Министерство Образования, Культуры и Исследований Республики Молдова
Технический Университет Молдовы
Факультет вычислительной техники, информатики и микроэлектроники
Департамент Программной Инженерии и Автоматики

Конспект лекций

по дисциплине: «Анализ и спецификация требований программного обеспечения»

Кишинёв 2020

Содержание

1 Лекция № 1

ТЕМА 1: Введение в проблему разработки требований. Информационная система. Характеристика современных информационных систем.

Определение ИС

Далее по тексту **информационной системой (ИС)** [1], либо автоматизированной ИС, **АИС**, будем называть программно-аппаратную систему, предназначенную для автоматизации целенаправленной деятельности конечных пользователей, обеспечивающую, в соответствии с заложенной в нее логикой обработки, возможность получения, модификации и хранения информации.

Ключевым моментом в этом определении является понятие "целенаправленной деятельности". Речь идет о деятельности, направленной на решение конкретной задачи, стоящей перед пользователем (коллективом пользователей).

Рассмотрим **примеры** некоторых программных средств, являющихся, либо не являющихся ИС.

- 1С-Бухгалтерия 8.0. Используется в целях формирования *бухгалтерской отчетности* предприятия перед налоговыми органами. Является информационной системой.
- MS Excel. Программное средство универсального характера, предназначенное для манипуляций с данными, представленными в табличной форме автоматизации расчетов, формирования разнообразных диаграмм для анализа данных. Не является информационной системой.
- Книга MS Excel, содержащая сведения о штатном расписании, работниках предприятия и оснащенная макросами, позволяющими рассчитывать заработную плату и формировать *платежные ведомости*. Является информационной системой.
- Система Axapta Retail комплексной автоматизации деятельности сети розничных магазинов. Является информационной системой.
- Реляционная база данных DB-2 фирмы IBM. Не является информационной системой.

Классификация ИС

Информационные системы могут быть классифицированы по различным признакам:

- по масштабу;
- по архитектуре;
- по характеру использования информации;
- по системе представления данных;
- по поддерживаемым стандартам управления и технологиям коммуникации;
- по степени автоматизации;
- и другие.

Рассмотрим наиболее важные из них.

Классификация по масштабу

По масштабу ИС будем подразделять на однопользовательские, групповые и корпоративные.

Однопользовательские ИС, как это ясно из названия, предназначены для использования на одном рабочем месте. В настоящее время на мировом и отечественном рынке представлено множество решений, предназначенных для автоматизации деятельности отдельно взятого пользователя. Как правило, это - решения, ориентированные на специалиста в той или иной области, будь то составление спецификаций для сборки изделий из комплектующих, планирование ремонтов оборудования, учет расходов и доходов частного предпринимателя оптовой торговли, либо составление расписаний занятий в деканате.

В настоящее время альтернативу таким узкоспециализированным системам составили табличные процессоры, не имеющие проблемной специализации, в первую очередь - MS Excel. Системы этого класса трудно отнести к классу ИС, но зачастую они позволяют непрограммирующему специалисту создать и, что очень важно, самостоятельно развивать собственные решения, заменяющие, а местами и перекрывающие функционал однопользовательских систем образца 90-х годов.

В основе большинства однопользовательских систем лежит стандарт X-Base (*Clipper*, FoxPro, dBase). Широко используются также решения на базе систем Paradox, Clarion, *MS Access*. Каждая из перечисленных конкурирующих систем обладает собственной высокоуровневой инструментальной средой, позволяющей спроектировать базу данных, логику обработки, пользовательский интерфейс, отчеты с помощью "помощников"-построителей. На рубеже тысячелетий появились также и однопользовательские решения на базе промышленных реляционных СУБД. В этом случае ПО сервера инсталлируется непосредственно на рабочую

станцию пользователя. Примером может служить Personal Oracle. Данные решения предъявляют значительные требования к ресурсам рабочей станции, однако несут в себе многие преимущества промышленных СУБД.

X-Base - семейство систем программирования, берущих начало с dBase (1980 г.). Были предназначены для разработки баз данных в архитектуре файл-сервер под управлением DOS, без поддержки ссылочной целостности. Типопредставители – Foxpro, Clipper, Clarion. Их объединяет общий язык программирования, с вариациями, присущими конкретной реализации и встроенные в этот язык средства доступа к базам данных формата DBF. Формат используется по сей день, как средство для импорта-экспорта данных, например в среде программирования 1С. Существуют современные коммерческие версии, созданные на базе идеологии X-Base, но с поддержкой современных технологий построения программного обеспечения, например X-Base++ (гиперссылка на http://www.alaska-software.com/).

Групповые системы предназначены для автоматизации деятельности в рабочей группе (отделе, кластере, группе проекта и т.д.). В отличие от однопользовательских ИС, групповые системы, как правило, представляют специализированные клиентские решения (их часто называют автоматизированными рабочими местами, АРМ) для различных участников группы. Например, для оптовой фирмы, ИС может представлять набор таких АРМ, как "Менеджер по продажам", "Кладовщик", "Снабженец", "Директор". Для учебного планирования - "Преподаватель", "Работник бюро планирования", "Работник учебного отдела", "Специалист по планированию на кафедре", "Работник деканата".

Групповое использование решений на базе табличных процессоров возможно, но имеет существенные ограничения, связанные с разграничением доступа, регламентацией и синхронизацией вносимых изменений. По сути, единственный режим их использования, обеспечивающий корректность данных - "файловый сервер, один автор, N читателей".

При создании групповых ИС в целом используются те же средства и инструментальные среды, что и при создании однопользовательских ИС. Следует, однако, отметить, что для использования в группе при выборе между системами с файловым и реляционным сервером следует отдавать предпочтение реляционному серверу, причем целесообразно использование выделенного сервера. Это может быть, например, сервер Oracle, DB2, MS SQL, *Sybase*, *Informix*.

Корпоративные ИС (КИС) предназначены для автоматизации деятельности предприятия. В англоязычной литературе понятие "КИС" неразрывно связано с понятием "ERP" (Enterprise Resource Planning). В основе ERP-систем лежит международный стандарт управления

предприятием *MRP*-II (Manufacture Resource Planning), обеспечивающий возможность учета, анализа и планирования основных ресурсов - финансов, человеческих, материальных. Соответственно, корпоративные ERP-системы - набор интегрированных приложений, которые комплексно, в едином информационном пространстве поддерживают все основные аспекты управленческой деятельности предприятий: планирование ресурсов (финансовых, человеческих, материальных) для производства товаров (услуг), оперативное управление выполнением планов (включая снабжение, сбыт, ведение договоров), все виды учета и анализ результатов хозяйственной деятельности.

ERP (Enterprise Resource Planning) – планирование ресурсов предприятия. Согласно словарю IT-терминологии Гарднер, http://blogs.gartner.com/it-glossary, ERP – это организационная стратегия интеграции производства и операций, управления трудовыми ресурсами, финансового менеджмента и управления активами, ориентированная на непрерывную балансировку и оптимизацию ресурсов предприятия посредством специализированного интегрированного пакета прикладного программного обеспечения, обеспечивающего общую модель данных и процессов для всех сфер деятельности. ERP-система — конкретный программный пакет, реализующий стратегию ERP.

Среди требований, предъявляемых к современным КИС:

- централизация данных в единой базе (в основе - всегда промышленная СУБД),
- близкий к реальному времени режим работы,
- сохранение общей модели управления для предприятий разных отраслей,
- поддержка территориально-распределенных структур,
- работа на широком круге аппаратно-программных платформ и СУБД.

Примеры ERP-систем - SAP R3, "Галактика", MS Navision Axapta.

Классификация по архитектуре

Рассмотрим классификацию ИС по архитектуре:

1. **Архитектура "Файл-сервер".** Исторически первая архитектура информационных систем. Как исполняемые модули, так и данные размещаются в отдельных файлах операционной системы. Доступ к данным осуществляется путем указания пути (path) и использования файловых операций (открыть, считать, записать). Для хранения данных используется выделенный сервер (отдельный компьютер), который и является файловым сервером. Исполняемые модули хранятся либо на рабочих станциях, либо на файловом

сервере. В последнем случае упрощается процедура их администрирования, но при этом возрастают требования к надежности сети.

2. **Архитектура "Клиент-сервер"**. Клиент-сервер — это не только архитектура, это - новая парадигма, пришедшая на смену устаревшим концепциям. Суть ее заключается в том, что клиент (исполняемый модуль) запрашивает те или иные сервисы в соответствии с определенным протоколом обмена данными. При этом, в отличие от ситуации с файловым сервером, нет необходимости в использовании прямых путей операционной системы: клиент их "не знает", ему "известны" лишь имя источника данных и другие специальные сведения, используемые для авторизации клиента на сервере. Сервер, который физически может находиться на том же компьютере, а может - на другом конце земного шара, обрабатывает запрос клиента и, произведя соответствующие манипуляции с данными, передает клиенту запрашиваемую порцию данных.

   В рамках направления "клиент-сервер" существуют два основных "диалекта": "тонкий" и "толстый" клиент.

   В системах на основе тонкого клиента используется мощный сервер баз данных, это - высокопроизводительный компьютер и библиотека так называемых хранимых процедур, позволяющих производить вычисления, реализующие основную логику обработки данных, непосредственно на сервере. Клиентское приложение, соответственно, предъявляет невысокие требования к аппаратному обеспечению рабочей станции. Основное достоинство таких систем - относительная дешевизна клиентских станций.

   Системы с толстым клиентом, напротив, реализуют основную логику обработки на клиенте, а сервер представляет собой в чистом виде сервер баз данных, обеспечивающий исполнение только стандартизованных запросов на манипуляцию с данными (как правило - чтение, запись, модификацию данных в таблицах реляционной базы данных). В системах такого класса требования к рабочей станции выше, а к серверу - ниже. Достоинство архитектуры - переносимость серверной компоненты на серверы различных производителей: все промышленные серверы баз данных реляционного типа поддерживают работу со стандартизованным языком манипулирования данными SQL, но внутренний встроенный язык обработки данных, необходимый для реализации логики обработки, на сервере у каждого из серверов свой.

3. **Трехслойная архитектура**. Базируется на дальнейшей специализации компонент архитектуры: клиент занимается только организацией интерфейса с пользователем,

сервер баз данных - только стандартизованной обработкой данных. Для реализации логики обработки данных архитектура предусматривает отдельный слой - слой бизнес-логики. Этот слой может представлять собой либо выделенный сервер (сервер приложений), либо размещаться на клиенте в качестве динамической библиотеки. Данная архитектура позволила соединить достоинства тонкого и толстого клиентов: хорошая переносимость соединяется в ней с невысокими требованиями к клиенту.

С развитием интранет-интернет технологий появилась разновидность трехслойной архитектуры на основании использования web-технологий. В этой разновидности роль сервера приложений играет web-сервер, а в качестве клиента используется стандартный web-браузер. Достоинства - в пониженных требованиях к клиенту и в легкой встраиваемости данной архитектуры в мировые информационные сети. Основной недостаток - известные ограничения, накладываемые на интерфейс пользователя web-браузерами.

Классификация по характеру использования информации

С некоторой степенью приближения все ИС можно разделить на 2 класса:

- информационно-поисковые;
- управляющие.

Конечные пользователи **информационно-поисковых систем** (ИПС), как правило, имеют доступ к хранимым данным только "по чтению" и используют данные системы для поиска ответов на те или иные вопросы. Доступ по модификации данных имеет администратор системы, в функции которого входит обеспечение актуальности информации, устранение ошибок.

Классические примеры ИПС - системы поиска в библиотеках, на транспорте (справки о наличии билетов). На современном этапе развития информационных технологий классические ИПС постепенно вытесняются поисковыми серверами Интернет - общего назначения и специализированными.

Альтернатива ИПС - **управляющие системы** автоматизируют (полностью или частично) деятельность, связанную с принятием решений. Действия конечных пользователей таких систем приводят к модификации информации, что, конечно, не исключает возможности просто получать информацию, как в ИПС.

Примеры управляющих систем - системы бухгалтерского учета, системы планирования производственных ресурсов и т.п.

Классификация по системе представления данных

Среди наиболее распространенных средств и моделей представления данных следует выделить:

- "самодельные" форматы хранения данных, хранящихся в файлах (текстовых, бинарных);
- специализированные форматы хранения данных, использовавшиеся в "дореляционный" период (например, x-Base, paradox);
- языки структурированной разметки на основе формата xml;
- реляционная модель; SQL сервер;
- объектная, объектно-реляционная модель;
- документоориентированное хранилище (IBM Lotus/Domino).

Классификация по поддерживаемым стандартам управления и технологиям коммуникации

Эпоха стихийной разработки АИС закончена. Современные автоматизированные информационные системы разрабатываются, исходя из сложившихся реалий автоматизированного управления бизнесом. Существует значительное количество концепций, технологий, подходов, нашедших свое эффективное применение в различных отраслях промышленности по всему миру. Некоторые из них приобрели статус международных стандартов. В спецификации АИС, разрабатываемой для массовой продажи, как правило, указывается - какие стандарты и технологии управления она поддерживает. Менее строги требования к АИС, создаваемым под заказ для конкретного предприятия. Однако и в этом случае не учитывать сложившийся в мире позитивный опыт просто неразумно. Ниже перечислены некоторые, наиболее важные, технологии и стандарты.

MRP (Material Requirements Planning) - планирование поставок материалов, исходя из данных о комплектации производимой продукции и плана продаж.

CRP (Capacity Requirements Planning) - планирование производственных мощностей, исходя из данных о технологии производимой продукции и прогноза спроса.

MRPII (Manufacture Resource Planning) - планирование материальных, мощностных и финансовых ресурсов, необходимых для производства. Стандартизовано ISO.

ERP (Enterprise Resource Planning) - финансово-ориентированное планирование ресурсов предприятия, необходимых для получения, изготовления, отгрузки и учета заказов потребителей на основе интеграции всех отделов и подразделений компании.

SCM (Supply Chain Management) - управление цепочками поставок. Реализация бизнес-процессов на базе внешних предприятий и торговых площадок Основано на референтной модели SCOR, стандартизованной Supply Chain Council.

CRM (Customer Relationship Management) - управление взаимоотношениями с заказчиками. Комплекс методов и средств, нацеленный на завоевание, удовлетворение требований и сохранение платежеспособных клиентов.

ERPII (Enterprise Resource & Relationship Processing) - управление ресурсами и взаимоотношениями предприятия. Объединяет в себе 3 вышеперечисленные технологии.

Workflow - технология, управляющая потоком работ при помощи программного обеспечения, способного интерпретировать описание процесса, взаимодействовать с его участниками и при необходимости вызывать соответствующие программные приложения.

Workflow - поток работ. Согласно определению Международной коалиции Worflow (http://www.wfmc.org/), поток работ – упорядоченное во времени множество заданий, которые получают сотрудники и которые обрабатываются ими вручную или с помощью средств автоматизации, но с той последовательностью и в рамках тех правил, которые определены для данного бизнес-процесса".

OLAP (Online Analytical Processing) - оперативный анализ данных. Технология поддержки принятия управленческих решений на основе концепции многомерных кубов информации.

Project Management - управление проектами. Поддерживается рядом международных стандартов.

CALS (Continuous Acquisition and Lifecycle Support) - непрерывная информационная поддержка поставок и жизненного цикла. Описывает совокупность принципов и технологий информационной поддержки жизненного цикла продукции на всех его стадиях. Объединяет в себе практически все вышеперечисленные подходы и технологии.

Столь лаконичные определения, конечно же, позволяют лишь ознакомиться с современной терминологией. Задача их детального изучения не входит в план занятий.

Классификация по степени автоматизации

Ручные ИС характеризуются отсутствием современных технических средств переработки информации и выполнением всех операций человеком. Например, о деятельности менеджера в фирме, где отсутствуют компьютеры, можно говорить, что он работает с ручной ИС.

Автоматические ИС выполняют все операции по переработке информации без участия человека.

Автоматизированные ИС предполагают участие в процессе обработки информации и человека, и технических средств, причем главная роль отводится компьютеру. В современном толковании в термин "информационная система", как правило, вкладывается понятие автоматизированной системы.

Основные характеристики информационных систем:

Требования.

- условия или возможности, необходимые пользователю для решения проблем или достижения целей;
- условия или возможности, которыми должна обладать система или системные компоненты, чтобы выполнить контракт или удовлетворить стандартам, спецификациям или другим формальным документам;
- документированное представление условий и возможностей для п1 и п2.

Процесс разработки программного обеспечения - структура, согласно которой построена разработка программного обеспечения (ПО). Существует несколько моделей такого процесса, каждая из которых описывает свой подход, в виде задач и/или деятельности, которые имеют место в ходе процесса.

Шаги процесса

Процесс разработки состоит из множества подпроцессов, или дисциплин, некоторые из которых показаны ниже. В модели водопада они идут одна за другой, в других аналогичных процессах их порядок или состав изменяется.

- Бизнес-моделирование.
- Анализ требований.
- Планирование.

- Разработка архитектуры.
- Кодирование.
- Тестирование и отладка.
- Документирование.
- Сертификация.
- Внедрение.
- Сопровождение.

Модели процесса:

- Модель водопада (Каскадная модель).
- Итеративный процесс.
- Гибкие методологии разработки.
- Экстремальное программирование.
- Формальные методы.

Методологии программирования:

- MCF (Microsoft Solutions Framework);
- Гибкая методология разработки Agile (англ. Agile software development) (XP (экстремальное программирование), FDD (Feature driven development) - разработка, управляемая функциональностью, SCRUM, Crystal Clear — гибкая методология разработки приложений и сайтов, DSDM (основан на концепции быстрой разработки приложений (Rapid Application Development, RAD) - представляет собой итеративный и инкрементный подход, который придаёт особое значение продолжительному участию в процессе пользователя/потребителя;
- RUP (Rational Unified Process);
- Индивидуальный процесс разработки (англ. Personal software process, PSP).
- Методологии PSP/TSP (Personal Software Process / Team Software Process). https://docplayer.ru/399267-Metodologiya-psp-i-tsp-opyt-vnedreniya-ispolzovaniya-i-adaptacii-processov-psp-i-tsp.html

2 Лекции № 2 и № 3

Тема 2: Понятие требования. Классификация требований. Требования к продукту и требования к проекту. Планирование проектных задач

Проблемы, которые приходится решать специалистам в процессе создания программного обеспечения, очень сложны. Природа этих проблем не всегда ясна, особенно если разрабатываемая программная система инновационная. В частности, трудно чётко описать те действия, которые должна выполнять система. Описание функциональных возможностей и ограничений, накладываемых на систему, называется требованиями к этой системе, а сам процесс формирования, анализа, документирования и проверки этих функциональных возможностей и ограничений – разработкой требований.

Анализ требований — часть процесса разработки программного обеспечения, включающая в себя сбор требований к программному обеспечению (ПО), их систематизацию, выявление взаимосвязей, а также документирование. Является частью общеинженерной дисциплины «инженерия требований» (англ. Requirements Engineering).

В процессе сбора требований важно принимать во внимание возможные противоречия требований различных заинтересованных лиц, таких как заказчики, разработчики или пользователи.

Полнота и качество анализа требований играют ключевую роль в успехе всего проекта. Требования к ПО должны быть документируемые, выполнимые, тестируемые, с уровнем детализации, достаточным для проектирования системы. Требования могут быть функциональными и нефункциональными.

Анализ требований включает три **типа деятельности**:

- Сбор требований — общение с клиентами и пользователями, чтобы определить, каковы их требования; анализ предметной области.
- Анализ требований — определение, являются ли собранные требования неясными, неполными, неоднозначными или противоречащими; решение этих проблем; выявление взаимосвязи требований.
- Документирование требований — требования могут быть задокументированы в различных формах, таких как простое описание, сценарии использования, пользовательские истории, или спецификации процессов.

Анализ требований может быть длинным и трудным процессом, во время которого вовлечены много тонких психологических навыков. Новые системы изменяют окружающую среду и отношения между людьми, поэтому важно определить все заинтересованные лица, принять во внимание все их потребности и гарантировать, что они понимают значения новых систем. Аналитики могут использовать несколько методов, чтобы выявить следующие требования от клиента: проведение интервью, использование фокус-групп или создание списков требований. Более современные методы включают создание прототипов и сценариев использования. Где необходимо, аналитик будет использовать комбинацию этих методов, чтобы выявить точные требования заинтересованных лиц так, чтобы была создана система, которая удовлетворяет деловые потребности.

Уровни и типы требований

Бизнес-требование — высокоуровневая бизнес-цель организации или заказчиков системы.

Бизнес-правило — политика, предписание, стандарт или правило, определяющее или ограничивающее некоторые стороны бизнес-процессов. По своей сути это не требование к ПО, но оно служит источником нескольких типов требований к ПО.

Ограничение – ограничение на выбор вариантов, доступных разработчику при проектировании и разработке продукта.

Внешнее требование к интерфейсу – описание взаимодействия между ПО и пользователем, другой программной системой или устройством.

Характеристика – одна или несколько логически связанных возможностей системы, которые представляют ценность для пользователя и описаны рядом функциональных требований.

Функциональное требование – описание требуемого поведения системы в определенных условиях.

Нефункциональное требование – описание свойства или особенности, которым должна обладать система, или ограничение, которое должна соблюдать система.

Атрибут качества – вид нефункционального требования, описывающего характеристику сервиса или производительности продукта.

Системное требование – требование верхнего уровня к продукту, состоящему из многих подсистем, которые могут представлять собой ПО или совокупность ПО и оборудования.

Пользовательское требование – задача, которую определенные классы пользователей должны иметь возможность выполнять в системе, или требуемый атрибут продукта.

Требования к ПО состоят из трех уровней — бизнес-требования, пользовательские и функциональные требования. Вдобавок в каждой системе есть свои нефункциональные

требования. Модель на рисунке 2.1 иллюстрирует способ представления этих типов требований. Как и все модели, она не полная, но схематично показывает организацию требований.

