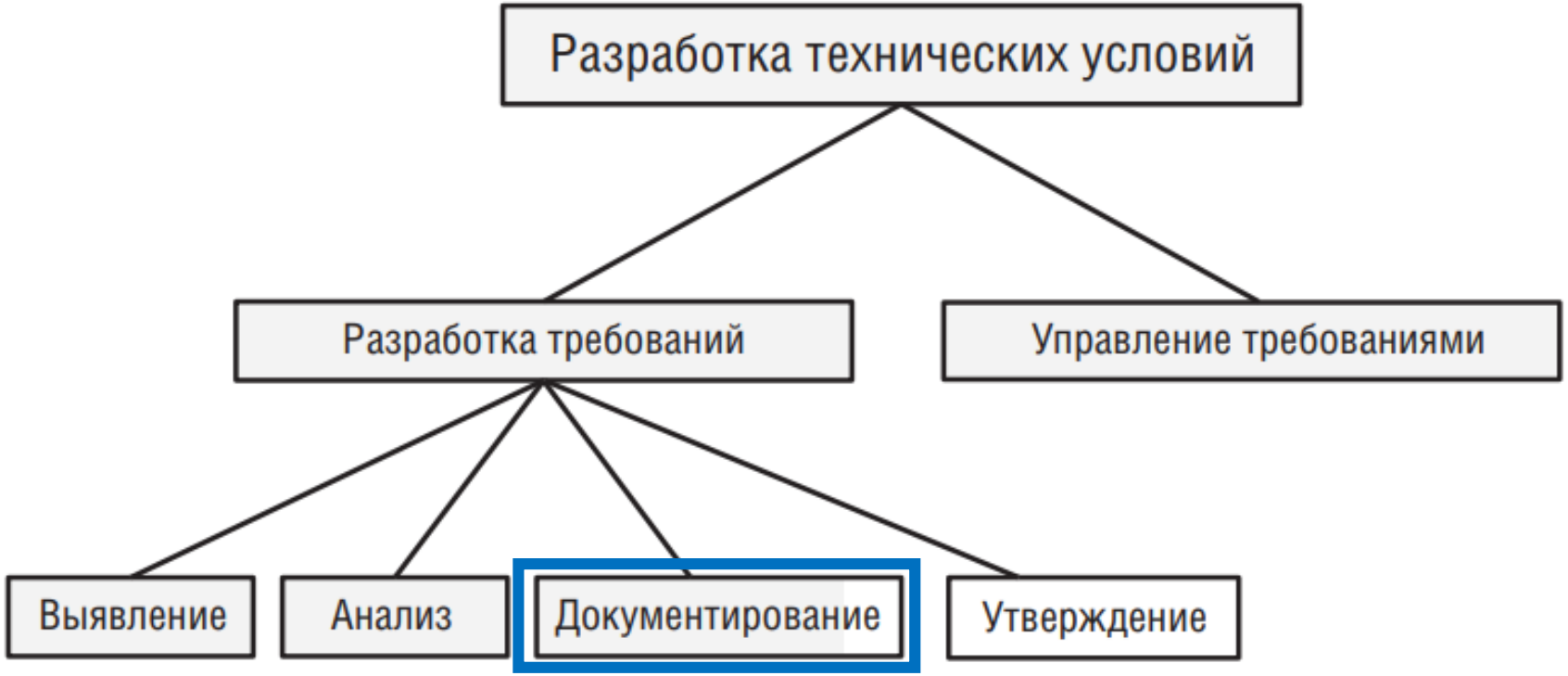
Лекция 4

Документирование требований



Способы представления требований

Подробные функциональные и нефункциональные требования к продукту записываются в спецификации к требованиям к ПО. **Спецификация предоставляется всем, кто должен проектировать, разрабатывать и проверять решение.**

Аналитики пишут требования со своей точки зрения, но они должны их писать так, чтобы требования были максимально понятными тем, кто их должен понимать и выполнять на их основе свою работу. Вот почему **важно, чтобы представители этих аудиторий рецензировали требования, проверяя, соответствуют ли требования их потребностям**.

Способов представления требований несколько:

- ✓ ***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.

обеспечивают наивысшую степень точности, однако в большинстве проектов не требуется такого уровня формализма.

Спецификация требований к ПО

Спецификацию требований в различных компаниях называют по-разному: документом бизнес-требований, функциональной спецификацией, спецификацией продукта или просто документом о требованиях.

> **Спецификация требований программного обеспечения** (англ. software requirements specification, SRS) — структурированный набор требований/запросов (функциональность, производительность, конструктивные ограничения и атрибуты) к программному обеспечению и его внешним интерфейсам. (Определение на основе IEEE Std 1012:2004). Предназначен для того, чтобы установить базу для соглашения между заказчиком и разработчиком (или подрядчиками) о том, как должен функционировать программный продукт. википедия

В спецификации требований к ПО указываются функции и возможности, которыми должно обладать ПО, а также необходимые ограничения.

Она должна содержать достаточно подробное описание поведения системы при различных условиях, а также необходимые атрибуты качества системы (производительность, безопасность и удобство использования).

Спецификация требований служит основой для дальнейшего планирования, дизайна и кодирования, а также базой для тестирования пользовательской документации.

Если нужной функциональности или качества нет в соглашении о требованиях, нет оснований ожидать, что они появятся в продукте.

