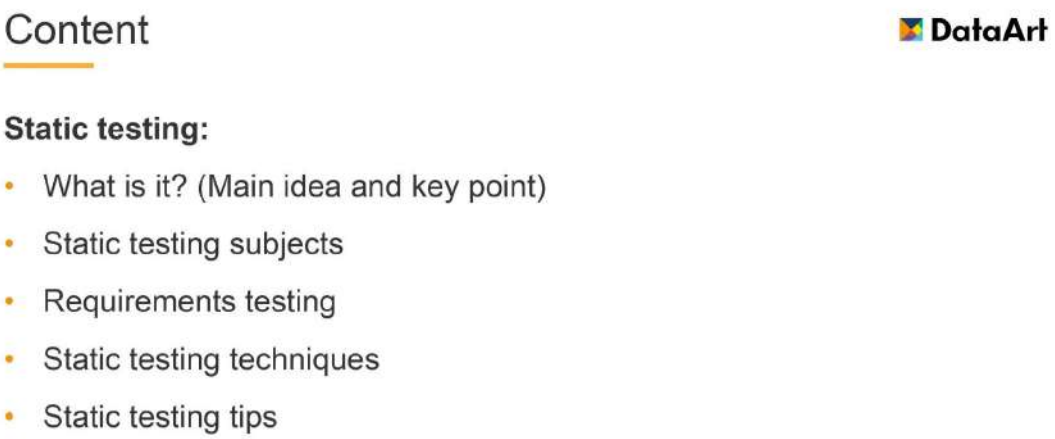
Статическое тестирование и тестирование требований.



- Что такое статическое тестирование? В частности, чем оно отличается от динамического?
- Что может быть объектом статического тестирования?
- Какие бывают техники статического тестирования?
- Какие полезные моменты стоит помнить, когда мы проводим статического тестирование?

What is it?

DataArt

Static testing is a software testing method that involves the examination of a program, along with any associated documents, but does not require the program to be executed.

In most cases the analysis is performed on some version of the source code.

Статического тестирование не предполагает выполнение программы, то есть это анализ документов, технической документации, исходников или какого-то статичного слепка системы. Мы не смотрим на поведение системы в действии.

Static VS Dynamic

DataArt

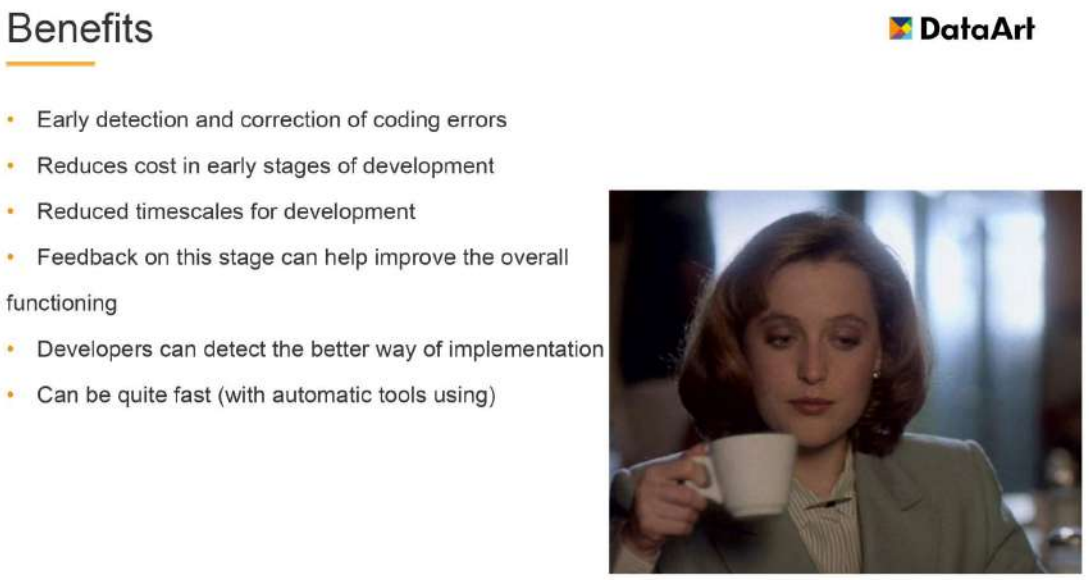
Static Testing	Dynamic Testing
It is performed in the early stage of the software development.	It is performed at the later stage of the software development.
In static testing whole code is not executed.	In dynamic testing whole code is executed.
Static testing prevents the defects.	Dynamic testing finds and fixes the defects.
Static testing is performed before code deployment.	Dynamic testing is performed after code deployment.
Static testing is less costly.	Dynamic testing is highly costly.
It generally takes shorter time.	It usually takes longer time as it involves running several test cases.
It can discover variety of bugs.	It expose the bugs that are explorable through execution hence discover only limited type of bugs.