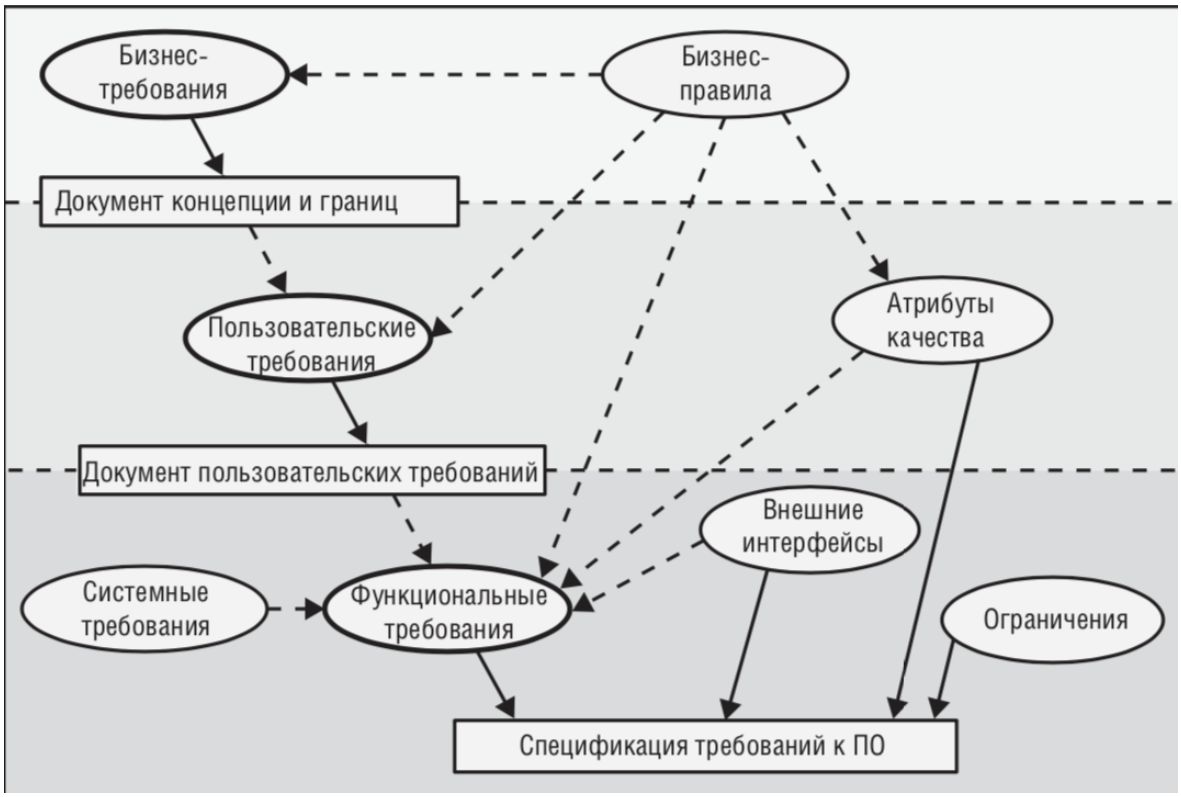

Рисунок 2.1 – Взаимосвязи нескольких типов информации для требований

Основные три уровня требований располагаются следующим образом:

- На верхнем уровне представлены бизнес-требования (business requirements). Примеры бизнес-требования: система должна сократить срок оборачиваемости обрабатываемых на предприятии заказов в три раза. Бизнес-требования обычно формулируются топ-менеджерами, либо акционерами предприятия.
- Следующий уровень – уровень требований пользователей (user requirements). Пример требования пользователя: система должна представлять диалоговые средства для ввода исчерпывающей информации о заказе, последующей фиксации информации в базе данных и маршрутизации информации о заказе к сотруднику, отвечающему за его планирование и исполнение. Требования пользователей часто бывают плохо структурированными, дублирующимися, противоречивыми. Поэтому для создания системы важен третий уровень, в котором осуществляется формализация требований.
- Третий уровень – функциональный (functional requirements). Пример функциональных требований (или просто функций) по работе с электронным

заказом: заказ может быть создан, отредактирован, удален и перемещен с участка на участок.

Планирование проектных задач

Основной задачей при планировании является определение WBS — Work Breakdown Structure (структуры распределения работ). Она составляется с помощью утилиты планирования проекта. Типовая WBS приведена на рисунке 2.2.

Первыми выполняемыми задачами являются системный анализ и анализ требований.

Системный анализ проводится с целью:

- выяснения потребностей заказчика;
- оценки выполнимости системы;
- выполнения экономического и технического анализа;
- распределения функций по элементам компьютерной системы (аппаратуре, программам, людям, базам данных и т. д.);
- определения стоимости и ограничений планирования;
- создания системной спецификации.

В системной спецификации описываются функции, характеристики системы, ограничения разработки, входная и выходная информация.

Анализ требований даёт возможность:

- определить функции и характеристики программного продукта;
- обозначить интерфейс продукта с другими системными элементами;
- определить проектные ограничения программного продукта;
- построить модели: процесса, данных, режимов функционирования продукта;
- создать такие формы представления информации и функций системы, которые можно использовать в ходе проектирования.

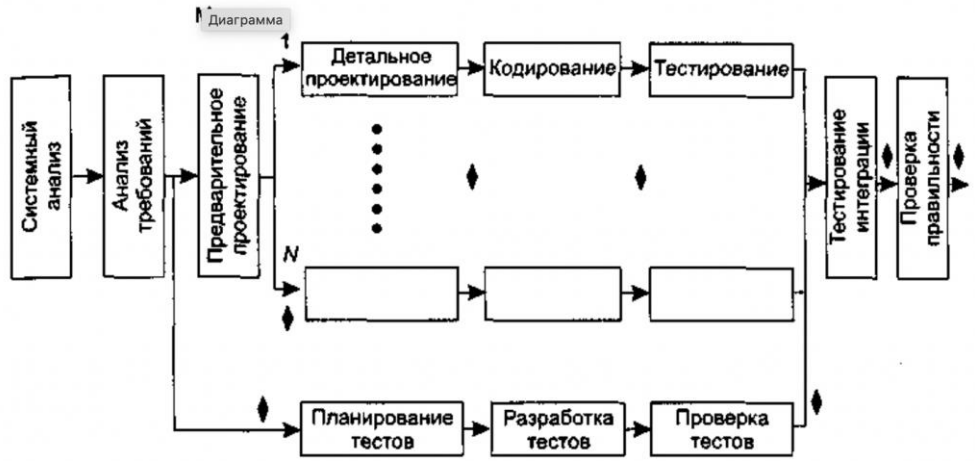

Рисунок 2.2 – Типовая структура распределения работ

Результаты анализа сводятся в **спецификацию требований** к программному продукту.

Как видно из типовой структуры, задачи по проектированию и планированию тестов могут быть распараллелены. Благодаря модульной природе ПО для каждого модуля можно предусмотреть параллельный путь для детального (процедурного) проектирования, кодирования и тестирования. После получения всех модулей ПО решается задача тестирования интеграции — объединения элементов в единое целое. Далее проводится тестирование правильности, которое обеспечивает проверку соответствия ПО требованиям заказчика.

Ромбиками на рисунке 2.2 обозначены вехи — процедуры контроля промежуточных результатов. Очень важно, чтобы вехи были расставлены через регулярные интервалы (вдоль всего процесса разработки ПО). Это даст руководителю возможность регулярно получать информацию о текущем положении дел. Вехи распространяются и на документацию как на один из результатов успешного решения задачи.

Анализ требований

Анализ требований — это процесс сбора требований к программному обеспечению (ПО), их систематизации, документирования, анализа, выявления противоречий, неполноты, разрешения конфликтов в процессе разработки программного обеспечения. В англоязычной среде также говорят о дисциплине «инженерия требований» (англ. Requirements Engineering). В процессе сбора требований важно принимать во внимание возможные противоречия требований различных заинтересованных лиц, таких как заказчики, разработчики или пользователи.

Полнота и качество анализа требований играют ключевую роль в успехе всего проекта. Требования к ПО должны быть документируемые, выполнимые, тестируемые, с уровнем детализации достаточным для проектирования системы. Требования могут быть функциональными и нефункциональными.

Виды требований

а) **Функциональный характер** — требования к поведению системы.

1) Бизнес-требования — определяют назначение ПО, описываются в документе о видении (vision) и границах проекта (scope).
2) Пользовательские требования — определяют набор пользовательских задач, которые должна решать программа, а также способы (сценарии) их решения в системе. Пользовательские требования могут выражаться в виде фраз утверждений, в виде способов применения (use case), пользовательских историй (user story), сценариев взаимодействия (scenario).

3) Функциональные требования — охватывают предполагаемое поведение системы, определяя действия, которые система способна выполнять. Описывается в системной спецификации (англ. system requirement specification, SRS).

b) **Нефункциональный характер** — требования к характеру поведения системы.

1) Бизнес-правила — определяют ограничения, проистекающие из предметной области и свойств автоматизируемого объекта (предприятия)
2) Системные требования и ограничения — определения элементарных операций, которые должна иметь система, а также различных условий, которым она может удовлетворять. К системным ограничениям относятся ограничения на программные интерфейсы, требования к атрибутам качества, требования к применяемому оборудованию и ПО.
3) Атрибуты качества – описание различных измерений характеристик продукта, которые важны для пользователей или для разработчиков и тех, кто будет обслуживать систему, таких как производительность, доступность и переносимость.
4) Внешние системы и интерфейсы – подключения другим к внешним программным системам аппаратным устройствам и пользователям, а также коммуникационные интерфейсы.
5) Ограничения проектирования – накладывают границы на возможности выбора разработчика при проектировании продукта.

Источники требований

Перечислим ниже источники требований:

- Федеральное и муниципальное отраслевое законодательство (конституция, законы, распоряжения).
- Нормативное обеспечение организации (регламенты, положения, уставы, приказы).
- Текущая организация деятельности объекта автоматизации.
- Модели деятельности (диаграммы бизнес-процессов).
- Представления и ожидания потребителей и пользователей системы.
- Журналы использования существующих программно-аппаратных систем.
- Конкурирующие программные продукты.

Характеристики требований

Характеристики качественных требований по-разному определены различными источниками. Однако, следующие характеристики являются общепризнанными:

- **Единичность** – требование описывает одну и только одну вещь.
- **Завершённость** – требование полностью определено в одном месте, и вся необходимая информация присутствует.
- **Последовательность** – требование не противоречит другим требованиям и полностью соответствует внешней документации.
- **Атомарность** – требование «атомарно». То есть оно не может быть разбито на ряд более детальных требований без потери завершённости.
- **Отслеживаемость** – требование полностью или частично соответствует деловым нуждам как заявлено заинтересованными лицами и документировано.
- **Актуальность** – требование не стало устаревшим с течением времени.
- **Выполнимость** – требование может быть реализовано в пределах проекта.
- **Недвусмысленность** – требование кратко определено без обращения к техническому жаргону, акронимам и другим скрытым формулировкам. Оно выражает объективные факты, не субъективные мнения. Возможна одна и только одна интерпретация. Определение содержит четкие фразы. Использование отрицательных утверждений и составных утверждений запрещено.
- **Обязательность** – требование представляет определённую заинтересованным лицом характеристику, отсутствие которой приведёт к неполноценности решения, которая не может быть проигнорирована. Необязательное требование — противоречие самому понятию требования.
- **Проверяемость** – реализованность требования может быть определена через один из четырёх возможных методов: осмотр, демонстрация, тест или анализ.

Проверка требований

Все требования должны быть поддающимися проверке. Наиболее общепринятая методика проверки — тесты. Если проверка тестами невозможна, тогда должен использоваться другой метод проверки (анализ, демонстрация, осмотр или обзор дизайна).

Определённые требования не являются поддающимися проверке. Они включают требования, которые говорят, что система никогда не должна или всегда должна показывать специфическое свойство. Надлежащее тестирование этих требований потребовало бы

бесконечного цикла тестирования. Такие требования должны быть переопределены так, чтобы они стали поддающимися проверке.

Нефункциональные требования, которые являются неподдающимися проверке на программном уровне, все равно должны быть сохранены как документация намерений клиента. Такие требования к продукту могут быть преобразованы в требования к процессу.

Требования к продукту – требования, описывающие свойства программной системы, которую планируется построить.

Требования к проекту

Требования к проекту – документ, где описана среда разработки, ограничения бюджета, руководство пользователя или требования для выпуска продукта и продвижения его в поддерживаемую среду.

К **требованиям проекта** относятся:

a) Физические ресурсы, необходимые команде разработки:
    1) рабочие станции;
    2) специальные аппаратные устройства;
    3) тестовые лаборатории;
    4) средства и оборудование тестирования;
    5) командные комнаты
    6) оборудование для видеоконференций.
b) Потребности в обучении персонала.
c) Пользовательская документация:
    1) обучающие материалы;
    2) пособия;
    3) справочные руководства;
    4) информация о выпусках ПО.
d) Документация для поддержки:
    1) ресурсы службы технической поддержки;
    2) информация о техническом обеспечении;
    3) обслуживание аппаратных устройств.
e) Инфраструктурные изменения.
f) Требования и процедуры для выпуска продукта.
g) Требования и процедуры для перехода со старой на новую систему.
h) Требования по сертификации продукта.

i) Скорректированные политики.

j) Сорсинг, приобретение и лицензирование ПО сторонних.

k) Требования по бета-тестированию, маркетингу и дистрибуции.

l) Соглашения об уровне обслуживания с клиентами.

m) Требования по правовой защите интеллектуальной собственности.

3 Лекция № 4

Тема 3: Свойства требований

Строжайшее и единственное правило построения систем программного обеспечения – решить точно, что же строить. Никакая другая часть концептуальной работы не является такой трудной, как выяснение деталей технических требований, в том числе и взаимодействие с людьми, с механизмами и с иными системами ПО.

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.

Остановимся на каждом из перечисленных выше свойств.

Полнота

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

Ясность

Недвусмысленность, определенность, однозначность спецификаций. Требование обладает свойством ясности, если оно сходным образом воспринимается всеми совладельцами системы. На практике ясность требований достигается в том числе и в процессе консультаций, в ходе которых происходит "выравнивание тезаурусов" совладельцев системы. Хорошим подспорьем в этом служит согласованный сторонами глоссарий ключевых понятий предметной области.

Корректность и согласованность (непротиворечивость)

Корректность – одно из важнейших свойств требований. Корректность требования представляется через точность описания функциональности. В этом смысле корректность в определенной степени конкурирует с полнотой. Свойство корректности носит оценочный

характер и задает дихотомию: каждое из требований либо корректно, либо нет. Если два требования вступают в конфликт, значит - как минимум одно из них некорректно.

Верифицируемость (пригодность к проверке)

Свойство верифицируемости существенно связано со свойствами ясности и полноты: если требование изложено на языке, понятном и одинаково воспринимаемом участниками процесса создания информационной системы, причем оно является полным, т.е. ни одна из важных для реализации деталей не упущена – значит, это требование можно проверить.

Необходимость и полезность при эксплуатации

Необходимость – свойство без выполнения которого, невозможно либо затруднительно выполнение бизнес-функций пользователя.

Полезность при эксплуатации – любое свойство, повышающее эргономичность продукта.

Осуществимость (выполнимость)

Выполнимость требования на практике определяется разумным балансом между ценностью (степенью необходимости и полезности) и потребными ресурсами. Отличной иллюстрацией балансировки между ценностью и выполнимостью требований являются треугольник компромиссов, изображенный на рисунке 3.1, и матрица компромиссов, представленная таблицей 3.1.

Рисунок 3.1 – Треугольник компромиссов

Таблица 3.1 – Матрица компромиссов

	Фиксированные	Оптимальные	Как могут
Ресурсы	+		
Время		+	
Функционал			+

Модифицируемость

Каждое требование должно быть записано лишь единожды.

Трассируемость

Трассируемость требования определяется возможностью отследить связь между ним и другими артефактами информационной системы (документами, моделями, текстами программ и пр.).

4 Лекция № 5

Тема 4: Разработка и управление требованиями. Выявление и сбор требований. Анализ. Документирование. Утверждение. Методы выявления требований. Проверка требований

Разработка и управление требований

Область разработки технических условий подразделяется на разработку требований и управление требованиями как показано на рисунке 2.3. Независимо от методологии проекта — водопадная, поэтапная, итеративная, гибкая или смешанная — есть вещи, которые надо выполнить в отношении всех требований. В зависимости от методологии эти операции могут выполняться в разное время жизненного цикла проекта и с разным уровнем глубины и детализации.

Как показано на рисунке 4.1 разработка технических условий подразделяется на выявление, анализ, документирование и утверждение. В эти составные части входят все действия, включающие сбор, оценку, документирование и утверждение требований для ПО.

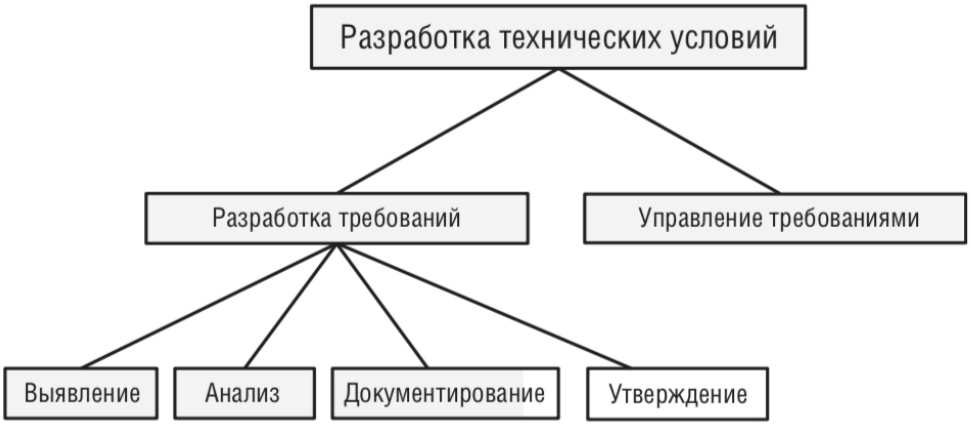

Рисунок 4.1 – Составные части предмета разработки технических условий

Разработка требований

Разработка требований — это процесс, включающий мероприятия, необходимые для создания и утверждения документа, содержащего спецификацию системных требований. Различают четыре основных этапа процесса разработки требований:

- анализ технической осуществимости создания системы,
- формирование и анализ требований,
- специфицирование требований и создание соответствующей документации,
- аттестация этих требований.

Анализ технической осуществимости создания системы

Выполнятся общее описание системы и ее назначения, в результате анализа формируется отчет, в котором должна быть четкая рекомендация, продолжать или нет процесс разработки требований проектируемой системы. Другими словами, анализ осуществимости должен осветить следующие вопросы.

- Отвечает ли система целям организации-заказчика и организации-разработчика?
- Можно ли реализовать систему, используя существующие на данный момент технологии и не выходя за пределы заданной стоимости?
- Можно ли объединить систему с другими системами, которые уже эксплуатируются?

Сначала следует определить, какая именно информация необходима, чтобы ответить на поставленные выше вопросы. Например, эту информацию можно получить, ответив на следующее:

- Что произойдет с организацией, если система не будет введена в эксплуатацию?
- Какие текущие проблемы существуют в организации и как новая система поможет их решить?
- Каким образом система будет способствовать целям бизнеса?
- Требует ли разработка системы технологии, которая до этого не использовалась в организации?

Далее необходимо определить источники информации. Это могут быть менеджеры отделов, где система будет использоваться, разработчики программного обеспечения, знакомые с типом будущей системы, технологи, конечные пользователи и т.д.

После обработки собранной информации готовится отчет по анализу осуществимости создания системы. В нем должны быть даны рекомендации относительно продолжения разработки системы. Могут быть предложены изменения бюджета и графика работ по созданию системы или предъявлены более высокие требования к системе.

Формирование и анализ требований

Обобщенная модель процесса формирования и анализа требований показана на рис. 1. Каждая организация использует собственный вариант этой модели, зависящий от “местных факторов”: опыта работы коллектива разработчиков, типа разрабатываемой системы, используемых стандартов и т.д.

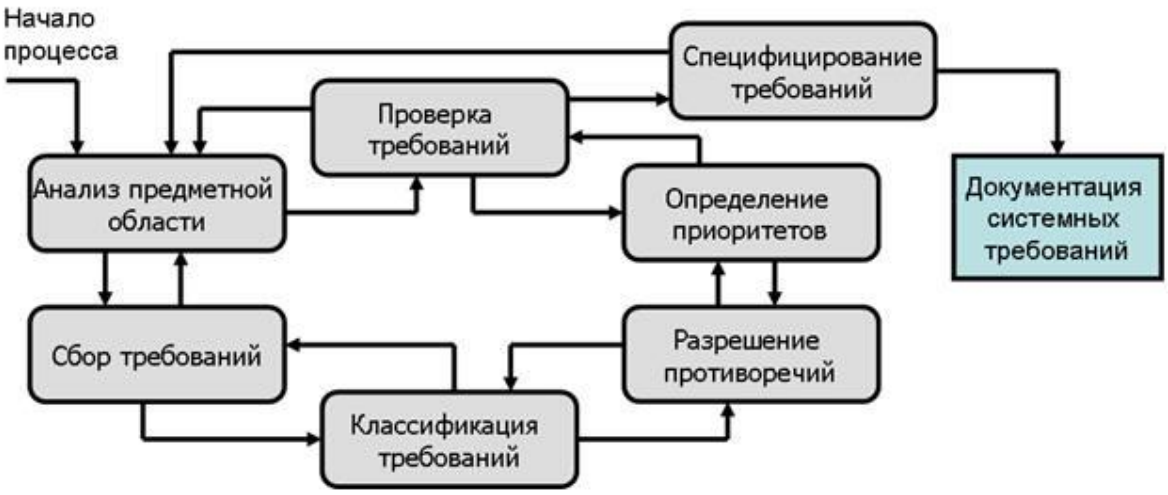

Рисунок 4.2 - Процесс формирования и анализа требований

Процесс формирования и анализа требований проходит через ряд этапов:

Этап 1. Анализ предметной области. Аналитики должны изучить предметную область, где будет эксплуатироваться система.

Этап 2. Сбор требований. Это процесс взаимодействия с лицами, формирующими требования. Во время этого процесса продолжается анализ предметной области.

Этап 3. Классификация требований. На этом этапе бесформенный набор требований преобразуется в логически связанные группы требований.

Этап 4. Разрешение противоречий. Без сомнения, требования многочисленных лиц, занятых в процессе формирования требований, будут противоречивыми. На этом этапе определяются и разрешаются противоречия различного рода.

Этап 5. Назначение приоритетов. В любом наборе требований одни из них будут более важны, чем другие. На этом этапе совместно с лицами, формирующими требования, определяются наиболее важные требования.