Спецификация требований к ПО необходима различным участникам проекта:

- **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;

Советы как сделать требования ясными и понятными:

- ✓ для структурирования всей необходимой информации **используйте** соответствующий **шаблон**;
- ✓ разделы, подразделы и отдельные требования должны именоваться единообразно;
- ✓ **используйте средства визуального выделения** (такие, как полужирное начертание, подчеркивание, курсив и различные шрифты); 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;
- ✓ если вы используете документы, **применяйте гиперссылки**, чтобы читатель смог быстро перейти к соответствующим разделам спецификации или другим файлам;
- ✓ **включайте графическое представление информации**, где это возможно для облегчения понимания.

Требования к именованию

У каждого требования должен быть уникальный и неизменный идентификатор.

Это позволит ссылаться на определенные требования в запросах на изменения, в перекрестных ссылках и т.д. При этом также упрощается повторное использование требований в нескольких проектах.

! Простые нумерованные или маркированные списки для этих целей не годятся.

Существует множество схем, которые используются для именования требований, например:

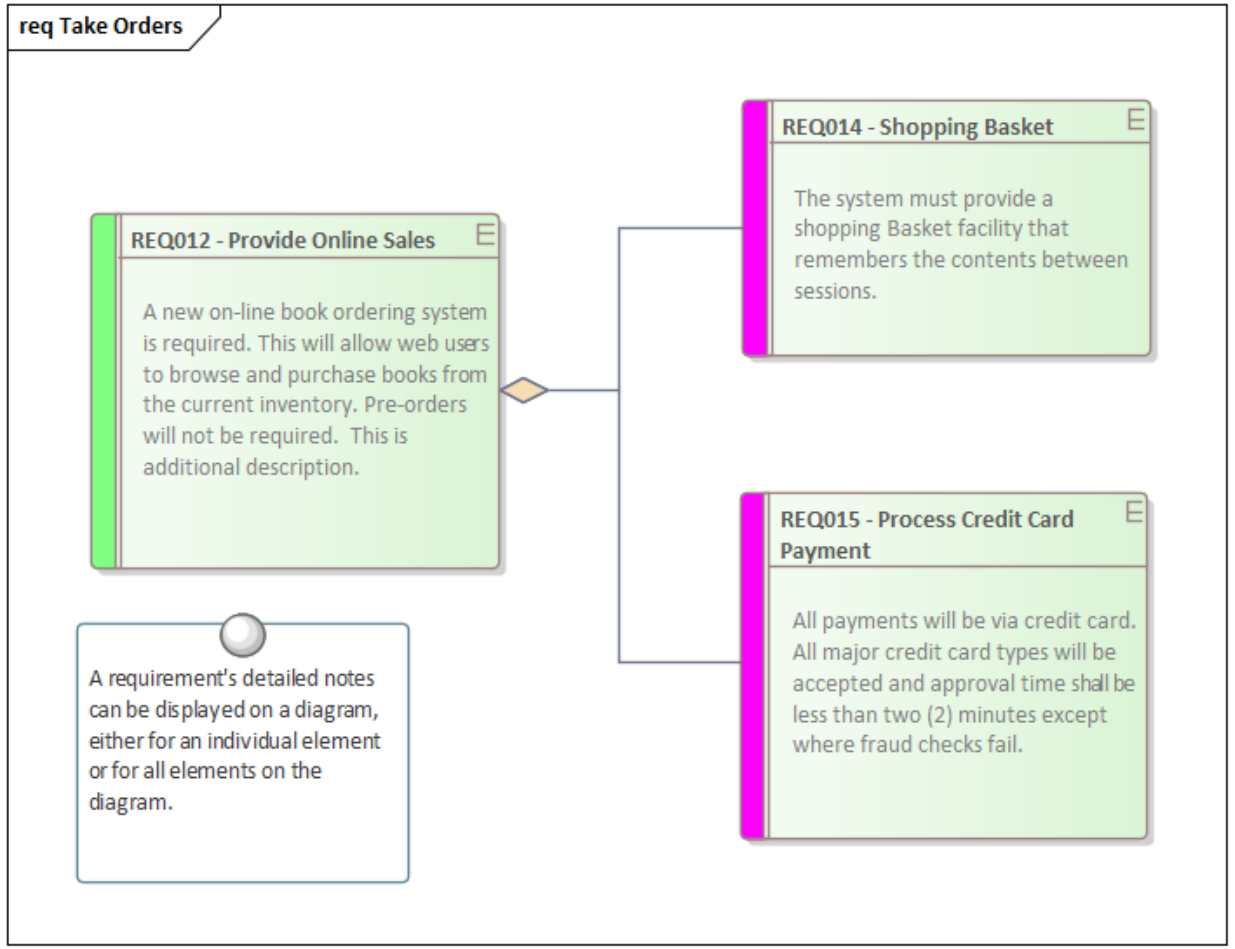
- ✓ Названия и описания требований
- ✓ Нумерация по порядку
- ✓ Иерархическая нумерация
- ✓ Иерархические текстовые теги
- ✓ . . .

Названия и описания требований

REQ116 -The system must email the client a copy of the receipt

The system must provide a copy of the receipt of payment to the customer within 15 minutes of the purchase. The receipt must itemize any applicable taxes and be in the client's local or chosen currency.

Этот метод создает выразительные диаграммы, раскрывающие детали требования и помогающие более полно понять требование.



Нумерация по порядку

Каждому требованию присвоен уникальный порядковый номер, например **UR-9** или **FR-26**. Префикс обозначает тип требования, например FR означает «functional requirement» (то есть «функциональное требование»).

При удалении требования номер повторно не используется, поэтому не придется путаться между исходным требованием FR-26 или новым требованием FR-26.

Минусы: Такой простой подход нумерации не обеспечивает логического или иерархического группирования связанных требований, число не подразумевает никакого упорядочения, а названия не раскрывают содержания требования.