Чем статического тестирования отличается от динамического?

Из определения следует, что статическое тестирование, не подразумевает выполнение кода. Тогда как при динамическом, мы смотрим на поведение системы в действии. Зачем вообще нужно статическое тестирование? Это один из ранних этапов разработки программного обеспечения, при котором мы оцениваем, что мы будем делать. На данном этапе мы занимаемся анализом технической документации и сопроводительных документов,исходного кода и требований, чтобы предотвратить дефект до того, как он будет реализован. Например, у вас есть документация, в которой

описано что данный модуль выводит определенный текст на экран. В то же время в требованиях сказано, что должен выводиться не текст, а картинка. Мы нашли явное несоответствие до того, как дошли до самой программы.

Статическое тестирования говорит о том, что мы смотрим на код, документацию, проводим сравнительный анализ и делаем выводы. Статическое тестирование позволяет сэкономить нам в будущем как время, так и деньги, потраченные на разработку, потому что при таком подходе есть возможность на раннем этапе предусмотреть лучший путь реализации того или иного модуля.



На данном слайде перечислены плюсы статического тестирования и основным из них является раннее выявления дефектов и ошибок реализации. С помощью статического тестирования мы можем уменьшить затраты на разработку, подобрать наилучший вариант реализации того или иного модуля, выбрать лучший путь реализации.

Здесь есть достаточно спорный момент, связанный с применением автоматизированного тестирования. В большинстве случаев, когда мы говорим об автоматизированном тестировании вкупе со статическим, мы говорим о статическом анализе кода - это когда берется исходный код определенного модуля и построчно анализируется.

Subjects

DataArt

- Requirements specifications
- Design documents
- Use cases
- Source code
- Test plan, test cases and test data
- Technical specifications and traceability matrix
- User documentation

Что может быть объектом статического тестирования?

Объектами статического тестирования являются любого рода требования, исходный код, а также пользовательские сценарии, тестовая документация (тест план, test suite, test case и тд), инструкции (user manual, user guide и тд), traceability matrix/матрица требований. Про требованиям поговорим чуть позже.

Requirements types

DataArt

- Business Requirements – high-level statements that describe the goals and objectives of the business
- Stakeholder/User Requirements – needs of a particular group of stakeholders and what they require of a particular solution
- Solution Requirements – describe what characteristics a solution will have to meet the needs of the stakeholders and business
  - Functional Requirements – describe the behavior of the solution and the information managed
  - Non-Functional Requirements – the qualities of the solution or the environmental conditions under which the solution will remain effective
- Transition Requirements – characteristics that a solution must have in order to transition from the current state to the desired future state

Одним из объектов тестирования могут быть требования/реквайменты. Есть хорошая классификация по международному стандарту BABOK, которая говорит, что есть 4 основных типа требований:

- бизнес требования;
- технические требования (solution requirements);
- требования конечного пользователя (user requirements);

- требование к переходу.

Бизнес требования - это требования направленные на извлечение выгоды для бизнеса. Например, хотим в этом месяце продать 100 машин - это бизнес реквайремент. Пользовательские требования, когда мы хотим, чтобы машина была красного цвета. Требования к реализации, как в случае с основными типами тестирования, бывает двух подвидов: функциональное и нефункциональное. Когда функциональные требования, мы проверяем, что именно мы хотим от системы. Нефункциональные требования о том, как мы хотим, чтобы это было реализовано, при каких условиях и каким образом должно работать в конечной среде. Требования к переходу, например, вы хотите заменить машину и подбираете максимально близко похожую по характеристикам к тому, что у вас есть. Допустим вы не хотите переучиваться с автомата на ручное управление, желаете оставить эту функцию.