Этап 6. Проверка требований. На этом этапе определяется их полнота, последовательность и непротиворечивость.

Процесс формирования и анализа требований циклический, с обратной связью от одного этапа к другому. Цикл начинается с анализа предметной области и заканчивается проверкой требований. Понимание требований предметной области увеличивается в каждом цикле процесса формирования требований.

Рассмотрим два основных подхода к формированию требований: метод, основанный на множестве опорных точек зрения и сценарии.

Опорные точки зрения

Подход с использованием различных *опорных* точек зрения к разработке требований признает различные (опорные) точки зрения на проблему и использует их в качестве основы

построения и организации как процесса формирования требований, так и непосредственно самих требований.

Различные методы предлагают разные трактовки выражения "точка зрения". Точки зрения можно трактовать следующим образом.

a) *Как источник информации о системных данных.* В этом случае на основе опорных точек зрения строится модель создания и использования данных в системе. В процессе формирования требований отбираются все такие точки зрения (и на их основе определяются данные), которые будут созданы или использованы при работе системы, а также способы обработки этих данных.

b) *Как структура представлений.* В этом случае точки зрения рассматриваются как особая часть модели системы. Например, на основе различных точек зрения могут разрабатываться модели "сущность-связь", модели конечного автомата и т.д.

c) *Как получатели системных сервисов.* В этом случае точки зрения являются внешними (относительно системы) получателями системных сервисов. Точки зрения помогают определить данные, необходимые для выполнения системных сервисов или их управления.

Наиболее эффективным подходом к анализу таких систем является использование внешних опорных точек зрения. На основе этого подхода разработан метод VORD (Viewpoint-Oriented Requirements Definition — определение требований на основе точек зрения) для формирования и анализа требований. Основные этапы метода VORD показаны на рисунке 4.3:

Этап 1. Идентификация точек зрения, получающих системные сервисы, и идентификация сервисов, соответствующих каждой точке зрения.

Этап 2. Структурирование точек зрения — создание иерархии сгруппированных точек зрения. Общесистемные сервисы предоставляются более высоким уровням иерархии и наследуются точками зрения низшего уровня.

Этап 3. Документирование опорных точек зрения, которое заключается в точном описании идентифицированных точек зрения и сервисов.

Этап 4. Отображение системы точек зрения, которая показывает системные объекты, определенные на основе информации, заключенной в опорных точках зрения.

Рисунок 4.3 - Метод VORD

Пример. Рассмотрим использование метода VORD на первых трех шагах анализа требований для системы поддержки заказа и учета товаров в бакалейной лавке. В бакалейной лавке для каждого товара фиксируется место хранения (определенная полка), количество товара и его поставщик. Система поддержки заказа и учета товаров должна обеспечивать добавление информации о новом товаре, изменение или удаление информации об имеющемся товаре, хранение (добавление, изменение и удаление) информации о поставщиках, включающей в себя название фирмы, ее адрес и телефон. При помощи системы составляются заказы поставщикам. Каждый заказ может содержать несколько позиций, в каждой позиции указываются наименование товара и его количество в заказе. Система по требованию пользователя формирует и выдает на печать следующую справочную информацию:

- список всех товаров;
- список товаров, имеющихся в наличии;
- список товаров, количество которых необходимо пополнить;
- список товаров, поставляемых данным поставщиком.

Первым шагом в формировании требований является идентификация опорных точек зрения. Во всех методах формирования требований, основанных на использовании точек зрения, начальная идентификация является наиболее трудной задачей. Один из подходов к идентификации точек зрения — метод "мозговой атаки", когда определяются потенциальные системные сервисы и организации, взаимодействующие с системой. Организуется встреча лиц, участвующих в формировании требований, которые предлагают свои точки зрения. Эти точки зрения представляются в виде диаграммы, состоящей из ряда круговых областей, отображающих возможные точки зрения (рисунок 4.5). Во время "мозговой атаки" необходимо идентифицировать потенциальные опорные точки зрения, системные сервисы, входные данные, нефункциональные требования, управляющие события и исключительные ситуации.

Следующей стадией процесса формирования требований будет идентификация опорных точек зрения (на рисунке 4.4 показаны в виде темных круговых областей) и сервисов (показаны в виде затененных областей). Сервисы должны соответствовать опорным точкам зрения. Но могут быть сервисы, которые не поставлены им в соответствие. Это означает, что на начальном этапе "мозговой атаки" некоторые опорные точки зрения не были идентифицированы.

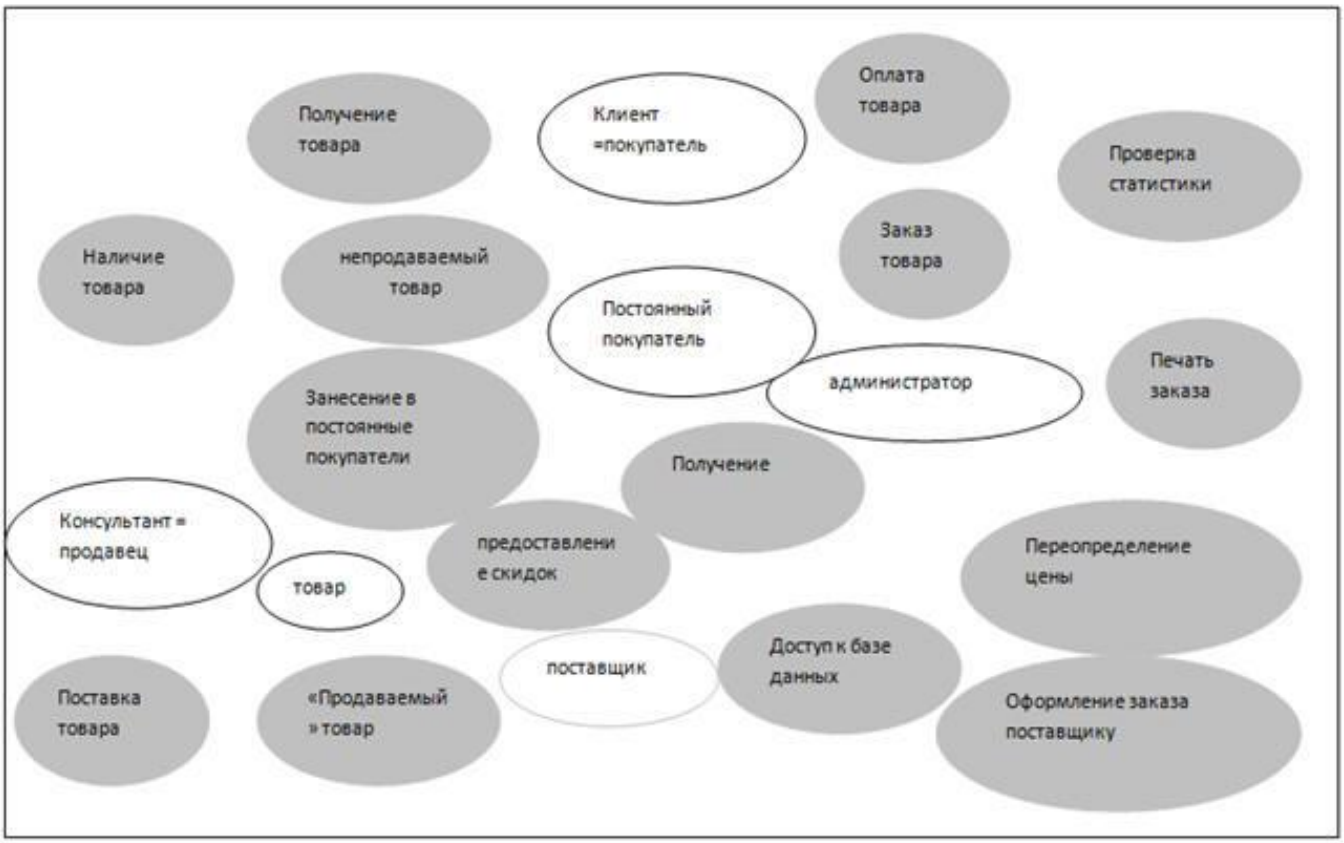

Рисунок 4.4 - Диаграмма идентификации точек зрения

В таблице 4.1 показано распределение сервисов для некоторых идентифицированных на рисунке 4.4 точек зрения. Один и тот же сервис может быть соотнесен с несколькими точками зрения.

Таблица 4.1 - Сервисы, соотнесенные с точками зрения

Клиент	**Постоянный покупатель**	**Товар**	**Поставщик**	**Продавец**	**Администратор**
Проверка наличия товара	Получение скидки	Прием товара	Занесение в базу данных (название, адрес, телефон и т.д.)	Продажа товара	Доступ к базе данных
Покупка товара	Получение информацию о новых поступлениях	Занесение в базу данных (данные о поставщике, кол-ве, месте хранения и.д.)		Печать чека	Проверка статистики

Продолжение таблицы 4.1

Получение чека		Назначение цены		Доступ к каталогу	Переопределение цены
Заказ товара		Переопределение цены		Проверка наличия товара	Оформление заказа поставщику
Занесение покупателя и суммы покупки в базу данных		«Покупаемый» или «непокупаемый» товар		Оформление заказа покупателю	Печать заказа

Информация, извлеченная из точек зрения, используется для заполнения форм шаблонов точек зрения и организации точек зрения в иерархию наследования. Это позволяет увидеть общие точки зрения и повторно использовать информацию в иерархии наследования. Сервисы, данные и управляющая информация наследуются подмножеством точек зрения. На рисунке 4.5 показана часть иерархии точек зрения для системы поддержки заказа и учета товаров.

Рисунок 4.5 - Иерархия точек зрения

Сценарии

Сценарии особенно полезны для детализации уже сформулированных требований, поскольку описывают последовательность интерактивной работы пользователя с системой. Каждый сценарий описывает одно или несколько возможных взаимодействий.

Сценарий начинается с общего описания, затем постепенно детализируется для создания полного описания взаимодействия пользователя с системой.

В большинстве случаев сценарий включает следующее:

– Описание состояния системы после завершения сценария.
– Информацию относительно других действий, которые можно осуществлять во время выполнения сценария.
– Описание исключительных ситуаций и способов их обработки.
– Описание нормального протекания событий.
– Описание состояния системы в начале сценария.

Сценарии событий используются для документирования поведения системы, представленного определенными событиями. Сценарии включают описание потоков данных, системных операций и исключительных ситуаций, которые могут возникнуть.

Этнографический подход

Этнографический подход к формированию системных требований используется для понимания и формирования социальных и организационных аспектов эксплуатации системы. Разработчик требований погружается в рабочую среду, где будет использоваться система. Его ежедневная работа связана с наблюдением и протоколированием реальных действий, выполняемых пользователями системы. Значение этнографического подхода заключается в том, что он помогает обнаружить неявные требования к системе, которые отражают реальные аспекты ее эксплуатации, а не формальные умозрительные процессы.

Этнографический подход позволяет детализировать требования для критических систем, чего не всегда можно добиться другими методами разработки требований. Однако, поскольку этот метод ориентирован на конечного пользователя, он не может охватить все требования предметной области и требования организационного характера.

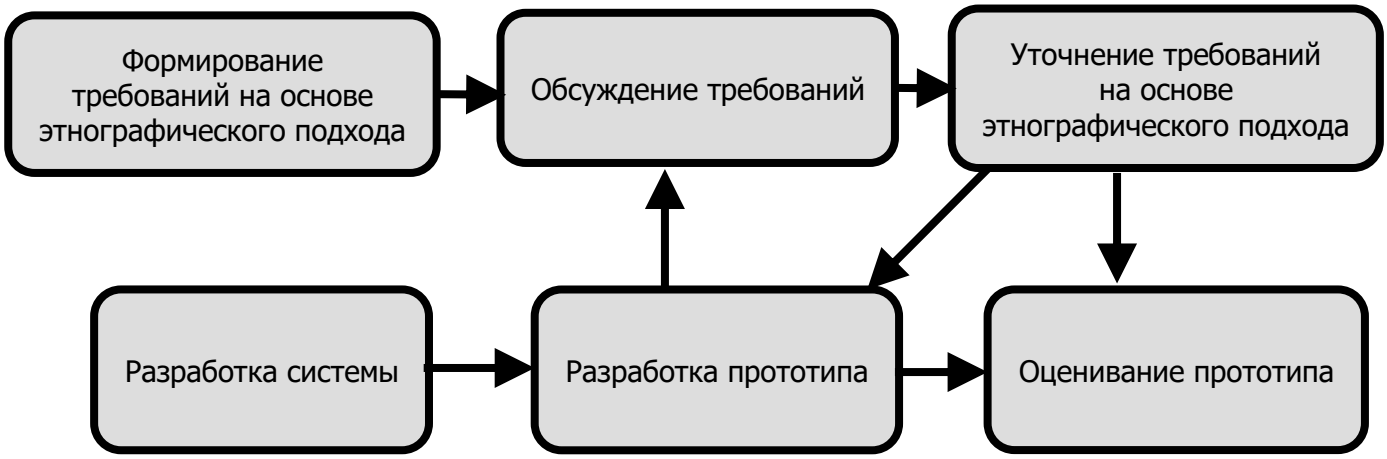

Рисунок 4.6 – Процесс разработки требований согласно этнографическому подходу

Аттестация требований

Аттестация должна продемонстрировать, что требования действительно определяют ту систему, которую хочет иметь заказчик. Проверка требований важна, так как ошибки в спецификации требований могут привести к переделке системы и большим затратам, если будут обнаружены во время процесса разработки системы или после введения ее в эксплуатацию. Стоимость внесения в систему изменений, необходимых для устранения ошибок в требованиях, намного выше, чем исправление ошибок проектирования или кодирования. Причина в том, что изменение требований обычно влечет за собой значительные изменения в системе, после внесения которых она должна пройти повторное тестирование.

Во время процесса аттестации должны быть выполнены различные типы проверок требований:

- **Проверка правильности требований**. Пользователь может считать, что система необходима для выполнения некоторых определенных функций. Однако дальнейшие размышления и анализ могут привести к необходимости введения дополнительных или новых функций. Системы предназначены для разных пользователей с различными потребностями, и поэтому набор требований будет представлять собой некоторый компромисс между требованиями пользователей системы.
- **Проверка на непротиворечивость.** Спецификация требований не должна содержать противоречий. Это означает, что в требованиях не должно быть противоречащих друг другу ограничений или различных описаний одной и той же системной функции.
- **Проверка на полноту.** Спецификация требований должна содержать требования, которые определяют все системные функции и ограничения, налагаемые на систему.
- **Проверка на выполнимость.** На основе знания существующих технологий требования должны быть проверены на возможность их реального выполнения. Здесь также проверяются возможности финансирования и график разработки системы.

Существует ряд методов аттестации требований, которые можно использовать совместно или каждый в отдельности:

a) **Обзор требований.** Требования системно анализируются рецензентами.

b) **Прототипирование.** На этом этапе прототип системы демонстрируется конечным пользователям и заказчику. Они могут экспериментировать с этим прототипом, чтобы убедиться, что он отвечает их потребностям.

c) **Генерация тестовых сценариев.** В идеале требования должны быть такими, чтобы их реализацию можно было протестировать. Если тесты для требований разрабатываются как часть процесса аттестации, то часто это позволяет обнаружить проблемы в

спецификации. Если такие тесты сложно или невозможно разработать, то обычно это означает, что требования трудно выполнить и поэтому необходимо их пересмотреть.

d) **Автоматизированный анализ непротиворечивости.** Если требования представлены в виде структурных или формальных системных моделей, можно использовать инструментальные CASE-средства для проверки непротиворечивости моделей. Для автоматизированной проверки непротиворечивости строят базу данных требований и затем проверяют все требования в этой базе данных. Анализатор требований готовит отчет обо всех обнаруженных противоречиях.

Выявление и сбор требований

Методы выявления требований:

- интервью, опросы, анкетирование;
- мозговой штурм, семинар;
- наблюдение за производственной деятельностью, «фотографирование» рабочего дня;
- анализ нормативной документации;
- анализ моделей деятельности;
- анализ конкурентных продуктов;
- анализ статистики использования предыдущих версий системы.

Ключевые действия при сборе информации:

- определение концепции продукта и границ проекта;
- определение классов ожидаемых пользователей продукта и других заинтересованных лиц;
- понимание задач и целей, а также бизнес-целей, которым соответствуют эти задачи;
- изучение среды, в которой будет использоваться новый продукт;
- работа с отдельными людьми, которые представляют каждый класс пользователей, чтобы понять их потребности и ожидания в отношении качества.

Подходы при сборе:

- подход, ориентированный на использование: упор делается на понимание и исследование задач пользователей, и на основе этой информации выводится необходимая функциональность системы;
- подход, ориентированный на продукт: сфокусирован на определение функций, которые, как ожидается, приведут к успеху на рынке или успеху бизнеса компании. В стратегиях, ориентированных на продукт, есть риск реализовать функции, которые не будут

активно использоваться несмотря на то, что во время сбора требований они казались очень нужными.

Приемы формулирования требований:

- выделите из пользователей ярых сторонников продукта;
- создайте фокус-группы;
- определение пользовательские требования;
- определение системные события и реакцию на них;
- проведите интервью для выявления требований;
- проведите семинары по выявлению требований;
- наблюдайте за пользователями на рабочих местах;
- раздайте опросные листы;
- выполните анализ документов;
- изучите отчеты о проблемах;
- повторно задействуйте существующие требования.

Анализ

Ключевые действия:

- анализ информации, полученной от пользователей, чтобы отделить их задачи от функциональных и нефункциональных требований, бизнес-правил, предполагаемых решений и другой информации;
- разложение высокоуровневых требований до нужного уровня детализации;
- выведение функциональных требований из информации других требований;
- понимание относительной важности атрибутов качества;
- распределение требований по компонентам ПО, определенным в системной архитектуре;
- согласование приоритетов реализации;
- выявление пробелов в требованиях или излишних требований, не соответствующих заданным рамкам.

Приемы формулирования требований:

- смоделируйте среду приложения;
- создайте прототипы;
- проанализируйте осуществимость;
- расставьте приоритеты для требований;
- создайте словарь данных;

– смоделируйте требования;
– проанализируйте интерфейсы;
– распределите требования по подсистемам.

Документирование

Ключевые действия:

– преобразование собранных потребностей пользователей в письменные требования и диаграммы, пригодные для понимания, анализа и использования целевой аудиторией.

Приемы формулирования требований:

– используйте шаблон спецификации требований;
– определение источники требований;
– задайте каждому требованию уникальный идентификатор;
– задокументируйте бизнес-правила;
– определение атрибуты качества.

Утверждение

Утверждение требований должно подтвердить правильность имеющегося набора требований, которые позволят разработчикам создать решение, удовлетворяющее бизнес-целям.

Ключевые действия:

– проверка задокументированных требований для устранения всех недостатков до принятия требований группой разработки;
– разработка приемочных тестов и критериев, которые должны подтвердить, что созданный на основе требований продукт будет отвечать потребностям заказчика и удовлетворять поставленным бизнес-целям.

Приемы формулирования требований:

– изучите документы с требованиями;
– протестируйте требования;
– определение критерии приемлемости;
– смоделируйте требования.

Управление требованиями

Ключевые действия:

- определение основной версии требований, моментальный снимок, который представляет согласованный, проверенный и одобренный набор функциональных и нефункциональных требований, обычно для конкретного выпуска продукта или итерации разработки;
- оценка влияния предлагаемых требований и внедрение одобренных изменений в проект управляемым образом;
- обновление планов проекта в соответствии с изменениями в требованиях;
- обсуждение новых обязательств, основанных на оцененном влиянии изменения требований;
- определение отношений и зависимостей, существующих между требованиями;
- отслеживание отдельных требований до их проектирования, исходного кода и тестов;
- отслеживание состояния требований и действий по изменению на протяжении всего проекта.

Приемы формулирования требований:

- определение процесс управления изменениями;
- проанализируйте, какое влияние оказывают изменения;
- определение базовую и контрольную версии наборов требований;
- отслеживайте хронологию изменений;
- отслеживайте состояние требований;
- отслеживайте проблемы с требованиями;
- создайте матрицу связей требований;
- используйте средство управления требованиями.

Задача управления требованиями состоит в предугадывании и приспосабливании к ожидаемым реальным изменениям, чтобы снизить их разрушительное влияние на проект.

На рисунке 4.10 показан другой способ разделения областей разработки требований и управления ими.

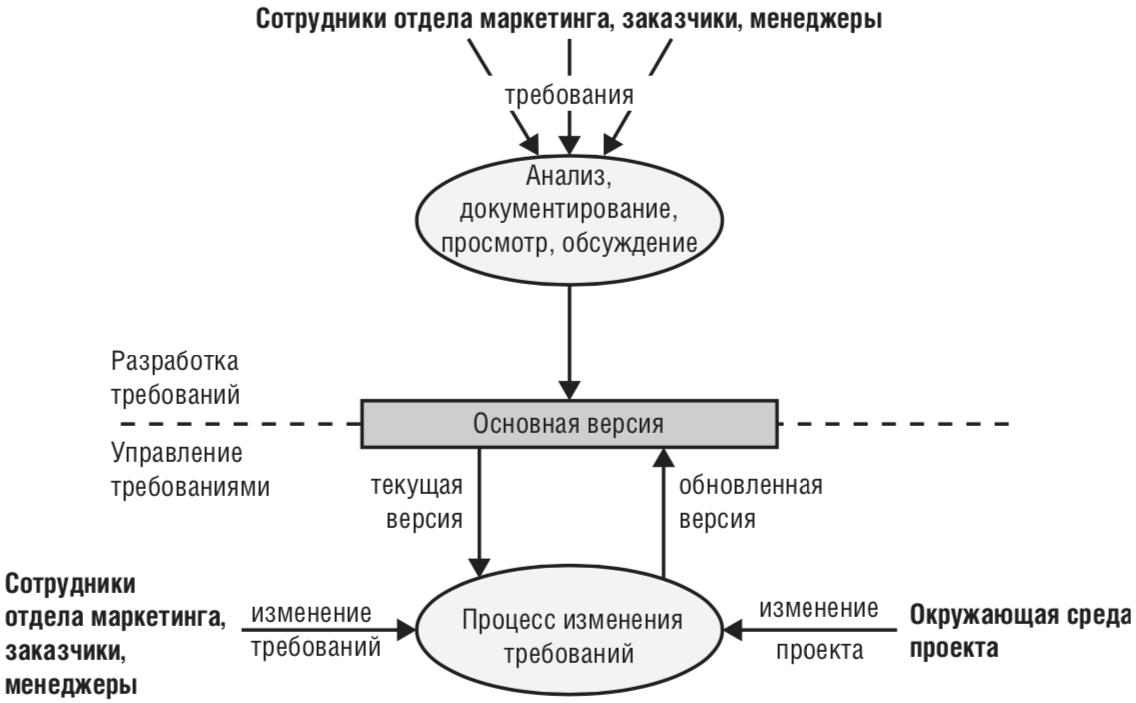

Рисунок 4.10 – Разделение областей разработки требований и управления ими

5 Лекция № 6

Тема 5: Риски. Когда появляются плохие требования? Требование с точки зрения клиента

Риски. Когда появляются плохие требования?

