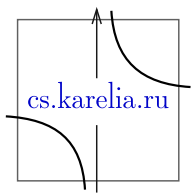
Петрозаводский государственный университет
Институт Математики и информационных технологий
Кафедра Информатики и математического обеспечения

Формирование и анализ требований. Управление и визуальные модели.

Лектор:

к.т.н., доцент Богоявленский Ю. А.
ybgv@cs.karelia.ru




Как пользователи видят программиста
Как программист видит пользователей

Разделы

- Способы представления границ проекта
- Характеристики детального требования
- Спецификация и управление требованиями
- Обзор и цели методов визуального моделирования требований
- Выбор правильного визуального представления

Способы представления границ проекта. Содержание

- Контекстная диаграмма.
- Карта экосистемы.
- Дерево функций.
- Список событий.

Способы представления границ проекта

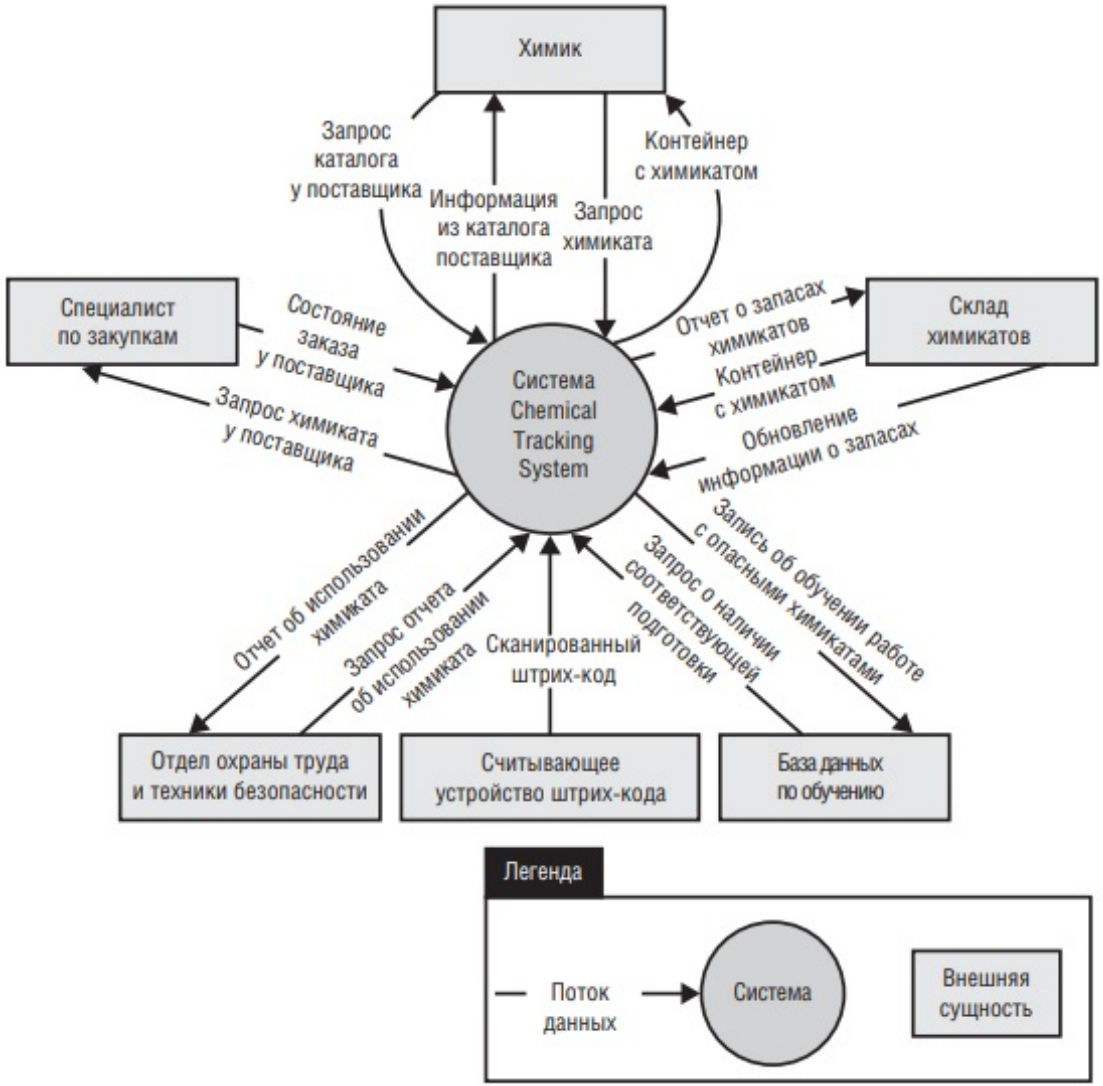
Задача таких инструментов, как контекстная диаграмма, карта экосистемы, дерево функций и список событий, — поощрять прозрачные и точные механизмы общения между заинтересованными лицами проекта.

Для наглядного представления границ проекта чаще всего используют контекстные диаграммы, карты экосистемы, дерева функций и списки событий. Но применяются и другие методы. Выявление затрагиваемых бизнеспроцессов также помогает определять границы проекта. Диаграммы вариантов использования позволяют проиллюстрировать границу между вариантами использования и действующими лицами.

Контекстная диаграмма

- графически иллюстрирует границу разрабатываемой системы со всем остальным миром;
- определяет оконечные элементы (terminators), расположенные вне системы, которые определенным образом взаимодействуют с ней, а также данные, элементы управления и материальные потоки, протекающие между оконечными элементами и системой;
- представляет собой высший уровень абстракции в диаграмме потока данных, разработанной по принципам структурного анализа (Robertson и Robertson, 1994), но эта модель полезна и в других проектах.