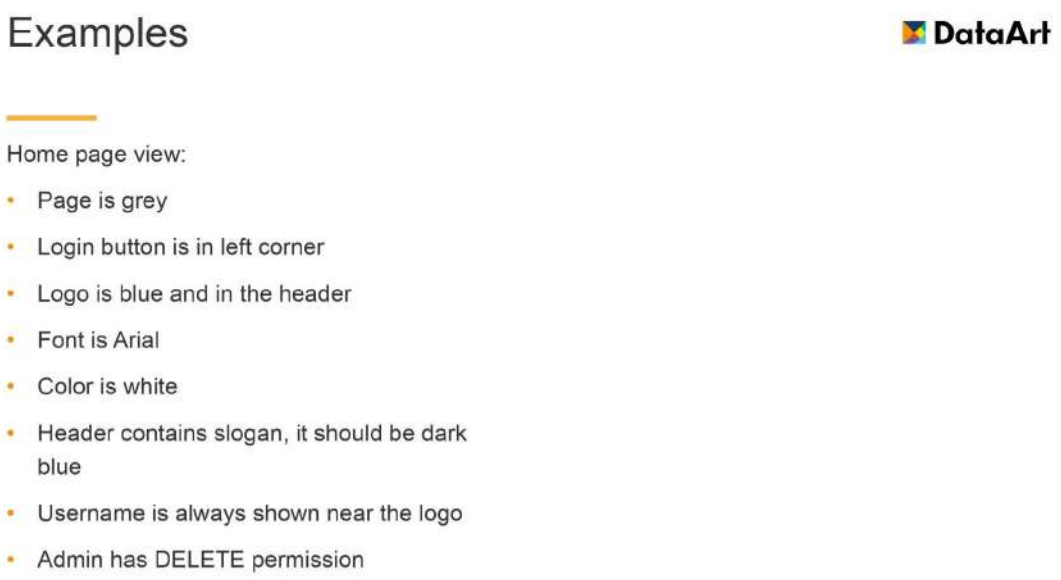
Следующий слайд нам рассказывает **о требованиях к требованиям.** Это те ключевые моменты, которым должно соответствовать хорошее требование. То есть вы можете сказать, хочу, чтобы все было красиво, это требование, но это самый плохой вариант. Потому что требование должно быть однозначным и ясным, если вы находите требование, которое может быть понято двусмысленно, то оно будет понято неправильно. Если вы говорите, что хотите, чтобы кнопка была голубой - она будет голубой, но не того оттенка, размера или располагаться не в том месте. Надо чётче формулировать требования.

Также каждое отдельное требование должно быть независимым или атомарным. Одно требование говорит об одном аспекте. Т.е. нельзя говорить: я хочу красную машину, у которой будут ксеноновые фары, потому что это требование можно разбить на два отдельных. Требования должно быть полным, то есть не должно быть такого, что вы прочли требования и вам нужны по нему уточнения. Вы не должны ходить к бизнес-аналитику, чтобы уточнять, что же имелось в виду. В идеале все требования должны быть тестируемы, не должно быть такого случая, что у вас есть требование, и у вас нет совершенно никакой возможности его проверить, вы не можете сказать выполнено оно или нет. При таком подходе вы не сможете потенциально найти дефект, и

это приведет к падению качества. Каждое требование должно быть необходимым, не должно быть излишним.

Когда мы говорим про требования, обычно мы имеем в виду, что это некоторый набор требований - как правило, все требования системы объединяются в несколько наборов, которые относятся к тому или иному аспекту. Полный набор требований должен быть последовательным, однообразным и не противоречивым, не написанным в разном стиле на разных языках. По возможности стоит этого избегать. Набор требований должен быть не избыточным, завершенным. Пример, у нас есть набор требований для описания внешнего вида автомобиля, т.е. после прочтения набора не должно оставаться сомнений, как в результате должен выглядеть автомобиль.

Каждый набор требований должен быть изменяемым. Каждый отдельный продукт отвечает нуждам бизнеса, и бизнес меняется, желания конечных пользователей тоже, поэтому нужно за всем этим поспевать и привносить изменения в свой продукт.



У нас набор требований домашней странице. Внимательно посмотрите, каждое требования не настолько хороши, насколько могли бы быть.

