эта модель определяет работников, их обязанности, и действия, которые они должны выполнять.

## **Методология и стандарты, регламентирующие работу с требованиями.**

Среди основополагающих нормативных документов в области работы с требованиями можно выделить отечественные и международные

Мы рассмотрим и те и другие

1) Отечественные ГОСТ:

- ГОСТ 34. 602-89. Информационная технология. Тех зад на создание автоматизированной системы

- ГОСТ 19.201-78. Единая система программной документации ТЗ. Требования к содержанию и оформлению

2) Разработки IEEE - Institute of Electrical and Electronics Engineers (общественная организация – международный институт инженеров электроники и энергетики):

- IEEE 1233 «Guide for Developing System Requirements Specifications»

- IEEE Standard 830-1998, “IEEE Recommended Practice for Software Requirements Specifications”

- IEEE Standard Glossary of Software Engineering Terminology / IEEE Std 610.12-1990.

### **Определение требования**

Неформальное определение требования – это условие или возможность, которая должна соответствовать системе.

Формальное определение: В соответствии стандартом глоссария технологий ПИ IEEE требования – это:

- 1) Условия или возможности, необходимые пользователю для решения проблем или достижения целей.
- 2) Условия или возможности, которыми должны обладать система или системные компоненты, чтобы выполнить контракт или удовлетворить стандартам, спецификациям или другим оригинальным документам
- 3) Документированное представление условий или возможностей для пунктов 1 и 2.

Техническое определение т – это исходящие данные на основании которых проектируются и создаются автоматические информационные системы. Первичные данные поступают из различных источников характеризуются противоречивостью неполнотой нечеткостью изменчивостью.

Требования нужны в частности для того чтобы разработчик мог определить и согласовать с Заказчиком временные и финансовые перспективы проекта автоматизации. Поэтому значительная часть требований должна быть собрана и обработана на ранних этапах создания АИС. Однако собрать на ранних стадиях все данные, необходимые для реализации АИС, удастся только в исключительных случаях.

На практике процесс сбора, анализа и обработки растянут во времени на протяжении всего жизненного цикла АИС, зачастую нетривиален и содержит множество подводных камней.

## **Классификация требований.**

Существует значительное количество различных методов классификации требований.

Наиболее существенные:

- требования к продукту и процессу
- уровни требований
- системные требования и требования к программному обеспечению
- функциональные, нефункциональные требования и характеристики продукта
- классификация RUP

## **Требования к продукту и процессу**

Требования к продукту – содержит в своей основе то, что формулирует Заказчик. Цель, которую преследует Заказчик – получить хороший конечный продукт: функциональный и удобный в использовании

Потому требования к продукту являются основополагающим классом требований

Требования к проекту – содержит вопросы формулирования требований к проекту, т е к тому, как Разработчик будет выполнять работы по созданию целевой системы. Заказчик, вступая в договорные отношения с Разработчиком, несет различные риски, главным из которых является риск получить продукт с опозданием, либо ненадлежащего качества.

Основные мероприятия по контролю и снижению риска – регламентация процесса создания ПО и его аудит. Заказчик регламентирует требования к проекту в зависимости от ценности конечного продукта для Заказчика, степени доверия Заказчика к разработчику, суммы подписанного контракта, увязки срока сдачи продукта в эксплуатацию с бизнес-рисками Заказчика и тд.

## **Уровни требований**

Современные ИС – это крупные программные системы, содержащие в себе множество модулей, функциональных интерфейсных элементов, отчетов и тд. Общепринятый прием борьбы со сложностью при моделировании сложных объектов – это абстракция и декомпозиция.

Применительно к анализу требований к программным системам эти принципы приводят к разделению требований по уровням. Уровни требований связаны с одной

стороны , уровнем абстракции системы, с другой – с уровнем управления на предприятии

Выделяют 3 уровня требований:

- 1) Верхний уровень – уровень бизнес-требований. Примеры бизнес-требований : система должна сократить срок обрачиваемости обрабатываемых на предприятии заказов в 3 раза. Бизнес-требования обычно формулируются топ-менеджерами, либо акционерами предприятия.
- 2) Средний уровень – уровень требований пользователей. Пример: система должна представлять диалоговые средства для ввода исчерпывающей информации о заказе, последующей фиксации информации в БД и маршрутизации информации о заказе к сотруднику, отвечающему за его планирование и исполнение. Требования пользователя часто бывают плохо структурированными, дублирующими, противоречивыми. Поэтому для создания системы важен третий уровень, в котором осуществляется формализация требований.
- 3) Нижний уровень – функциональный. Пример по работе с электронным заказом: заказ может быть создан, отредактирован, удален и перемещен с участка на участок.

Существуют объективные противоречия между требованиями различных уровней.

Пример:

- 1) Бизнес-требования полноты информации противоречит требованиям конкретного пользователя системы, которые включают исполнение только той части информации, которая влияет на выполнение его основных функций.

## **Системные требования и требования к ПО**

Системные требования являются подмножеством функциональных требований к ИС. Это наиболее важные, существенные требования, которые относятся в целом к системе и не содержат избыточной детализации.

Различаются:

-общие системные требования к ИС

-требования к ПО, как подмножеству системных требований, направленных исключительно на программные компоненты системы

В практике компьютерной инженерии бытует более узкий подход: под системными требованиями понимаются требования, выдвигаемые прикладной программной системой (в частности - информационной) к среде своего функционирования (системной, аппаратной).

Пример таких требований:

- тактовая частота процессора

-объем памяти

-требования к выбору ОС и т.д.