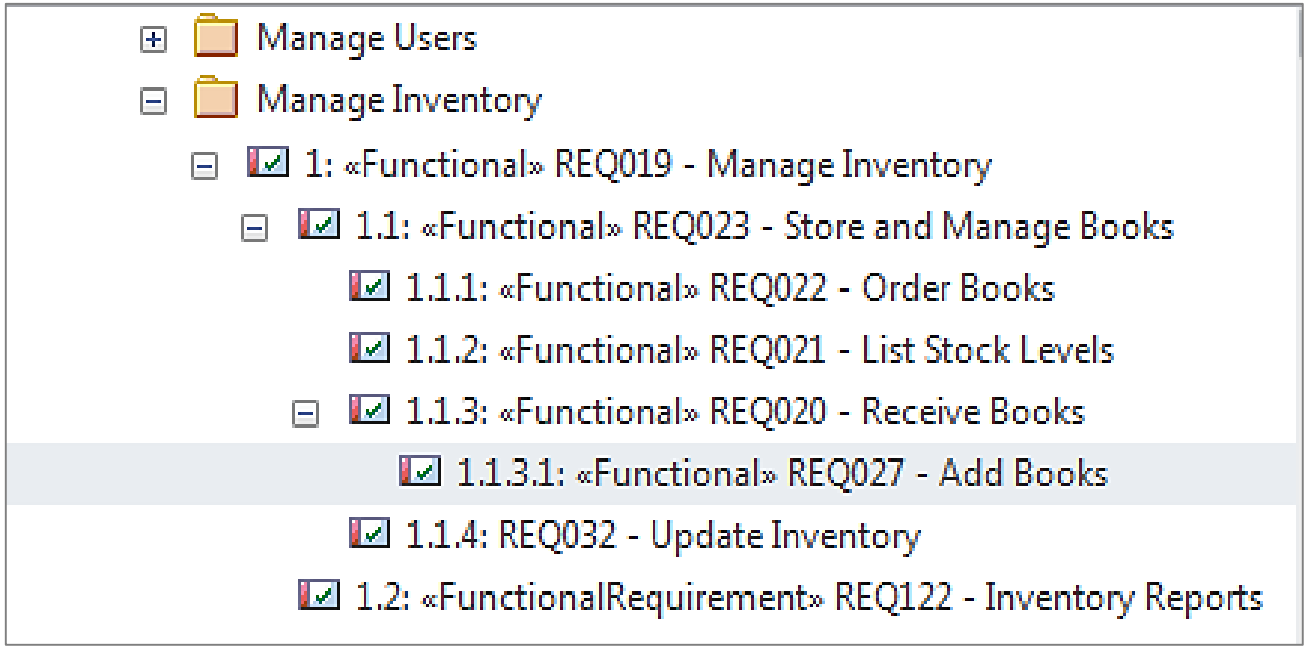
Плюсы: Позволяет сохранять уникальные идентификаторы при перемещении требований внутри документа.

Иерархическая нумерация

Если функциональные требования приводятся в разделе 3.2 спецификации, то все их номера будут начинаться с 3.2. Чем больше цифр, тем больше уровень детализации требования, так что сразу понятно, что 3.2.4.3 является потомком по отношению к 3.2.4.

Плюсы: Это способ отличается простотой, компактностью и очевидностью.

Минусы: Даже в спецификации среднего размера нумерация может быть весьма длинной. Названия, состоящие только из цифр, ничего не говорят о назначении требований. Удаление, вставка, слияние или перемещение целых разделов приводит к изменению многих фрагментов. При этом нарушаются все ссылки на эти фрагменты.



Иерархические текстовые теги

Рассмотрим такое требование: «*Система должна запрашивать у пользователя подтверждение запроса, когда тот хочет печатать более 10 копий*». Этому требованию можно присвоить тег **Print.ConfirmCopies**, который означает, что это требование является частью функции печати и связано с количеством печатаемых копий.

Плюсы: Иерархические текстовые тэги структурированы, их названия осмысленны и не меняются при добавлении, удалении или перемещении остальных требований.

При иерархической организации между требованиями возникают отношения «родитель–потомок». Родительское требование должно выглядеть как заголовок или название функции.

Вот пример, содержащий заголовок и четыре функциональных требования.

Product:	Заказ товаров в интернет-магазине
.Cart	В интернет-магазине должна быть корзина, в которой размещаются все выбранные клиентом товары
.Discount	В корзине должно присутствовать поле для кода скидки. Код скидки принимает вид определенного процента на все содержимое корзины или фиксированную долларовую скидку на конкретные товары в корзине
.Error	При вводе клиентом неверного кода скидки интернет-магазин должен отображать сообщение об ошибке
.Shipping	Корзина должна прибавлять к заказам клиентов стоимость доставки, если клиент заказывает физические товары, которые нужно доставлять

Полный уникальный идентификатор каждого требования создается путем присоединения метки требования к строке, состоящей из меток вышестоящих родителей. Первое функциональное требование обозначено как ***Product.Cart***. Полный идентификатор третьего требования — ***Product. Discount.Error***. Такая иерархическая схема избавляет от проблем с обслуживанием иерархической нумерации.

Может быть сложным обеспечить уникальность, особенно если над набором требований трудится несколько человек. Один из способов: можно упростить схему, совместив иерархическую нумерацию с суффиксом порядкового номера: *Product.Cart.01*, *Product.Cart.02* и т. д.

Пользовательские интерфейсы и спецификация требований к ПО

Включение элементов пользовательского интерфейса в спецификацию имеет как преимущества, так и недостатки.