Недостаточное вовлечение пользователей:

- к обнаружению ошибок в требованиях на поздних стадиях проекта, а значит, к задержке завершения проекта;
- бизнес-аналитик может не понять и неправильно задокументировать реальные бизнес-требования или потребности клиента.

Небрежное планирование:

- неопределенные, плохо понятые требования порождают слишком оптимистические оценки, которые возвращаются и не дают нам покоя, когда возникает перерасход;
- наибольшие вклады в проект при плохо просчитанной смете составляют затраты на частые изменения требований, пропущенные требования, недостаточное взаимодействие с пользователями, не детализированную спецификацию требований и плохой анализ.

«Разрастание» требований пользователей:

- в процессе разработки требования могут меняться, из-за чего проект часто выходит за установленные рамки как по срокам, так и по бюджету.

В проектах гибкой методологии объем итерации корректируется так, чтобы вписаться в заданный бюджет и длительность итерации. При появлении новых требований они размещаются в резерве (backlog) и назначаются в будущие итерации на основе приоритета. Изменения зачастую критически важны для успеха, однако они всегда имеют цену.

Двусмысленные требования:

- пользователь может интерпретировать одно и то же положение по-разному;
- у нескольких читателей требования возникает разное понимание, что оно означает;
- формирование всевозможных ожиданий у заинтересованных лиц. впоследствии некоторые из них удивляются результату. Разработчики же впустую тратят время, занимаясь не теми задачами. А тестировщики готовятся к проверке не тех особенностей поведения системы.

Требования-«бантики»

Под «бантиками» (gold plating) понимают отсутствующие в спецификации требований функции, добавленные разработчиками, потому что им кажется, что это понравится пользователям.

Если избыточные возможности оказываются ненужными для клиентов, получается, что время, отведенное на реализацию, тратится впустую. Прежде чем просто вставлять новые функции, разработчики и бизнес-аналитики должны представить свои творческие идеи на суд заказчиков.

Задача команды — четко соблюдать требования спецификации, а не действовать за спиной клиентов, без их одобрения.

Все возможные к добавлению функции стоят времени и денег, поэтому следует акцентировать внимание на максимизации пользы от будущего продукта. Чтобы снизить количество «бантиков», следует отслеживать каждый элемент функциональности до его источника, чтобы четко понимать, почему именно он включен в продукт. Необходимо убедиться, что все специфицируемое и разрабатываемое находится в рамках проекта.

Пропущенные классы пользователей:

- если не определить важные классы пользователей для продукта заранее, некоторые потребности клиентов не будут учтены;
- помимо очевидных пользователей необходимо помнить о сотрудниках поддержки, у которых есть собственные требования, функциональные и нефункциональные. У сотрудников, которые конвертируют данные из унаследованных систем, есть требования по переходу, которые не влияют на конечный продукт, но определенно влияют на успех решения;
- заинтересованные лица, которые даже не знают о существовании проекта, например государственные учреждения, определяющие стандарты, которые могут влиять на систему.

Требования с точки зрения клиента

Клиенты, которым требуется новая информационная система, зачастую не понимают, насколько важно непосредственно опросить будущих пользователей и других заинтересованных лиц. «успех проекта зависит от согласованных действий клиента и программистов».

Одна из проблем при формировании требований в том, что люди путают разные уровни требований: бизнес-уровень, уровень пользователя и функциональный.

С другой стороны, пользователи могут описать необходимые им возможности системы, но не способны грамотно перечислить функциональные требования, которые должны реализовать разработчики для предоставления им таких возможностей. Бизнес-аналитики должны поработать с пользователями, чтобы лучше понять будущую систему.

Разрыв ожиданий

Без адекватного участия клиента в конце проекта неизбежно возникает разрыв ожиданий — пробел между тем, что клиенту реально нужно, и тем, что предоставили разработчики, основываясь на том, что они знали в начале проекта (рисунок 5.1).

Требования устаревают по мере изменения в бизнесе, поэтому взаимодействие с клиентами жизненно важно.

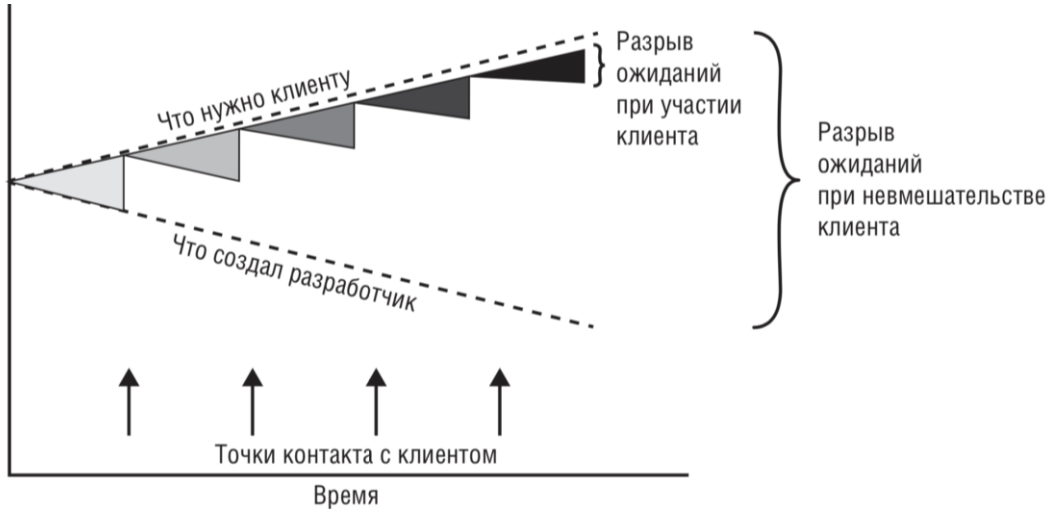

Рисунок 5.1 – Частое взаимодействие с клиентом сокращает разрыв ожиданий

Наилучший способ минимизации разрыва ожиданий — организация частых точек контакта с представителями клиента.

Виды точек контакта:

- интервью;
- собеседование;
- просмотр требований;
- просмотра дизайна пользовательского интерфейса;
- оценки прототипа;
- откликов клиентов на небольшие расширения функциональности исполняемого ПО.

Каждая точка контакта предоставляет возможность закрыть разрыв ожиданий

Основным принципом гибкой разработки является постоянный диалог между разработчиками и клиентами.

Кто клиент?

Клиенты – подмножества заинтересованных лиц. Клиент (customer) — человек или организация, получающая от продукта прямую или косвенную выгоду. Клиенты — это заинтересованные в проекте лица, запрашивающие, оплачивающие, выбирающие, определяющие, использующие и получающие результаты работы программного продукта.

На рисунке 5.2 указаны возможные заинтересованные лица в команде проекта, в разрабатывающей организации и за пределами организации.

Рисунок 5.2 – Возможные заинтересованные лица в команде проекта, в разрабатывающей организации и за пределами организации

Требования пользователей определяют те, кто прямо или косвенно взаимодействуют с продуктом. Конечные являются подмножеством клиентов. Прямые пользователи непосредственно работают с продуктом. Непрямые пользователи могут получать результаты работы системы, не входя в непосредственный контакт с ней, например менеджер хранилища данных, получающий по почте ежедневный отчёт о деятельности хранилища данных. Пользователи способны описать, какие задачи им нужно выполнять, какие результаты они ожидают и какие ожидаются качественные характеристики продукта.

Бизнес- требования должен определять тот, кто в конечном итоге отвечает за пользу для бизнеса, ожидаемую от продукта.

Перечисленные права и обязанности распространяются непосредственно на клиентов в случае, когда программный продукт разрабатывается для внутрикорпоративного использования, на заказ или для определенной группы крупных клиентов.

Билль о правах клиента ПО. Права клиента ПО при формировании требований

Право № 1. Иметь дело с аналитиком, который разговаривает на вашем языке

Выяснить потребности и задачи бизнеса, используя при этом принятую в бизнесе терминологию. Аналитик должен говорить на обычном языке. При разговоре с аналитиками не надо переходить на технический жаргон.

Право № 2. Иметь дело с аналитиком, хорошо изучившим бизнес и цели, для которых создается система

Пригласите разработчиков и аналитиков посмотреть на вашу работу. Если заказанная система заменит существующее приложение, предложите аналитикам поработать с ним. Таким образом, им будет легче понять его сильные и слабые стороны и то, как его можно усовершенствовать. Не надо предполагать, что аналитик уже знает все ваши бизнес-операции и терминологию.

Право № 3. Потребовать, чтобы аналитик зафиксировал требования в надлежащей форме

Требования должны быть написаны и организованы так, чтобы их было легко понимать.

Право № 4. Получить подробный отчёт о будущих процедурах и результатах процесса формулирования требований

Знание требований может представляться в самых разных формах. Аналитик должен объяснить рекомендуемые процедуры. Для иллюстрации текста аналитик может создать ряд диаграмм. Аналитик должен объяснить назначение каждой из них, смысл обозначений и процедуру проверки диаграмм на предмет ошибок.

Право № 5. Изменить свои требования

Выработайте простой, но эффективный процесс работы с изменениями в своем проекте.

Право № 6. На взаимное уважение

Право № 7. Знать о вариантах и альтернативах требований и их решении

Чтобы гарантировать, что новая система не будет автоматизировать неэффективные или устаревшие процессы, аналитик должен знать, почему существующие системы не годятся для ваших бизнес-процессов.

Право № 8. Описать характеристики, упрощающие работу с продуктом

Аналитик должен выяснить конкретные характеристики, означающие для вас дружественность или надежность.

Право № 9. Узнать о способах корректировки требований для ускорения разработки за счет повторного использования

Отношение к требованиям должны быть гибкими. Разумные возможности применения существующих модулей сэкономит ваши время и деньги.

Право № 10. Получить систему, функциональность и качество которой соответствует вашим ожиданиям

Это самое главное право клиента, но оно осуществимо, только если вы сумеете донести до разработчиков всю информацию, которая поможет им создать подходящий продукт, если разработчики четко изложат вам все варианты и ограничения и, если все участники достигнут соглашения.

Билль об обязанностях клиента ПО. Обязанности клиента ПО при формировании требований

Обязанность № 1. Ознакомить аналитиков и разработчиков с особенностями вашего бизнеса

Основная задача — помочь им понять ваши проблемы и цели.

Обязанность № 2. Потратить столько времени, сколько необходимо на предоставление и уточнение требований

Обязанность № 3. Точно и конкретно описать требования к системе

В спецификацию требований к программному обеспечению рекомендуется включить маркеры «требует прояснения» (to be determined, TBD), указывающие на необходимость дополнительных исследований, анализа и информации. Попытайтесь прояснить цель каждого требования, чтобы аналитик мог точно выразить его.

Обязанность № 4. По запросу принимать своевременные решения относительно требований

Согласование противоречивых запросов от разных клиентов, выбор между конфликтующими атрибутами качества и оценка точности информации.

Обязанность № 5. Уважать определенную разработчиком оценку стоимости и возможности реализации ваших требований

Обязанность № 6. Определять реалистичные приоритеты требований совместно с разработчиками

Определить, какие возможности необходимы, какие полезны и без каких пользователи обойдутся — важная составляющая анализа требований.

Определив приоритеты, вы поможете разработчикам вовремя и с минимальными затратами создать максимально эффективный продукт. позволяет разработчикам максимально быстро поставлять полезный программный продукт.

Обязанность № 7. Проверять требования и оценивать прототипы

Рецензирование требований — одна из наиболее значимых операций, обеспечивающих качество ПО. Участие клиентов в таких мероприятиях — единственный способ узнать, отражают ли требования потребности клиента наиболее полно и точно.

Чтобы лучше понять потребности клиента и выявить оптимальные способы их удовлетворения, аналитики и разработчики иногда создают прототипы предполагаемого продукта.

Обязанность № 8. Определить критерии приемки

Одна из обязанностей клиента — установить критерии приемки, то есть заданные условия, которым должен удовлетворять продукт, чтобы его можно было считать приемлемым. Такие критерии включают приемочные тесты, которые оценивают, позволяет ли продукт пользователям правильно выполнять те или иные их операции. Другие критерии приемки могут определять, допустимый уровень оставшихся в продукте дефектов, производительность определенных действий в рабочей среде или способность удовлетворить определенным внешним сертификационным требованиям.

Обязанность № 9. Своевременно сообщать об изменениях требований

Постоянное изменение требований — серьезная угроза для возможности своевременной сдачи проекта командой разработчиков.

Обязанность № 10. Уважительно относиться к процессам создания требований

Выявление и спецификация требований — одни из самых трудных задач в разработке ПО. Взаимопонимание и уважение приемов и потребностей друг друга имеют огромное значение для организации эффективного и даже доставляющего удовольствие сотрудничества.

Определение ответственных за принятие решений

В группе принятия решений нужно определить главного ответственного за принятие решений.

6 Лекция № 7

Тема 6: Выгоды высококачественного продукта. Интерактивный процесс формирования требований. Распределение работ с требованиями на протяжении жизненного цикла проекта в разных моделях разработки

Выгоды высококачественного продукта

Ниже перечислим выгоды высококачественного продукта:

- привлечение множества заинтересованных лиц на протяжении всего проекта;
- упор на задачи пользователей, а не на внешне привлекательные функции, команда избежит необходимости переписывать код, который даже не понадобится;
- ясное разделение требований на те, что относятся к по, оборудованию или подсистемам, взаимодействующим с людьми, позволяет применять системный подход к разработке продукта;
- эффективные процессы управления изменениями минимизируют неблагоприятные последствия от изменения требований;
- недвусмысленно составленные документы облегчают тестирование продукта.

Возможные выгоды:

- меньше дефектов в требованиях и в готовом продукте;
- меньше переделок;
- быстрее разработка и поставка готового продукта;
- меньше ненужных и неиспользуемых функций;
- ниже стоимость модификации;
- меньше недопонимания;
- меньше расползание границ проекта;
- меньше беспорядок в проекте;
- выше удовлетворение заказчиков и членов команды;
- продукты, которые делают то, что от них ожидается.

7 Лекция № 8

Тема 7: Контекст задачи анализа требований. Анализ требований. Бизнес-анализ. Анализ предметной области. Требования и архитектура АИС. Анализ требований и другие рабочие потоки программной инженерии

Контекст задачи анализа требований

В каком случае следует применять анализ требований, бизнес-анализ или бизнес-моделирование?

Анализ требований, бизнес-анализ, анализ проблемной области

Существуют сотни методик, методологий, процессов, стандартов, регламентирующих те или иные детали выбора и комплексирования потоков работ при разработке автоматизированных информационных систем. Анализ требований стоит в начале цепочки работ и его результаты во многом определяют успех проекта. Работы, связанные с бизнес-анализом и бизнес-моделированием: их роль не столь очевидна, и принимается далеко не всеми методологиями.

Вопрос – проводить или не проводить бизнес-анализ, решается в зависимости от конкретной задачи.

Анализ требований стоит в начале цепочки работ и его результаты во многом определяют успех проекта.

Роль глоссария при АТ

Заказчик и Разработчик говорят на разных языках. Успешная реализация проекта в области внедрения АИС во многом зависит от того, удастся ли выработать и документировать их общее представление о предмете разработки. Глоссарий – документ, удостоверяющий общее понимание основной терминологии Заказчиком и Разработчиком.

Задачу анализа бизнес-процессов следует рассматривать как часть более общей задачи систем для анализа проблемной области.

Вначале надо определить цели и задачи самого бизнес-анализа, как этапа построения КИС. С позиций моделирования, анализ требований (АТ) и анализ проблемной области (АПО) - принципиально разные процессы. Классические цели создания модели: существует объект и задача аналитика - отразить этот объект в создаваемой модели с требуемой степенью точности (рисунок 7.1).

Документ АТ является ничем иным, как моделью модели ОС.

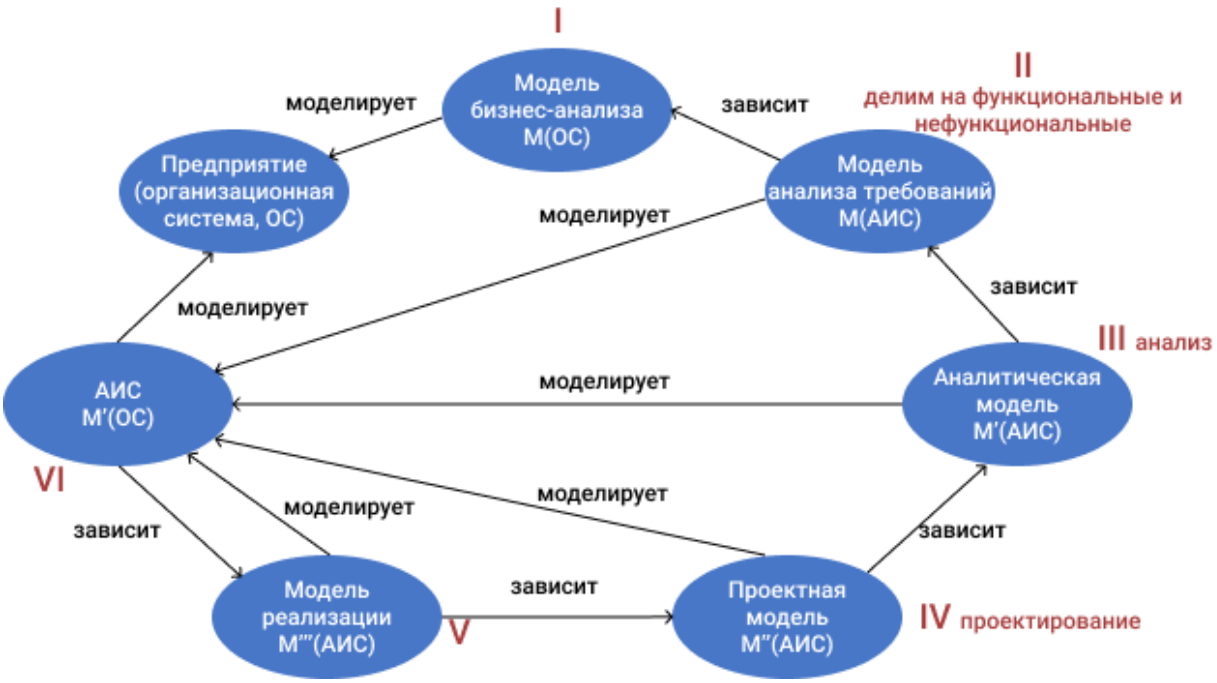

Рисунок 7.1 - Обобщенная "формула" создания АИС

Обобщенная "формула" создания АИС: ОС → М(ОС) → М(АИС) → М'(АИС) → М"(АИС) → М'"(АИС) → АИС.

Анализ организационной системы позволяет создать ее модель М(ОС). Это – модель бизнес-анализа (проблемной области).

Анализируя модель проблемной области, в ней можно выяснить задачи и функции, реализуемые внутри ОС и функции коммуникации ОС и среды (I, рисунок 7.1).

Выделив среди функций те, которые подлежат автоматизации, мы получаем основу для выявления функциональных требований. Остальная собранная на этапе АПО информация служит для поиска нефункциональных требований. В результате получаем модель АТ, как первое приближение модели АИС, М(АИС).

Затем, путем углубленного анализа и проектирования, формируются, соответственно, аналитическая модель М'(АИС), проектная модель М"(АИС) и модель реализации М'"(АИС).

Модель уровня реализации позволяет синтезировать собственно АИС, как совокупность программных, информационных, организационных и др. артефактов.

АИС в свою очередь представляет собой модель организационной системы М'(ОС), замыкая цикл.

Методологии бизнес-анализа

Методологии бизнес анализа можно разделить на три категории по типам моделей:

- модели, преследующие цель анализа и улучшения организационной системы (например, SWOT, VCM, BPR, CPI/TQM/ISO9000, BSC);
- модели общего назначения, такие, как SADT, DFD, IDEF1, IDEF3, IDEF5 и другие;
- модели, специально разработанные для использования при автоматизации (например, ISA, BSP, ARIS, RUP).

CPI (Continuous Process Improvement) – непрерывный процесс усовершенствования. Относится ко всем аспектам деятельности компании; предусматривает постоянный поиск способов улучшения работы организации и использование этих способов для совершенствования продукции компании, создания более благоприятного рабочего климата на предприятии и т. п.

Наиболее развитая модель описания проблемной области предлагается в методологии ARIS.

Архитектура ARIS выделяет в организации следующие подсистемы:

- **Организационная.** Определяет структуру организации - иерархию подразделений, должностей и конкретных лиц, многообразие связей между ними, а также территориальную привязку структурных подразделений.
- **Функциональная.** Определяет функции, выполняемые в организации.
- **Подсистемы входов/выходов.** Определяют потоки используемых и производимых продуктов и услуг.
- **Информационная (подсистема данных).** Описывает получение, распространение и доступ к информации (данным).
- **Подсистема процессов управления.** Определяет логическую последовательность выполнения функций посредством событий и сообщений. Можно сказать, что подсистема управления — это совокупность разнесенных во времени сообщений разного рода.
- **Подсистема целей организации.** Описывает иерархию целей, достигаемых в ходе выполнения того или иного процесса.
- **Подсистема средств производства.** Описывает жизненный цикл основных и вспомогательных средств производства.
- **Подсистема человеческих ресурсов.** Описывает прием на работу, обучение и продвижение по службе персонала организации.

– **Подсистема расположения организационных структур.** Описывает территориальное расположение организационных единиц.

