**Требования к ПО**

Коротко о главном, с точки зрения разработки ПО и тестирования в частности. Будет интересно почитать начинающим тестировщикам, аналитикам, ПО, заказчикам.

**Требование** — слово, которое часто встречается в книгах, статьях, на конференциях, при общении с коллегами, заказчиками, в общем много где, но значимость которого часто недооценивается, так как все “понимают, о чем речь”.

**НО,** как показывает практика, **это совсем не так**.

***Требование (requirement) —*** *описание функции и/или условия, которое должно соблюдаться приложением в процессе решения пользовательской задачи.*

Зачем нужны требования?

Вопрос, который часто можно услышать от заказчиков / менеджеров / ПО, которые знают :) определение, но не понимают его значимости.

На этот вопрос есть несколько ответов.

Требования необходимы, потому что они:

* Позволяют понять **что** система **должна** делать
* Являются **основой** для формирования **плана проекта**
* Помогают **предотвращать** и **разрешать** конфликтные ситуации
* Упрощают **расстановку приоритетов**
* Дают возможность **оценить масштаб** изменений
* Позволяют **оценить степень прогресса** в разработке

Требования — это “скелет” каждого проекта. Проект без требований — это эквивалент [амёбы](https://ru.wikipedia.org/wiki/%D0%90%D0%BC%D1%91%D0%B1%D1%8B). Надеюсь нет такого заказчикика, который хотел бы, чтоб его продукт был таким :)

Итак,

1. Что такое — разобрались
2. Зачем нужны — разобрались
3. Припустим написали — …

Дефекты в требованиях

Наличие требований, которые **написаны не правильно** — **не спасет вас от проблем**, а может их только **усугубить**.

Многие **не уделяют** внимания написанию требований, откладывают их на потом или пишут их “на коленке за 5 минут”. Из-за этого в **таких** требованиях часто встречаются ошибки, а как все понимают — **ошибки стоят денег**.

Давайте рассмотрим на примере, к чему может привести и сколько будет стоить **одна и таже ошибка в требованиях,** найденная на разных этапах разработки.

Припустим, заказчик не верно указал перечень поддерживаемых форматов файлов для проверки их содержимого.

На первый взгляд — ничего страшного не произошло.

Одним форматом больше, одним меньше — какая разница. Ведь это быстро делается.

Заказчик

Давайте проверять, на сколько эта ошибка страшна на самом деле.

Рассмотрим 3 варианта:

**1)** Если дефект в требованиях найден на **раннем** этапе разработки (можно назвать его *подготовительным этапом*) — стоимость исправления будет эквивалентна сумме:

* стоимости исправления требования (*указать верные форматы*)
* стоимости оповещения об исправлении требования

**2)** Если дефект в требованиях найден на **познем** этапе разработки (какая-то часть требований уже реализована)— стоимость исправления будет эквивалентна сумме:

* стоимости исправления требования
* стоимости оповещения об исправлении требования
* стоимости исправления дизайна (*замена сообщений на сайте, в нотификациях, кнопках и тд.*)
* стоимости исправления кода (*реализация поддержки новых форматов + правки по дизайну*)
* стоимости исправления тестов (*исправление связанных тест-кейсов*)
* стоимости повторного тестирования

**3)** В **самом худшем** сценарии, ошибка приведет к:

* невозможности реализации исправленного требования при помощи выбранных технологий, что эквивалентно полной переработке продукта — все ресурсы потрачены зря. (*указанный формат не может быть проверен при помощи выбранной технологии*)
* не полной реализации требования, отказа от него или отказа от его части, что ухудшает продукт с точки зрения клиента, особенно если он ждет, что требование будет реализованно. В таком случае часть ресурсов, которая пошла на анализ, подготовку, реализацию, потрачена зря + получаем не довольного клиента.

Итоги:

* Требования нужно писать, и писать их нужно **внимательно** и **вдумчиво.**
* **Требования нужно тестировать**, и как можно раньше.

**Техники тестирования требований**

Тестирование требований относится к нефункциональному тестированию (non-functional testing).

Основными техниками являются:

1. **Взаимный просмотр**. Требования вдумчиво перечитывают несколько человек, высказывая вопросы, замечания.
2. **Вопросы. Уточняющие вопросы.** Любая неточность или размытость в требованиях может содержать основу для будущих дефектов. Вопросы нужно задавать до тех пор, пока требование не будет точным, понятным и **однозначным**.
3. **Продумывание тест-кейсов (чек-листов).** По ходу прочтения требований нужно представлять, как их нужно будет проверять. Если в голову ничего не приходит — значит с требованием что то не так. Нужно задавать вопросы.