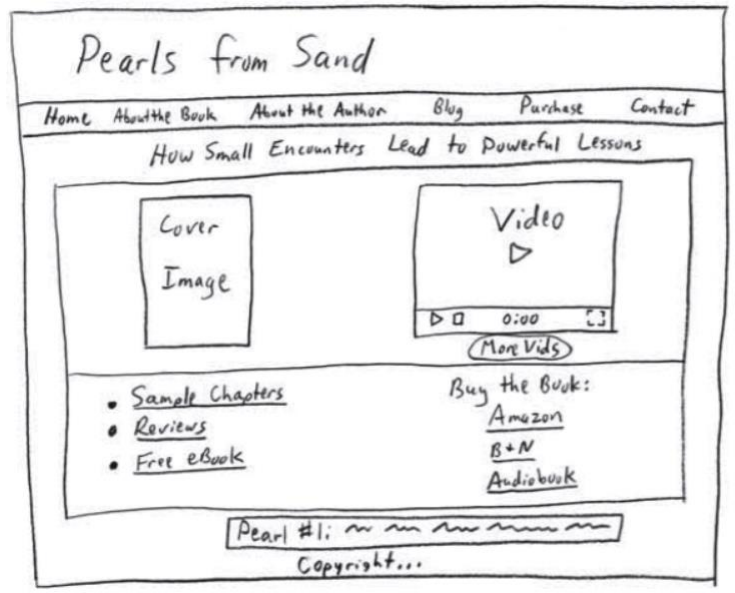
Положительным моментом можно считать то, что прототипы делают требования осязаемыми как для пользователей, так и разработчиков.

Отрицательным моментами можно считать следующие утверждения:

- ✓ Изображения и архитектура пользовательского интерфейса отображают решения (дизайн), а не требования.
- ✓ Их включение в спецификацию требований к ПО делает документ тяжеловесным.
- ✓ Откладывая создание базового соглашения до завершения разработки пользовательского интерфейса, ведет к замедлению разработки.
- ✓ Включение дизайна пользовательского интерфейса может привести к тому, что визуальный дизайн будет определять требования, что часто ведет к пропуску функций.
- ✓ Заинтересованные лица увидят пользовательский интерфейс в спецификации требований и им сложно будет его «забыть», что приведет к затруднению улучшения пользовательского интерфейса со временем.
- ✓ Не следует ожидать, что разработчики смогут сделать вывод о базовой функциональности и взаимосвязи данных по снимкам экрана.

Если у пользователей продукта есть ожидания насчет того, как должны выглядеть и вести себя те или иные части продукта, а, значит, они будут разочарованы отсутствием реализации своих ожиданий, эти ожидания относятся к требованиям.

«Золотая середина» подразумевает включение в требования концептуальных изображений выбранных экранов (набросков) без обязательного точного соблюдения этих моделей при реализации.



Стандарты и шаблоны для ТЗ на разработку ПО

Основные стандарты, методологии и своды знаний, где упоминается ТЗ или SRS (Software (or System) Requirements Specification):

- **ГОСТ 34**
- **ГОСТ 19**
- **IEEE STD 830-1998**
- **ISO/IEC/ IEEE 29148-2011**
- **RUP**
- **SWEBOK, BABOK и пр.**

Основными документами, регламентирующими порядок разработки систем для государственных органов, являются ГОСТы

ГОСТ 34

ГОСТ 34.602-89 Техническое задание на создание автоматизированной системы рекомендует структуру ТЗ на **создание именно СИСТЕМЫ**, в которую входят ПО, аппаратное обеспечение, люди, которые работают с ПО, и автоматизируемые процессы.

При разработке ТЗ для государственных проектов Заказчики, как правило, требуют соблюдение именно этого стандарта.

ГОСТ 19

ГОСТ 19.ххх Единая система программной документации (ЕСПД)— это комплекс государственных стандартов, устанавливающих взаимоувязанные правила разработки, оформления и обращения программ (или ПО) и программной документации. Т.е**. этот стандарт относится к разработке именно ПО.**

Стандарты и шаблоны для ТЗ на разработку ПО (продолжение)

IEEE STD 830-1998

Стандарт IEEE STD 830-1998 — IEEE Recommended Practice for Software Requirements Specifications.
Описывается содержание и качественные характеристики правильно составленной спецификации требований к программному обеспечению (SRS) и приводится несколько шаблонов SRS.

Согласно данному стандарту техническое задание должно включать следующие разделы:

1. Введение
    1. Назначение
    2. Область действия
    3. Определения, акронимы и сокращения
    4. Ссылки
    5. Краткий обзор
2. Общее описание
    1. Взаимодействие продукта (с другими продуктами и компонентами)
    2. Функции продукта (краткое описание)
    3. Характеристики пользователя
    4. Ограничения
    5. Допущения и зависимости
3. Детальные требования (могут быть организованы по разному, н-р, так)
    1. Требования к внешним интерфейсам
    2. Функциональные требования
    3. Требования к производительности
    4. Проектные ограничения (и ссылки на стандарты)
    5. Нефункциональные требования (надежность, доступность, безопасность и пр.)
    6. Другие требования
4. Приложения
5. Алфавитный указатель

Стандарты и шаблоны для ТЗ на разработку ПО (продолжение)

ISO/IEC/ IEEE 29148-2011

Стандарт IEEE 29148-2011 обеспечивает единую трактовку процессов и продуктов, используемых при разработке требований на протяжении всего жизненного цикла систем и программного обеспечения. Он приходит на смену стандартов IEEE 830-1998, IEEE 1233-1998, IEEE 1362-1998.

Стандарт содержит два шаблона спецификации требований:

- System requirements specification (SyRS)
- Software requirements specification (SRS)

SyRS определяет технические требования для выбранной системы и удобства взаимодействия предполагаемой системы и человека.