Частичная контекстная диаграмма для системы Chemical Tracking System



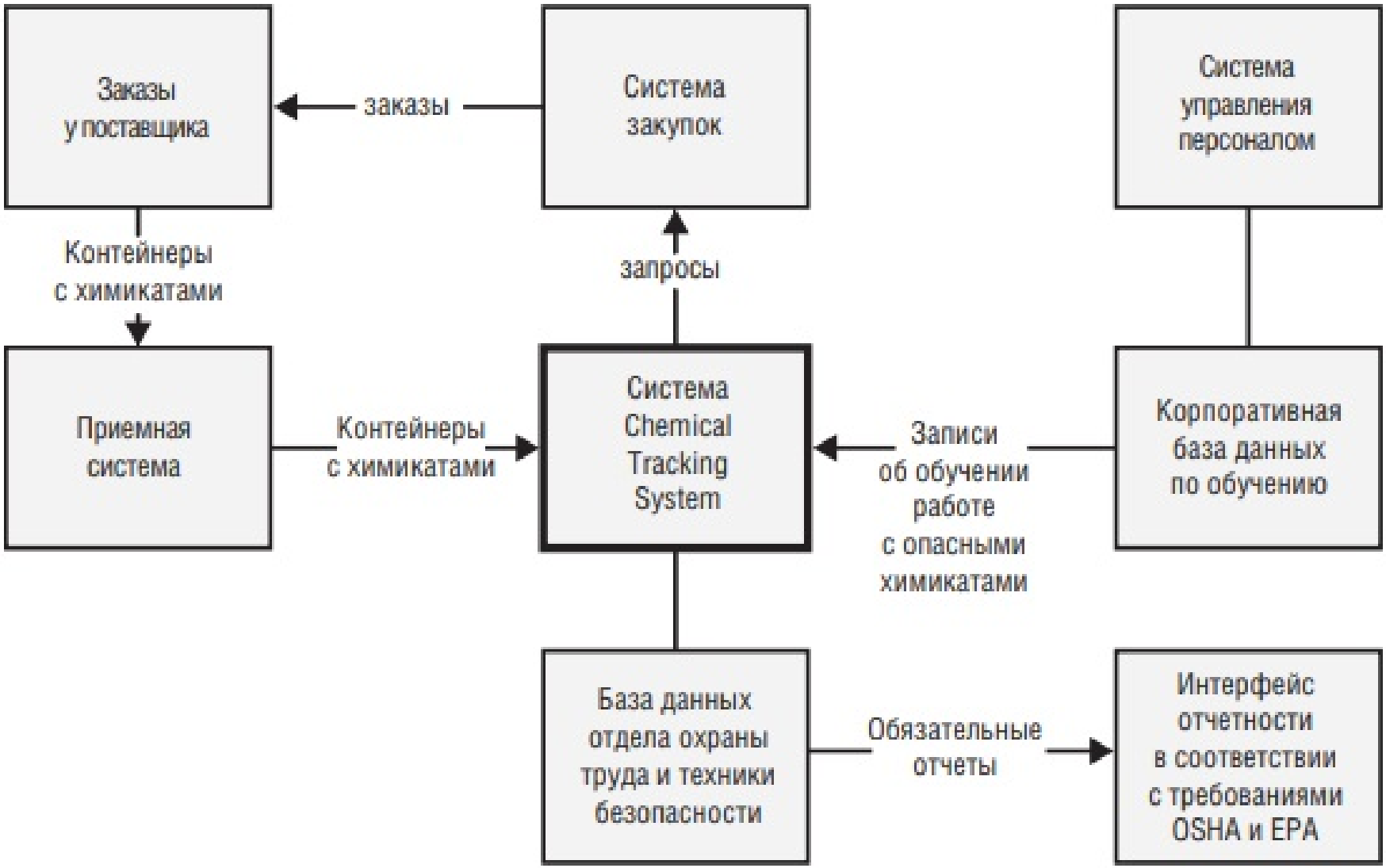
Частичная контекстная диаграмма для системы Chemical Tracking System

- вся система изображена кружком;
- «система» внутри кружка может иметь любую комбинацию ПО, оборудования или человеческих ресурсов. Поэтому она может содержать ручные операции в составе системы в целом;
- оконечные элементы в прямоугольниках представляют классы пользователей («Химик» или «Специалист по закупкам»), отделы («Отдел охраны труда и техники безопасности»), другие системы («База данных по обучению») или аппаратные устройства («Считывающее устройство штрих-кода»);
- стрелками показаны потоки данных («запрос химиката») или физические элементы («контейнер с химикатом») между системой и оконечными элементами.

Карта экосистемы

- показывает все системы, связанные с создаваемой системой и взаимодействующие друг с другом, а также природу этих взаимодействий (Beatty и Chen, 2012);
- представляет рамки путем отображения всех систем, которые взаимосвязаны друг с другом и которые может потребоваться изменить при создании вашей системы;
- отличается от контекстных диаграмм тем, что показывает другие системы, связанные с создаваемой вами, в том числе без непосредственных интерфейсов.
  Зависимые системы можно определить путем выявления тех, что потребляют данные, поступающие из вашей системы.

Частичная карта экосистемы для Chemical Tracking System



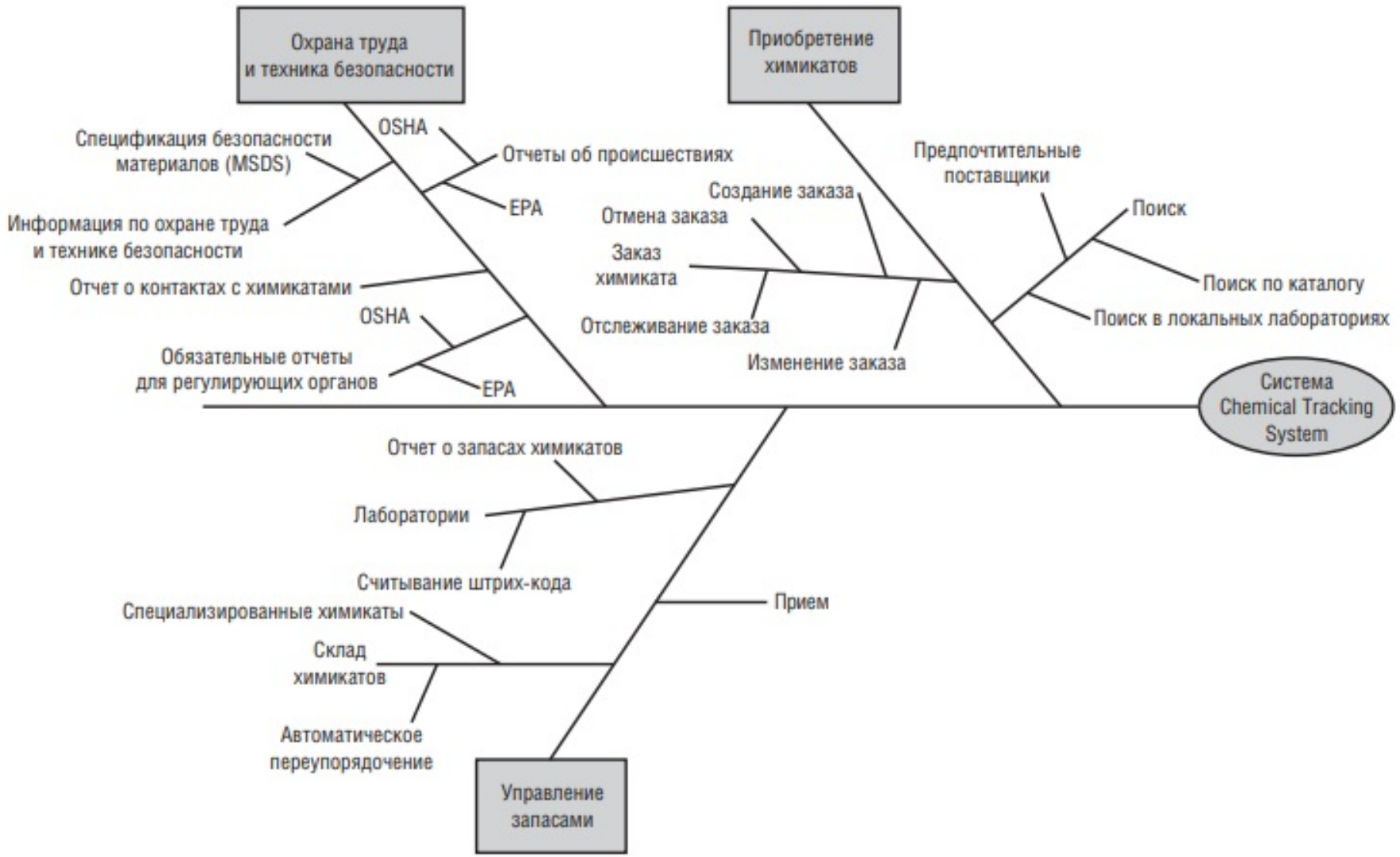
Частичная карта экосистемы для Chemical Tracking System

- системы изображены в виде прямоугольников (например, «Система закупок» или «Приемная система»). В этом примере основная система, над которой мы работаем, выделена жирным прямоугольником («Chemical Tracking System»), но если у всех систем равный статус, можно применить такой стиль и к ним;
- линии обозначают интерфейсы между системами (например, от системы закупок к Chemical Tracking System);
- линии со стрелками и надписями показывают важные порции данных, переходящих от одной системы к другой (например, «Записи об обучении работе с опасными химикатами» передаются от «Корпоративная база данных по обучению» в «Chemical Tracking System»). Некоторых из этих потоков также могут присутствовать на контекстной диаграмме.