ARIS (Architecture of Integrated Information Systems) — архитектура интегрированных информационных систем. Методология и CASE-средство для моделирования бизнес-процессов организаций с целью их автоматизации от компании Softwareag. Основные черты методологии – поддержка 5 различных, но взаимосвязанных между собой точек зрения на предприятие; наличие сверхмощного, не имеющего аналогов графического языка, насчитывающего десятки различных диаграмм и сотни специальных символов для всевозможных аспектов деятельности предприятий и его автоматизации. Сюда вошли многие наработки из мирового арсенала бизнес-моделирования, инкапсулирован также и UML. Наиболее известная диаграмма ARIS – EPC.

Требования и архитектура АИС

Говоря об архитектуре АИС, обычно подчеркивают деление на аппаратные, программные, информационные, организационные компоненты, их связность и детализацию.

Метафора архитектуры RUP описывается в виде 4+1 представлений: логическое, представление процессов, представление реализации и физическое представление связываются между собой представлением вариантов использования (Use case), которое играет центральную роль в выработке архитектуры системы (рисунок 7.2).

Требования первичны по отношению к архитектуре. Но это не значит, что требования и архитектура должны разрабатываться последовательно.

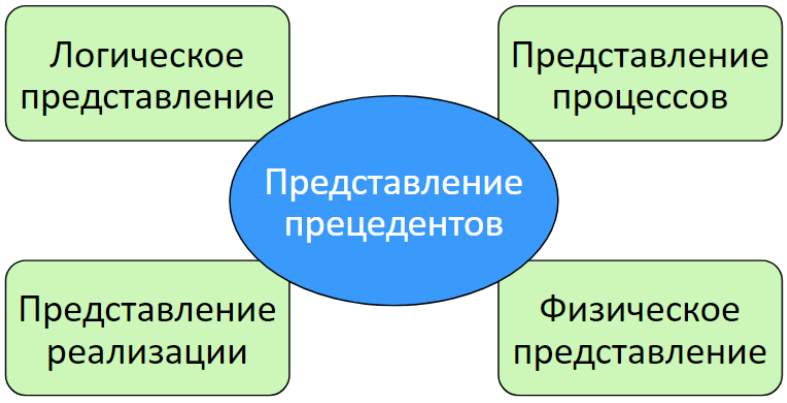

Рисунок 7.2 – Метафора архитектуры RUP

Процессы взаимосвязаны и должны быть существенно параллельны. В крупных, ответственных проектах обычно рассматривается несколько альтернативных архитектур, их достоинства и недостатки применительно к исходным требованиям.

В процессе работы с требованиями они детализируются, детализируется и архитектура. На определенном уровне детализации необходимо остановиться на одной, чтобы не распылять

ресурсы. Но природа требований такова, что, помимо детализации они еще и изменяются. С изменением требований изменяются и детали архитектуры. Устойчивость архитектуры проявляется в незначительных ее изменениях при добавлении, детализации и изменении требований. Если наступил момент, при котором появление новой информации о требованиях перестало оказывать влияние на архитектуру — значит, архитектура стабилизировалась.

Анализ требований и другие рабочие потоки программной инженерии

Обзор рабочих потоков RUP и их связь с потоком работ АТ (рисунок 7.3).

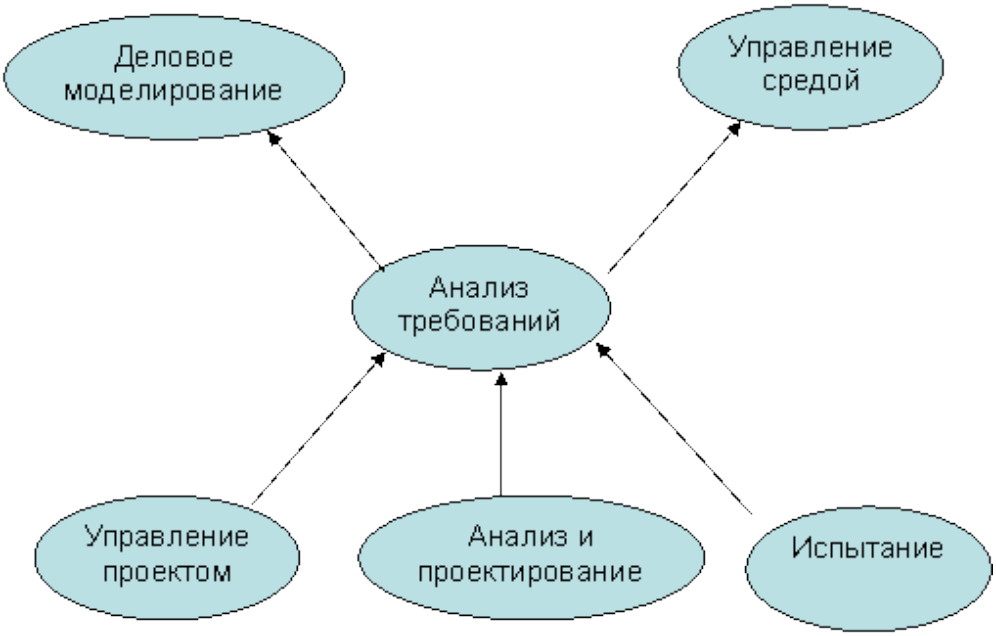

Рисунок 7.3 - Обзор рабочих потоков RUP и их связь с потоком работ АТ

Поток работ "деловое моделирование" служит основой для анализа и формирования требований к АИС, позволяет избежать ошибок.

Поток работ "управление средой" предоставляет исходную информацию для рабочей группы АТ, регламентирующую форматы оформления, CASE-средства, регламенты работы.

Поток работ "управление проектом" основывается на спецификации требований. Стратегическое и тактическое планирование, формирование промежуточных вех (ожидаемых результатов) тесно увязано с требованиями к системе.

Поток работ "анализ и проектирование" осуществляется на основе исходных данных, предоставленных АТ. В определенной мере эти потоки работ проводятся параллельно. При обнаружении проблем, связанных с требованиями, возникает обратная связь от этого потока работ к потоку работ АТ.

Поток работ "испытание" во многом базируется на модели требований и дополнительных спецификациях, регламентирующих процесс тестирования (тестовые сценарии и пр.).

Требования должны анализироваться и учитываться во ВСЕХ рабочих потоках проекта.

8 Лекция № 9

Тема 8: Бизнес аналитик. Роль бизнес-аналитика. Задачи аналитика. Навыки, необходимые аналитику. Знания, необходимые аналитику

Бизнес-аналитик

Роль бизнес-аналитика

Бизнес-аналитик — это основное лицо, отвечающее за выявление, анализ, документирование и проверку требований к проекту. Это основной коммуникативный канал между группой клиентов и командой разработчиков, хотя, конечно, не единственный: есть и другие. Аналитик отвечает за сбор и распространение информации о продукте, а менеджер проекта — за обмен информацией о проекте.

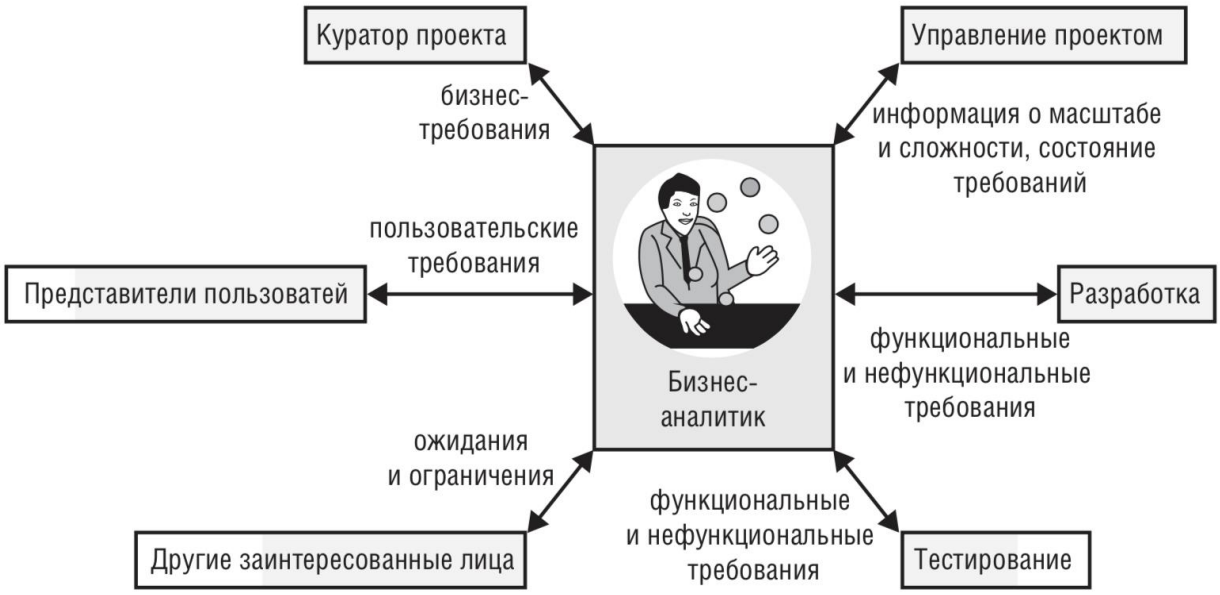

Рисунок 8.1 – Наведение коммуникативных мостов между клиентом и разработчиком

Бизнес-аналитик — это одна из ролей участников проекта, а не обязательно название должности. Их также называют аналитиками по требованиям, системными аналитиками, инженерами по требованиям, менеджерами по требованиям, прикладными аналитиками, аналитиками бизнес-систем, ИТ- аналитиками и просто аналитиками.

В организациях, разрабатывающих потребительские продукты, роль аналитика часто выполняют менеджер продукта или специалисты по маркетингу. В сущности, менеджер продукта действует как бизнес-аналитик, часто с дополнительным ударением на понимании рыночного ландшафта и предугадывании потребностей внешних пользователей. Если в проекте есть и

менеджер продукта, и бизнес-аналитик, первый обычно фокусируется на внешнем рынке и запросах пользователей, а второй преобразовывает эту информацию в функциональные требования.

Задача аналитика

Задача аналитика:

- выяснить, для чего нужна пользователям новая система;
- определить пользовательские требования;
- определить функциональные требования;
- определить качественные требования;
- спроектировать, построить и проверить продукт.

Аналитик — это посредник в общении, проясняющий смутные представления пользователей и обращающий их в четкие спецификации, которыми руководствуется команда разработчиков продукта. В этом разделе описаны некоторые стандартные обязанности аналитика.

Определить бизнес-требования

Работа начинается с помощи куратору, менеджеру продукта или менеджеру по маркетингу с определением бизнес-требований к проекту. Можно разработать шаблон документа о концепции и границах и, расспросив людей, имеющих представление о концепции системы, получить у них необходимую информацию.

Спланировать подход к работе с требованиями

Аналитик должен разработать планы выявления, анализа, документирования, проверки и управления требованиям на всем протяжении проекта и тесно сотрудничать с менеджером проекта над согласованием этих планов с общим планом проекта для достижения поставленных целей.

Определить заинтересованных лиц и классы пользователей

Совместно с кураторами следует выбрать соответствующих представителей каждого класса, заручиться их поддержкой и согласовать обязанности.

Выявить требования

Профессиональный аналитик помогает пользователям четко обрисовать функции системы, необходимые им для достижения бизнес-целей, используя приемы сбора информации.

Анализировать требования

Аналитик должен искать производные требования, логически проистекающие из запросов клиентов, а также невысказанные ожидания, которые, как считают клиенты, и так будут реализованы. Он использует модели требований, чтобы выявить паттерны, пробелы в требованиях, конфликтующие требования и убедиться, что все требования укладываются в границы проекта.

Документировать требования

Аналитик отвечает за создание хорошо организованного и написанного документа требований, который описывает решение, удовлетворяющее потребности клиента.

Применение стандартных шаблонов ускоряет разработку требований, поскольку у аналитика перед глазами будет постоянно находиться перечень тем, которые нужно обсудить с представителями пользователей.

Доводить требования до заинтересованных лиц

Аналитик должен эффективно довести требования до всех участников. Аналитик должен определить, полезно ли представлять требования не текстовыми средствами, например с помощью разнообразных моделей графического анализа, таблиц, математических уравнений, раскадровок и прототипов. Доведение требований — это постоянное взаимодействие с командой, чтобы быть уверенным в правильности понимания доводимой аналитиком информации.

Управлять проверкой требований

Аналитик должен гарантировать, что система, созданная на основе требований, устроит пользователей. Аналитики — ключевые участники рецензирования документов с требованиями. Им также следует изучить архитектуру, код и варианты тестирования, спроектированные на основе спецификаций требований, и убедиться, что требования интерпретированы правильно.

Обеспечить расстановку приоритетов требований

Аналитик обеспечивает общение и взаимодействие различных классов заинтересованных лиц с разработчиками, чтобы расставить приоритеты требований в соответствии с бизнес-целями.

Управлять требованиями

Бизнес-аналитик вовлечен во все этапы разработки ПО, его задача — помочь создать, обсудить и осуществить план управления требованиями к проекту. Определив базовую версию требований для текущего выпуска продукта или итерации разработки, бизнес-аналитик переходит к отслеживанию состояния этих требований, проверкой того, как они реализуются в продукте и управлением изменениями базовых требований.

Навыки, необходимые аналитику

Умение слушать

Чтобы стать специалистом, аналитик должен научиться эффективно слушать. Активное слушание подразумевает устранение помех, сохранение вежливой позы и зрительный контакт, а также повторение основных моментов для закрепления их понимания.

Умение опрашивать и задавать вопросы

Аналитик должен уметь общаться с разными людьми и группами — только так ему удастся выявить их потребности. Необходимо задавать правильные вопросы.

Способность соображать на ходу

Бизнес-аналитики всегда находятся в курсе существующей информации и обрабатывают на ее основе новую информацию. Они должны выявлять противоречия, неясности, неопределенность и допущения и своевременно обсуждать их. Можно создать идеальный список вопросов для интервью, но всегда приходится задавать дополнительные вопросы.

Навыки анализа

Эффективный бизнес-аналитик способен думать на нескольких уровнях абстракции и знать, когда переходить между ними. Иногда требуется перейти от сведений высшего порядка к подробностям. В некоторых случаях на основе потребности одного из пользователей надо обобщить набор требований, которые удовлетворят большинство заинтересованных лиц.

Навыки системного мышления

Ориентируясь на детали, аналитик обязан не упускать общей картины и проверять требования на основе своих знаний о предприятии в целом, бизнес-среде и приложении, чтобы вовремя обнаружить несоответствия и влияние различных компонентов

Навыки обучения

Аналитики должны быстро усваивать новый материал. Аналитик должен отлично владеть языком и ясно выражать сложные идеи. Аналитик же должен уметь читать критично и эффективно.

Навыки создания комфортных условий общения

Умение организовывать дружескую атмосферу на семинарах для уточнения требований — один из необходимых навыков аналитика. Создание доверительных отношений позволяет вести группу к успеху.

Лидерские качества

Сильный аналитик может подталкивать группу заинтересованных лиц в определенном направлении для достижения общей цели. Лидерство подразумевает набор приемов достижения согласия между заинтересованными лицами проекта, разрешения конфликтов и принятия решений.

Умение наблюдать

Наблюдательность помогает направить дискуссию в новое русло, чтобы выявить дополнительные требования, о которых никто не упоминал.

Навыки общения

Основной итог процесса создания требований — письменная спецификация с информацией для клиентов, отдела маркетинга и технического персонала. Аналитик должен отлично владеть языком и ясно выражать сложные идеи как письменной, так и устной форме.

Организационные навыки

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

Навыки моделирования

Аналитик должен уметь работать с разнообразными средствами, начиная с древних блок-схем и структурированных моделей анализа и заканчивая современным языком UML.

Навыки межличностного общения

Аналитик должен уметь организовать людей с разными интересами для совместной работы, и уверенно чувствовать себя в разговорах с сотрудниками, занимающими разные должности в организации. Бизнес-аналитик должен разговаривать на языке целевой аудитории, не прибегая к техническому жаргону при общении с заинтересованными лицами.

Творческий подход

Отличный аналитик творчески подходит к делу: рассказывая о системе, ему удается удивить клиента — тот даже не всегда подозревает, что такая функциональность возможна.

Знания, необходимые аналитику

Помимо специальных навыков и личных качеств, бизнес-аналитик должен обладать обширными знаниями, большая часть которых приходит с опытом.

- понимать современные приемы работы с требованиями;
- знать, как применять современные приемы работы на различных стадиях разработки ПО;
- обучать и убеждать тех, кто не знаком с принятыми приемами работы с требованиями;
- проводить линию разработки и управления требованиями на протяжении всего жизненного цикла проекта;
- защищать проект от проблем, возникающих из-за неправильной работы с требованиями;
- знание принципов управления продуктом;
- базовые знания об архитектуре и эксплуатационных условиях.

Рисунок 8.2 – Основа для формирования эффективного бизнес-аналитика

9 Лекция № 10

Тема 9: Определение бизнес-требований. Источники. Концепция продукта и границы проекта. Противоречивые бизнес-требования. Документ о концепции и границах. Формирование видения в различных моделях разработки проекта.

Формулировка бизнес-требований

Документ о концепции и границ — результирующий документ, содержащий бизнес-требования проекта.

Бизнес-требования относится к информации, которая в совокупности описывает потребность, которая инициирует один или больше проектов, призванных предоставить решение и получить требуемый конечный бизнес-результат. В основе бизнес-требований лежат бизнес-возможности, бизнес-цели, критерии успеха и положение о концепции.

Вопросы бизнес-требований должны решаться до окончательного определения функциональных и нефункциональных требований.

Источники

Бизнес-требования могут исходить от:

- финансирующих проект заказчиков;
- топ-менеджеров;
- менеджеров по маркетингу;
- ответственных за концепцию продукта;
- клиент;
- менеджер по продукту;
- эксперт предметной области.

Бизнес-аналитик должен обеспечить, чтобы бизнес-требования определяли правильные заинтересованные лица и организовать сбор, приоритизацию и разрешение конфликтов.

Концепция продукта и границы проекта

Концепция и границы — два базовых элемента бизнес-требований.

- сжато описать конечный продукт;
- представление продукта сейчас и каким он станет в последствии;
- границы проекта показывают, на какую часть конечной концепции продукта будет направлен текущий проект или итерация.

Противоречивые бизнес-требования

Трения между заинтересованными лицами с разными целями и ограничениями ведут к конфликтующим бизнес-требованиям. Лица, ответственные за принятие решений, должны разрешить эти конфликты до того, как аналитик создаст подробные требования к терминалу. Основное ударение должно быть сделано на предоставлении максимальных бизнес-преимуществ основным заинтересованными лицам.

Бизнес-аналитик должен выявить области возможных конфликтов и различий в предположениях, отметить противоречивые бизнес-цели, отметить функции, не соответствующие этим целям и организовать процесс разрешения конфликтов.

В длительных проектах часто происходит смена ответственных за принятие решений.

- нужно немедленно уточнить базовые требования с новыми ответственными;
- менеджеру проекта потребуется скорректировать бюджет, график и ресурсы;
- бизнес-аналитику может понадобиться поработать с заинтересованными лицами над обновлением пользовательских и функциональных требований и их приоритетов.

Документ о концепции и границах

Собирает бизнес-требования в единый документ, который подготавливает основу для последующей разработки продукта, создают устав проекта или положение о бизнес-задачах.

Владельцем документа о концепции и границах считается куратор проекта, тот, кто финансирует проект, или имеет аналогичную ответственность. Бизнес-аналитик может вместе с этим человеком разрабатывать документ о концепции и границах. На рисунке 9.1 показан шаблон документа о концепции и границах, его нужно изменить в соответствии со спецификой проекта.

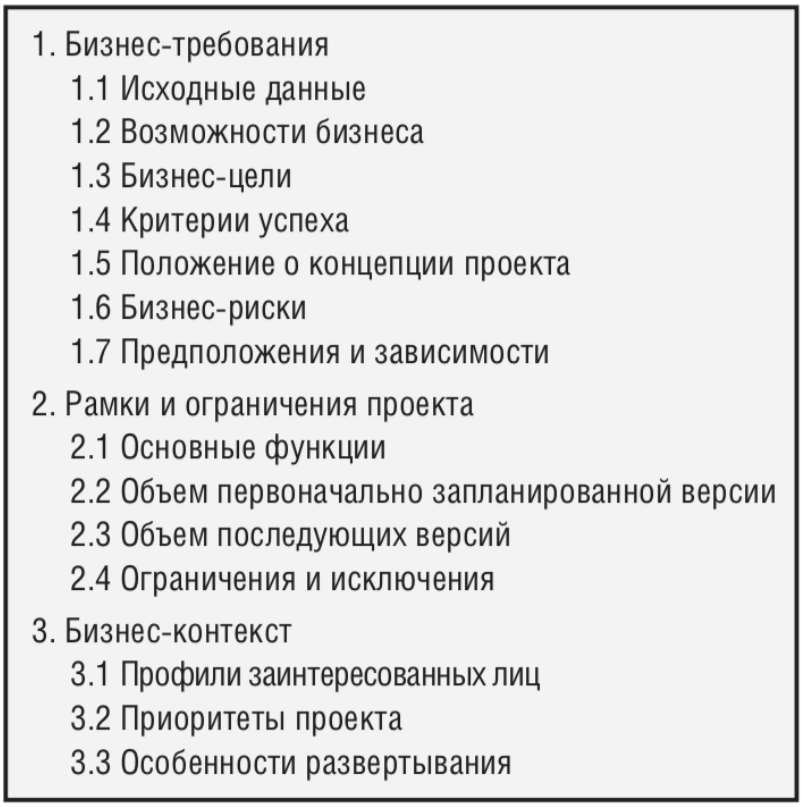

Рисунок 9.1 – Шаблон документа о концепции и границах

Шаблон может помочь вам учесть всю необходимую информацию требований, которую нужно собрать для проекта.

1. Бизнес-требования

Проекты запускаются с полным убеждением, что новый продукт сделает мир для кого-то лучше и обеспечит прибыль. Бизнес-требования описывают основные преимущества, которые новая система даст ее заказчикам, покупателям и пользователям. Бизнес-требования непосредственно влияют на то, какие пользовательские требования будут реализованы и в какой последовательности.

1.1 Исходные данные

Они суммируют обоснование и содержание нового продукта или изменения, которые нужно внести в существующий продукт. Здесь помещают общее описание предыстории или ситуации, в результате чего было принято решение о создании продукта.