SRS это спецификация требований для определенного программного изделия, программы или набора программ (продукт), которые выполняют определенные функции в конкретном окружении. .

RUP(Rational Unified Process)

Структура SRS в RUP(Rational Unified Process) представляет собой документ, в котором необходимо описать артефакты, полученные в процессе специфицирования требований.

Шаблон SRS в RUP адаптирован из стандарта IEEE STD 830 и содержит два варианта:

- Традиционный шаблон SRS со структурированными функциональными требованиями по функциям Системы, максимально похож на 830 стандарт.
- Упрощенный шаблон SRS со структурированными функциональными требованиями в виде вариантов использования (use cases)

Шаблон спецификации требований к ПО по Вигерсу

Для упрощения составления спецификации можно воспользоваться адаптированным шаблоном Карла Вигерса.

Данный шаблон не служит строгим руководством, а позволяет проконтролировать собрана ли вся информация, необходимая для успеха проекта.

В книге “Разработка требований к программному обеспечению” Карла Вигерса приведен рекомендуемый шаблон спецификации требований, который подходит для многих проектов.

1. Введение
   - 1.1 Назначение
   - 1.2 Соглашения, принятые в документах
   - 1.3 Границы проекта
   - 1.4 Ссылки
2. Общее описание
   - 2.1 Общий взгляд на продукт
   - 2.2 Классы и характеристики пользователей
   - 2.3 Операционная среда
   - 2.4 Ограничения дизайна и реализации
   - 2.5 Предположения и зависимости
3. Функции системы
   - 3.x Функция системы X
     - 3.x.1 Описание
     - 3.x.2 Функциональные требования
4. Требования к данным
   - 4.1 Логическая модель данных
   - 4.2 Словарь данных
   - 4.3 Отчеты
   - 4.4 Получение, целостность, хранение и утилизация данных
5. Требования к внешним интерфейсам
   - 5.1 Пользовательские интерфейсы
   - 5.2 Интерфейсы ПО
   - 5.3 Интерфейсы оборудования
   - 5.4 Коммуникационные интерфейсы
6. Атрибуты качества
   - 6.1 Удобство использования
   - 6.2 Производительность
   - 6.3 Безопасность
   - 6.4 Техника безопасности
   - 6.x [Другие]
7. Требования по интернационализации и локализации
8. Остальные требования

Приложение А. Словарь терминов

Приложение Б. Модели анализа

<u>Рекомендации</u>:

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.

Далее рассмотрим подробнее содержание данного шаблона

1. Введение

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

1.1 Назначение

Определите 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. Опишите типы читателей, которым адресован этот документ, например разработчикам, менеджерам проектов, маркетологам, пользователям, тестировщикам или составителям документации.

1.2 Соглашения, принятые в документах

Опишите все стандарты или типографические соглашения, особенности выделения или нотацию.

1.3 Границы проекта

Кратко опишите ПО и его назначение. Покажите, как связан продукт с пользователями или корпоративными целями, а также с бизнес-целями и стратегиями. Если имеется отдельный документ о концепции и границах проекта, просто сошлитесь на него. Если спецификацию требований к ПО предполагается разрабатывать постепенно, она должна содержать собственное положение о концепции и границах продукта в качестве подраздела долгосрочной стратегической концепции. Можно предоставить высокоуровневую сводку главной функциональности выпуска или функций, которые он должен выполнять.

1.4 Ссылки

Перечислите все документы или другие ресурсы, на которые вы ссылаетесь в этой спецификации, в том числе гиперссылки на них, если их местоположение меняться не будет. Это могут быть руководства по стилям пользовательского интерфейса, контракты, стандарты, спецификации к системным требованиям, спецификации интерфейса и спецификации требований к ПО связанных продуктов.

2. Общее описание

В этом разделе представлен 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

Опишите контекст и происхождение продукта. Поясните, 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, укажите, как это ПО соотносится со всей системой и определите основные интерфейсы между ними. Неплохо также включить визуальные модели, такие как контекстную диаграмму или карту экосистемы, чтобы показать взаимосвязь продукта с другими системами.

2.2 Классы и характеристики пользователей

Определите различные классы пользователей, которые, как предполагается, будут работать с вашим продуктом, и опишите их соответствующие характеристики. Некоторые требования могут относиться только к определенным классам пользователей. Определите 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

Опишите рабочую среду, в которой будет работать ПО, включая аппаратную платформу, операционные системы и их версии, а также географическое местоположение пользователей, серверов и баз данных вместе с организациями, в которых располагаются соответствующие базы данных, серверы и вебсайты.

2.4 Ограничения дизайна и реализации

Опишите все факторы, которые ограничат возможности, доступные разработчикам, и логически обоснуйте каждое положение. 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. Разработчик может думать, что определенный набор функций написан специально для этого приложения, бизнес-аналитик — что он будет взят из предыдущего проекта, а менеджер проекта — что предполагается приобрести коммерческую библиотеку функций. Включаемые здесь предположения относятся к системной функциональности; предположения относящиеся к бизнесу представлены в документе концепции и границ проекта.

Определите все зависимости проекта или создаваемой системы от внешних факторов или компонентов вне ее контроля. Например, до установки продукта может требоваться установить Microsoft .NET Framework 4.5 или более позднюю версию — это зависимость.

3. Функции системы

Данный шаблон спецификации структурирован по функциям системы. Другие методы классификации — по функциональным областям, рабочим потокам, вариантам использования, режимам работы, классам пользователей. Возможны также иерархические комбинации этих элементов, например варианты использования внутри классов пользователей. Не существует единственно правильного метода организации; выберите тот, при котором пользователям будет легче понять предполагаемые возможности продукта. Мы опишем схему функций на примере.

