Собирая требования и проектируя систему, мы должны формировать её модель: отдельные артефакты (тексты, схемы, таблицы и прочее) должны укладываться в связанную, непротиворечивую модель будущей системы, которая по мере разработки должна обогащаться подробностями и техническими деталями, становясь в итоге документацией уже разработанной системы.

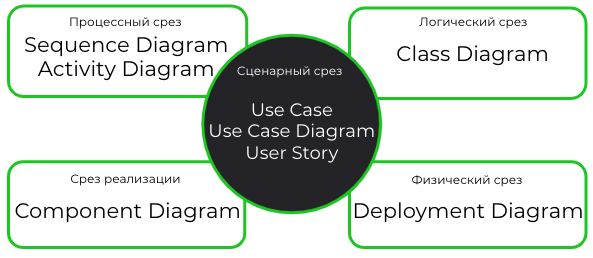
Какие именно аспекты будущей системы мы хотим описать в артефактах определяет то, какие срезы (*view*) есть в модели — то есть артефакты каких типов мы должны создавать. Вариантов таких срезов придумано много, один из самых простых и известных — «модель 4+1».



* + - Процессный — описание того, как ведут себя сами компоненты — какое у них поведение, как они взаимодействуют между собой.
    - Реализации — из каких компонентов состоит наша система, за какие функции они отвечают.
    - Логический — какие данные есть в системе: какие сущности и как они между собой связаны.
    - Физический — как *физически* устроена система: какое аппаратное обеспечение она использует, на каких серверах развёрнута.
    - Сценарный — дополнительный, «плюс один», срез который помогает связать остальные между собой.

Для каждого среза мы можем использовать свои артефакты. Если говорить про *UML* (он потому и унифицированный в том числе, что для каждого среза в нём есть свой тип диаграмм), то, например, таким образом:

* + - Процессный: *Sequence* *Diagram*, *Activity* *Diagram*
    - Реализации: *Component* *Diagram*
    - Логический: *Class* *Diagram*
    - Физический: *Deployment* *Diagram*
    - Сценарный: *Use* *Case* *Diagram*



Сами по себе эти артефакты можно считать представлением требований, разрабатываемых системой (а затем и спецификацией работы уже созданной системы) — они могут быть, например, разделами базы знаний в *Confluence* или *Wiki*. Но из них мы можем получить и другие представления — например, задачу в разработку или документ, где артефакты станут наполнением разделов в нём. При этом важно, чтобы это всё равно были срезы одной модели — связанные между собой и непротиворечивые.

В зависимости от того, как устроена команда и конкретный проект, процесс передачи требований — **постановка задач в разработку** — может быть устроен по-разному.

В разработку могут уходить сразу пользовательские требования, например, **пользовательские истории**.

Подробнее



Если вы принимаете участие в проектировании вместе с разработкой — детализируйте задачу, описывая как нужно сделать, это стоит это делать после вдумчивого проектирования и обсуждения. Иначе оставьте только пользовательское требование, либо совсем верхнеуровневое системное и дайте разработчикам самим определить техническое решение и декомпозировать задачу, но максимально понятно и подробно опишите, **что** нужно сделать. Здесь помогут пользовательские истории с критериями приёмки, сценарии, описания бизнес-процессов.

Общие рекомендации при постановке задачи в разработку:

* + 1. Приложите все нужные материалы: дайте ссылки или приложите на конкретные материалы, на которые вы ссылаетесь (спецификации, описания *API*, схемы, макеты, бизнес-правила и так далее).
    2. Опишите задачу подробно.
    3. Если задача из нескольких шагов — приложите или опишите сценарий работы (алгоритм).
    4. Обязательно дайте контекст (смысл) задачи, который поможет понять, зачем мы её реализуем.

**Пример** кратко описанной задач, которая оставляет много вопросов:

Сервис пользовательского профиля должен отправлять команду на отправку СМС при появлении новой записи в БД.

**Пример** подробно описанной задачи, которая содержит ссылки на материалы, даёт контекст, подробно описывает, как должна работать система:

Новые пользователи часто не знают о возможностях продукта, поэтому им после регистрации стоит отправлять поясняющую СМС.

Сервис пользовательского профиля должен отправлять вызывать метод POST /send-sms сервиса нотификаций (ссылка на спецификацию) при появлении новой записи в таблице users БД:

* + 1. При появлении новой записи срабатывает триггер на отправку сообщения.
    2. Формируется текст сообщения исходя из шаблона, хранящегося в таблице notification-template.
    3. Определяется номер телефона пользователя, на который нужно отправить СМС: по последней дате last-used записи таблицы phone-numbers, привязанной к этому пользователю.
    4. С номером телефона и текстом вызывается метод POST /send-sms (ссылка на спецификацию).