1.2 Возможности бизнеса

Для корпоративной информационной системы:

– бизнес-задачи;
– бизнес-процессы;
– среда.

Для коммерческого продукта:

- существующие рыночные возможности;
- рынок, на котором продукту придется конкурировать с другими продуктами.

Возможности бизнеса могут содержать:

- сравнительную оценку существующих продуктов и возможных решений, указывая, в чем заключается привлекательность продукта и его преимущества;
- задачи, которые не удается разрешить без предлагаемого решения;
- демонстрация того насколько оно соответствует тенденциям рынка, развитию технологий или корпоративной стратегии;
- краткое описание других технологии, процессов или ресурсов, необходимых для удовлетворения клиента;
- описание потребности типичных клиентов или целевого рынка;
- представление задачи клиента, которые будет решать новый продукт;
- примеры того, как клиенты будут использовать продукт;
- известные критичные требования к качеству или интерфейсам.

1.3 Бизнес-цели

Суммирует важные преимущества бизнеса, предоставляемые продуктом, в количественном и измеряемом виде. Модель бизнес-целей отображает иерархию связанных бизнес-проблем и измеряемых бизнес-целей. Бизнес-цели определяют способы измерения достижения требуемых ориентиров. Задачи и цели взаимосвязаны: понимание одной раскрывают суть второй. При наличии бизнес-задачи спросите себя: «Как определить, что задача решена?», чтобы определить измеряемую цель.

1.4 Критерии успеха

Необходимо определить, как заинтересованные лица будут определять и измерять успех проекта. Необходимо установить факторы, которые максимально влияют на успех проекта — те, которые организация может контролировать, и те, которые находятся вне сферы ее влияния.

1.5 Положение о концепции

Сжатое положение о концепции проекта, обобщающее долгосрочные цели и назначение нового продукта. В этом документе следует отразить сбалансированную концепцию, удовлетворяющую различные заинтересованные лица.

Шаблон, состоящий из ключевых слов для документа о концепции продукта:

- *Для* [целевая аудитория покупателей];

- *Который* [положение о потребностях или возможностях];
- *Эта* (этот) [имя продукта];
- *Является* [категория продукта];
- *Который (ая)* [основные функции, ключевое преимущество, основная причина для покупки или использования];
- *В отличие от* [основной конкурирующий продукт, текущая система или текущий бизнес-процесс];
- *Наш продукт* [положение об основном отличии и преимуществе нового
- продукта].
- Разработка положения о концепции обеспечит правильное направление оставшейся части проекта.

1.6 Бизнес-риски

В категорию рисков входят:

- рыночная конкуренция;
- временные факторы;
- приемлемость для пользователей;
- проблемы, связанные с реализацией;
- возможные негативные факторы, влияющие на бизнес.

1.7 Предположения и зависимости

Предположение (assumption) — это утверждение, которое предполагается верным в отсутствие знаний или доказательств иного. Бизнес-предположения тесно связаны с бизнес-требованиями. Неверные предположения могут не позволить достичь поставленных бизнес-целей.

2 Рамки и ограничения проекта

Необходимо указать, что может делать система, а что не может. Первый шаг на пути к обузданию распухания границ — определение рамок проекта.

Рамки и ограничения помогают установить реалистичные ожидания заинтересованных лиц. Рамки проекта могут представляться различными способами. На самом высоком уровне границы определяются, когда клиент решает, какие бизнес-цели преследовать. На низком уровне границы определяются на уровне функций, пользовательских историй, вариантов использования или событий и реакции на них. В конечном итоге границы определяются через набор функциональных требований, которые планируется реализовать в определенном выпуске или итерации.

2.1 Основные функции

Необходимо описать основные функции продукта или возможности пользователей, уделив основное внимание тому, что отличает продукт от предыдущей версии или конкурирующих продуктов.

2.2 Объем первоначально запланированной версии

Необходимо сосредоточится на наиболее ценных функциях, имеющих максимально приемлемую стоимость, годных для самой широкой целевой аудитории, которые удастся создать как можно раньше.

2.3 Объем последующих версий

Необходимо создать план выпуска, в котором будет указано, какие функции будут отложены и желательные сроки последующих выпусков.

Чем дальше заглядывать, тем более расплывчатыми будут границы проекта.

2.4 Ограничения и исключения

Перечислить все возможности или характеристики, которых могут ожидать заинтересованные в проекте лица, но включение которых в продукт или в определенную версию не запланировано.

3 Бизнес-контекст

В этом разделе представлены профили основных категорий заинтересованных лиц, приоритеты руководства в проекте, а также сводка некоторых обстоятельств, которые надо учесть при планировании развертывания решения.

3.1 Профили заинтересованных лиц

Заинтересованными в проекте лицами (stakeholders) называются отдельные лица, группы или организации, которые активно вовлечены в проект, на которых влияет результат проекта и которые сами могут влиять на этот результат.

В профиль каждого заинтересованного в проекте лица включается следующая информация:

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

1) повышенная производительность;
2) меньшее количество переделок;

3) снижение себестоимости;
4) ускорение бизнес-процессов;
5) автоматизация задач, ранее выполнявшихся вручную;
6) возможность выполнять совершенно новые задачи;
7) соответствие соответствующим стандартам и правилам;
8) лучшая, по сравнению с текущими продуктами, легкость и простота использования;

b) их вероятное отношение к продукту;
c) самые важные для них функции и характеристики;
d) все известные ограничения, которые должны быть соблюдены.

3.2 Приоритеты проекта

Для принятия эффективных решений, заинтересованные лица должны договориться о приоритетах проекта. Один из подходов к этому заключается в рассмотрении пяти измерений: функции (или объем), качество, график, затраты и кадры. Каждое из этих измерений относится к одной из трех категорий:

- ограничение — сдерживающий фактор, в рамках которого должен оперировать менеджер проекта;
- ведущий фактор — важный фактор успеха, ограниченно гибкий при изменениях;
- степень свободы — возможность для менеджера проекта до определенной степени менять измерение и балансировать относительно других измерений.

10 Лекции № 12 и № 13

Тема 10: Классификация и специфицирование требований. Согласование и документирование требований в различных моделях разработки проекта

Документирование требований

Итог разработки требований — задокументированное соглашение между заинтересованными лицами о создаваемом продукте. В документе об образе и границах проекта содержатся бизнес-требования, а пользовательские требования часто фиксируются в виде вариантов использования продукта и пользовательских историй. Подробные функциональные и нефункциональные требования к продукту записаны в спецификации к требованиям к ПО.

Спецификацию требований к ПО описывается как документ.

Шаблон спецификации требований к ПО — удобное средство напоминания о том, какую информацию надо собрать и как ее организовать. Акт спецификации и моделирования помогает участникам проекта обдумывать и точно формулировать важные вещи, которые в устном обсуждении могут остаться невыясненными.

Требования пишутся для определенной аудитории. Объем подробностей, типы предоставляемой вами информации и способ ее структурирования должны соответствовать потребностям целевой аудитории. Требования должны быть максимально понятными тем, кто их должен понимать и выполнять на их основе свою работу.

Последовательное уточнение деталей — ключевой принцип эффективной разработки требований. Нужно узнать о требованиях достаточно, чтобы можно было в первом приближении определить их приоритеты и назначить на конкретные будущие выпуски и итерации. Не стоит надеяться, что даже самая точная документация по требованиям сможет заменить текущие обсуждения в процессе проекта.

Способы представления требований:

- документация, в которой используется четко структурированный и аккуратно используемый естественный язык;
- графические модели, иллюстрирующие процессы преобразования, состояния системы и их изменения, отношения данных, а также логические потоки и т.п.;
- формальные спецификации, где требования определены с помощью математически точных, формальных логических языков.

Таблицы, рабочие модели, фотографии и математические выражения – самые практичные способы документирования требований в большинстве проектов по разработке ПО.

Спецификация требований к ПО

В спецификации требований к ПО указываются функции и возможности, а также необходимые ограничения. Она должна содержать достаточно подробное описание поведения системы при различных условиях, а также необходимые качества системы: производительность, безопасность и удобство использования. Спецификация требований служит основой для дальнейшего планирования, дизайна и кодирования, а также базой для тестирования пользовательской документации.

Список участников проекта, использующих спецификацию требований к ПО:

- клиенты, отдел маркетинга и специалисты по продажам хотят иметь представление о конечном продукте;
- менеджеры проекта по данным спецификации рассчитывают графики, затраты и ресурсы;
- команда разработчиков ПО получает представление о том, какой продукт надо создавать;
- тестировщики составляют основанные на требованиях тесты, планы тестирования и процедуры;
- специалисты по обслуживанию и поддержке получают представление о функциональности каждой составной части продукта;
- составители документации создают руководства для пользователей и окна справки на основании спецификации требований к ПО и дизайна пользовательского интерфейса;
- специалистам по обучению спецификация требований к ПО и пользовательская документация необходима для разработки обучающих материалов;
- персонал, занимающийся юридической стороной проекта, проверяет, соответствуют ли требования к продукту существующим законам и постановлениям;
- субподрядчики строят свою работу и могут нести юридическую ответственность также согласно спецификации требований к ПО.

Советы, как сделать требования ясными и понятными:

- для структурирования всей необходимой информации используйте соответствующий шаблон;
- разделы, подразделы и отдельные требования должны быть именоваться единообразно;
- используйте средства визуального выделения (такие, как полужирное начертание, подчеркивание, курсив и различные шрифты) последовательно и в разумных пределах; помните, что цветовые выделения могут быть невидны дальтоникам или при черно-белой печати;

- создайте оглавление, чтобы облегчить пользователям поиск необходимой информации;
- пронумеруйте все рисунки и таблицы, озаглавьте их и, ссылаясь на них, используйте присвоенные номера;
- если при хранении требований в документе вы ссылаетесь в документе на другие его части, используйте возможности работы с перекрестными ссылками в вашем редакторе, а не сложную кодировку страниц;
- если вы используете документы, применяйте гиперссылки, чтобы читатель смог быстро перейти к соответствующим разделам спецификации или другим фиалам;
- при хранении требований в специализированном средстве используйте ссылки, чтобы облегчить читателю навигацию по нужной информации;
- включайте графическоё представление информации, где это возможно для облегчения понимания;
- воспользуетесь услугами опытного редактора, чтобы обеспечить последовательность документа и единообразие терминологии и структуры.

Требования к именованию требований

У каждого требования должен быть уникальный и неизменный идентификатор. Уникальная идентификация требований упрощает взаимодействие между членами команды при обсуждении.

Нумерация по порядку

Самый простой способ.

Иерархическая нумерация

Это наиболее распространённый способ. Нумерация может быть весьма длинной.

Иерархические текстовые теги

Иерархические текстовые тэги структурированы, их названия осмысленны и не меняются при добавлении, удалении или перемещении остальных требований. Этот метод также подходит для нумерации бизнес-правил, если они управляются вручную, а не с помощью специализированного инструмента или хранилища бизнес-правил.

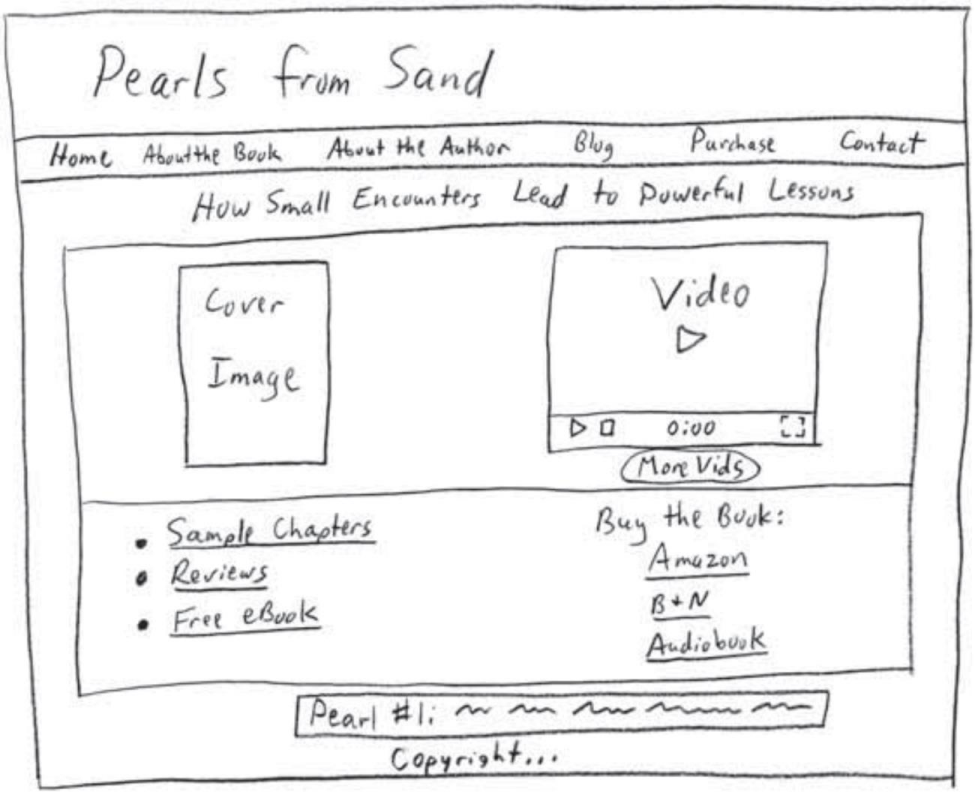

Рисунок 10.1 – Пример наброска пользовательского интерфейса, подходящего для включения в документ требований

Шаблон спецификации требований к ПО

Каждая организация, специализирующаяся на разработке ПО, должна принять один или несколько стандартных шаблонов спецификации требований к ПО для использования в проектах. Доступны различные шаблоны спецификации. При конструировании систем, независимо от типа и размеров проекта для каждого крупного класса необходим отдельный шаблон спецификации. На рисунке 10.2 показан шаблон спецификации требований, который подходит для многих проектов.

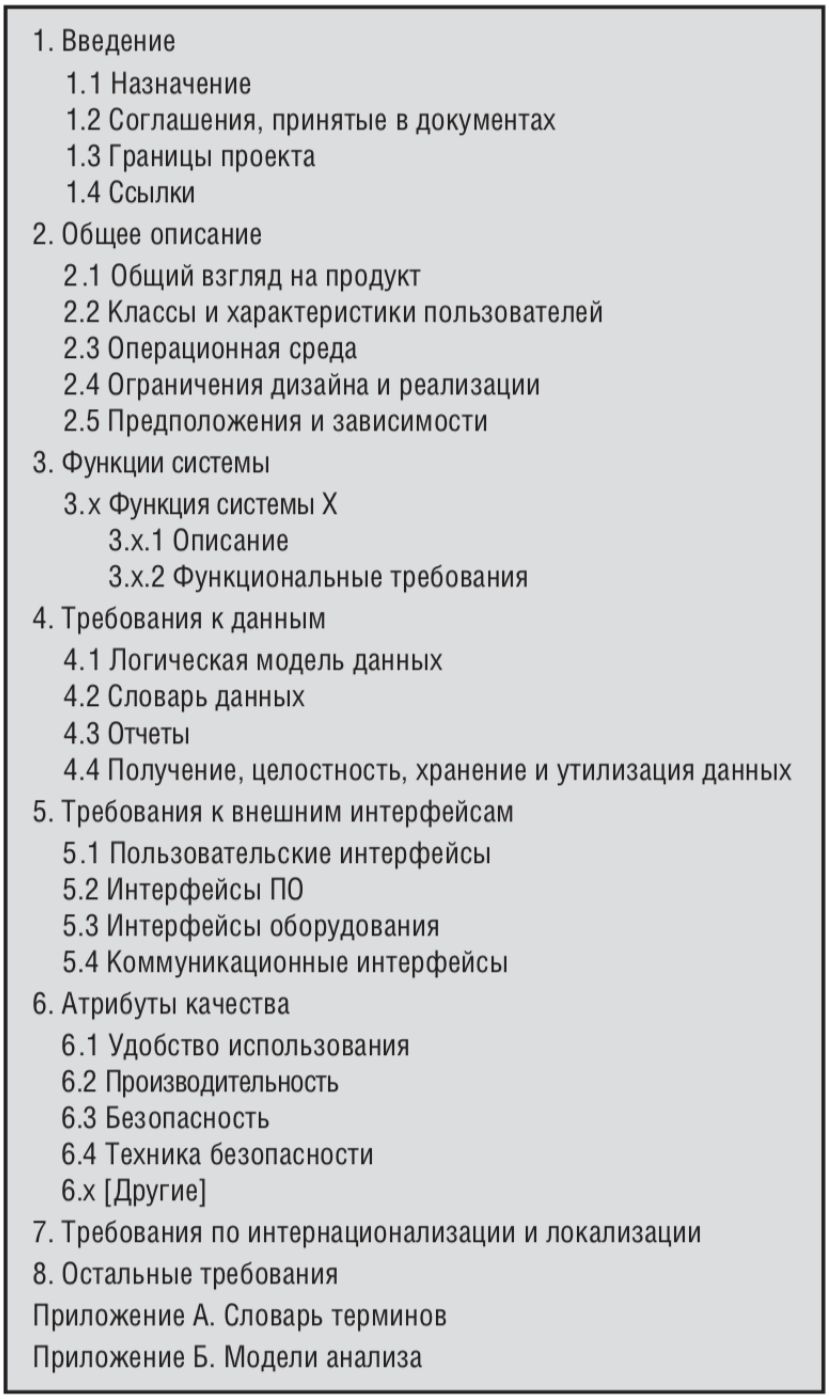

Рисунок 10.2 – Предлагаемый шаблон для спецификации требований к ПО

Иногда фрагмент информации логически подходит для нескольких разделов шаблона. Выберите один раздел и используйте именно его для информации такого типа в своем проекте. Не дублируйте информацию в нескольких разделах, даже если логически она ложится в эти разделы. Используйте перекрестные ссылки и гиперссылки, чтобы облегчить читателям поиск нужной информации.

При создании документов требований применяйте эффективные приемы и средства управления версиями, чтобы все читатели четко понимали, какую версию они читают в тот или

иной момент времени. Ведите журнал изменений, в котором фиксируются суть изменений, автор, дата и причина. В оставшейся части этой главы описывается информация, которая должна присутствовать в отдельных разделах спецификации требований к ПО.

1. Введение

Обзор, помогающий читателям разобраться в структуре и принципе использования спецификации требований к ПО.

1.1 Назначение

Определение продукт или приложение, требования для которого указаны в этом документе, в том числе редакцию или номер выпуска. Если эта спецификация требований к ПО относится только к части системы, идентифицируйте эту часть или подсистему. Описание типов читателей, которым адресован этот документ.

1.2 Соглашения, принятые в документах

Описание всех стандартов или типографических соглашений, включая значение стилей текста, особенности выделения или нотацию. При нумерации требовани2 вручную, нужно определить принятый формат на случай, если кому-нибудь позже понадобиться добавить требование.

1.3 Границы проекта

Краткое описание ПО и его назначения. Связь продукта с пользователями или корпоративными целями, а также с бизнес-целями и стратегиями. Не повторять содержание документа о концепции и границах проекта, если он есть сослаться на него. При постепенной разработке спецификаций требований к ПО, она должна содержать собственное положение о концепции и границах продукта в качестве подраздела долгосрочной стратегической концепции.

1.4 Ссылки

Перечисление всех документов или других ресурсов, на которые есть в этой спецификации, в том числе гиперссылки на них, если их местоположение меняться не будет. Руководства по стилям пользовательского интерфейса, контракты, стандарты, спецификации к системным требованиям, спецификации интерфейса и спецификации требований к ПО связанных продуктов.

2. Общее описание

Общий обзор продукта и среды, в которой он будет применяться, предполагаемая пользовательская аудитория, а также известные ограничения, предположения и зависимости.

2.1 Общий взгляд на продукт

Описание контекста и происхождения продукта. Указать, является он новым членом растущего семейства продуктов, новой версией существующей системы, заменой существующего приложения или совершенно новым продуктом. Если спецификация требований определяет

компонент более крупной системы, указать, как это ПО соотносится со всей системой и определить основные интерфейсы между ними.

2.2 Классы и характеристики пользователей

Необходимо определить различные классы пользователей, которые, как предполагается, будут работать с продуктом, и описать их соответствующие характеристики. Определить привилегированные классы пользователей. Классы пользователей представляют подмножество заинтересованных в проекте лиц, их описание приводится в документе концепции и границ проекта. Описания классов пользователей являются повторно используемым ресурсом. Если есть главный каталог классов пользователей, можно включить описания классов пользователей, просто указав их в каталоге, не дублируя информацию.

2.3 Операционная среда

Описание рабочей среды, в которой будет работать ПО: аппаратную платформу, операционные системы и их версии, географическое местоположение пользователей, серверов и баз данных вместе с организациями, в которых располагаются соответствующие базы данных, серверы и веб-сайты. Перечислить все остальные компоненты ПО или приложения, с которыми система должна быть совместима.

2.4 Ограничения дизайна и реализации

Описание всех факторов, которые ограничат возможности, доступные разработчикам, и логическое обоснование каждого положения. Требования, которые включают или написаны в форме идей по решению, а не потребностей накладывают ограничения на дизайн, часто неоправданные, поэтому за этим надо следить.

2.5 Предположения и зависимости

Предположение (assumption) — это утверждение, которое предполагается верным в отсутствие знаний или доказательств иного. Проблемы возможны в том случае, если предположение неверны, устарели, не находятся в совместном использовании или изменяются, поэтому определенные предположения можно отнести к группе рисков проекта.

Определить все зависимости проекта или создаваемой системы от внешних факторов или компонентов вне ее контроля.

3. Функции системы

Шаблон на рисунке 10.2 структурирован по функциям системы — это еще один способ систематизации функциональных требований.

Методы классификации по:

- функциональным областям;
- рабочим потокам;
- вариантам использования;

– режимам работы;

– классам пользователей;

– стимулам;

– реакциям.

Не существует единственно правильного метода организации; выбирается тот, при котором пользователям будет легче понять предполагаемые возможности продукта.

3.x Функция системы X

Описание названия особенности несколькими словами, например «3.1 Проверка правописания». Так же называются подразделы с 3.x.1 по 3.x.3 для каждой функции системы.

3.x.1 Описание

Краткое описание функции системы и указание ее приоритета: высокий, средний или низкий. Приоритеты являются динамичной характеристикой, они могут изменяться в ходе проекта.