3.x Функция системы X

Опишите название особенности несколькими словами, например «3.1 Проверка правописания». Так же назовите подразделы с 3.x.1 по 3.x.3 для каждой функции системы.

3.x.1 Описание

Кратко опишите функцию системы и укажите ее приоритет: высокий, средний или низкий приоритетом . Приоритеты являются динамичной характеристикой, они могут изменяться в ходе проекта. Если вы используете средство управления требованиями, определите атрибут требований для обозначения приоритета.

3.x.2 Функциональные требования

Перечислите по пунктам конкретные функциональные требования, которые связаны с этой функцией. Именно эти функции ПО нужно реализовать, чтобы пользователь мог использовать сервисы этой функции или реализовать вариант использования. Опишите, как продукт должен реагировать на ожидаемые ошибки, неправильный ввод информации или неверные действия. Присвойте каждому функциональному требованию.

4. Требования к данным

Используйте этот раздел шаблона для описания 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.

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.

4.3 Отчеты

Если приложение будет генерировать отчеты, перечислите их здесь и опишите их характеристики. Сосредоточьтесь на логических описаниях, порядке сортировки, уровнях суммирования и т. п., отложив подробный макет до этапа дизайна.

4.4 Получение, целостность, хранение и утилизация данных

Если это важно, опишите, как получают и обслуживают данные. Укажите все требования, относящиеся к защите целостности данных системы. Укажите все процедуры, которые могут потребоваться, например резервное копирование, создание контрольных точек, зеркальное отображение или проверка корректности данных. Укажите политики, которые должна применять система для хранения или утилизации данных, в том числе временных данных, метаданных, остаточных данных (таких как удаленные записи), данных в кеше, локальных копий, архивов и промежуточных архивов.

5. Требования к внешним интерфейсам

В этом разделе 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. Например, ссылка может указать на руководство по работе с устройством, где перечислены коды ошибок, которые устройство может отправить программе.

5.1 Пользовательские интерфейсы

Опишите логические характеристики каждого пользовательского интерфейса, который необходим системе.

Некоторые особенные характеристики пользовательских интерфейсов:

- ссылки на стандарты графического интерфейса пользователей или стилевые рекомендации для семейства продуктов;
- стандарты шрифтов, значков, названий кнопок, изображений, цветовых схем, 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 (такие как 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

Опишите связи продукта и других компонентов ПО, в том числе другие приложения, базы данных, операционные системы, средства, библиотеки, веб-сайты и интегрированные серийные компоненты. Укажите назначение, форматы и содержимое сообщений, данных и контрольных значений, обмен которыми происходит между компонентами ПО. Опишите соответствия между входными и выходными данными между системами и все преобразования, которые должны происходить с данными при перемещении между системами. Опишите службы, необходимые внешним компонентам ПО, и природу взаимодействия между компонентами. Определите данные, которыми будут обмениваться и к которым будут иметь общий доступ компоненты ПО. Определите нефункциональные требования, влияющие на интерфейс, такие как уровни обслуживания для времени и частоты отклика или меры и ограничения безопасности. Часть этой информации может быть определена как требования к данным в разделе 4 или как требования к взаимодействию в разделе «6. Атрибуты качества».

5.3 Интерфейсы оборудования

Опишите 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.

5.4 Коммуникационные интерфейсы

Укажите требования для любых функций взаимодействия, которые будут использоваться продуктом, включая электронную почту, веб-браузер, сетевые протоколы и электронные формы. Определите соответствующие форматы сообщений. Опишите особенности безопасности взаимодействия или шифрования, скорости передачи данных и механизмов согласования и синхронизации. Укажите все ограничения этих интерфейсов, например допустимость тех или иных типов вложений в сообщениях электронной почты.

6. Атрибуты качества

В этом разделе описываются нефункциональные требования. Эти характеристики должны быть точно определены и поддаваться проверке и измерению. Укажите 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.

6.2 Производительность

Укажите 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

Укажите все требования, касающиеся безопасности или конфиденциальности, которые ограничивают доступ или возможности использования продукта. Это может быть физическая безопасность, а также защита данных или ПО. Источником требований к безопасности являются бизнес-правила и юридические законы, например:

GDPR (Общее положение о защите данных)
HIPAA (Закон о переносимости и подотчетности медицинского страхования)
PCI DSS (Стандарт безопасности данных индустрии платежных карт)
CCPA (Калифорнийский закон о конфиденциальности потребителей)

6.4 Техника безопасности

В этом разделе укажите требования, связанные с возможными потерями, повреждениями или ущербом, которые могут быть результатом использования продукта. Определите упреждающие действия, которые можно предпринять.

6.x [Другие]

Создайте 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

Такие требования могут быть направлены на разрешение различий в валютах, форматировании дат, чисел, адресов и телефонных номеров, языках, в том числе различных вариантах одного языка (например, американского и британского вариантов английского), 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.

8. [Остальные требования]

Определите все другие требования, 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, к мониторингу.

Приложение А. Словарь терминов

Определите все специальные термины, которые читателю необходимо знать для правильного понимания спецификации требований к ПО, включая сокращения и аббревиатуры. Расшифруйте каждое сокращение и приведите его определение.

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». Часто читателю удобнее, когда определенные модели внедрены в соответствующие разделы спецификации, а не собраны скопом в конце.

Спецификация требований в проектах гибкой разработки

В проектах гибкой разработки (agile) применяется ряд способов определения требований, которые отличаются от только что описанного метода.

Во многих проектах гибкой разработки на этапе выявления требований используются пользовательские истории.