Дерево функций

- представляет собой наглядную картину функций, объединенных в логические группы с иерархическим разбиением каждой функций на более мелкие (Beatty и Chen, 2012);
- предоставляет сжатую иллюстрацию всех запланированных к реализации в проекте функций, что отлично подходит для показа топ-менеджерам, желающим увидеть общую картину всего проекта;
- может содержать до трех уровней функций, которые обычно называют уровень 1 (L1), уровень 2 (L2) и уровень 3 (L3). Функции уровня L2 являются подфункциями L1, а функции L3 — подфункциями L2.

Частичная карта функций для системы Chemical Tracking System



Частичная карта функций для системы Chemical Tracking System

- ствол дерева в середине представляет реализуемый продукт. У каждой функции собственная линия или «ветка», отходящая от ствола;
- серые прямоугольники представляют функции уровня L1, такие как «Приобретение химикатов» и «Управление запасами»;
- линии, отходящие от L1, представляют функции уровня L2: «Поиск» и «Заказ химикатов» являются подфункциями функции «Приобретение химикатов»;
- подветками ветки L2 являются функции уровня L3: «Поиск в локальных лабораториях» является подфункцией функции «Поиск».

Список событий

- перечисляет внешние события, которые могут инициировать определенное поведение в системе;
- определяет границы системы путем перечисления возможных бизнес-событий, инициируемых пользователями или инициируемых временем (срабатывание по времени), или сигналов от внешних компонентов, таких как аппаратные устройства;
- в списке находятся только названия событий — функциональные требования, описывающие, как система реагирует на события, должны описываться в спецификации SRS с использование таблиц событий и реакций на них.

Список событий

Внешние события для Chemical Tracking System.
Частичный список событий для Chemical Tracking System:

- Химик разместил заказ химиката;
- Просканирован штрих-код контейнера с химикатом;
- Наступило время генерации отчетов OSHA;
- Поставщик выпустил новый каталог химикатов;
- Новый специализированный химикат добавлен в систему;
- Поставщик отменил заказ химиката;
- Химик запросил свой отчет о контактах с химикатами;
- Получена спецификация безопасности материалов из Управления по охране окружающей среды (EPA);
- В список предпочтительных поставщиков добавлен новый поставщик;
- Получен контейнер с химикатами от поставщика.

Список событий

Список событий можно сверить на предмет корректности и полноты с контекстной диаграммой и картой экосистемы следующим образом:

- Определите, какие внешние сущности в контекстной диаграмме могут являться источниками событий: «Могут ли какие-либо действия химика инициировать определенное поведение системы Chemical Tracking System?»;
- Посмотрите, нет ли в карте экосистемы системы, которая может инициировать события в вашей системе;
- Для каждого события определите, если соответствующие ему внешние сущности в контекстной диаграмме или системы в карте экосистемы: «Если контейнер с химикатом может поступить от поставщика, может ли поставщик фигурировать в контекстной диаграмме и/или в карте экосистемы?»

Характеристики детального требования. Содержание

- Желаемые характеристики детального требования.
- Прослеживаемость.
  - Определение.
  - Матрица прослеживания.
  - Прямая и обратная прослеживаемость.
  - Полная прослеживаемость
- Тестируемость.
  - Пример
  - Основные правила
- Однозначность.
- Приоритетность.
  - Причины выставления требованиям приоритетов
  - Порядок сокращения требований
- Полнота.
- Согласованность.

Желаемые характеристики детального требования

В ходе создания детальных требований ориентируются на формирование целого набора желаемых характеристик. К этим характеристикам относятся:

- прослеживаемость;
- тестируемость;
- однозначность;
- приоритетность;
- полнота;
- согласованность.

Прослеживаемость: определение

Прослеживаемостью называют отображение каждого требования на соответствующие части проекта и системы. Как показано на следующем рисунке, каждое функциональное детальное требование отображают на артефакты разработки.

Рис.: Прослеживание детального требования

Прослеживаемость: матрица прослеживания

По мере продвижения проекта должна сохраняться согласованность спецификации требований с архитектурой системы и программным кодом.

Когда код, реализующий требование, находится в нескольких местах, прослеживание достигается с помощью матрицы прослеживания требований:

Требование	Модуль 1	Модуль 2	Модуль 3
M_1415	showData()	AverageMark()	getInfo()
M_1416	showGroup()	showSemester()	showData()

Рис.: Пример матрицы прослеживания требований

Изменения в требованиях следует четко контролировать, поскольку одни и те же функции могут участвовать в удовлетворении разных требований. В результате изменения, сделанные для удовлетворения одного требования, могут нарушить другое.

Прослеживаемость: прямая и обратная

Должна быть возможность проследить каждое детальное требование прямо и обратно:

- Прямое отслеживание означает отслеживание от детальных требований до реализации.
- Обратное отслеживание детальных требований означает, что требование представляет собой четкую последовательность одного или нескольких первичных требований.

Обратное отслеживание является основой проверки детальных требований.

Полная прослеживаемость

Возможность полного прослеживания означает, что каждое детальное требование связано с конкретным элементом программной системы, а также с тестом элемента.