3.x.2 Функциональные требования

Перечисление по пунктам конкретных функциональных требований, которые связаны с этой функцией. Именно эти функции ПО нужно реализовать, чтобы пользователь мог использовать сервисы этой функции или реализовать вариант использования. Описание, как продукт должен реагировать на ожидаемые ошибки, неправильный ввод информации или неверные действия.

4. Требования к данным

Описание различных аспектов данных, которые будет потреблять система в качестве входной информации, как-то обрабатывать и возвращать в виде выходной информации.

4.1 Логическая модель данных

Модель данных — это визуальное представление объектов и наборов данных, которые будет обрабатывать система, а также отношений между ними. Существует много видов нотации для моделирования данных, в том числе диаграммы отношений «сущность–связь» и диаграммы классов UML. Можно включить модель данных для бизнес-операций, выполняемых системой или логическое представление данных, с которыми будет работать система. Это не то же самое, что модель данных реализации, которая реализуется в виде дизайна базы данных.

4.2 Словарь данных

Словарь данных определяет состав структур данных, а также их значение, тип данных, длину, формат и разрешенные значения элементов данных, из которых состоят эти структуры. Серийные средства моделирования данных часто включают компонент-словарь данных. Во многих случаях словарь данных лучше хранить как отдельный артефакт, не внедряя его в

спецификацию требований к ПО. Это повышает возможности повторного использования в других проектах.

4.3 Отчёты

Перечисление отчётов, если приложение будет их генерировать и описание их характеристик. Если отчет должен соответствовать определенному готовому макету, можно указать это как ограничение.

4.4 Получение, целостность, хранение и утилизация данных

Описание, как получают и обслуживают данные. Указание всех требований, относящихся к защите целостности данных системы. Указание всех процедур, которые могут потребоваться: резервное копирование, создание контрольных точек, зеркальное отображение или проверка корректности данных. Указание политик, которые должна применять система для хранения или утилизации данных.

5. Требования к внешним интерфейсам

Указывается информация, которая гарантирует, что система будет правильно взаимодействовать с пользователями и компонентам внешнего оборудования и ПО. В сложной системе с множеством подкомпонентов следует использовать раздельные спецификации для интерфейсов или спецификацию системной архитектуры. В документацию по интерфейсу можно включить ссылки на материал из других документов.

5.1 Пользовательские интерфейсы

Описание логических характеристик каждого пользовательского интерфейса, который необходим системе. Некоторые особенные характеристики пользовательских интерфейсов могут упоминаться в разделе «6.1 Удобство использования».

Некоторые из них перечислены здесь:

- ссылки на стандарты графического интерфейса пользователей или стилевые рекомендации для семейства продуктов, которые необходимо соблюдать;
- стандарты шрифтов, значков, названий кнопок, изображений, цветовых схем, последовательностей полей вкладок, часто используемых элементов управления, графики фирменного стиля, уведомления о зарегистрированных товарных знаках и о конфиденциальности и т.п.;
- размер и конфигурация экрана или ограничения разрешения;
- стандартные кнопки, функции или ссылки перемещения, одинаковые для всех экранов, например кнопка справки;
- сочетания клавиш;
- стандарты отображения и текста сообщений;

– стандарты проверки данных (ограничения на вводимые значения и когда нужно проверять содержимое полей);
– стандарты конфигурации интерфейса для упрощения локализации ПО;
– специальные возможности для пользователей с проблемами со зрением, различением цвета и другими ограничениями.

5.2 Интерфейсы ПО

Описание связи продукта и других компонентов ПО, в том числе другие приложения, базы данных, операционные системы, средства, библиотеки, веб-сайты и интегрированные серийные компоненты. Необходимо указать назначение, форматы и содержимое сообщений, данных и контрольных значений, обмен которыми происходит между компонентами ПО. Описание соответствия между входными и выходными данными между системами и все преобразования, которые должны происходить с данными при перемещении между системами. Описание служб, необходимых внешним компонентам ПО, и природу взаимодействия между компонентами. Определение данных, которыми будут обмениваться и к которым будут иметь общий доступ компоненты ПО. Определение нефункциональных требований, влияющих на интерфейс, такие как уровни обслуживания для времени и частоты отклика или меры и ограничения безопасности.

5.3 Интерфейсы оборудования

Описание характеристики каждого интерфейса между компонентами ПО и оборудования системы. В описание могут входить типы поддерживаемых устройств, взаимодействия данных и элементов управлений между ПО и оборудованием, а также протоколы взаимодействия, которые будут использоваться. Перечисление входных и выходных данные, их формат, разрешенные значения или их диапазоны, а также все временные характеристики, о которых должны знать разработчики.

5.4 Коммуникационные интерфейсы

Необходимо указать требования для любых функций взаимодействия, которые будут использоваться продуктом, включая электронную почту, веб-браузер, сетевые протоколы и электронные формы. Необходимо определить соответствующие форматы сообщений. Описание особенностей безопасности взаимодействия или шифрования, скорости передачи данных и механизмов согласования и синхронизации. Необходимо указать все ограничения этих интерфейсов.

6. Атрибуты качества

В этом разделе описываются нефункциональные требования помимо ограничений и требований к внешним интерфейсам. Эти характеристики должны быть точно определены и поддаваться проверке и измерению. Необходимо указать относительные приоритеты различных

атрибутов, например приоритет простоты использования над легкостью изучения или приоритет безопасности над производительностью.

6.1 Удобство использования

Требования к удобству использования подразумевают легкость изучения, простоту использования, предотвращение ошибок и восстановление, эффективности взаимодействия и специальные возможности. Указанные в этом разделе требования к удобству использования помогут дизайнеру интерфейсов создать максимально удобную для пользователя рабочую среду.

6.2 Производительность

Необходимо указать конкретные требования к производительности для различных системных операций. Если у различных функциональных требований или функций имеются разные требования к производительности, то следует указывать задачи, связанные с производительностью, там же, в разделе соответствующих функциональных требований, а не включать их все в этот раздел.

6.3 Безопасность

Необходимо указать все требования, касающиеся безопасности или конфиденциальности, которые ограничивают доступ или возможности использования продукта. Источником требований к безопасности являются бизнес-правила, поэтому необходимо определить политики или положения, касающиеся защиты или конфиденциальности, которым продукт должен соответствовать.

6.4 Техника безопасности

В этом разделе необходимо указать требования, связанные с возможными потерями, повреждениями или ущербом, которые могут быть результатом использования продукта. Определение мер безопасности или упреждающих действий, которые можно предпринять, так же как и потенциально опасные действия, которые можно предотвратить. Определение сертификатов по безопасности, политики или положения, которым продукт должен соответствовать.

6.x [Другие]

Создание в спецификации требований к ПО отдельный раздел для каждого дополнительного атрибута качества продукта, чтобы описать характеристики, которые будут важны для клиентов или для разработчиков и людей, ответственных за поддержку. Это может быть доступность, возможность установки, целостность, возможность модификации, переносимость, надежность, устойчивость, масштабируемость и контролируемость

7. Требования по интернационализации и локализации

Требования по интернационализации и локализации обеспечивают возможность использовать продукт в других странах, региональных стандартах и географических районах, отличающихся от тех, в которых он был создан. Такие требования могут быть направлены на

разрешение различий в валютах, форматировании дат, чисел, адресов и телефонных номеров, языках, в том числе различных вариантах одного, используемых символах и наборах символов, личных именах и фамилиях, часовых поясах, международных нормативных актах и законах, культурных и политических традициях, размере используемой бумаги, единицах веса и меры, электрическом напряжении и конфигурации электрических разъемов и во многом другом. Требования по интернационализации и локализации вполне могут повторно использоваться во многих проектах.

8. [Остальные требования]

Определение всех других требований, которые еще не были описаны в спецификации требований к ПО. Примером могут служить юридические, законодательные или финансовые требования и требования стандартов, требования к установке, конфигурированию, запуску и остановке продукта, а также к журналированию, мониторингу и контрольному следую. Вместо того чтобы сливать всю эту информацию в один раздел, лучше добавить любые новые разделы к шаблону проекта

Приложение А. Словарь терминов

Определение всех специальных терминов, которые читателю необходимо знать для правильного понимания спецификации требований к ПО, включая сокращения и аббревиатуры. Расшифровка каждого сокращения и приведение его определения. Создании расширенного общекорпоративного словаря для нескольких проектов, который включает по ссылке все термины, относящиеся к данному проекту. В этом случае в спецификации требований к ПО будут определены только те термины, которые относятся лишь к данному проекту и которых нет в общекорпоративном словаре.

Приложение Б. Модели анализа

В этом необязательном разделе описывается, а точнее напоминается, о таких моделях анализа, как диаграммы потоков данных, деревья функций, диаграммы переходов состояния и диаграммы «сущность-связь».

11 Лекция № 13

Тема 11: Расширенный анализ требований. Моделирование

В этой лекции будет идти разговор о моделировании анализа требований. Будут рассмотрены подробно диаграммы UML, поясняющие функциональность системы и внутреннее устройство системы, а также альтернативные языки моделирования.

Какие модели использовать

Вербальные описания вариантов использования системына сегодня являются стандартом де-факто для формулировки соглашения между Заказчиком и Исполнителем. Если обе стороны готовы выделить достаточное количество времени на внимательный и всесторонний *анализ* требований к системе и на начальной фазе ее создания сформулировали 80% всех возможных сценариев использования системы на понятном сторонами языке - значит, ключевые риски создания системы - риск различного понимания проблемы и риск размытия границ во многом преодолены.

Однако, далеко не всякий Заказчик готов скрупулезно обсуждать скучные тома описания вариантов использования, которые даже для систем среднего размера могут достигать сотни страниц.

Чтобы облегчить процесс формулировки и понимания требований для Заказчика, существует ряд приемов:

Во-первых, требования можно формулировать на разных уровнях абстракции. Так, уровень описания требований, поддерживаемый в документе "Видение", является достаточно сбалансированным. То же можно сказать и про краткие (в один абзац) описания ключевой функциональности системы. Действуя таким образом, мы, очевидно, решим проблему вовлечения Заказчика в *анализ задач,* однако указанные выше риски будут снижены недостаточно: концептуальные описания функциональности оставляют много свободы для толкования в ту или иную сторону.

Хорошим подспорьем в решении задачи является **применение визуальных средств** описания требований. Как известно, у большинства людей правополушарное (образное) *мышление* позволяет воспринимать информацию в разы и порядки более ускоренном темпе, чем левополушарное (вербальное).

На сегодня в арсенале аналитика существуют десятки методик, языков, визуальных представлений, позволяющих моделировать требования к системе. При создании информационных систем стандартом де-факто является универсальный *язык моделирования, UML*.

Как уже отмечалось в лекции, посвященной анализу контекста АТ, процесс анализа требований тесно связан, с одной стороны, с анализом проблемной области, с другой - с архитектурным анализом и проектированием. Часто на практике бывает трудно вычленить границы компетенций этих потоков *работ*. Так, модель *анализа потоков данных,* широко используемая в анализе проблемной области, упоминается многими авторами, как модель, полезная в анализе требований. Ряд исследователей считает целесообразным иллюстрировать описания требований диаграммами классов, хотя, строго говоря, выделение классов относится не к анализу требований, а к архитектурному анализу.

Как определить целесообразность использования тех или иных приемов, методик, языков моделирования при анализе требований? Здесь можно предложить **три практические рекомендации**.

Во-первых, *анализ* требований призван изучать взаимодействия *автоматизированной информационной системы* и ее среды, т.е. пользователей, сетевых и системных *компонент,* находящихся вне системы. Следовательно, *бизнес-модели,* описывающие взаимодействия между компонентами организационной системы, строго говоря, можно рассматривать лишь как "сырье" для *извлечения требований,* но не как модели, описывающие требования.

Во-вторых, *анализ* требований должен находить ответ на то, ЧТО делает система, абстрагируясь от деталей реализации, т.е. того, КАК она это делает. Поэтому, допустим, *диаграмму взаимодействия* объектов, реализующих тот или иной *вариант использования,* можно рассматривать скорее, как иллюстрацию внутреннего устройства системы, полезную для программиста, чем модель для заказчика.

Однако, наиболее важным является третье соображение, в чем-то "оппозиционное" по отношению к первым двум. Для моделирования анализа требований следует применять модели, наиболее адекватно проясняющие функциональность системы и особенности ее использования. Однако, *аналитик* волен выбирать те языки и методики, которые позволят добиться целевой функции: снижения рисков непонимания между Исполнителем и Заказчиком и размытия границ. Поэтому, иллюстрируя варианты использования, начинайте с "канонических" способов, которые

будут рассмотрены чуть ниже, но, если посчитаете целесообразным отклониться от них - экспериментируйте смело.

Модели UML, поясняющие функциональность системы

Диаграмма вариантов использования

Диаграмма вариантов использования UML, *Use case* Diagram - одно из самых простых представлений системы. Ее базовые "строительные элементы" - актёры 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 актёром и вариантом использования.

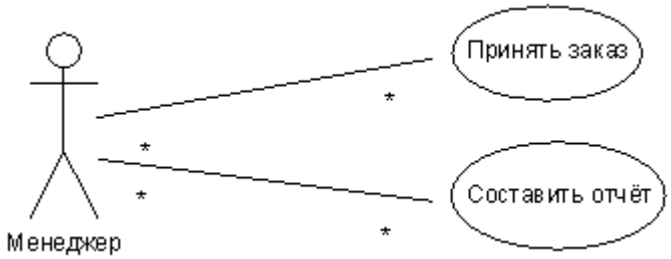

Рисунок 9.1

Другие виды отношений:

- отношение включения (include);
- расширения (extend)
- обобщения/генерализации.

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

Рисунок 9.2

Расширение в точности соответствует точке расширения, используемой при описании варианта использования, см. "Классификация и специфицирование требований" .

Рисунок 9.3

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

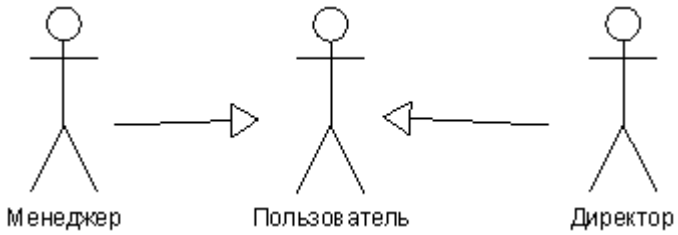

Рисунок 9.4

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.

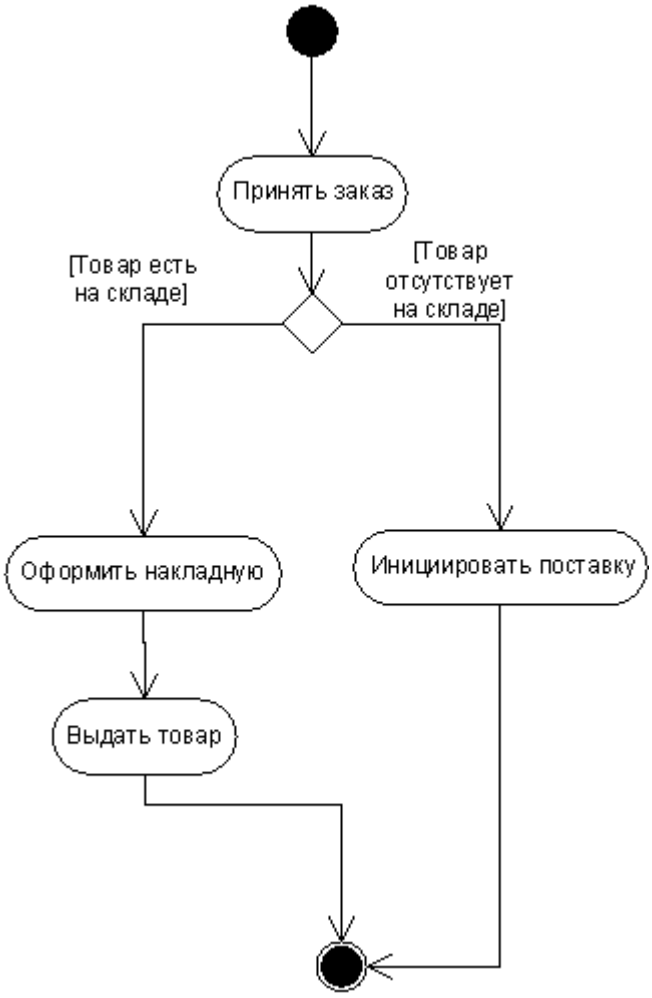

Рисунок 9.5

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.

Помимо стандартного формата описания, UML предлагает вариант с "плавательными дорожками". Этот формат удобен для описания случая, когда в варианте использования участвуют несколько актёров.

Диаграмма состояний

Диаграмма состояний в анализе требований используется, когда требуется исследовать поведение системы, как конечного автомата. Это представление пришло в UML из теории систем.

В общем случае диаграмма состояний описывает, как система себя ведет в более, чем одном варианте использования. Синтаксис диаграмм состояний во многом совпадает с синтаксисом диаграмм действий.

Основные компоненты описания системы:

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

В языке UML под состоянием понимается абстрактный *метакласс*, используемый для моделирования отдельной ситуации, в течение которой имеет место выполнение некоторого условия. Состояние может быть задано в виде набора конкретных значений *атрибутов класса* или объекта, при этом изменение их отдельных значений будет отражать изменение состояния моделируемого класса или объекта.

Рисунок 9.6

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. Полный синтаксис описания перехода (надписи на стрелке) следующий:

Событие [сторожевое условие] / выражение действия

Иногда бывает полезным объединить часть состояний в одно мета-состояние. Графически это выглядит, как символ состояния (прямоугольник со скругленными углами), содержащий внутри себя несколько символов состояний. При этом возможны переходы между подчиненными состояниями, переходы между подчиненным и внешним состояниями и переходы между составным и внешним состоянием.

Диаграммы UML, поясняющие внутреннее устройство системы

Некоторые авторы рекомендуют использовать при описании требований диаграммы *UML*, описывающие создаваемую систему через ее компоненты (классы, объекты), отношения и взаимодействия между ними. Данный подход имеет свои ограничения (см. Принцип 2).

Диаграмма классов

Для создания диаграммы классов необходимо:

1. Осуществить поиск классов (ключевых компонент проблемной области).
2. Для каждого найденного класса определить его имя, основные атрибуты, операции и (или) ответственности.
3. 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.

Принято выделять 3 уровня абстракции классов:

– концептуальный уровень,
– уровень спецификации,

– уровень реализации.

Анализ требований разумно сопровождать диаграммами, отражающими концептуальный уровень. На данном уровне при описании классов целесообразно указать их наименования и ответственности. Атрибуты и операции можно не указывать, либо ввести только самые основные, отложив их исследование на более поздние стадии детализации.

Отношения, подлежащие анализу на концептуальном уровне — это:

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

Рисунок 9.7

Диаграмма классов показывает статическую структуру проблемной области. Для анализа взаимодействия объектов - экземпляров класса в ходе реализации варианта использования в UML предусмотрены две диаграммы взаимодействия: диаграмма кооперации и диаграмма последовательности.

Если диаграмма классов в ряде случаев и может рассматриваться, как артефакт, поясняющий структуру проблемной области, то диаграммы взаимодействия вряд ли стоит рассматривать в качестве выразительного языкового средства, иллюстрирующего требования к системе в диалоге "Заказчик-Исполнитель". Тем не менее, в соответствии с Принципом 3 свободы выбора языковых средств, аналитик решает использовать данные диаграммы.

Альтернативные языки моделирования

Диаграмма потоков данных

Диаграмма потоков данных (data flow diagram, DFD) - один из основных инструментов структурного анализа и проектирования информационных систем, существовавших в "доюмээльную" эпоху. Несмотря на имеющее место в современных условиях смещение акцентов от структурного к объектно-ориентированному подходу к анализу и проектированию систем, "старинные" структурные нотации по-прежнему широко и эффективно используются как в бизнес-анализе, так и в анализе информационных систем.

DFD (Data Flow Diagram) – диаграмма потоков данных [41]. Методология графического структурного анализа, описывающая взаимосвязи функций системы с потоками и хранилищами данных, а также с внешними по отношению к системе источниками, к которым осуществляется доступ. Диаграмма DFD – один из основных инструментов структурного анализа и проектирования информационных систем. Авторы SWEBOK 3.0 http://www.computer.org/portal/web/swebok/swebokv3 отмечают полезность DFD при анализе безопасности информационных систем, поскольку в нем удобно проводить идентификацию возможных путей атак и раскрытия конфиденциальной информации.

Исторически сложилось так, что для описания диаграмм DFD используются две нотации - Йодана (Yourdon) и Гейна-Сарсона (Gane-Sarson), отличающиеся синтаксисом. На приведенной ниже иллюстрации использована нотация Гейна-Сарсона.

Рисунок 9.8

Информационная система принимает извне потоки данных. Для обозначения элементов среды функционирования системы используется понятие внешней сущности. Внутри системы существуют процессы преобразования информации, порождающие новые потоки данных. Потоки данных могут поступать на вход к другим процессам, помещаться (и извлекаться) в накопители данных, передаваться к внешним сущностям.

Модель DFD, как и большинство других структурных моделей - иерархическая модель. Каждый процесс может быть подвергнут декомпозиции, т.е. разбиению на структурные составляющие, отношения между которыми в той же нотации могут быть показаны на отдельной диаграмме. Когда достигнута требуемая глубина декомпозиции - процесс нижнего уровня сопровождается мини-спецификацией (текстовым описанием).

Кроме того, нотация DFD поддерживает понятие подсистемы - структурной компоненты разрабатываемой системы.

Нотация DFD - удобное средство для формирования контекстной диаграммы, т.е. диаграммы, показывающей разрабатываемую АИС в коммуникации с внешней средой. Это - диаграмма верхнего уровня в иерархии диаграмм DFD. Ее назначение - ограничить рамки системы, определить, где заканчивается разрабатываемая система и начинается среда. Другие нотации, часто используемые при формировании контекстной диаграммы - диаграмма *SADT* [1], диаграмма *Диаграмма вариантов использования*.

Другие виды моделей

Среди многообразия моделей, использующихся в анализе систем, хочется особо отметить еще две нотации, позволяющие описать сложные многоальтернативные взаимодействия компонент информационной системы - нотацию IDEF3 и EPC-диаграмму *ARIS*.