Из отдельных артефактов можно собрать разные документы — с разным составом разделов и глубиной детальности в них, а стандартная структура таких документов, фактически — **шаблон**, может помочь не забыть полезные артефакты.

Разные стандарты и своды знаний предлагают писать разные документы на разных этапах жизненного цикла системы, например, вот такой состав документа предлагает вести *BABoK*:

* + - *Current* *state* *analysis* *document*
    - *Project* *vision* *document*
    - *Solution* *vision* *document*
    - *Business* *requirements* *document*
    - *Business* *process* *design* *document*
    - *Use* *case* *model* *document*
    - *Use* *case* *specification* *document*
    - *System*-*wide* *requirements* *document*
    - *Solution* *glossary*
    - *Functional* *requirement* *specification* (*FRS*)
    - *Software* *requirement* *specification* (*SRS*)

С точки зрения системного анализа, наиболее важные **два документа** — это *Vision*, концепция (*Project* *vision* *document*) и *SRS*, спецификация требований (*Software* *requirement* *specification*). Плюс, есть ещё один немного специфичный отечественный тип документов: техническое задание.

* + 1. **Концепция решения** (*Vision*) — документ первичной стадии анализа.
    2. **Спецификация требований** (*Software* *Requirements* *Specification*) — документ более детальной проработки.
    3. **Техническое задание** — специфичный отечественный документ, совмещающий спецификацию требований и проектные разделы (стадии создания, порядок приёмки, экономическое обоснование внедрения и другие).

**Концепция** составляется на этапе первичного сбора требований, содержит бизнес- и пользовательские требования, описывает цели системы, раскрывает предметную область, но практически не затрагивает реализацию и фиксирует лишь основные верхнеуровневые системные требования.

Документ-концепция обычно состоит из разделов, описывающих:

* + - назначение системы
    - границы системы
    - бизнес-требования (цели создания системы)
    - пользовательские требования
    - описание предметной области: глоссарий и основные сущности
    - основные функции, которые видны на этом уровне проработки
    - список стейкхолдеров и их описание
    - список типов пользователей и их описание
    - ограничения, в которых будет работать система
    - основные общие нефункциональные требования, которые видны на этом уровне проработки
    - требования к документации системы
    - бизнес-процессы *AS* *IS* и *TO* *BE*

Состав разделов документа может меняться, например бизнес-процессы зачастую на этом этапе проработки могут быть не формализованы, но если вдруг они проработаны, хотя бы верхнеуровнево — это сильно поможет в дальнейшем проектировании и их стоит включить в концепцию.

**Спецификация требований** — это большой документ, описывающий системные требования к будущему решению. В целом спецификация требований может быть для системы (*SysRS*) и программного обеспечения (*SRS*) — система состоит не только из ПО, но ещё, например, из аппаратного обеспечения, на котором оно работает.

Спецификация требований, в соответствии со стандартом *IEEE*-29148, состоит, не считая ссылок на внешние документы и приложений, из **двух основных разделов**:

* + - введение с кратким описанием системы;
    - самих требований разного типа.

Сами разделы уже содержат подразделы с подробностями:

**Введение**

* + - Назначение
    - Границы системы и бизнес-требования
    - Краткое описание контекста использования
    - Типы пользователей
    - Основные требования (пользовательские)
    - Ограничения
    - Определения

**Требования**

* + - Функциональные
    - Производительность
    - Удобство использования
    - Логическая модель данных
    - Программные интерфейсы
    - Пользовательские интерфейсы
    - Атрибуты качества

Подразделы введения фактически используют ту информацию, которую мы собрали на предыдущем этапе, создавая документ-концепцию, так как в целом спецификация требований развивает концепцию и детализирует её системные требования, собранные во второй раздел.