## **Определение функциональных требований**

Подход к выявлению системных требований основан на использовании вариантов использования системы (Use Cases), которые охватывают как функциональные, так и нефункциональные требования, которые специфичны для конкретного варианта использования.

Для пользователя важно, чтобы система выполняла определенные действия для него, при этом пользователь определенным образом взаимодействует с системой, использует ее для своих целей. Таким образом, если определить все возможные варианты использования системы пользователями или другими внешними процессами, то мы получим функциональные требования к ней.

Однако каждый конкретный пользователь работает с системой по-своему, поэтому для определения действительных вариантов использования системы необходимо досконально знать потребности всех заинтересованных пользователей, провести анализ полученной информации и систематизировать ее в действительные варианты использования системы, т.е. абстрагироваться от конкретных пользователей и исходить от бизнес-задач.

В дополнение к вариантам использования необходимо определить, как должен выглядеть пользовательский интерфейс для реализации того или иного варианта использования. Набросать эскизы, обсудить их с пользователями, создать прототипы и отдать их на тестирование пользователям.

Функциональные требования регламентируют функционирование или поведение системы и отвечают на вопрос «что должна делать система?» в тех или иных ситуациях.

Функциональные требования определяют для Разработчика цели, задачи и сервисы, предоставляемые системой Заказчику.

Функциональные требования записываются обычно при посредстве предписывающих правил: «система должна позволять кладовщику формулировать приходные и расходные накладные». Другим способом являются так называемые варианты использования (user cases).

Это – основной, определяющий вид требований.

Характеристика продукта – это подмножество важнейших логически связанных функциональных требований, которые обеспечивают возможности пользователя и удовлетворяют бизнес-цели.

Роль характеристик проявляется в отрасли маркетинга: не всякий потребитель продукта станет читать его функциональные описания, а набор ключевых характеристик, характеризующих конкурентные преимущества, можно сделать

лаконичным и уместить на одной страничке рекламной листовки, либо напечатать на компакт-диске.

## **Определение нефункциональных требований**

К нефункциональным требованиям относятся такие свойства системы, как ограничения среды и реализации, производительность, зависимость от платформы, расширяемость, надежность и т.д. Под надежностью понимаются такие характеристики, как пригодность, точность, средняя наработка на отказ, число ошибок на тысячу строк программы, число ошибок на класс.

Требования по производительности – это скорость, пропускная способность, время отклика, используемая память. Многие требования, связанные с производительностью должны быть описаны в конкретных вариантах использования, а не в разделе относящейся ко всей системе.

Также можно отметить, что часто нефункциональные требования не могут быть привязаны к конкретному варианту использования и должны быть вынесены в отдельный список дополнительных требований к системе.

Нефункциональные требования, соответственно, регламентируют внутренние и внешние условия или атрибуты функционирования системы.

Основные группы неф. требований:

- внешние интерфейсы (External Interfaces)
- атрибуты качества (Quality Attributes)
- ограничения (Constraints)

Рассмотрим их детальнее

### **1) Внешние интерфейсы**

Среди внешних интерфейсов наиболее важным является интерфейс пользователя UI.

Кроме выделяются:

- интерфейсы с внешними устройствами (аппаратные интерфейсы)
- программные интерфейсы
- интерфейсы передачи информации (коммуникационные интерфейсы)

### **2) Основные атрибуты качества:**

- применимость
- надежность

-производительность

-эксплуатационная пригодность

3) Ограничения – формулировки условий, модифицирующих требования или наборы требований, сужая выбор возможных решений по их реализации.

Например: Выбор платформы реализации и/или развертывания (протоколы, серверы приложений, баз данных ...), которые, в свою очередь, могут относиться, например, к внешним интерфейсам.

## **Классификация RUP**

В спецификациях Rational Unified Process при классификации требований используется модель FUR PS+ со ссылкой на стандарт IEEE Std 610.12.1990.

Акроним FUR PS обозначает следующие категории требований:

-Functionality (функциональность)

-Usability (применимость)

- Reliability(надежность)

-performance(производительность)

-supportability(эксплуатационная пригодность)

Символ «+» расширяет FURPS-модель, добавляет к ней:

-ограничения проекта

-требования выполнения

-требования к интерфейсу

-физические требования

Кроме того, в спецификациях RUP выделяются такие категории требований, как:

- требования, указывающие на необходимость согласованности с некоторыми юридическими и нормативными актами

-требования к лицензированию

-требования к документированию

## **ТЗ ВЫЯВЛЕНИЕ ТРЕБОВАНИЙ**

Рассмотрим основные источники и стратегии выявления требований к ПО.



## **Источники требований**

Источниками, образующими «вход» процесса выявления требований, являются требования, высказанные совладельцами, а также данные, описывающие объект исследования

Основным источником требования информационной системы являются соображения, высказанные представителями Заказчика.

Однако, представители Заказчика могут быть некомпетентны в данном вопросе. Поэтому, наряду с требованиями, высказанными Заказчиком, необходимо собирать и требования от других совладельцев системы: сотрудников аналитической группы исполнителя, внешних экспертов и т.д.

Результатом, часто достаточно сырой материал рассматривается как документ «Требования совладельцев». На требования совладельцев обычно не накладывают никаких специальных ограничений.

Модель создаваемой информационной системы в определенной мере должна отражать модель организации системы (ОС).

Соответственно, другими важными источниками информации, являются артефакты, описывающие предметную область.