Пользовательские истории накапливаются и приоритизируются в *резерве продукта (product backlog)**, который меняется на протяжении всего проекта. Крупные истории, которые охватывают значительную функциональность и которые нельзя реализовать в одной итерации, разбиваются на более мелкие, которые назначаются конкретным итерациям.

Когда команда приступает к очередной итерации, каждая история, назначенная на эту итерацию, уточняется и наполняется деталями в процессе обсуждения между владельцем продукта, людьми, выполняющими роль бизнес-аналитика, аналитиками, тестировщиками и пользователями. То есть спецификация подразумевает постепенное уточнение подробностей на правильном этапе проекта.

Однако в проектах гибкой разработки эти подробности часто представляются в форме приемочных тестов, описывающих, как система должна вести себя при правильной реализации истории. Тесты истории проводятся во время итерации, в которой реализуется эта история, и в будущих итерациях в рамках регрессионного тестирования.

* Бэклог продукта — это **список рабочих задач, расположенных в порядке важности, для команды разработчиков**. Наиболее важные задачи расположены в начале бэклога продукта, чтобы команда понимала, какую работу следует выполнить в первую очередь. Иногда этот список называют списком дел и считают «артефактом» в методе разработки программного обеспечения scrum.

Спецификация требований в проектах гибкой разработки

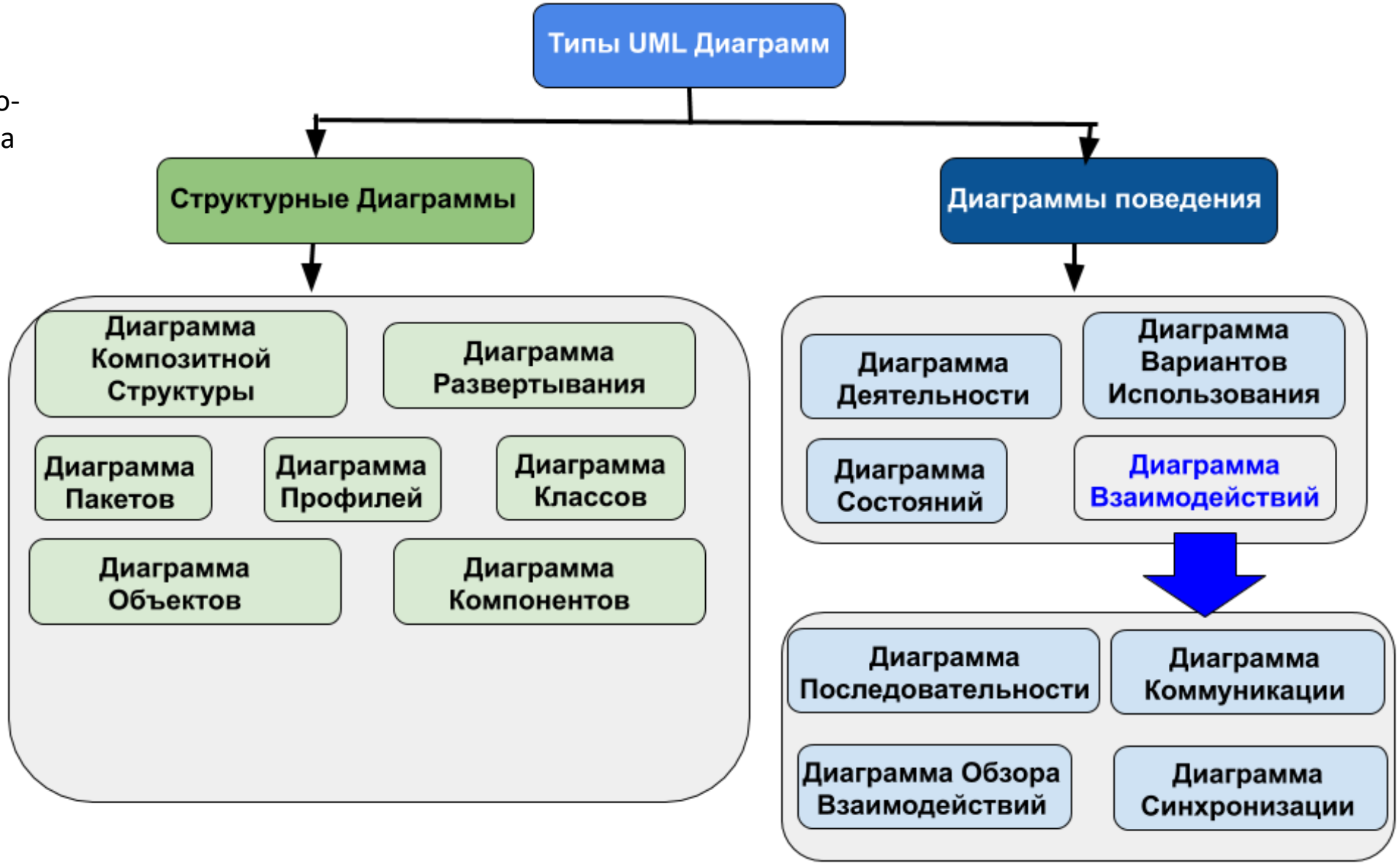
Нефункциональные требования относящиеся к определенной пользовательской истории можно записывать в форме критериев приемки.