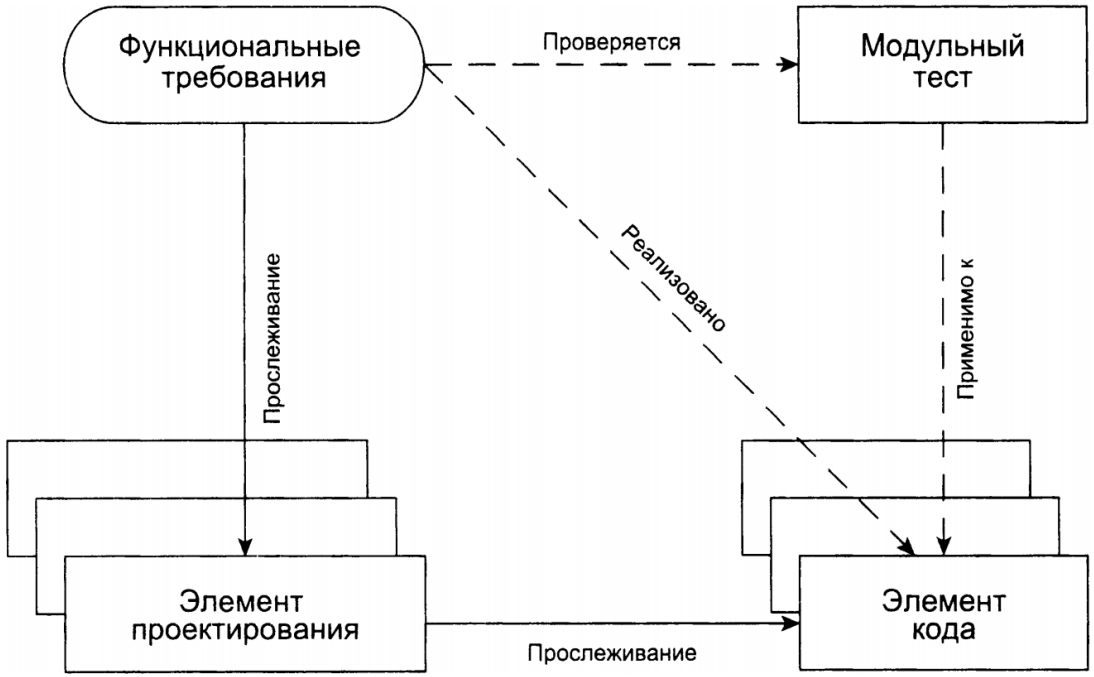


Рис.: Прослеживание и тестирование детальных требований

Отметим, что прослеживать нефункциональное требование обычно труднее, поскольку ему соответствуют несколько частей проектного решения и реализации.

Тестируемость: пример

Должна существовать возможность протестировать реализацию требования. Рассмотрим требование:

Коэффициент интеллекта сотрудника должен превышать средний уровень.

Это нетестируемое требование, поскольку средний уровень не задан. Его нужно модифицировать, например, так:

Коэффициент интеллекта сотрудника должен превышать средний уровень выпускника американского вуза (115 единиц).

Тестируемость: основные правила

Основные правила для достижения тестируемости:

- Готовность к тестированию будет полной, если для требования определено «свидетельство об ошибке».
- Появление свидетельства об ошибке говорит о некорректности реализации требования (или о некорректных условиях работы).
- Многие требования зависят от конкретных данных, и надо описывать, как реализация требования должна работать в случае неправильных данных.

Однозначность

Если требование записано нечетко или двусмысленно, нельзя определить правильность его реализации. Следующее требование неоднозначно:

Центральный процессор космического самолета должен быть идеален.

Что означает здесь «идеал»? Самый производительный, или самый надежный, или совместимый по системе команд с ЦП корабля Space Shuttle? А может, нужна какая-то комбинация этих показателей? Однозначное требование может быть записано в следующей форме:

Центральный процессор космического самолета должен иметь следующие параметры: среднее быстродействие на задачах управления полетом не менее 2 млн оп./ с, вероятность безотказной работы не менее 0,9999, рабочий диапазон температур -80 °C ... +140 °C, общая доза радиации 1000 крад.

Приоритетность: причины выставления требованиям приоритетов

Приоритетность необходима по следующим причинам:

- Часто бывает трудно реализовать все запланированные требования в срок.
- В процессе реализации можно выйти за рамки бюджета.

В этом случае уменьшают количество реализованных требований.

Одна из процедур отсеивания базируется на расстановке приоритетов между детальными требованиями. Обычно выделяют три категории требований:

- существенные;
- желательные;
- необязательные.

Приоритетность: порядок сокращения требований

Порядок сокращения прост:

1. Вначале сокращают необязательные требования.
2. Если этого недостаточно, переходят к сокращению категории желательных требований.
3. Категорию существенных требований стремятся сохранить, сокращая её только в самых крайних случаях.

Оптимальное сокращение достигается с помощью правила Парето — 20% требований реализуют 80% желаемой функциональности.

Полнота

Полнота набора требований является гарантией охвата всей функциональности и установки всех характеристик системы.

Для достижения полноты выполняют следующие действия:

- Оценивают полноту эксперты предметной области.
- Добавляют недостающие требования, если они были обнаружены при экспертизе предметной области.

Если полнота требований не была достигнута, в ходе реализации может получиться программная система, возможности которой недостаточны для заказчика.

Согласованность

Набор требований согласован, если между требованиями нет противоречий. По мере роста числа требований вероятность противоречий возрастает:

Тр_221: *Для повышения надежности следует вводить многоканальную избыточность вычислений основных управляющих воздействий и их голосование перед подачей на исполнительные органы ЛА.*

...

Тр_14247: *Для повышения скорости отработки возмущений не следует использовать схемы голосования управляющих воздействий в каналах угловой стабилизации и стабилизации центра масс ЛА.*

Организация детальных требований помогает минимизировать противоречия благодаря группировке родственных требований. Однако и при такой организации противоречия возможны, поэтому следует проверять согласованность наряду с другими характеристиками.

Спецификация и управление требованиями. Содержание

- Введение.
- Стандарт IEEE Std 830-1998.
- Структура спецификации требований к программному обеспечению (SRS).
- Управление требованиями.
- Программные утилиты для управления требованиями.
- Шаги процесса управления изменениями.

Введение

Спецификация требований — это документ, являющийся официальным предписанием для разработчиков ПО. Он содержит описание требований заказчика (первичных требований) и разработчика (детальных требований).

Первичные требования документируются при формировании требований, а детальные требования — при выполнении анализа требований.

Многие организации разрабатывали стандарты документирования требований. Наиболее полным и авторитетным считают стандарт Института инженеров но электротехнике и радиоэлектронике IEEE Std 830-1998.

Стандарт IEEE Std 830-1998

Стандарт IEEE Std 830-1998 должен помочь:

- заказчикам программного обеспечения точно описать, что они хотят получить;
- разработчикам программного обеспечения точно понять, что хочет заказчик.

Качественно составленная спецификация должна принести заказчикам, разработчикам и другим лицам определенные выгоды:

1. Создать основу для соглашения между заказчиками и разработчиками по поводу функций, которые должен выполнять программный продукт.
2. Уменьшить объем работ по разработке.
3. Обеспечить основу для оценки расходов и графиков работ.
4. Обеспечить основу для аттестации (валидации) и верификации.
5. Облегчить передачу программного изделия пользователям.
6. Служить в качестве основы для расширения.

Структура спецификации требований к программному обеспечению (SRS)

1. Введение,
2. Полное описание,
3. Конкретные требования,
4. Приложения,
5. Алфавитный указатель.

В стандарте указано, что он может использоваться частными организациями и для создания собственных стандартов по требованиям.

Структура спецификации требований к программному обеспечению (SRS). Введение

1. Введение
    1. Назначение (назначение спецификации, аудитория),
    2. Область действия (название ПО, его задачи, применение),
    3. Определения и сокращения (терминология),
    4. Публикации (список литературы),
    5. Краткий обзор (характеристика всех разделов).

Структура спецификации требований к программному обеспечению (SRS). Полное описание, Конкретные трбования, Приложения

1. Полное описание
    1. Перспектива изделия (оценка продукта, связи с другими продуктами),
    2. Функции изделия (основные функции продукта),
    3. Характеристики пользователя (общие характеристики пользователей продукта),
    4. Ограничения (ограничения возможностей разработчика),
    5. Допущения и зависимости (факторы, влияющие на требования),
    6. Распределение требований (требования, которые откладываются до появления будущих версий продукта).