Вот небольшой пример (слайд ниже) решения того, каким образом можно эти требования поменять. Я не претендую на абсолютную правоту правого списка, но как минимум левый список можно переработать, улучшить в соответствии с правым. Рассмотрим подробнее:

- “Страница серая”, но можно уточнить, что именно серое, вся страница, элементы или фон?
- “Кнопка логина”, здесь можно уточнить позицию.
- “Логотип голубой и должен быть в заголовке” - это можно разделить на два требования.
- “Шрифт Arial” - указать конкретные элементы с этим шрифтом.
- “Белый цвет”, но не указано, чего именно белый цвет.
- “Заголовок содержит слоган, должен быть темно-голубым”. Непонятно, что именно должно быть голубым заголовок или слоган. Нужно расписать, уточнить, что именно какого цвета.

- “Юзер-нейм показывается рядом с логотипом”. Но если мы не залогинены в системы, то какой будет юзер-нейм? никакого, тоже можно уточнить.

Examples

DataArt

Home page view:

- Page is grey
- Login button is in left corner
- Logo is blue and in the header
- Font is Arial
- Color is white
- Header contains slogan, it should be dark blue
- Username is always shown near the logo
- Admin has DELETE permission

- Page background is solid grey
- Login button is in top left corner
- Logo is blue
- Logo is in the header of page in top right corner
- Font is Arial for all texts on the home page
- Color of all texts is white
- Header contains slogan at the left of the logo
- Header color is dark blue
- Username is shown under the logo for all authorized users
- ~~Admin has DELETE permission~~

Review roles

DataArt

Moderator

- the person who leads the review of the document(s) including planning the review, running the meeting, and following-up after the meeting

Author

- the writer or person who is responsible for the document(s) to be reviewed

Reviewers

- people with a specific technical or business background who will check document(s)

Recorder

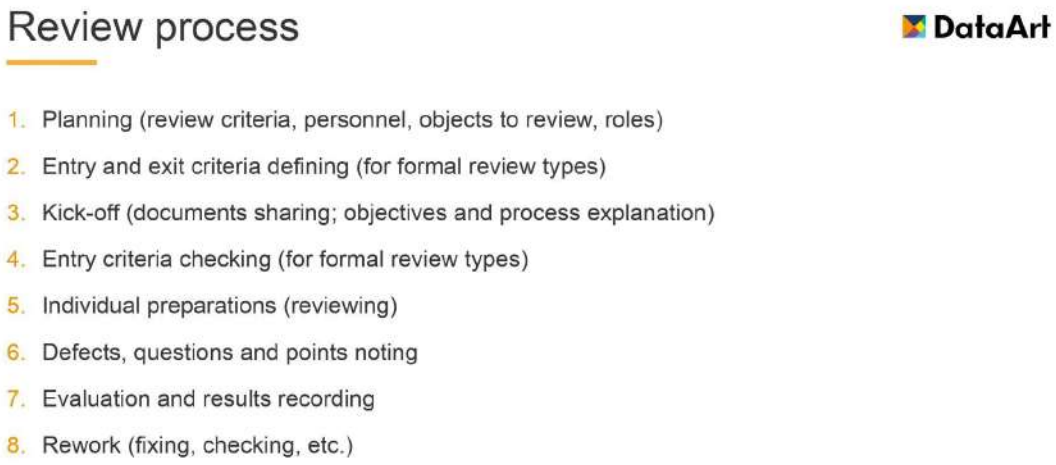
- Makes notes about all found issues, key points, open questions, etc.

Техники статического тестирования.

Для начала посмотрим, кто может участвовать в статическом тестировании. Далее статическое тестирование мы будем называть процессом ревью. Как правило, мы говорим о тестировании требований, а для требований предусматривается такой процесс, как ревью.