EPC Event-Driven Process Chain - цепочка функций, направляемая событиями [24]. Событийно-ориентированная графическая диаграмма функций; используется в методологии *ARIS*. Позволяет строить сложные комбинации вызовов функций на основе генерируемых ими событий с использованием логических операторов. Содержит широкие иллюстративные возможности для описания контекста каждой функции (например – источники данных, оргтехника, пользователи и т.д.).

Для моделирования требований к системам с разветвленной логикой К.Вигерс рекомендует использовать таблицы и деревья решений. Часто на практике бывают полезны диаграммы сущность-связь и SADT-диаграммы

12 Лекция № 14

Тема 12: Расширенный анализ требований. Иллюстрированные сценарии и прототипы

Прототипирование

Основные цели, требующие применения:

- прояснить неясные требования к системе;
- выбрать одно из различных концептуальных решений;
- проанализировать осуществимость.

Неясные требования

Часто Заказчику бывает трудно сформулировать требования к тому, что он ожидает от системы. В этом случае прототип интерфейса пользователя, оперативно созданный по результатам интервью, дает ему возможность увидеть схематичную реализацию того, как Исполнитель увидел соответствующую часть системы.

Разные варианты решения

Любую техническую задачу можно решить различными способами. Это касается как задачи формулировки требований, так и ее реализации в UI.

После реализации прототипов UI по разным сценариям Заказчик, оценив их достоинства и недостатки и в диалоге с Разработчиком формулирует другой, комбинированный сценарий, сочетающий достоинства предыдущих.

Анализ осуществимости

Часто бывает так, что комбинация функциональных, нефункциональных требований и ограничений такова, что возникает риск невозможности их реализации. Как правило, такой риск связан с требованиями к быстродействию системы при известных ограничениях среды ее реализации. В этом случае создаются прототипы, реализующие соответствующую часть системы, имитирующие потоки данных, поступающие на ее вход и их обработку.

Классификация прототипов

Классификации прототипов:

- горизонтальные и вертикальные
- одноразовые и эволюционные
- бумажные и электронные, раскадровки

Горизонтальный прототип

Горизонтальный или поведенческий прототип моделирует интерфейс пользователя приложения, не затрагивая логику обработки и базу данных.

Если в разработанном интерфейсе используются фрагменты базы данных - они имитируются в программном коде. При этом тексты, отображаемые на экране, должны отражать реальную специфику проблемной области, иначе пользователю будет трудно сосредоточиться. Если при нажатии на элемент управления должны производиться какие-то расчеты или запросы во внешние системы - результаты запросов также имитируются. Желательно реализовать ту часть кода, которая отвечает за перемещение между экранами в *процессе исполнения* вариантов использования, чтобы пользователь смог понять, как будет действовать система в ответ на его действия.

Горизонтальные прототипы следует использовать для достижения цели прояснения неясных, либо многоальтернативных требований.

Вертикальный прототип

Вертикальный или структурный прототип не ограничивается интерфейсом пользователя. Он реализует вертикальный "срез" системы, затрагивая все уровни ее реализации. При создании такого рода прототипов рекомендуется использовать те языки и среды реализации, что и при изготовлении целевой системы.

Основные цели применения такого рода прототипов - анализ применимости, проверка архитектурных концепций.

Одноразовый прототип

Одноразовый или исследовательский прототип создается, когда нужно быстро макетировать те или иные аспекты и компоненты системы.

Целям создания исследовательских прототипов служит технология *RAD* (*rapid application development*) - быстрая разработка приложений.

Одноразовый прототип должен создаваться быстро. При его разработке не следует уделять внимание вопросам повторного использования кода, качества, быстродействия, технологичности и т.п.

В результате получается "сырой" код, который может содержать значительное количество дефектов. Необходимо принять меры к тому, чтобы фрагменты кода, реализующие такого рода прототипы, не стали частью целевой системы.

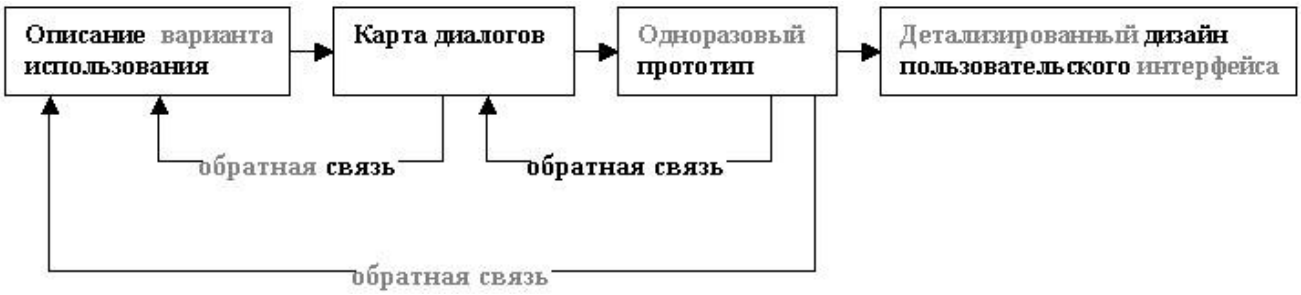

Рисунок 12.1 – Схема перехода от одноразового прототипа к детально проработанному UI

Эволюционный прототип

Эволюционный прототип создается, как первое приближение системы, призванное стать впоследствии самой системой.

Код эволюционного прототипа должен последовательно, в течении одной или более итераций, перерасти в код целевого приложения. Поэтому данный вид прототипов требует всего того, от чего следует отказаться при создании одноразовых прототипов: скрупулезной разработки, применения технологических методов и приемов, тестирования результатов и т.п.

Таблица 12.1 – Отношение между рассмотренными выше 4 видами прототипов

	Одноразовые прототипы	**Эволюционные прототипы**
Горизонтальные	– Прояснение и уточнение примеров использования и функциональных требований. – Выявление пропущенных требований. – Исследование возможных вариантов интерфейса пользователя.	– Реализация базовых вариантов использования. – Реализация дополнительных вариантов использования по приоритетам. – Реализация и доработка web-сайтов. – Адаптация системы к быстро меняющимся требованиям бизнеса.
Вертикальные	– Демонстрация технической осуществимости.	– Реализация и наращивание ключевой клиент-серверной функциональности и уровней коммуникации. – Реализация и оптимизация основных алгоритмов. – Тестирование и настройка производительности.

Бумажный прототип

Бумажный прототип – альтернатива рассмотренным разновидностям электронных прототипов в случае, когда Разработчик ограничен в ресурсах. Наброски интерфейсов на бумаге, конечно, не заменят интерфейс, созданный в среде разработки. Однако, при всех недостатках, у таких прототипов есть два существенных достоинства:

- Заказчик не станет акцентировать внимание на цветовом решении, форме кнопок и т.п., отвлекаясь от анализа функциональности.
- Заказчик никогда не скажет, глядя на бумажный интерфейс: "Да вы, я вижу, уже создали систему на 85%! Давайте закончим ее в течении недели".

Раскадровка

Типы раскадровки:

- пассивная раскадровка;
- активная раскадровка;
- интерактивная раскадровка.

Решением промежуточного между электронным и бумажным вариантами прототипов UI класса являются презентации, изготовленные при помощи средств электронного офиса. В этом случае пользователь лишен свободы выбора, предоставляемой ему поведенческим прототипом. Но идею пошаговой смены экранов в процессе реализации сценария варианта использования вполне можно реализовать. Данный вид решения определяется, как пассивная раскадровка. Активная раскадровка является дальнейшим развитием понятия пассивной раскадровки, с применением средств анимации и т.п. Третий вид раскадровки – интерактивная, представляет собой электронный одноразовый горизонтальный прототип.

13 Лекция № 15

Тема 13: Проверка требований

Верификация и валидация

Термин "*верификация*" (*verification*) в русскоязычной литературе обычно переводят, как "проверка". Термин "*валидация*" - как "проверка правильности", "*аттестация*", "утверждение".

Согласно стандарту IEEE 1012-1986, *верификация* представляет собой процесс оценивания системы или компонента с целью определить, удовлетворяют ли результаты некой фазы условиям, наложенным в начале данной фазы. *Валидация* в этом же стандарте определяется, как процесс оценивания системы или компонента во время или *по* окончании процесса разработки с целью определить, удовлетворяет ли она указанным требованиям.

Трудно ожидать от читателя, впервые столкнувшегося с этой терминологией и ее определениями в этом и других стандартах, ясности понимания. *По* крайней мере, у автора настоящего курса лекций при первом знакомстве с определениями тех же понятий в *ISO IEC* 12207 возникло четкое ощущение, что авторы стандарта явно чего-то не договаривают. Посудите сами: согласно данному стандарту, *верификация* — это подтверждение экспертизой и представлением объективных доказательств того, что конкретные требования полностью реализованы. С другой стороны, *валидация* — это подтверждение экспертизой и представлением объективных доказательств того, что конкретные требования к конкретным объектам полностью реализованы. Правда, к чести авторов последнего стандарта, они приводят примечание, которое несколько приближает читателя к пониманию: "*валидация* связана с экспертизой продукта в целях определения его соответствия потребностям пользователя". В этом и заключается суть отличия: если *верификация* связана с выяснением того, удовлетворяет ли разрабатываемый *объект,* либо процесс его создания сформулированным требованиям, то *валидация* отвечает на вопрос - правильно ли разработан целевой *объект* (продукт), удовлетворяет ли он потребностям заказчика. Другой аспект валидации заключается в том, что она обычно увязывается с формальной приемкой (аттестацией) системы.

Некоторые стандарты, например *SWEBOK,* IEEE 1059-93 "IEEE Guide for Software *Verification* and Validation Plans", вводят для этих двух процессов обобщающее понятие V&V (Validation and *Verification*). Согласно IEEE 1059-93, верификация и валидация программного обеспечения - упорядоченный подход в оценке программных продуктов, применяемый на протяжении всего жизненного цикла. Усилия, прилагаемые в рамках работ по верификации и

валидации, направлены на обеспечение качества как неотъемлемой характеристики программного обеспечения и удовлетворение *пользовательских требований*.

Из вышесказанного ясно, как осуществить верификацию и валидацию АИС и (или) процесса ее создания: в первом случае необходимо убедиться, что АИС (компонента, процесс) соответствует сформулированным требованиям, во втором - что АИС действительно работает! Но если критерием проверки АИС служат требования, то что может послужить критерием проверки самих требований? Ответ заключается в том, что требования должны удовлетворять свойствам, сформулированным в "Свойства требований" , Кроме того, следует убедиться в том, что:

- в спецификации требований к ПО должным образом описаны предполагаемые возможности и характеристики системы, которые удовлетворят потребности различных заинтересованных в проекте лиц;
- требования к ПО точно отражают системные требования, бизнес-правила и др.;
- требования обеспечивают качественную основу для проектирования и сборки ПО.

Некоторые типичные проблемные ситуации процесса формирования и оценки требований

Двусмысленность требований

В ряду проблем и недостатков требований двусмысленность, является, пожалуй, наиболее критичным фактором риска проекта, закладываемого в фазе формирования требований. Двусмысленность (несоответствие свойству ясности, определенности) закладывает под проект "бомбу замедленного действия". На практике требование, сформулированное двусмысленным образом, может привести к различным его интерпретациям представителями Разработчика и Заказчика. Разработчик, руководствуясь своей интерпретацией, определит на ее основе архитектурную основу, создаст аналитические и проектные модели и в конечном итоге создаст программный код. Как показывают исследования, цена исправления ошибки вырастает примерно на порядок при переходе между рабочими потоками (от анализа требований к проектированию, от проектирования к реализации и т.д.).

Тем самым, если не заложить средства на проверку требований на предмет двусмысленности в момент их формирования - существует риск непринятия готовой системы в момент приемо-сдаточных испытаний, т.к. каждая из сторон будет придерживаться своей версии

интерпретации требований, что ведет к убыткам, судебным процессам и т.п. и тому есть масса примеров.

"Золочение" продукта

Под "золочением" понимают такие ситуации, когда разработчики добавляют функции, которых нет в спецификации, но им кажется, что это понравится пользователям. Зачастую же клиентам не нужны такие избыточные возможности, получается, что время, отведенное на реализацию, тратится впустую.

Эта ситуация возникает в случае, когда, во-первых, в коллективе Разработчика присутствуют творческие личности (ведь далеко не всякая команда станет проявлять инициативу и делать сверх того, о чем ее просили), во вторых - существует разрыв в прохождении информации от Заказчика к Разработчику. Инициативный разработчик "золотит" продукт из самых лучших побуждений, но, возможно, он плохо знаком с бизнес-процессом Заказчика и заложенные им "фичи" попросту не будут востребованы.

Другая сторона "золочения" заключается в том, что группа представителей Заказчика неоднородна по своей структуре и может возникнуть ситуация, когда представитель Заказчика, формулирующий "дорогие" требования, не обладает соответствующими полномочиями. Это - специфика российских предприятий, где часто все бывает устроено существенно неформально.

Автору пришлось однажды присутствовать в одном очень показательном диалоге, где он участвовал в качестве Разработчика, сдающего продукт (АИС для склада материалов), еще присутствовали пользователь новой системы и инвестор проекта. Ирина (пользователь): мне неудобно работать в этой программе. Она не учитывает размеры баночек в литрах. Павел (инвестор): без этого спокойно можно обойтись, достаточно того, что есть учет материалов по классификатору и 2 единицы измерения - в литрах и килограммах. Ирина: а мне этого недостаточно, я веду учет в баночках! Павел: а я тут хозяин, здесь все мое и мне твой баночный учет не нужен! Ирина: а я все равно буду работать, как я привыкла! Диалог длился около получаса и последнее слово осталось, конечно, за Павлом, программу переписывать не пришлось.

Минимальная спецификация

Создавать полную документацию требований в соответствии с вышеизложенными принципами, или ограничиться наброском требований на 2-3 страницы, как это зачастую делает автор лекций в небольших проектах - как говорится, дело вкуса.

Однако, для работы "не по правилам", во-первых, должны быть объективные предпосылки, во-вторых - следует отдавать себе отчет в выгодах и рисках этого выбора.

Минимальная спецификация уместна, если имеет место наличие одновременно трех обстоятельств (объединение по "И"):

а) цена контракта и размеры проекта таковы, что разработка развернутого ТЗ экономически нецелесообразна;

б) коллектив Разработчика обладает достаточной степенью профессионализма и опыта выполнения проектов в смежных областях, чтобы уметь создавать по краткой спецификации продукт, который пройдет приемку Заказчиком;

в) между Заказчиком и Разработчиком существуют конструктивные отношения и обе стороны понимают и принимают риски мини-спецификации.

Другой вариант работы по мини-спецификации: Заказчик и Разработчик понимают, что создание развернутой спецификации оттягивает окончание выпуска готового продукта, что главная цель проекта - продукт, а не документация и готовы к плотному сотрудничеству в процессе его создания. Это - путь так называемых *Agile*-методологий разработки ПО, подробнее см. в http://www.agile.org.

Основной риск применения мини-спецификации заключаются в том, что они базируются на человеческом факторе. Хорошие и конструктивные отношения между сторонами "на берегу" должны сохраниться на всем протяжении проекта, в противном случае у сторон возникнут существенные проблемы в формальном доказательстве того - что должна делать программа, т.к. мини-спецификация для этого недостаточно полна.

Пропуск типов пользователей

Корпоративные АИС создаются для того, чтобы быть использованными различными группами пользователей. Может сложиться ситуация, в которой в группу представителей Заказчика, участвующих в формировании требований, попадут наиболее инициативные персоны предприятия, которые, по всей видимости, смогут донести свой голос до представителей Разработчика. Те же категории пользователей, у которых не найдется активных представителей, могут оказаться "за бортом" автоматизации. Именно эта ошибка формирования требований называется "пропуск классов пользователей". Чтобы ее избежать, представитель Разработчика должен объективно оценить организационную структуру предприятия и его бизнес-процессы и

вдумчиво подойти к выбору ключевых персон, проведение интервью с которыми поможет сформировать целостную картину требований к создаваемой АИС.

Методы и средства проверки требований

Наработано значительное количество методов и средств проверки требований. Они разнятся *по* ряду параметров. Так, различают:

- по широте анализа - просмотр (выборочная проверка) и сквозной контроль (тотальная проверка);
- по степени формализации - неофициальные процедуры, процедуры, проводимые по формальным правилам (инспекции, экспертизы);
- по составу группы проверки - с (без) участием автора, с (без) участием менеджера проекта, с (без) участием представителей внешних организаций;
- по используемым средствам - тексты требований, тестовые сценарии, критерии приемлемости, прототипы.

Понятие и методы прототипирования будут рассмотрены ниже. Некоторые другие, наиболее важные из перечисленного выше, методы и средства, рассмотрены далее *по* тексту.

Неофициальные просмотры требований

Различают несколько способов неофициальных просмотров требований:

- просмотр "за столом",
- коллективная проверка,
- критический анализ.

В первых двух случаях автор требований обращается за помощью к коллегам (соответственно, к одному, либо к нескольким) с целью выдачи практических рекомендаций по улучшению продукта. В третьем случае автор осуществляет презентацию разработанных им требований на совещании с последующим обсуждением.

Неофициальные просмотры используют для знакомства с разработкой, сбора отзывов, формирования обратной связи. По статистике неофициальные просмотры позволяют выявить до 60% ошибок в требованиях.

Инспекции

Понятие инспекции, применительно к IT-индустрии, впервые было сформулировано Майклом Фэганом (Michael Pagan) из IBM в середине 70-х гг.

Согласно стандарту IEEE[2], проведение инспекций, в отличие от неформальных просмотров, базируется на своде формальных требований и правил.

Лица, занимающие управленческие позиции (менеджеры) в отношении к любым членам команды инспектирования, не должны участвовать в инспекциях.

Инспекция должна вестись под руководством непредвзятого (независимого от проекта и его целей) лидера, обученного техникам инспектирования.

Инспектирование всегда вовлекает авторов промежуточного или конечного продукта.

В группу инспекции входят лидер, регистратор, рецензент и несколько (от 2 до 5) инспекторов. Члены команды инспектирования могут специализироваться в различных областях экспертизы (обладать различными областями компетенции), например, предметной области, методах проектирования, языке и т.п. В заданный момент (промежуток) времени инспекции проводятся в отношении отдельного небольшого фрагмента продукта (в большинстве случаев, фокусируясь на отдельных функциональных или других характеристиках; часто, отталкиваясь от отдельных бизнес-правил, функциональных требований или атрибутов качества, прим. автора). Каждый член команды должен исследовать оцениваемый продукт и другие входные данные до проведения инспекционной встречи, применяя, возможно, те или иные аналитические техники к небольшим фрагментам продукта или к продукту, в целом, рассматривая в последнем случае только один его аспект, например, интерфейсы. Любая найденная аномалия должна документироваться, а информация передаваться лидеру инспекции. В процессе инспекции лидер руководит сессией и проверяет, что все подготовились к инспектированию. Общим инструментом, используемым при инспектировании, является проверочный лист (checklist), содержащий аномалии и вопросы, связанные с аспектами, вызывающими интерес. Результирующий лист часто классифицирует аномалии и оценивается командой с точки зрения его завершенности и точности. Решение о завершении инспекции принимается в соответствии с одним (любым) из трех критериев:

1. Принятие с отсутствием либо малой необходимостью переработки
2. Принятие с проверкой переработанных фрагментов
3. Необходимость повторной инспекции.

Список использованных источников

1. Вигерс Карл. Разработка требований к программному обеспечению/Пер, с англ. — М.: Издательство-торговый дом «Русская Редакция», 2004. —576с.
2. Леффингуелл Д., Уидриг Д. Принципы работы с требованиями к программному обеспечению. М.: ИД “Вильямс”, 2002.
3. Маглинец Ю.А. Разработка информационных систем. Учебное пособие. Часть 1, Структурные методы. – Красноярск.: Кларитеанум, 2004. – 120с.
4. Маглинец Ю.А. Анализ требований к автоматизированным информационным системамю- [Электронный ресурс].- Режим доступа: https://www.intuit.ru/studies/courses/2188/174/lecture/4712
5. Мацяшек Л. Анализ требований и проектирование систем. Разработка информационных систем. пер. с англ.- М.: ИД.: Вильямс,2002.- 428 с.
6. Орлов С.А. Технологии разработки программного обеспечения. Учебник для вузов. - СПб.: Питер, 2002, с. 463.
7. Петров В.Н. Информационные Системы. Учебник для вузов. - СПб.: Питер, 2003, с. 688.
8. Светлов Н. М. Информационные технологии управления проектами: Учебное пособие / Н.М. Светлов, Г.Н. Светлова. - 2 изд., перераб. и доп. - М.: НИЦ ИНФРА-М, 2015. - 232с.
9. Гагарина Л.Г., Кокорева Е.В., Виснадул Б.Д. Технология разработки программного обеспечения: Учеб. пос. Под ред. проф. Л.Г. Гагариной - М.: ИД ФОРУМ: НИЦ Инфра-М, 2013. - 400 с.- [Электронный ресурс]. - Режим доступа: http://znanium.com/bookread2.php?book=389963
10. Гвоздева В.А., Лаврентьева И.Ю. Основы построения автоматизированных информационных систем: Учебник - М.: ИД ФОРУМ: НИЦ Инфра-М, 2013. - 320 с. URL: http://znanium.com/bookread.php?book=392285.
11. Академия Microsoft: Анализ требований к автоматизированным информационным системам: - Руководство по применению стандарта ИСО 9001:2000 при разработке программного обеспечения/ Пер. с англ. А.Л. Раскина. – М.: РИА “Стандарты и качество”, 2002. – 104 с. – (“Дом качества”, вып. 9 (18)).
12. Вендров А.М. Проектирование программного обеспечения экономических информационных систем. – М.: – Финансы и статистика, 2002 – 352 с.
13. Петров В. Н. Информационные системы. – СПб.: Питер, 2002. – 688 с.

14. Грекул В. И., Денищенко Г.Н., Коровкина Н.Л. Проектирование информационных систем: курс лекций: учеб пособие для студентов вузов, обучающихся в области информационных технологий. - М.: Интернт-Ун-т Информ- технологий, 2005.- 304 с

Буч Г., Рамбо Дж., Джекобсон А. Язык UML. Руководство пользователя.: Пер. с англ. -- М.: ДМК, 2000.