Это могут быть документы с описанием бизнес-процессов предприятия, выполненные консалтинговым агентством, либо просто документы (должностные инструкции, распоряжения, своды бизнес-правил), принятые на предприятии.

Одной из немногих методологий, в которой специально выделяется рабочий поток делового моделирования, является RUP.

Также модно использовать так называемые «лучшие практики». Они представляют собой описания моделей деятельности успешных компаний отрасли, используемые длительное время в сотнях и тысячах компаний по всему миру.

## **Стратегия выявления требований**

На практике аналитики требований могут принимать такие стратегии:

- интервью
- анкетирование
- наблюдение
- самостоятельное описание требований
- совместные семинары
- прототипирование

Рассмотрим их детальнее.

## **2.1 Интервью**

Ключевой стратегией выявления требований было и остается интервью с экспертами.

В процессе проведения интервью, как правило, имеются 3 подчиненных процесса:

- подготовка
- проведение интервью (опроса)
- завершение

### **2.1.1 Подготовка**

Подготовка позволяет спланировать процесс опроса и выработать стратегию управления этим процессом. При подготовке рекомендуются следующие шаги:

- выберите нужного собеседника
- договоритесь о встрече
- установите предварительную программу встречи
- изучите соответствующую информацию.

Полезно сформировать программу беседы и знакомить с ней респондента, подробно планировать беседу вплоть до записи подготовленных вопросов.

Подготовленное таким образом интервью называется структурированным.

### **2.1.2 Проведение опроса**

При проведении опроса самое важное:

- правильно организовать и поддерживать поток информации от эксперта к вам
- рекомендуется потратить время на обдумывание верного начала опроса
- не возражайте
- никогда не задавайте наводящих вопросов или вопросов с короткими ответами «да/нет»

Вы узнаете больше если дать эксперту говорить то, что он хочет сказать, а не то, что вы хотите услышать.

## **2.2 Анкетирование**

Анкетирование – самый малозатратный для аналитика способ извлечения информации, он же – и наименее эффективный.

Недостатки анкетирования: респонденты часто неспособны, либо слабо мотивированы хорошо и информативно заполнять анкету.

Рекомендуется формулировать вопросы в одной из таких форм:

1) Многоальтернативные вопросы

2) Рейтинговый набор типов ответов на сформулированные вопросы: «абсолютно согласен», «согласен», «относительно нейтрален», «не согласен», «абсолютно не согласен», «не знаю».

3) Вопросы с ранжированием предусматривают ранжирование (упорядочение) ответов путем присваивания им порядковых номеров, процентных значений.

## **2.3 Наблюдение**

Наблюдение за работой моделируемой организационной системы – полезная стратегия получения информации.

Различают пассивное и активное наблюдение.

При активном наблюдении аналитик работает как участник команды, что позволяет улучшить понимание процессов.

Через наблюдение, а возможно, и участие аналитики получают информацию о происходящем день за днем, получая данные об операциях из первых рук.

Недостаток этой стратегии:

- наблюдатель, как «измерительный прибор», вносит помехи в результат исследования

- сотрудники организации, находящиеся «под колпаком» могут вести себя по другому

## **Самостоятельное описание требований**

Документы – хороший источник информации, потому что они чаще всего доступны и их можно «опрашивать» в удобном для себя темпе.

Чтение документов – прекрасный способ получить первоначальное представление о системе и сформулировать вопросы к экспертам.

Если опытный аналитик уже исследовал большое число систем такого же типа, что и предприятие внедрения, он обладает фундаментальными знаниями относительно определенного класса систем.

По результатам анализа документов и собственных знаний аналитик может составить описания требований и предложить его представителям Заказчика в качестве информации к размышлению, либо - основы для формирования ТЗ.

Недостаток этой стратегии – опасность пропуска знаний, специфичных для объекта исследования (в случае самоопроса), либо – неформализованных знаний, эмпирических правил и процедур, широко используемых на практике, но не вошедших в документы.

### **Совместные семинары**

Помимо классического интервью существует значительное количество методик, предполагающих широкое участие представителей Заказчика и Исполнителя.

Например, мозговой штурм. Правила мозгового штурма предполагают полную раскрепощенность и свободу мнений, даже самых вычурных и на первый взгляд «бредовых». Первое правило мозгового штурма – «полный запрет на любую критику». Всякое высказанное мнение представляет ценность, а полное отсутствие запретов позволяет полноценным образом подключить творческую фантазию.

Затем все высказанные мнения тщательным образом обсуждаются, заведомо неприемлемые варианты отсеиваются, формируются коллективные предложения. Более сложный вариант – это JAD-метод, который был сформулирован в конце 1970-х годов компанией IBM.

## **2.6 Прототипирование**

Прототипирование, т.е. создание эскиза программы, - это ключевая стратегия выявления требований в большинстве современных методологий.

Документный способ выявления требований всегда уступает живому общению.

Анализ того, что сделано в виде интерфейсов пользователя дает еще больший эффект.

### **Видение продукта и границы проекта**

Понятие видения широко употребляется в бизнес-анализе. Если у топ-менеджмента компании имеется представление о том, какие ключевые цели, сегменты рынка, товарные позиции, прибыль должны быть достигнуты, допустим, через 5 лет – значит, компания имеет долгосрочное видение.

Границы проекта (рамки, контекст) обсуждают такие вопросы, как граница системы и среды, требуемые ресурсы на создание системы, сроки и т.д.

Построив «ничем не ограниченное видение», рано или поздно приходится вернуться к таким вещам, как бюджет, календарное планирование, подбор персонала, вехи проекта.