Структура спецификации требований к программному обеспечению (SRS). Конкретные требования, Приложения

Конкретные требования охватывают:

- функциональные требования;
- нефункциональные требования;
- интерфейсные требования.

Конкретные требования являются наиболее значимой частью документа, описывают детальные требования, организованные выбранным способом.

В разделе Приложения дается оценка себестоимости, описываются форматы ввода-вывода, проблемы.

Управление требованиями

Известно, что требования в жизненном цикле разработки программной системы изменяются. И процесс этот имеет объективную основу. После создания начальной версии требований приходит новое, более глубокое понимание предметной области ПО, прорисовываются новые детали, которые раньше были незаметны.

В ходе управления требованиями нужно решить ряд вопросов:

- *Распознавание и учет требований.* Каждое требование должно быть индивидуально учтено.
- *Управление внесением изменений.* Должна предусматриваться последовательность защитных действий для оценки воздействия изменения и стоимости изменения.
- *Стратегия трассировки.* Трассировка должна обнаруживать зависимые требования, запоминать эти зависимости и отслеживать влияние требований друг на друга и на проектные решения.

Программные утилиты для управления требованиями

Управление требованиями нуждается в автоматизированной поддержке, которую обеспечивают программные утилиты. Утилиты поддерживают следующие действия:

- Хранение требований. Информация о требованиях должна сохраняться в защищенной управляемой памяти, доступной для участников процесса разработки требований.
- Реализация цикла изменения требования. Изменение требования происходит в интерактивном режиме: организуется диалог сотрудника с программной утилитой.
- Управление трассировкой. Утилита обнаруживает зависимые требования, выполняя действия трассировки.

Шаги процесса управления изменениями

1. Шаг 1. Распознавание проблемы.

   Фиксируется проблема в требованиях или прямой запрос на внесение изменения. Проверяется обоснованность проблемы или запроса. Если обоснованность подтверждена, переходят к следующему шагу. В противном случае процесс прекращается.
2. Шаг 2. Анализ изменения.
3. Шаг 3. Выполнение изменения.

Шаги процесса управления изменениями

1. Шаг 1. Распознавание проблемы.
2. Шаг 2. Анализ изменения.

   При определении возможности изменения исходят из информации трассировки и общих представлений о требованиях к системе. Стоимость изменения определяется двумя параметрами: стоимостью изменения спецификации и (если это необходимо) стоимостью изменения проектного решения системы и программного кода. По окончании анализа принимается решение об изменении требования.
3. Шаг 3. Выполнение изменения.

Шаги процесса управления изменениями

1. Шаг 1. Распознавание проблемы.
2. Шаг 2. Анализ изменения.
3. Шаг 3. Выполнение изменения.

   Вносится изменение в спецификацию требований и, если необходимо, в проектное решение и программный код. Спецификация требований должна быть организована так, чтобы внесение изменения носило локальный характер и не потребовало реорганизации всего документа. Как и в случае программ, изменяемость документов достигается минимизацией внешних ссылок и обеспечением модульности разделов. Это означает, что разделы могут изменяться без влияния на остальные части документа.

Вывод

- Спецификация требований содержит описание требований заказчика и разработчика.
- Наиболее полным и авторитетным стандартом документирования требований считают стандарт IEEE Std830-1998.
- В связи с многократными изменениями требований в жизненном цикле разработки программной системы следует пользоваться программными утилитами, обсепечивающими автоматизрованную поддержку, включающую в себя хранение требований, реализацию изменений требований, управление трассировкой.

Обзор и цели методов визуального моделирования требований. Содержание

- Введение
- Моделирование требований
  - Модели визуального представления требований
  - Применение и возможности визуальных моделей
  - Инструменты моделирования и их преимущества
  - Преимущества и недостатки моделей требований
- От желания клиента к модели анализа
  - Ключевые слова для построения модели
  - Пример выделения ключевых слов

Введение

«Никакое представление требований в одном виде не дает их полной картины» (c) Alan Davis, 1995

Необходима комбинация текстовых и визуальных способов представления требований на различных уровнях абстракции, чтобы получилась полная картина создаваемой системы. К таким способам относятся:

- списки функциональных требований;
- таблицы;
- графические модели анализа;
- прототипы пользовательского интерфейса;
- приемочные тесты;
- деревья и таблицы решений;
- видеоклипы;
- математические формулы.

Введение

- В идеале различные представления требований должны создавать **разные специалисты**. Это помогает выявить несоответствия, неясности, допущения и упущения, которые трудно обнаружить, когда требования представлены в одном формате.
- Определенные типы информации с помощью **диаграмм** удается передать гораздо эффективнее, чем с помощью текста.
- **Изображения** помогают преодолеть языковые барьеры и терминологические разногласия между членами команды.

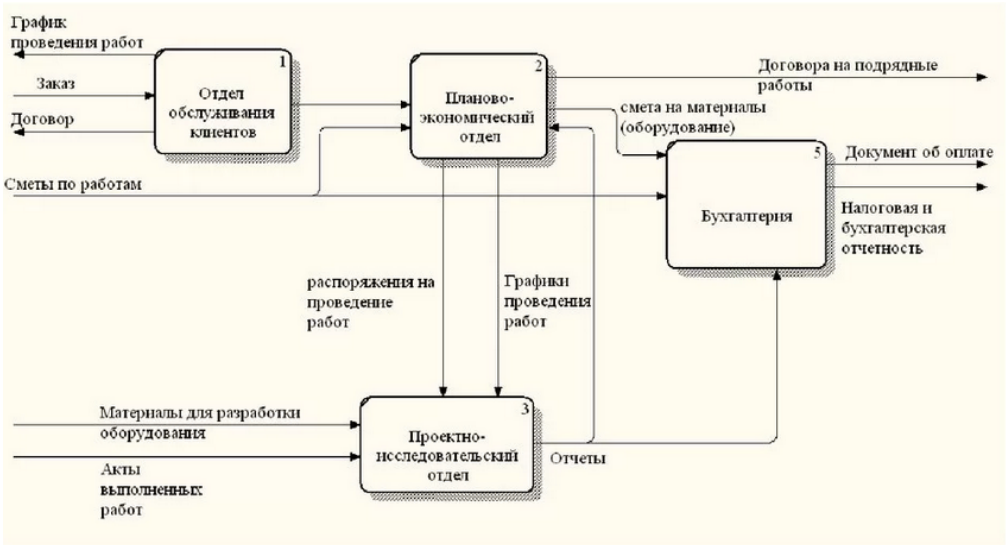
Моделирование требований: Модели визуального представления требований