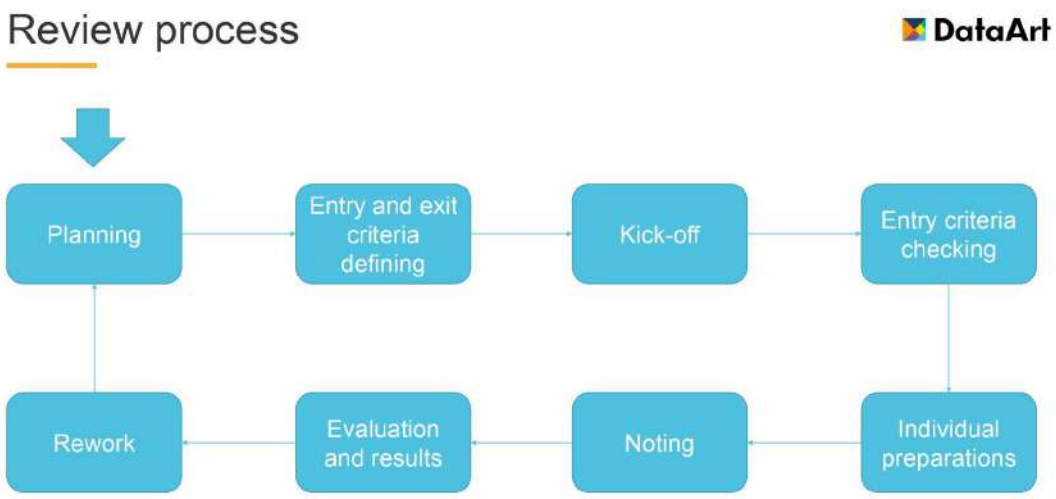
Модератор (администратор) - человек ответственный за всё, организует процесс и звонки, определяет роли, загоняют всех заинтересованных, заставляет прийти на митинге, на встречи, он ответственен за весь процесс. *Автор* - человек, который написал требования или кусок кода, который будет отдуваться в процессе ревью. *Инспекторы* -

люди, которые будут заниматься обзором, проверять требования, кусочек кода, документы, любой объект, который предусмотрен. Человек, который делает записи и ответственен за ведение документации по процессу ревью - это *писарь*.



Процесс ревью состоит из следующих ключевых моментов.

Мы планируем, *что* мы будем рассматривать, *когда*, какие у нас *входные и выходные критерии* процесса. Кто какие *роли* будет выполнять, *зачем* нам это нужно. В большинстве случаев это относится к формальным типам ревью, но может применяться и для неформальных. Когда достигнут входной критерий (в большинстве случаев это готовность документации или требований), начинается индивидуальная подготовка, изучение документов и процесса, что мы будем делать: как, когда, с кем. Проводятся подготовительные работы, составление документов и заметок. Происходит процесс ревью, когда мы все встречаемся и анализируем состояние документа. Составляем список вопросов, документируем дефекты, проводим уточнения, обсуждаем и оцениваем полученный результат. Затем делаем работу над ошибками.



Чтобы не зацикливаться на постоянном анализе требований и улучшении до идеального состояния, мы определяем выходной критерий,

Рассмотрим разные типы ревью или техники ревью: неформальный обзор, беглый просмотр или ознакомления, технический обзор и инспекция.

Informal Review

DataArt

- No formal process
- May take the form of pair programming or code review
- Varies in usefulness depending on the reviewers
- Results may be documented
- Main purpose: inexpensive way to get some benefit

Неформальное обсуждение требований может происходить между двумя людьми в команде, когда есть непонятные моменты или пробелы.

Например, есть нечёткие требования, пойдём сходим к аналитику и уточним у него, что он нам скажет. Вроде бы обычная рабочая ситуация, совершенно неформальная, повседневная, но так или иначе это тоже процесс ревью. Результаты этого ревью очень сильно зависят от того, кто его проводит, если это человек, который только пришел на проект, не знает ничего про продукт, возможно джуниор, которой очень слабо понимает, что происходит - в этом случае ревью будет не таким полезным, как если бы его проводил ваш опытный коллега, который в проекте 10 лет, знает все слабые места, подводные камни и может вам подробно и четко всё рассказать. Можно задокументировать свои работы и результаты по найденной ошибке, написать письмо аналитику, а можете выводы оставить при себе - здесь всё зависит от конкретной ситуации, зачем вам нужно было ревью.

В основном такой вид ревью проводят, когда хотят получить результат здесь и сейчас, то есть встает актуальный рабочий момент, и мы пытаемся его разрешить. Из минусов - плохая документация хода ревью и полученных результатов.

Walkthroughs

DataArt

- The author will explain the document to the team; participants are asking questions and making notes
- Optional pre-meeting preparation of reviewers
- Optional preparation of a review report including list of findings
- Optional participation
- May vary from quite informal to very formal
- Main purposes: learning, gaining understanding, finding defects

Ознакомление.