Давайте более подробно рассмотрим эти подразделы:

* + 1. **Назначение**: описывает назначение системы — какую общую бизнес-проблему система должна решать.
    2. **Границы системы**: кратко описывает, что система будет делать — что будет в границе её зоны ответственности — определяет, какие бизнес-цели система предназначена реализовывать. При необходимости можно отдельно указать, что находится вне границ системы. Определить границы системы помогает проработка сценариев и создание диаграммы сценариев.
    3. **Контекст использования**: кратко перечисляет, какие *API* и пользовательские интерфейсы система предоставляет и использует, какое аппаратное обеспечение нужно для её работы. При этом детальное описание требований к программным и пользовательским интерфейсам должно быть дано дальше в разделе требований. Для иллюстрации контекста использования системы хорошо подходит контекстная диаграмма (*DFD*-0).
    4. **Типы пользователей**: кратко перечисляет типы пользователей, требования которых система будет реализовывать, с их характеристиками. За основу можно взять информацию из документа-концепции.
    5. **Основные функции**: кратко перечисляет основные функции системы с точки зрения пользователей — фактически, пользовательские функциональные требования. Здесь можно использовать информацию из пользовательских историй.
    6. **Ограничения**: основные ограничения, накладываемые на систему или её разработку внешней средой — технические ограничения, ограничения регуляторов, лицензионные ограничения и т. д.
    7. **Определения**. Глоссарий терминов и расшифровка аббревиатур. Эти термины должны использоваться единообразно дальше по всему документу.
    8. **Функциональные требования**: список функциональных системных требований. Их стоит сгруппировать, например, по одному из признаков: типам пользователей, из требований которых вытекают системные требования к функциям, исходным пользовательским требованиям, сценариям использования, режимам работы. На практике, самая удобная группировка — по сценариям, с расписыванием самих сценариев и группировкой функциональных требований, нужных для реализации этих сценариев, под ними. В предельном случае, когда нет нужды или времени выделять отдельные более детальные функциональные требования, можно оставить только сценарии использования.
    9. **Требования к производительности**: требования к количеству обрабатываемых запросов в единицу времени, объёму данных, быстродействию и так далее. Нужно стараться указывать измеримые характеристики.
    10. **Требования к удобству использования**: должны быть конкретными, если их невозможно сформулировать — раздел не пишут.
    11. Требования к модели данных: включает описание основных сущностей, их атрибутов и связей — как минимум на концептуальном уровне, если уже началось проектирование — то и логическую структуру БД. Здесь пригодятся *UML* *Class* *Diagram* или *ERD*.
    12. **Программные интерфейсы**: описание *API*, которое система будет предоставлять. Стоит расписать принципы, по которым будет строиться *API* (*REST*/*RPC*/*SOAP*, файловый обмен и т. д.), методы, их входящие и возвращаемые параметры. Здесь можно ссылаться на объекты логической структуры данных. При этом если детальное проектирование ещё не началось — можно описывать смысл методов, без всех подробностей.
    13. **Пользовательские интерфейсы**: структура и описание экранов пользовательского интерфейса. Здесь очень полезно добавлять макеты и снабжать их кратким описанием экранов.
    14. **Атрибуты качествам**: остальные нефункциональные требования, описывающие качества работы системы, такие как надёжность, доступность, информационная безопасность, поддерживаемость и так далее.

Если проект уже на стадии проектирования, то можно описывать в спецификации больше подробностей — уходить в проектирование и включить в документ архитектурные решения:

* + - архитектурные принципы, по которым строится системами,
    - какие основные компоненты она будет включать,
    - как устроены потоки данных между компонентами,
    - как распределяются функции между компонентами.

Но в этом случае спецификация требований начинает содержать не только сами требования, но фактически становится проектировочным документом.

В России есть свой набор стандартов на разработку программного обеспечения и систем родом из Советского Союза, который, к сожалению, очень давно не обновлялся. Здесь про эти стандарты мы не будем много рассказывать, если кому-то интересны подробности, можно посмотреть серию открытых вебинаров:

* + - [«ГОСТ 19 в современной разработке»](https://www.youtube.com/watch?v=1P18VkS7ORQ);
    - [«ГОСТ 34 в современной разработке, часть 1»](https://www.youtube.com/watch?v=Cb7oyeIjWZ8).

Эти стандарты собраны в две группы — серии: 34-я и 19-я:

* + - ГОСТ 19 — стандарты на программное обеспечение;
    - ГОСТ 34 — стандарты на автоматизированную систему.