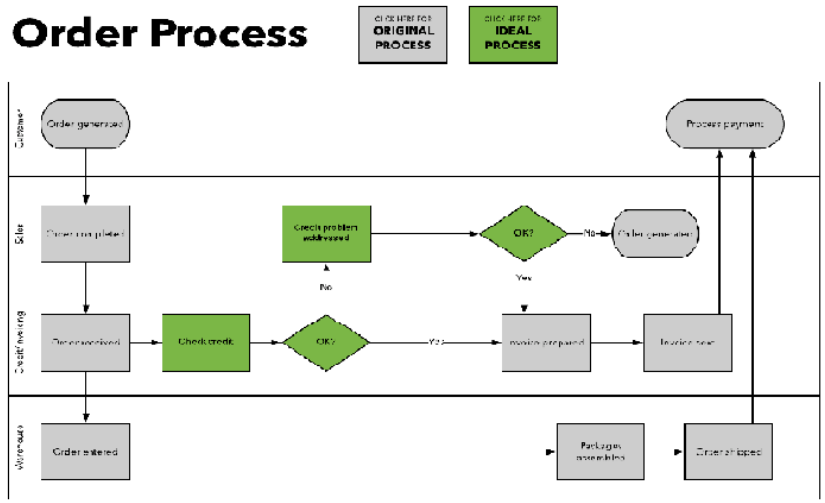
Первоначальной целью структурированных систем анализа была полная замена классической функциональной спецификации графическими диаграммами и системами обозначений, более формальными, чем текстовые комментарии. Однако опыт показал, что модели анализа должны *дополнять* — а *не заменять* — спецификации требований на естественном языке.

Визуальные модели требований могут помочь выявить отсутствующие, излишние и несовместимые требования. Ввиду ограничений краткосрочной памяти человека, анализ тысячи требований на предмет несоответствий, дублирования и посторонних требований практически невозможен.

Моделирование требований: Модели визуального представления требований

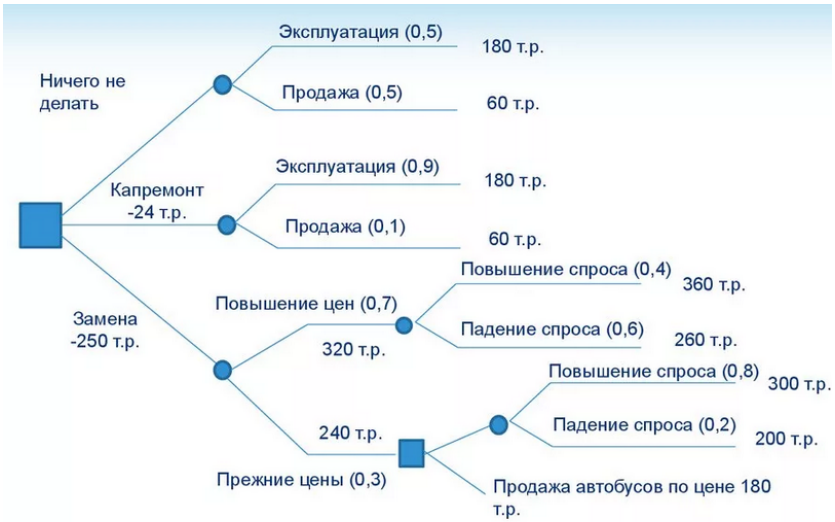
- диаграммы потоков данных (data flow diagrams, DFD);
- диаграммы рабочих потоков, такие как диаграммы swimlane;
- диаграммы переходов состояний (state-transition diagrams, STD) и таблицы состояний;
- карты диалоговых окон;

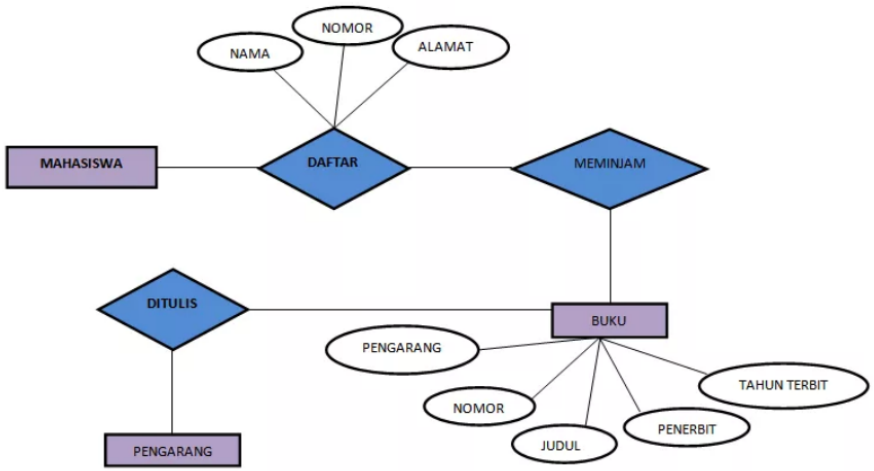




Моделирование требований: Модели визуального представления требований

- таблицы и деревья решений;
- таблицы событий и реакций;
- деревья функций;
- диаграммы вариантов использования;
- диаграммы процессов;
- диаграммы «сущность-связь» (entity-relationship diagrams, ERD).





Моделирование требований: Применение и возможности визуальных моделей

Модели годятся для **разработки и изучения требований**, а также для **дизайна ПО**. Вариант использования зависит от временных рамок и целей моделирования.

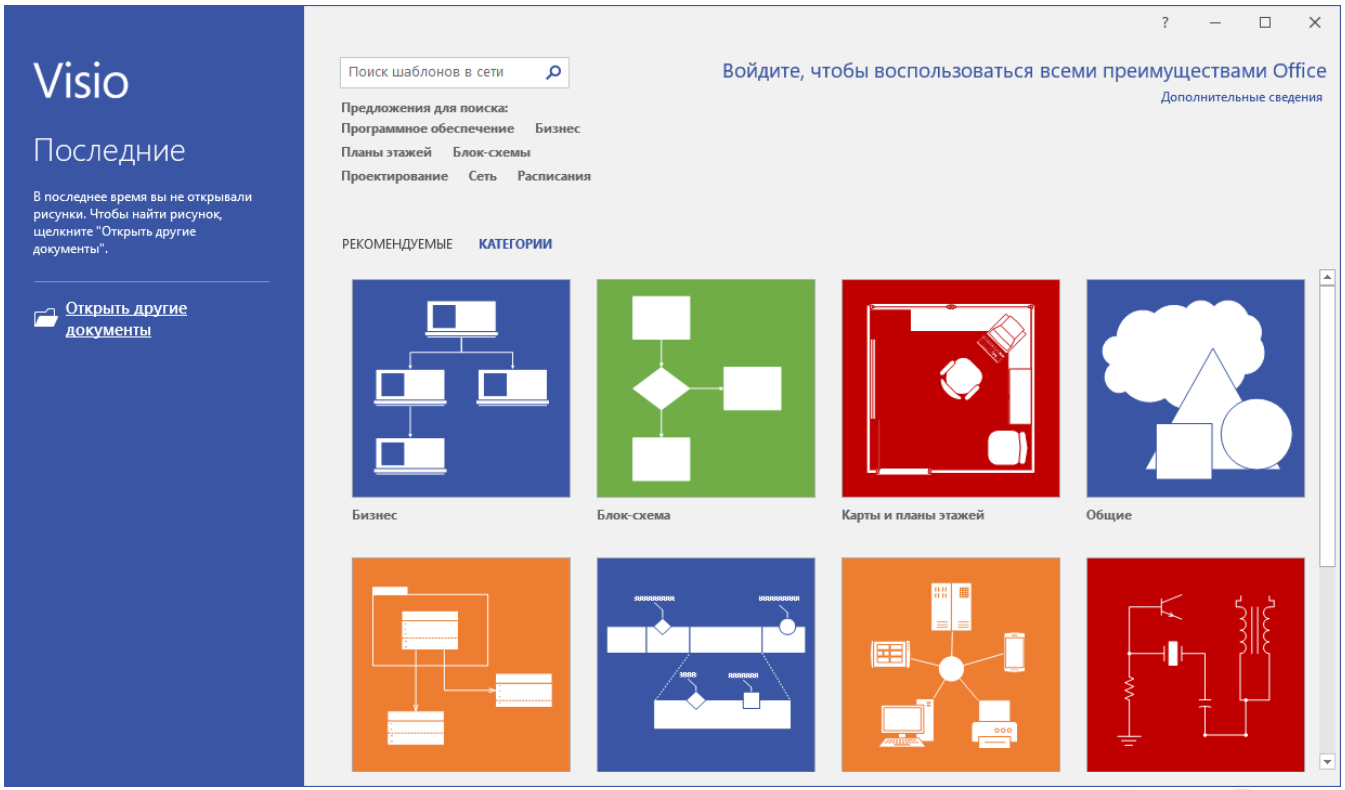
Применение диаграмм **для анализа требований** позволит смоделировать предметную область или создать концептуальные представления новой системы. Они отображают логические аспекты компонентов данных предметной области, транзакции и преобразования, объекты реального мира и изменения состояния системы.