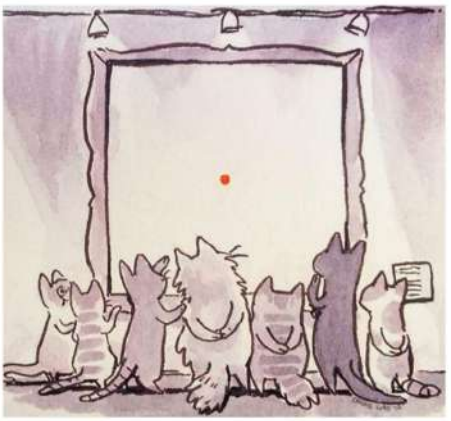
При *ознакомлении* обязательно присутствуют автор документации/реквайментов/кода и члены команды (ревьюеры). Объект ревью предоставляется для ознакомления всей команде, кто хочет высказывает своё мнение, что стоит поправить, какие есть спорные моменты. Т.е. идёт некий процесс, направленный на ознакомление остальных членов команды с объектом, учёт мнений и пожеланий, в споре зачастую рождается истина. При данном типе ревью высока вероятность потенциально возможного нахождения и исправления дефектов. Процесс при этом может быть совершенно неформальным.

Например, у вас был тяжелый спринт, была интересная фича, написанием кода занималась отдельная подкоманда девелоперов, и ребята в конце спринта рассказывают остальной части команды о том, что была сделано, показывают код. Но в то же время, разработчики соседнего модуля говорят, что ребята, конечно, классно, но вы не учли возможность интеграции с нашим модулем. Т.е. при просмотре кода потенциально может быть найден дефект, в новом модуле, который сделали ребята из первой команды, есть возможность оперативно учесть это и исправить, и эти изменения пойдут уже в следующую ревизию, так и не приведя к возникновению бага поведения..

Technical Reviews

DataArt

- Document is reviewed by the technical experts in order to detect any discrepancies or issues
- Pre-meeting preparation by reviewers
- Optional use of checklists
- Preparation of a review report which includes the list of findings, the verdict whether the software product meets requirements and standards
- May vary in practice from quite informal to very formal
- Main purposes: discussing, making decisions, finding defects, solving technical problems and checking conformance to specifications and standards



Техническое ревью - процесс, когда есть сугубо технический документ, написанный сухим языком, достаточно низкоуровневым, то есть это могут быть уже не бизнес или пользовательские требования, а уже формально описанные для технических специалистов, девелоперов, тестировщиков требования или код. Анализом данных документов занимаются технические специалисты, люди, которые каждый день встречаются и сталкиваются с подобным. Процесс может варьироваться от очень неформального до крайне документированного формального, могут использоваться чек-листы.

В техническом ревью мы занимаемся ознакомлением, но не в плане того, что мы хотим понять, что это такое и как это работает, но хотим вынести финальное решение о жизнеспособности. Конечно, в процессе любого ревью вы можете найти какие-то дефекты и решить технические проблемы еще до того, как они наступят. При данном виде ревью мы можем заниматься анализом, например, архитектуры приложения. И если мы сейчас сделаем точно так, как у нас написано в требованиях, то потом любые изменения нам будет стоить очень больших сил в будущем. Например, у нас система будет стабильная, классная, будет отвечать текущим требованиям, но она будет очень неповоротливая, плохо тестируемая, и в конечном итоге, эта система будет разработана на один раз. Чтобы это избежать, мы пытаемся учесть все потенциально слабые места еще на бумаге.

Inspections

DataArt

- Moderator provides a strict review to find defects
- Roles are strongly defined
- Might includes metrics gathering
- Formal process based on rules and checklists
- Specified entry and exit criteria for acceptance
- Pre-meeting preparation
- Main purpose: finding defects

Инспекция - очень формализованный процесс, когда анализом занимается внешний человек, какой-то аудитор, например, визит налоговой инспекции в бухгалтерию. Данный процесс ревью направлен на то, чтобы найти дефекты, здесь не идет никакой речи об исправлении, о предотвращении проблем, главная задача - найти дефект. Такой тип ревью возможен на финальном этапе разработки требований или написания документации, здесь мы все дотошно записываем, составляем отчёт со списком всех проблем, делаем заключение и выдаем это решение как финальное.