Однако способ снятия ограничений при разработке видения позволяет выработать новый взгляд на вещи, «подняться над ситуацией», планировать будущее, отталкиваясь не от текущих ресурсов и ограничений, а от стратегических целей, применяя инновации, ноу-хау и т.д.

Опыт формирования видения переносим и на процесс разработки ИС: нужно «увидеть» в горизонте средне- и (или) долгосрочного планирования, как АИС впишется в организационные процессы предприятия, какие ключевые выгоды она даст, какие проблемы позволит решить.

При поиске новых методов и средств управления предприятием на основании информационных технологий приходится часто «перекраивать» существующие бизнес-процессы.

Видение АИС, затрагивающей существенный процент процессов предприятия, неизбежно приводит к перестройке этих процессов с целью оптимизации деятельности предприятия, достижения ключевых факторов эффективности и т.д.

Зачастую заказчик осознает необходимость автоматизации как способ решения накопившихся проблем.

Сформулировав для себя проблему, заказчик часто видит и вариант ее решения, с которым приходит к исполнителю.

Квалифицированный исполнитель не должен, сломя голову, спешить решить задачу в формулировке заказчика.

По известному образному выражению: «автоматизировать процессы «как есть» - все равно, что асфальтировать дорожки, по которым ходят коровы».

В нотации RUP присутствует метафора «Увидеть проблему за проблемой».

Видение как раз и служит для того, чтобы помочь заказчику выявить именно те требования к системе, которые помогут ему оптимизировать работу своего предприятия в долгосрочной перспективе.

Этап формирования видения важен, но он предъявляет и к заказчику и к исполнителю свои требования:

1) Заказчик должен выделить ресурсы и быть готовым к трудозатратам на совместный поиск решений

2) Исполнитель должен обладать достаточной квалификацией как в сфере IT, так и в сфере управления предприятием.

## **Видения в RUP**

Шаги, которые необходимо пройти для формирования документа «видения»:

- формулировка проблем
- идентификация совладельцев
- определение границ системы
- идентификация ограничений
- формулировка постановки задач
- определение возможностей системы
- оценка результатов

## **T5 СВОЙСТВА ТРЕБОВАНИЙ**

Свойства требований подвергаются анализу на этапе проверки требований: определение, являются ли собранные требования неясными, неполными, неоднозначными или противоречащими, и затем – решение этих проблем.

Имеются такие свойства требований к программной системе:

- полнота
- ясность
- корректность
- согласованность
- верифицируемость
- необходимость
- полезность при эксплуатации
- осуществимость
- модифицируемость
- трассируемость
- упорядоченность по важности и стабильности

- наличие количественной метрики

Большинство из этих свойств раскрыто в первом разделе стандарта IEEE.

## **Полнота**

Требование полноты нужно рассматривать в 2 аспектах: полнота отдельного требования и полнота системных требований.

Полнота отдельного требования – свойство, означающее, что текст требования не требует дополнительной детализации, то есть в нем предусмотрены все необходимые нюансы, особенности и детали данного требования.

Полнота системных требований – свойство, означающее, что совокупность артефактов, описывающих требования, исчерпывающим образом описывает все то, что требуется от разрабатываемой системы.

## **Ясность (недвусмысленность, определенность, однозначность спецификаций).**

Каждый совладелец разрабатываемой системы обладает личным опытом восприятия событий внешнего мира. То, что является ясным для заказчика, необязательно будет ясным для специалиста в области программной инженерии.

Требования обладают свойством ясности, если оно сходным образом воспринимается всеми совладельцами системы. Ясность требования достигается в процессе консультаций, в ходе которых происходит «выравнивание понятий» совладельцев системы, хорошим подспорьем в этом служит согласованный сторонами глоссарий ключевых понятий предметной области.

## **Корректность и согласованность (непротиворечивость)**

Корректность – точность описания функциональности.

Свойство корректности носит оценочный характер: каждое из требований либо корректно, либо нет. Взаимная корректность требований или согласованность: если 2 требования вступают в конфликт, значит, - как минимум одно из них некорректно.

В иерархии требования имеется вертикальная и горизонтальная согласованность, т.е. требования не должны противоречить соответственно требованиям своего уровня иерархии и требованиям «родительского уровня». Так, требования пользователя не должны противоречить бизнес-требованиям, а функциональные требования – требованиям пользователя.

## **Верифицируемость (пригодность к проверке).**

Все признаки (свойства) требований являются зависимыми. В математической статистике такие признаки называются коррелируемыми.

Свойство верифицируемости существенно связано со свойствами ясности и полноты: если требование изложено на языке, понятном и одинаково

воспринимаемом участником процесса создания информационной системы, причем оно является полным, т.е. ни одна из важных для реализации деталей не упущена – значит, это требование можно проверить.

В ходе проверки у сторон (принимающей и сдающей работу) не должно возникнуть неразрешимых противоречий в оценках, т.к. хорошо сформулированные требования составляют основу успешного создания системы – роль верифицируемости трудно переоценить.

Требования к системе представляют основу контракта между заказчиком и исполнителем.

Если данные требования нельзя проверить – значит, и контракт не имеет никакого смысла, следовательно, успех или неудача проекта будут зависеть только от эмоциональных оценок сторон и их способности договориться, а это – слишком шаткая основа для осуществления работ.

### **Необходимость и полезность**

При эксплуатации являются одним из самых субъективных и трудно проверяемых свойств требований.

Необходимость свойства – свойство, без выполнения которого невозможно, либо затруднительно выполнение автоматизированных бизнес-функций пользователей.