Моделирование требований: Применение и возможности визуальных моделей

- Возможно создавать модели на основании *текстовых требований*, чтобы представить их с различных точек зрения, или вывести детализированные функциональные требования из моделей высокого уровня, созданных на основе предоставленной пользователями информации.
- В процессе разработки модели демонстрируют, как вы намерены реализовать систему.
- Так как в диаграммах анализа и дизайна используется одинаковая нотация, необходимо указывать *вид диаграммы*: модель анализа (концепции) или модель дизайна (что планируется создать).

Моделирование требований: Инструменты моделирования и их преимущества

Приемы моделирования анализа поддерживаются различными серийными **инструментами моделирования**, управления требованиями и построения графических диаграмм, такими как Microsoft Visio.



Моделирование требований: Инструменты моделирования и их преимущества

- Инструменты моделирования легко позволяют улучшить качество диаграмм при повторных просмотрах требований.
- Инструменты автоматизированного проектирования ПО применяют правила для каждого метода моделирования, который они поддерживают.
- Средства управления требованиями позволяют отслеживать связи между требованиями и моделями.
- Некоторые инструменты связывают различные диаграммы друг с другом и с их связанными функциональными требованиями и требованиями к данным.
- Использование средства со стандартными обозначениями позволяет обеспечивать совместимость моделей друг с другом.

Моделирование требований: Преимущества и недостатки моделей требований

Аргументы против использования моделей требований

- «Наша система слишком сложна для моделирования»;
- «У нас плотный график проекта и у нас нет времени на моделирование требований».

Преимущества моделей требований

- Модель проще системы, которая моделируется.
- Создание большинства моделей требует не больше времени, чем было бы потрачено на написание документа требований и анализа его на предмет проблем.
- Создание большинства моделей требует не больше времени, чем было бы потрачено на написание документа требований и анализа его на предмет проблем.
- Модели или их части иногда можно повторно использовать между проектами или по крайней мере использовать их в качестве

От желания клиента к модели анализа: Ключевые слова для построения модели

Тщательно слушая, как клиенты представляют свои требования, бизнесаналитик может выделить ключевые слова, которые преобразуются в определенные элементы модели.

Тип	Примеры	Компоненты модели анализа
Существительное	Люди, организации, системы ПО, элементы данных или существующие объекты	Внешние сущности, хранилища данных или потоки данных (диаграммы потоков данных) Действующие лица (диаграммы вариантов использования) Сущности или их атрибуты (диаграммы «сущность–связь») Дорожки (диаграммы swimlane) Объекты с состояниями (STD)

От желания клиента к модели анализа: Ключевые слова для построения модели

Тип	Примеры	Компоненты модели анализа
Глагол	Действия, задачи, которые пользователь может выполнить, или события, которые могут произойти	Процессы (диаграммы потоков данных)
		Шаги процессов (диаграммы swim-lane)
		Варианты использования (диаграммы вариантов использования)
		Взаимосвязи (диаграммы «сущность–связь»)
		Переходы (диаграммы переходов состояний)
		Действия (диаграммы действий)
		События (таблица событий и реакций)
Условие	Выражения условной логики, такие как «если, то»	Решения (деревья решений, таблицы решений или диаграмма действий)
		Ветвление (диаграмма swimlane или диаграмма действий)

От желания клиента к модели анализа: Пример выделения ключевых слов (Chemical Tracking System)

«**Химик** или **сотрудник склада химикатов** может *разместить* **заказ** на один или несколько **химикатов**, ***если пользователь уполномочен делать такие заказы***. Заказ может быть *выполнен* или посредством *доставки* **контейнера** с химикатом, который числится в **инвентарной описи товаров** на складе химикатов, или посредством размещения **заказа** на химикат у стороннего **поставщика**. ***Если химикат относится к опасным***, он может быть доставлен, ***только если пользователь прошел соответствующее обучение***. **Сотрудник**, размещающий заказ, *подготавливая* заказ, должен иметь возможность *искать* в интерактивном режиме нужный химикат в **каталогах поставщиков**. Системе необходимо *отслеживать* **состояние** каждого заказа с момента его подготовки и до момента его выполнения или *отмены*. Ей также необходимо отслеживать **историю** каждого контейнера с химикатом с момента его *получения* **компанией** до его полного *расходования* или *утилизации*».

Выбор правильного визуального представления. Содержание

- Выбор правильного визуального представления.
- Способы отображения информации.
  - Внешние интерфейсы системы
  - Поток бизнес-процессов
  - Определения данных и связи объектов данных
  - Состояния системы и объектов
  - Сложная логика
  - Пользовательские интерфейсы
  - Описания задач пользователей
  - Нефункциональные требования (атрибуты качества, ограничения)

Выбор правильного визуального представления

Команде необходимо создать полный набор моделей анализа для всей системы.

При моделировании нужно сосредоточиться на:

- наиболее сложных и опасных участках системы;
- на участках, где наиболее вероятны неясности и неопределенности;

Кандидатами в моделирование являются:

- элементы системы, от которых зависит ее безопасность и защита;
- элементы, влияющие на выполнение критически важных задач;

В них последствие дефектов окажутся особенно тяжелыми.

Выбор правильного визуального представления

Следует выбирать модели, которые будут использоваться вместе, чтобы гарантировать полноту всех моделей.

Например:

- Анализ объектов данных в диаграмме потоков данных помогает выявить отсутствующие сущности в диаграмме «сущность–связь»;
- Анализ всех процессов в диаграмме потоков данных может оказаться полезным для построения нужных диаграмм swimlane.