* **Исследование системы.** Иногда требования могут быть протеворечивыми, по этому нужно “представлять” систему и пытаться сопоставлять требования между собой.
* **Визуализация**. Если существуют алгоритмы работы системы, будущие бизнес процессы, UML диаграммы — их тоже нужно анализировать. Они помогают определять места, которые не продуманы, либо не стыкуются между собой.
* **Прототипирование**. Анализ прототипов дает теже результаты, что и визуализация, только со стороны конечного пользователя системы.

Выбор техник зависит от проекта, по этому их набор может изменяться.

Главное не забывать, что они существуют, и что ими нужно пользоваться.

# Источники требований

Все требования приходят от заказчиков, или людей связанных с ними (сотрудников, пользователей и тп.)

Для выявления требований чаще всего используются следующие техники:

* **Интервью** (либо переписка) – опрос, часто в формате вопрос ответ между аналитиком и заказчиком / пользователем
* **Фокусные группы** – расширенное интервью с несколькими пользователями
* **Мозговой штурм** – позволяет за короткий промежуток времени собрать большое количество идей, которые в дальнейшем изучаются и анализируются
* **Наблюдение** – позволяет выявить процессы, о которых не упомянули в интервью, занимает много времени
* **Прототипирование** – один из лучших способов понять и уточнить требования

Так же существуют более сложные методы, при котором аналитик должен «сам во всем разобраться», и уточнить собранную информацию у заказчиков:

* Анализ документов
* Моделирование процессов
* Самостоятельное описание

**Типы требований**

Почти в каждом проекте существует 3 заинтересованных стороны:

1. Заказчики продукта
2. Пользователи продукта
3. Разработчики продукта

Все они смотрят на «продукт» со своей колокольни, следовательно требования разделяются на соответствующие группы или уровни.

1. **Уровень заказчика (бизнес-требований)**

На этом уровне находится только один тип требований – бизнес требования (business requirements).

Они выражают цель, ради которой создается продукт (зачем он вообще нужен, каким образом он будет приносить прибыль). Они часто представлены простым текстом, без каких либо технических подробностей.

*Нужен инструмент, позволяющий помочь отделу поддержки проверять качество выполнения заказов.*

*Нужно автоматизировать процесс начисления бонусов за приглашенных друзей.*

Основываясь на этих требованиях можно получить общее видение проекта.

1. **Уровень пользователя**

Описывают “взгляд” на продукт глазами пользователя.

Включает в себя:

**Пользовательские требования** (user requirements) — описание задач, которые может выполнять пользователь при помощи системы. Оформляются в виде пользовательских историй (user stories, cases, scenarios). Эти требования могут быть использованы для оценки времени, сложности, стоимости разработки.

После первого входа сотрудника отдела поддержки в систему отображается обучающее видео для ознакомления с функциями приложения.

Аналитик должен иметь возможность в любой момент получить отчет о приглашенных друзьях за указанный промежуток времени.

Пример оформления этих же требований в виде **User Stories**

Как сотрудник отдела поддержки, я могу просмотреть обучающее видео для ознакомления с функциями приложения после первого входа в систему, чтоб понимать, что я могу в ней делать

Как аналитик, я могу заполнить и отправить форму “получить отчет о друзьях”, и мне на почту придет письмо с .xls отчетом, содержащий информацию о приглашенных друзьях за указанный мной промежуток времени

**Бизнес правила** (business rules) — описывают особенности принятых в предметной области процессов, ограничений, правил.

*Доступ к инструменту предоставляется только сотрудникам поддержки уровня Main support и выше.*

*Аналитики не должны иметь возможности изменять полученные отчеты.*

**Атрибуты качества** (quality attributes) — описывают ключевые показатели качества продукта.

*Не зависимо от количества заказов, максимальное время открытия списка проверок не должно превышать 5 секунд.*

*Система должна работать с большими объёмами данных (сотни тысяч записей).*

1. **Уровень разработки (продуктных требований)**

Содержит наиболее детализированное описание функций продукта, которые должны быть реализованными.

Конечным документом, содержащим все требования уровня разработки является **Спецификация требований** (software requirements specification, SRS). Часто это объемный документ, содержащий сотни страниц.

К уровню разработки относятся следующие типы требований:

**Функциональные требования** (functional requirements) — описывают что **должна** и что **НЕ** **должна** делать система.