Полезные свойства – свойства повышающие эргономические качества продукта при эксплуатации.

1) Бизнес-требования – наиболее бесспорные. Данные требования формулируют первые лица, представляющие заказчика, определяющие каким условиям должна соответствовать создаваемая ИС, чтобы соответствовать бизнес-целям предприятия.

Однако, если у исполнителя возникают сомнения в необходимости того или иного бизнес-требования, вызванные интуитивными соображениями, либо опытом внедрения ИС на аналогичных предприятиях, он должен проявлять инициативу и собрать совместное совещание сторон.

2) Необходимость требований пользователя может вытекать из соответствующих бизнес-требований.

Кроме того, требования пользователя могут мотивироваться эргономичностью продукта и особенностями функциональности его отдела (подразделения), недостаточно полно раскрытыми на предыдущем уровне иерархии требований.

3) Большинство функциональных требований вытекают из требований первых 2 уровней.

Другие функциональные требования могут лежать вне сферы компетенции заказчика (который вообще говоря, не обязан быть экспертом в области ИТ) и их



должен сформулировать исполнитель. Например, наличие функции архивирования информации.

## **Осуществимость**

Требует исключать абсурдные требования и рассмотрение тех, которые выполнимы принципиально. Однако, не все требования, выполнимые принципиально, являются осуществимыми.

Выполнимость требования на практике определяется разумным балансом между ценностью (степенью необходимости и полезности) и потребными ресурсами.

Иллюстрацией балансировки между ценностью и выполнимостью требований является так называемый треугольник компромиссов.



После достижения равновесия в этом треугольнике изменение на любой из его сторон для поддержания баланса требует модификации на другой (двух других) сторонах и/или на изначально измененной стороне.

Необходимо обеспечить возможность переработки требований, если понадобится, и поддерживать историю изменений для каждого положения.

## **7. Трассируемость.**

Описывается возможностью отследить связь между ним и другими компонентами ИС (документами, моделями, текстами программ и пр.).

Отдельная трасса представляет собой направленное бинарное отношение, заданное на множестве компонентов ИС, где 1-ый элемент отношения представляет соответствующее требование, а 2-ой компонент зависит от данного требования.

На практике трассировка анализируется при посредстве графовых либо табличных моделей. Процесс трассировки позволяет с одной стороны выявить уже на стадии проектирования системы проектные компоненты, к которым не ведет связь от одного из компонентов, описывающих требования, с другой – компоненты, описывающие требования, не связанные с проектными компонентами.

В первом случае надо убедиться в том, что проектный компонент действительно имеет право на существование, а не является избыточным. Во втором случае необходимо проанализировать полезность выявленных требований:

- либо эти требования несут достаточно полезную нагрузку и могут быть игнорированы
- либо имеют место ошибки проектирования: пропущены соответствующие компоненты

Другая цель трассировки – повысить управляемость проектом.

### **Упорядоченность по важности и стабильности**

Приоритет требования представляет собой количественную оценку степени значимости требования. Приоритеты требования обычно назначает представитель заказчика. Разработчик, отталкиваясь от приоритетности требований, управляет процессом реализации ИС.

Стабильность требований характеризует прогнозную оценку неизменности требований во времени.

### **Наличие количественной метрики.**

Количественные метрики играют важную роль в верификации и аттестации ИС. В первую очередь это относится к нефункциональным требованиям, которые, как правило, должны иметь под собой количественную основу. Например:

- запрос должен обрабатывать не более, чем \_ секунд
- средняя наработка на отказ должна составлять не менее, чем \_ часов

Формульные требования также могут расширяться количественными мерами при помощи так называемых аспектов применимости.

### **Каких требований не должно быть.**

Согласно установившемуся подходу, спецификация требований не должна содержать деталей проектирования или реализации (кроме известных ограничений).

Требования должны отвечать на вопрос: «что должна делать система?», и не касаться вопроса «как она должна это делать?».

Стремление принимать детальные проектные решения на этапе анализа требований – одна из «ловушек», типичных для неопытных команд разработчиков.

### **ТЕМА6: Классификация и специфицирование требований**

Результат анализа предметной области, как правило, завершается построением глоссария, то есть списком терминов предметной области. Повысить уровень

информативности требований к ПО возможно с помощью оформления их в виде вариантов использования.

Прежде, чем приступить к специфицированию требований в форме вариантов использования RUP рекомендует выявить реестр актеров и вариантов использования.

## **Глоссарий**

Помимо формирования требований совладельцев другим результатом фазы выявления требований является концептуальный анализ предметной области.

Самым первым его результатом является формирование глоссария (словаря) основных используемых терминов.

Значение глоссария трудно переоценить: он является основой, ключом для единообразного понимания требований заказчиком и разработчиком.

Кроме того, глоссарий является отправной точкой для построения более развернутых моделей проблемной области, которые на стадии реализации информационной системы ложатся в основу объектной модели (для объектно-ориентированного приложения) и модели данных (для генерации схемы БД).

Глоссарий оформляется как текст, состоящий из абзацев, каждый из которых определяет значение одного из терминов проблемной области.

Термин обычно выделяют полужирным кеглем. Иногда проблемную область целесообразно сегментировать на ряд подобластей. Тогда каждой из них выделяется отдельный параграф.

## **2. Актеры и варианты использования.**

Результатом выявления требований является реестр требований.

Требования совладельцев обычно оформляют в простой письменной форме, без какой-либо особой регламентации. Типовой пример оформления требования к программе электронной почты: Система должна позволять набирать текст сообщения с возможностью форматирования текста и вставки картинок.