**Автоматизированная система** состоит из:

* + 1. Объекта автоматизации — предприятия или бизнес-процесса, который и должна автоматизировать система.
    2. Персонала, который будет обслуживать систему (и весь автоматизируемый бизнес-процесс).
    3. Технических средств — аппаратного обеспечения, которое задействовано в системе. Это оборудование, которое необходимо для работы ПО (вроде серверов), или любое оборудование в целом, находящееся на объекте автоматизации (например, станки, которыми управляет системами). Документация на технические средства создаётся в соответствии со стандартами отдельной серии — «Единой системы конструкторской документации» (ЕСКД).
    4. Программного обеспечения — набора программ, который будет использоваться в системе. Документация к программам описывается как раз по стандартам ГОСТ 19 (Единая система программной документации — ЕСПД). Таким образом, ПО — это лишь часть автоматизированной системы.
    5. Информационного обеспечения — данных, которые хранятся и обрабатываются в системе.

**Процесс разработки** систем и программного обеспечения по ГОСТ строго водопадный (но в самом стандарте можно найти оговорки о том, что стадии могут изменяться в зависимости от проекта).

Он состоит из последовательных этапов:

* + 1. Анализ объекта автоматизации и создание концепции.
    2. Сбор системных требований и написание ТЗ.
    3. Создание эскизного проекта.
    4. Создание технического проекта.
    5. Создание рабочей документации.

Для каждого из этапов ГОСТ предполагает **написание большого набора документов** и регламентирует их содержание.

Нас интересует в первую очередь техническое задание. ГОСТ различает ТЗ для программного обеспечения и автоматизированной системы — они в разных сериях:

* + - ГОСТ 19.201-78 «Техническое задание, требования к содержанию и оформлению»;
    - ГОСТ 34.602-89 «Техническое задание на создание автоматизированной системы».

ТЗ по ГОСТ-19 должно содержать разделы:

* + - «Введение»;
    - «Основания для разработки»;
    - «Назначение разработки»;
    - «Требования к программе или программному изделию»;
    - «Требования к программной документации»;
    - «Технико-экономические показатели»;
    - «Стадии и этапы разработки»;
    - «Порядок контроля и приёмки».

Как можно заметить, ТЗ — это не только спецификация требований, но и во многом **проектный документ**.

Требований в нём касается раздел «Требования к программе или программному изделию», содержащий подразделы для разных типов требований:

* + - «Требования к функциональным характеристикам»;
    - «Требования к надёжности»;
    - «Условия эксплуатации»;
    - «Требования к составу и параметрам технических средств»;
    - «Требования к информационной и программной совместимости»;
    - «Требования к маркировке и упаковке»;
    - «Требования к транспортированию и хранению»;
    - «Специальные требования».

Техническое задание по ГОСТ 34 более обширный документ, содержащий разделы:

* + - «Общие сведения»;
    - «Назначение и цели создания (развития) системы»;
    - «Характеристика объектов автоматизации»;
    - «Требования к системе»;
    - «Состав и содержание работ по созданию системы»;
    - «Порядок контроля и приёмки системы»;
    - «Требования к составу и содержанию работ по подготовке объекта автоматизации к вводу системы в действие»;
    - «Требования к документированию»;
    - «Источники разработки».

Так же как в ТЗ по ГОСТ 19, здесь много разделов не относятся непосредственно к самим требованиям, а описывают **объект автоматизации** (что близко к содержанию документа-концепции) и различную проектную информацию в целом.

Сам же раздел требований к системе содержит подразделы:

* + - «Требования к системе в целом»;
    - «Требования к структуре и функционированию системы»;
    - «Требования к численности и квалификации персонала системы и режиму его работы»;
    - «Показатели назначения»;
    - «Требования к надёжности»;
    - «Требования безопасности»;
    - «Требования к эргономике и технической эстетике»;
    - «Требования к эксплуатации, техническому обслуживанию, ремонту и хранению компонентов системы»;
    - «Требования к защите информации от несанкционированного доступа»;
    - «Требования по сохранности информации при авариях»;
    - «Требования к защите от влияния внешних воздействий»;
    - «Требования к патентной чистоте»;
    - «Требования по стандартизации и унификации»;
    - «Дополнительные требования»;
    - «Требования к функциям (задачам), выполняемым системой»;
    - «Требования к видам обеспечения: информационному, лингвистическому, программному, техническому, метрологическому, организационному, методическому и другим видам обеспечения системы».

При работе с документацией по ГОСТ стоит помнить:

* + - различия между ПО и системой и выбирать правильный документ;
    - состав документов и разделов в них может меняться;
    - есть проблемы в стандарте — устаревшая терминология и подход к рассмотрению системы, нечёткие формулировки, нет многих важных понятий;
    - кроме самой спецификации требований содержит ещё проектные разделы;
    - работать стоит от модели, встраивая в структуру документа артефакты требований.