*Список проверок должен быть отсортирован по конечной дате выполнения (deadline) заказа.*

*Никакая личная информация пользователя (логин, пароль, номера телефонов, и тд.) не должна отображаться в отчетах.*

**Нефункциональные требования** (non-functional requirements) — описывают свойства системы при реализации своего поведения. (По сути это более техническое и детальное описание атрибутов качества)

*Приложение должно поддерживать работу с мобильных устройств (минимальная ширина экрана – 320 px).*

*Обьем используемой оперативной памяти не должен превышать 256 Мб .*

Как показывает практика, не функциональных требований часто больше, чем функциональных, по-этому их можно разбивать на подгруппы.

**Не функциональные требования**

Основными подгруппами являются:

* **Ограничения** (limitations) — факторы, которые ограничивают выбор способов и средств реализации продукта.

*Приложение должно работать в последних версиях браузеров Chrome, Firefox, Safari.*

*Приложение должно работать на Raspberry PI 3b+.*

* **Требования к интерфейсам** (external interfaces requirements) — особенности взаимодействия системы с другими системами

*Весь трафик между браузером и сервером должен быть зашифрован (HTTPS соединение).*

*Отправка письма с отчетом на почту аналитиков должно выполняться согласно RFC3207 (SMTP over TLS).*

* **Требования к данным** (data requirements) — описывают структуры данных, описания баз данных, особенности их использования.

*Все данные системы должны храниться в БД под управлением СУБД MySQL.*

*Для ускорения поиска данных по определенному пользователю должны быть предусмотрены индексы по соответствующим полям таблицы.*

… можно выделять и другие подгруппы, так как эта группа требований основывается на атрибутах качества, которых может быть очень-очень много :)

Свойства качественных требований

Качественными можно называть требования, которые соответствует перечисленным ниже свойствам:

* **Завершенность (completeness)**. Требование содержит полный набор информации для его однозначного понимания. В нем не должно встречаться «очевидных», «и так всем понятных», «само собой разумеющееся» фраз, «размытых» формулировок (и т.п., и т.д., и других…, ну и в том же духе).
* **Атомарность (atomicity)**. Требование является атомарным, если его не можно разбить на отдельные требования, и оно описывает только одну функцию.
* **Последовательность (consistency)**. Требование не должно противоречить самому себе (да-да, и так бывает) или другим требованиям.
* **Недвусмысленность (clearness)**. Требование не должно содержать жаргона, неочевидных аббревиатур, расплывчатых формулировок. (Кстати, для улучшение этой характеристики есть хорошая практика – вести словарь проекта. Тогда понятий «клиент», «пользователь», «кастомер», «юзер» – пропадут, и появиться одно, однозначное – «студент», (если сервис предназначен для них) и все будут понимать друг друга с первого раза и 100% однозначно)
* **Выполнимость (feasibility)**. Требование должно быть реализуемым в рамках бюджета и срока разработки проекта. (Построить самолет для 150 человек за 2 недели – не возможно)
* **Обязательность (obligatoriness)**. Если требование не обязательное – его просто нужно выкинуть. **НО** если требование **нужное, но не критичное** – ему просто присваивают меньший приоритет.
* **Актуальность (up-to-date)**. Если требование не актуально (например описанный функционал убрали с проекта) – его нужно удалить.
* **Прослеживаемость (traceability)**. Бывает вертикальной – показывает связь с требованиями других уровней, или горизонтальной – показывает связь с тест-планом, тест-кейсами. Для обеспечения этой характеристики часто применяются специальные инструменты (requirements management tool) но даже без них у требования должны быть как минимум: уникальный номер, оглавление, валидные ссылки на другие требования / тест-кейсы.
* **Модифицируемость (modifiability)**. Исправление / внесение правок в требования должно быть простой операцией. Если с этим возникают проблемы – значит требование изначально не соответствует описанным характеристикам, и это нужно исправлять.
* **Важность (importance)**. Показывает на сколько требование важно для успеха проекта.
* **Стабильность (stability)**. Описывает вероятность правок в будущем. (Лично – никогда не встречал. Интересно, кто в команде может адекватно оценить будущее, астрологи??)
* **Срочность (priority)**. Показывает на сколько срочно должно быть реализовано требование (самое важное – делаем в первую очередь)
* **Проверяемость (verifiability)**. Показывает, что можно создать тест-кейс, который однозначно покажет, что требование реализовано верно.

Писать качественные требования, будучи не знакомым с этими характеристиками— не возможно. Для того, чтоб было проще — **можно создать чек-лист** с перечнем всех свойств и перепроверять себя по нему.