Данные требования далеко не во всем могут удовлетворять критериям, сформированным ранее; они могут противоречить друг другу, быть неясными, неточными.

Тем не менее, документ «Требования совладельцев», несмотря на невысокий уровень формализации, играет очень важную роль: содержит мнения всех заинтересованных сторон, а также как можно более полный набор требований.

Для повышения уровня информативности требований необходимо устранить взаимные противоречия и добиться выполнения их других основных характеристик, осуществляется:

- 1) переход от полностью неформализованных текстов к частично регламентированным текстам;
- 2) классификация;
- 3) присвоение наборов атрибутов;
- 4) построение моделей;
- 5) прототипирование.

Самым популярным и крайне эффективным способом повышения информативности требований является оформление их в виде вариантов использования.

Прежде чем приступить непосредственно к специфицированию требований в форме вариантов использования, RUP рекомендует выявить реестр актеров и вариантов использования.

Актер – некто/нечто, обладающее активностью относительно системы.

Поиск актеров корпоративной информационной системы обычно сводится к анализу ролей всех пользователей.

### **3. Спецификация вариантов использования.**

Существуют различные способы описания вариантов использования.

- свободный формат;
- полный формат;
- таблица в 3 колонки;
- таблица в 2 колонки;
- стиль RUP.

Кроме того, иногда целесообразно использовать:

- диаграммы активности UML;
- диаграммы графической модели.

#### **Свободный формат**

Свободный формат предполагает описаний действий системы и пользователя в повествовательном стиле.

Пример: Менеджер запросил у системы список заказов за период. Системы отображает на экран найденные заказы данного менеджера с указанием их основных атрибутов.

Свободный стиль допускает использование конструкций «Если, то».

### **3.2 Шаблон полного описания вариантов использования.**

Название /краткая фраза в виде глагола в неопределенной форме совершенного вида, отражающая цель/

Контекст использования /уточнение цели, при необходимости – условия её нормального завершения/

Область действия /ссылка на рамки проекта/ (Пример: подсистема бухгалтерского учета)

Уровень /один из 3: обобщенный, цели пользователя, подфункции/ (Автор задает предопределенную трехуровневую классификацию требований, в целом соответствующую классификации требований на бизнес-требования, требования пользователя и функциональные требования.)

Участники и интересы /описание других актеров-участников прецедента с указанием их интересов/

Предусловие /то, что ожидается, уже произошло/

Минимальные гарантии /что точно гарантируется актерам-участникам/ (Например, в случае транзакции все данные, имевшиеся в системе до её начала, сохраняются неизменными/

Гарантии успеха /то, что получают актеры в случае успешного достижения целей/ (Пример: то, что запускает вариант использования, обычно – событие во времени.)

Основной сценарий /тут перечисляются шаги основного сценария, начиная от триггера и вплоть до достижения гарантии успеха/ (Формат описания: /№шага/ /описание действия/)

Расширения /тут последовательно описываются все альтернативные сценарии, каждая из которых привязана к шагу основного сценария/ (Формат описания: /№шага/ /№расширения/

/условие/: /действие или ссылка на подчиненный вариант использования/

Любой из шагов основного сценария может иметь более одного ветвления.

Каждое ветвление оформляется в виде расширения. В блоке «Расширения» все расширения описываются последовательностью. В случае, если альтернативный сценарий не удастся описать одной строкой, применяется следующий формат: начиная со строки, следующей после описания расширения, идет описание его действий в формате основного сценария:

/№шага/ /№расширения/ /№шага расширения/ /действие/

Описание расширения заканчивается описанием выхода из расширения. Основные варианты выхода из расширения:

- возврат к очередному шагу основного сценария;
- переход к другому шагу основного сценария)

Вспомогательная информация /дополнительная информация, полезная при описании варианта использования/

### **3.3 Табличные представления варианта использования.**

Иногда представляется удобным помещать сценарии вариантов использования в таблицу, как это показано ниже.

Преимущества: информация становится структурированной.

Актер	Действие
Пользователь	Формирует запрос на поиск заказов
Система	Отображает список заказов
Пользователь	Выбирает требуемый заказ
Система	Показывает подробную информацию о заказе

Второй вариант:

№ шага	Пользователь	Система
	Делает запрос на поиск заказов	Отображает список заказов
	Выбирает заказ	Показ. подробную инф. о заказе

### **3.4 Шаблон варианта использования RUP.**

Приведем краткий обзор разделов шаблона описания варианта использования с помощью RUP.

1. Наименование и краткое описание. В этом разделе указывается: наименование варианта использования, актеры, краткое описание.

2. Поток событий.

2.1 Основной поток событий – аналогично основному сценарию из 3.2

2.2 Альтернативные потоки событий. Каждый из альтернативных сценариев описывается в отдельном параграфе, в том же стиле, что и основной поток событий. Альтернативные сценарии описывают поведение системы при любом отклонении от основного сценария.

3. Специальные требования – перечисление нефункциональных требований, имеющих непосредственное отношение к данному варианту использования.

4. Предусловие – состояние, в котором должна находиться система до начала прецедента.

(События, описывающие предусловия или постусловия, должны быть состояниями, которые пользователь может наблюдать.)

5. Постусловия – по сути описывают то же, что и минимальная гарантия в пункте 3.2. Корректно сформулированное постусловие должно быть истинным при любом возможном сценарии прецедента.

6. Точки расширения – положение точек, расширяющих поток событий.