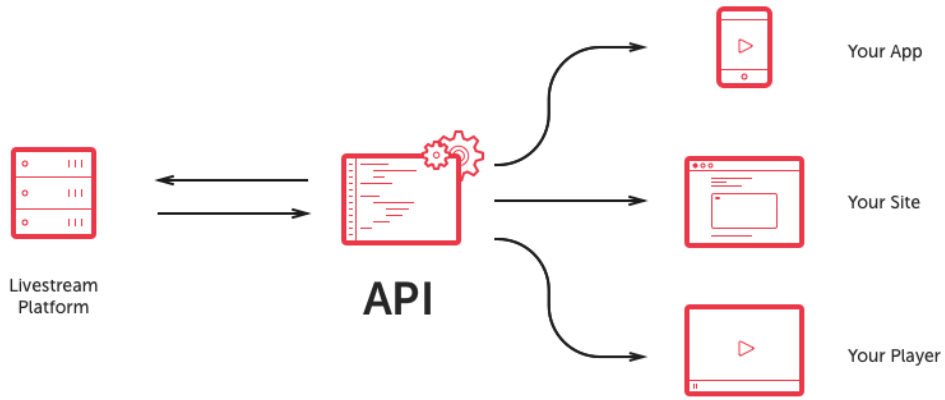
Далее приводятся рекомендации, какие приемы представления следует использовать в зависимости от типа имеющейся информации.

Внешние интерфейсы системы

- *Контекстная диаграмма* и *диаграмма вариантов использования* указывают внешние объекты, подключающиеся к системе;
- Контекстная диаграмма и *диаграмма потоков данных* иллюстрируют входную и выходную информацию системы на высоком уровне абстракции;
- *Карта экосистемы* указывает возможные системы, которые взаимодействуют, но также включает те, которые взаимодействуют не напрямую;
- *Диаграммы swimlane* показывают, что происходит в процессе взаимодействия между системами.

Внешние интерфейсы системы

- Подробности внешних интерфейсов могут фиксироваться во входных и выходных *форматах файлов* и *макетов отчетов* ;
- У продуктов, состоящих как из программных, так и аппаратных компонентов, часто есть *спецификации интерфейсов* с определениями атрибутов данных, обычно в форме API-интерфейса или конкретных входных и выходных сигналах аппаратного устройства;



Поток бизнес-процессов

- *Диаграмма потоков данных* верхнего уровня представляет на высоком уровне абстракции, как бизнес-процесс работает с данными;
- *Диаграмма swimlane* показывают роли, участвующие в выполнении различных этапов в потоке бизнес-процессов;
- Уточняющие уровни *диаграмм потоков данных* или *диаграмм swimlane* могут очень детально представлять потоки бизнес-процессов;
- *Диаграммы потоков* и *диаграммы действий* также могут использоваться как на высоком, так и низком уровне абстракции, хотя чаще всего они используются для определения деталей процесса;

Определения данных и связи объектов данных

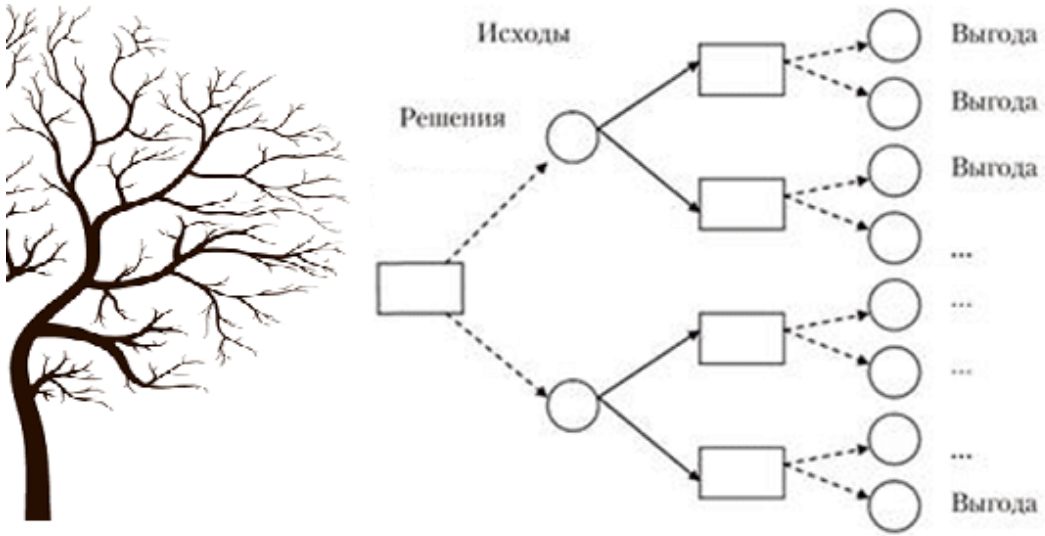
- *Диаграмма «сущность–связь»* показывает логические отношения между объектами данных (сущностями);
- *Диаграмма классов* показывает логические связи между классами объектов и связанными с ними данными;
- *Словарь данных* содержит подробные определения структур данных и отдельных элементов данных. Сложные объекты данных последовательно разбиваются на составляющие элементы данных;

Состояния системы и объектов

- *Диаграммы переходов состояния* и *таблицы состояния* дают представление высокого уровня абстракции возможных состояний системы или объекта и изменений между состояниями, которые могут происходить в определенных обстоятельствах;
- Некоторые аналитики создают *таблицу событий и реакций* в качестве инструмента определения границ продукта на основе внешних событий, в которой также могут быть указаны отдельные функциональные требования;
- *Функциональные требования* предоставляют подробности того, как в точности поведение системы и пользователя должно приводить к изменениям состояний.

Сложная логика

- *Дерево решений* показывает возможные результаты связанных решений и условий;
- *Таблица решений* определяет уникальные функциональные требования, связанные с различными комбинациями результатов true и false для наборов решений или условий.



Пользовательские интерфейсы

- *Карта диалоговых окон* предоставляет высокоуровневое представление предполагаемого или фактического пользовательского интерфейса и показывает разные элементы отображения и пути навигации между ними;
- *Раскадровки и бумажные прототипы* дополняют карту диалоговых окон, показывая, что должно содержать каждое окно, но не предоставляя точных деталей;
- *Модели «отображение-действие-реакция»* описывают отображение и поведение требований на каждом экране;
- *Подробные макеты экрана и бумажные прототипы* показывают в точности, как должны выглядеть элементы интерфейса;
- *Определения полей данных* и *описания элементов управления пользовательским интерфейсом* предоставляют дополнительные подробности.

Описания задач пользователей

- *Пользовательские истории, спецификации сценариев и вариантов использования* описывают задачи пользователей с различной степенью детализации;
- *Диаграммы swimlane* иллюстрируют бизнес-процесс или взаимодействие между многими действующими лицами и системой;
- *Блок-схемы* и *диаграммы действий* наглядно отображают направление диалога варианта использования и разветвляются на альтернативные направления и исключения;

Описания задач пользователей

- *Функциональные требования* предоставляют подробные описания, как система и пользователь будут взаимодействовать для достижения полезных результатов;
- *Варианты тестирования* предоставляют альтернативное представление низкого уровня абстракции, описывающее какое точно поведение следует ожидать от системы при определенных условиях входных данных, состоянии системы и действиях.

Спасибо за внимание.