Например, *тесты безопасности могут демонстрировать, что только определенным пользователям разрешен доступ к функциональности, описанной в соответствующей пользовательской истории, а остальным пользователям доступ закрыт.*

Ограничения/нефункциональные требования тоже можно представить **в виде пользовательских историй:**

- ✓ Как клиент, я хочу иметь возможность запускать систему во всех версиях Windows, начиная с Windows 95.
- ✓ Как технический директор, я хочу, чтобы система использовала нашу существующую базу данных заказов, а не создавала новую, чтобы нам не приходилось поддерживать еще одну базу данных.
- ✓ Как пользователь, я хочу, чтобы сайт был доступен 99,999% времени, когда я пытаюсь получить к нему доступ, чтобы я не расстраивался и не искал другой сайт для использования.
- ✓ Как человек, говорящий на японском языке, я, возможно, когда-нибудь захочу запустить ваше программное обеспечение.

UML – (Unified Modeling Language) – это система обозначений, которая применяется для объектно-ориентированного анализа и проектирования и **используется для визуализации, конструирования и документирования программных систем.**



https://habr.com/ru/articles/508710/

статья: UML как альтернатива техническим заданиям

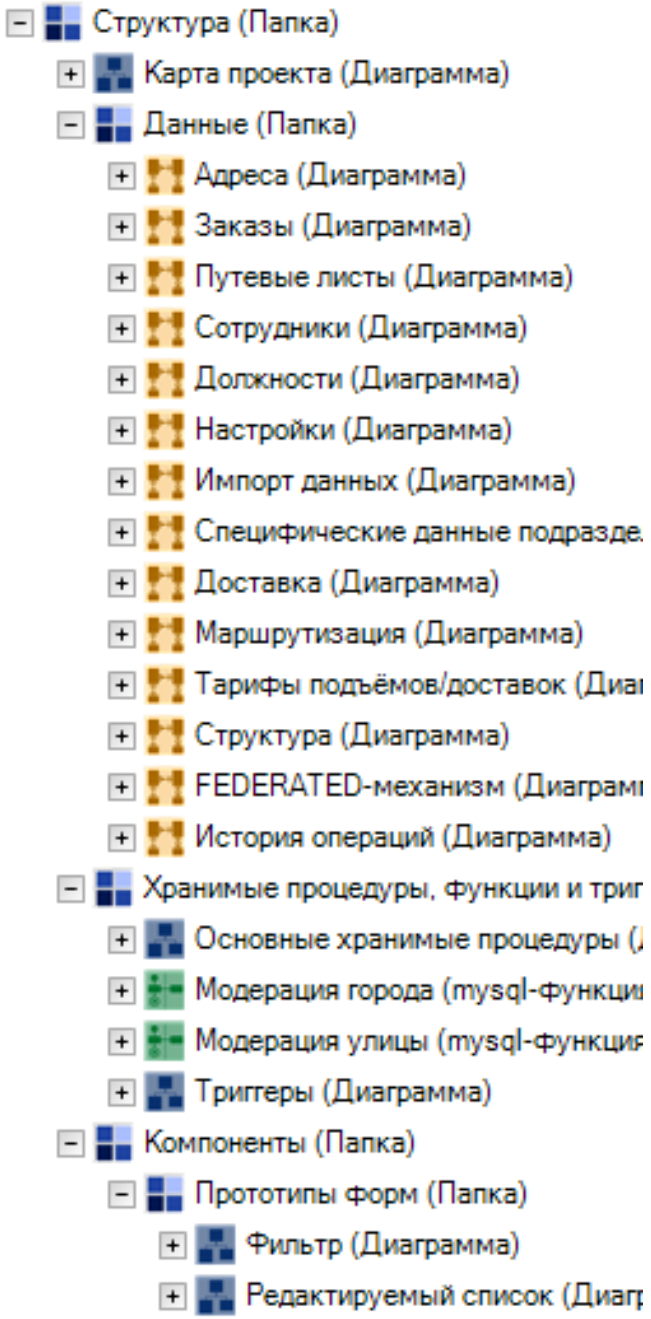
https://www.webformat.ru/blog/it-special/uml-kak-alternativa-tekhnicheskim-zadaniyam/

Если речь идет о **сложных логистических процессах, объектах и связях между ними, то UML позволяет описать задачу оптимальным образом**. В то время как техническое задание – это преимущественно текстовое описание задачи, которое хорошо работает для простых задач.

Преимущества:

- ✓ просто ориентироваться;
- ✓ можно легко изменять “масштаб” описания блоков: сворачивать лишние и разворачивать нужные.
- ✓ можно использовать как графическое, так и текстовое описание одновременно (и каждое из них именно там, где нужно);
- ✓ можно объединять и структурно организовывать десятки диаграмм;
- ✓ можно видеть крупные блоки “сверху”, с основными связями, а также иметь возможность опуститься до базового уровня с полным описанием, структурой таблиц, детализацией связей и чуть ли не примерами кода.

Вот так, например, выглядит часть дерева диаграмм проекта



Заключение

Выбор надлежащих форм спецификации требований к ПО — исключительная прерогатива команды проекта.

Г**лавная цель разработки требований** — собрать согласованное понимание требований, качество которых *достаточно*, чтобы приступить к разработке следующей части продукта и продолжить работу при приемлемом уровне риска.

Независимо от типа создаваемого командой продукта, используемой методики разработки или используемых бизнес-аналитиком приемов выявления требований, **эффективная спецификация требований жизненно важна для успеха.**

Примеры

Пример спецификации онлайн-системы для управления рейсами и пассажирами по рассмотренному шаблону https://krazytech.com/projects/sample-software-requirements-specificationsrs-report-airline-database

Пример спецификации веб-приложения «Электронная запись кредитный аукцион» https://drive.google.com/file/d/0B5xz4Vc-CUY1NjVoT0pFcnlEQU0/view?resourcekey=0-nlfqQymSe_F_NotsPEQGaA

Пример спецификации приложения «Система учёта запчастей» https://akiselev87.files.wordpress.com/2011/03/d0b5d181d183d0b7.pdf