### **3.5 Выбор формы описания варианта использования.**

При выборе формы и степени подробности описания варианта использования следует учитывать такие факторы, как:

- размеры проекта;
- важность проекта и варианта использования;
- традиции, сложившиеся в коллективе.

Для небольшого проекта вряд ли будет целесообразным применение описания вариантов использования в развернутом формате, достаточно использовать краткую форму свободного стиля.

Для проекта, в котором задействовано более 10 участников, когда возникают проблемы разбиения на микроколлективы, координации участников, следует выбрать более формализованный и подробный документ.

Степень подробности зависит также от критичности проекта в целом и критичности варианта использования в данном проекте.

А. Коберн делит программные проекты по степени критичности на категории, исходя из цены ошибки – проекты, ошибки в которых могут привести к...:

- опасности для жизни;
- невосполнимым финансовым потерям;
- финансовым потерям в ограниченном объеме;

- снижению комфортности конечного пользователя.

Очевидно, что военные системы, или системы управления сложными техническими объектами требуют более скрупулезного документирования, в том числе и на уровне вариантов использования.

Кроме того, в одном и том же проекте может встречаться разные по важности прецеденты с позиции:

- частоты и массовости использования, технических рисков;
- сложности для понимания, ...

В этом случае для разных прецедентов одного и того же проекта вполне допускаются описания с разной степенью подробности.

Наконец, спецификация вариантов использования в стиле Коберна, стиле RUP, табличном стиле, с использованием псевдокодов или графических конструкций во многом определяется субъективно выбором автора прецедентов и сложившимся опытом работы с заказчиком проекта.

## **Описание нефункциональных требований**

Описание нефункциональных требований обычно осуществляется в форме, близкой к свободному формату описания вариантов использования. Рекомендуется концентрировать нефункциональные требования в документе, описывающем варианты использования, во всех случаях, когда это возможно.

В случае, если нефункциональные требования носят общий характер и не могут быть привязаны к конкретному прецеденту, они выносятся в документ «дополнительная спецификация».

### **Атрибуты требований.**

Описание требований должны быть операбельными.

Для этого все требования должны учитываться в той или иной учетной системе, будь то:

- таблица в Excel;
- спецификация БД;
- интегрированная среда управления изменениями.

При регистрации требования оно проходит классификацию в соответствии с определенной системой признаков. Основные признаки (атрибуты) требований были рассмотрены ранее.



Кроме того, для оперативного управления требованиями бывает полезно назначить им такие свойства, как проект, ответственное лицо, статус, риск, степень законченности и т.д.. В RUP для управления атрибутами требований предусмотрен артефакт «Атрибуты требований».

Артефакт «Атрибуты требований», предлагаемый RUP, представляет собой репозиторий текстов требований, их атрибутов и трассируемости.

Атрибуты требований представляются матрицей обратных??? требований для каждого типа требований.

Для каждого требования указана функция его соответствия атрибутов.

Примеры: статус во времени, приоритет, важность, риск...

## **ТЕМА7: Расширенный анализ требований. Моделирование**

### **Цели моделирования**

Вербальные описания вариантов использования системы, рассмотренные ранее, являются стандартами для формулировки соглашения между заказчиком и исполнителем.

Чтобы облегчить процесс формулировки и понимания требований для заказчика существует ряд приемов.

- 1) Требования можно формулировать на разных уровнях абстракции
- 2) Применение визуальных средств описания требований (моделирование)

Имеются 3 рекомендации по целям моделирования:

- анализ требований призван изучать взаимоотношения АИС и ее среды, т.е. пользователей, сетевых и системных компонент, находящихся вне системы.
- анализ требований должен находить ответ на то, что делает система, абстрагируясь от деталей реализации, т.е. как она это делает
- анализ требований должен позволять добавляться целевой функции снижения рисков непонимания между исполнителем и заказчиком и размытия границ

### **Модели UML**

Рассмотрим пример различных типов отношений

## **2.1 Диаграмма вариантов использования**

одно из самых простых представлений системы. Ее базовые "строительные элементы" - акторы и варианты использования. Диаграмма задумана так, чтобы дать наиболее общее представление о функциональности системы (ее компоненты), не вдаваясь в детали взаимосвязей функций. Поэтому основной вид отношения, используемый в диаграмме - **ассоциация** между актором и вариантом использования.

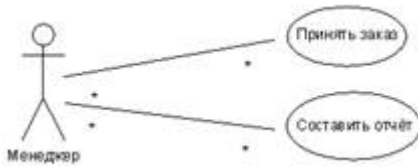


Рисунок - Отношение ассоциации

Другие виды отношений - отношение включения (include), расширения (extend) и обобщения/генерализации.

**Включение** служит для обозначения подчиненных вариантов использования (когда один или более вариантов использования содержат вызовы одной и той же функциональности).



Рисунок - Отношение включения

**Расширение** в точности соответствует точке расширения, используемой при описании варианта использования



Рисунок - Отношение расширения

**Отношение обобщения** может применяться как к акторам, так и к вариантам использования, с целью указания специализации одних относительно других.



## Рисунок – отношения обобщения

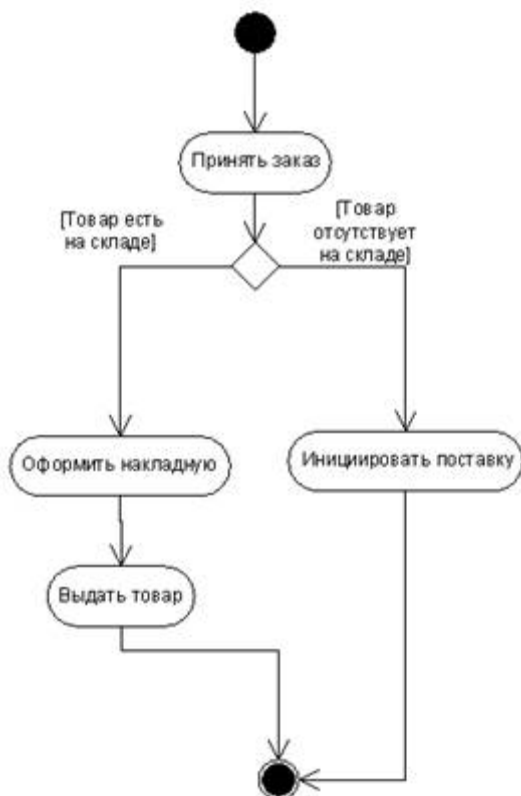
### Диаграмма действий

Если диаграмма вариантов использования дает «вид сверху» на функциональность системы, диаграмма действий UML напротив позволяет подробно иллюстрировать отдельный вариант использования и его сценарии.

Основные компоненты описания системы:

- Функции (действия),
- Символы "старт" и "стоп",
- Потоки управления,
- Разветвители,
- Линейки синхронизации.

Диаграмма действий позволяет проиллюстрировать вариант использования с требуемой степенью подробности. Линейный вариант использования приводит к диаграмме действий с линейным потоком управления между действиями. Действия варианта использования с альтернативными сценариями реализуется через разветвители. Линейки синхронизации позволяют описывать такие сложные конструкции, как синхронизацию начала (окончания) параллельных во времени процессов.



### Диаграмма состояний

В общем случае описывает как система себя ведет в более чем 1 варианте использования. Переход системы из состояния в состояние осуществляется при наступлении событий.

Основные компоненты описания системы:

- Простые состояния,
- Составные состояния,
- Символы "старт" и "стоп",
- Переходы,
- Линейки синхронизации.

Переход системы из состояния в состояние осуществляется при наступлении событий. При этом говорится, что переход срабатывает. Переход может быть безальтернативным, либо содержать альтернативы. Во втором случае переход обусловлен наступлением сторожевых условий. Наконец, событие может сопровождаться выражением действия, которое происходит в случае, если срабатывает переход.



## Диаграмма классов






Для создания необходимо:

- осуществить поиск классов (ключевые компоненты проблемной области)
- для каждого класса определить его имя и основные атрибуты, операции и/или ответственности

- исследовать отношения найденных классов

Классы на диаграмме представляются в виде прямоугольников, отношения - в виде линий, связывающих прямоугольники. Линии разного типа графически отличаются различной штриховкой и стрелками.

Отношения, подлежащие анализу на концептуальном уровне - это:

	ассоциация (именованная связь),
	зависимость (изменения в одном классе приводят к изменениям в другом),
	обобщение / генерализация (родовидовое отношение),
	агрегация (отношение «часть-целое»),
	композиция (отношение «часть-целое», однозначно регламентирующее количество и состав частей целого).

## Прототипирование

Прототипы позволяют увидеть фрагменты реальной системы.

Рассмотрим основные причины для применения прототипов:

- прояснение неясных требований к системе;
- выбор одного из концептуальных решений;
- выполнить анализ осуществимости.

## Неясные требования

Часто заказчику бывает трудно сформулировать требования к тому, что он ожидает от системы. В этом случае прототип интерфейса пользователя дает ему возможность увидеть схематичную реализацию того, как исполнитель видит соответствующую часть системы.

## 2) Разные варианты решения.

Любую техническую задачу можно решить различными способами. Это касается как формулировки требований, так и реализации их в пользовательском интерфейсе.

После реализации прототипов пользовательского интерфейса по различным сценариям заказчик, оценив их достоинства и недостатки, сможет в диалоге с разработчиком сформулировать комбинированный сценарий, сочетающий в себе самое лучшее от предлагаемых.

## Анализ осуществимости.

Часто комбинация функциональных, нефункциональных требований и ограничений такова, что возникает риск невозможности их реализации.

### **Классификация прототипов.**

Рассмотрим классификацию прототипов:

- горизонтальные и вертикальные;
- одноразовые и эволюционирующие;
- бумажные и электронные.

2.1) Горизонтальный прототип – или поведенческий прототип, - моделирует интерфейс пользовательского приложения, не затрагивая логику обработки и БД.

Следует использовать для достижения цели прояснения неясных либо имеющих много альтернатив требований.

2.2) Вертикальный прототип – или структурный прототип, - не ограничивается интерфейсом пользователя, он реализует вертикальный «срез» системы, затрагивая все уровни реализации. При создании такого рода прототипов рекомендуется использовать те языки и среды реализации, что и при изготовлении целевой системы.

Основная цель – анализ применимости, проверка архитектуры концепции.

2.3) Одноразовый прототип – или исследовательский, - предназначен для создания быстрого прототипа с использованием технологии RAD.

Ключевым моментом является скорость создания.

2.4) Эволюционирующий прототип – создается как первое приближение системы, призванное стать системой.

Код прототипа перерастает в код приложения.

2.5) Бумажный – альтернатива при ограничении разработчика в ресурсах. Наброски интерфейса на бумаге позволяют быстро создать прототип.

2.6) Раскадровка – промежуточный вариант между электронным и бумажным вариантами. Представляет собой электронную презентацию.