Tips

DataArt

- Focus only on things that really count
- Explicitly plan and track review activities
- Train participants with examples
- Keep process formal as the project culture
- Continuous Improvement

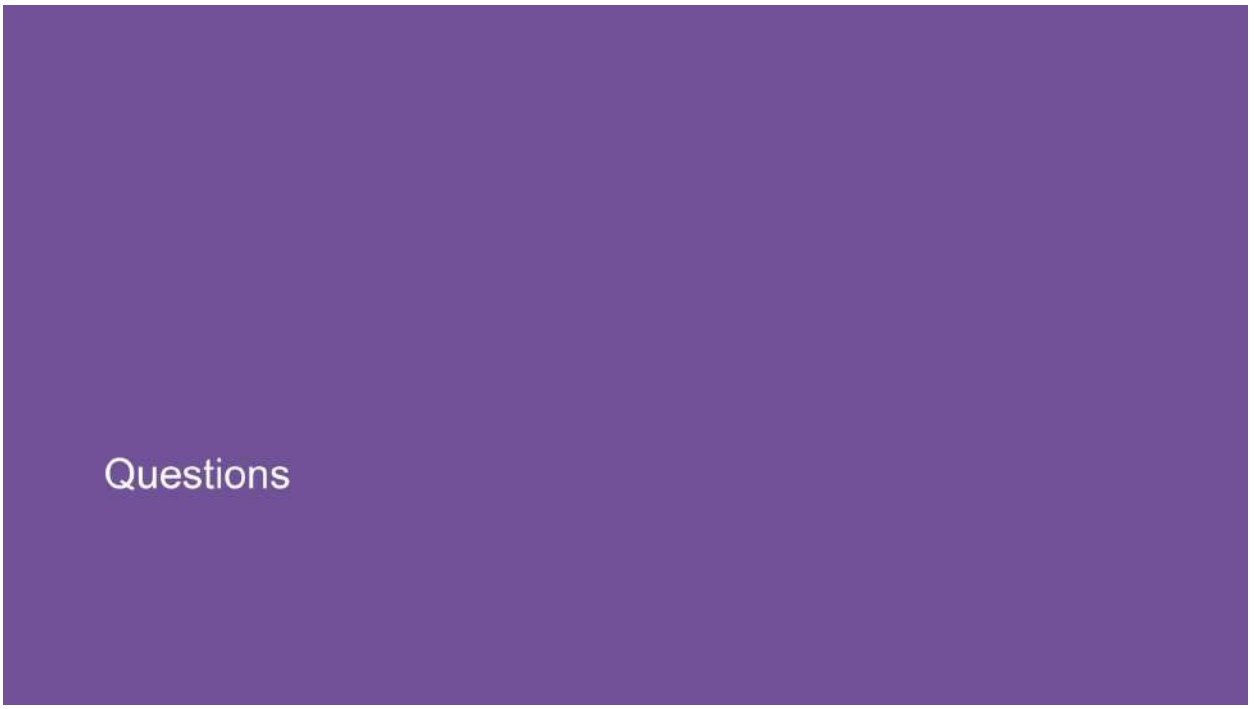
Зачем нужны все эти процессы ревью? Зачем нужно тестировать требования? Когда мы занимаемся статическим тестированием, мы сами лучше знакомимся с тем, с чем нам предстоит работать. Зачастую бывает так, что на первый взгляд система работает достаточно очевидно, просто и понятно, а когда начинаешь ее тестировать,

понимаешь, какое там болото. И чтобы не провалиться сразу в трясину, ты можешь сначала открыть сопроводительную документацию, изучить ее, понять что происходит - если что-то непонятно, то это скорее всего проблемы документации. Практически не бывает такого идеального случая, когда вы открыли любой документ, код и требования, для вас все стопроцентно понятно и четко. Это бывает очень редко, и вообще весь процесс ревью направлен на то, чтобы сделать последующие процессы более прозрачными, четкими, понятными.

Из всех перечисленных типов ревью на контроль качества ориентирована только инспекция. То есть это формальный процесс, когда мы нашли проблемы и сделали предписания, попросили поправить. Здесь не идет речи о том, чтобы предотвратить эту ситуацию будущем. Во всех остальных случаях, мы разговариваем с остальными членами команды, выстраиваем процесс и коммуникации, обращаем внимание на те моменты, которые действительно важны.

Уделяйте внимание тому, что действительно важно. Вы можете закапываться в процесс до бесконечности, если хотим, чтобы все было идеально. Но чисто физически никогда нет столько физических ресурсов, чтобы всё привести в образцовый порядок. Поэтому стоит уделять внимание самому процессу ревью и четко его организовывать. Не будет лишним давать понять остальным членам команды, что вы именно делаете: вы разъясняете команде, какой конечный результат должен быть достигнут.

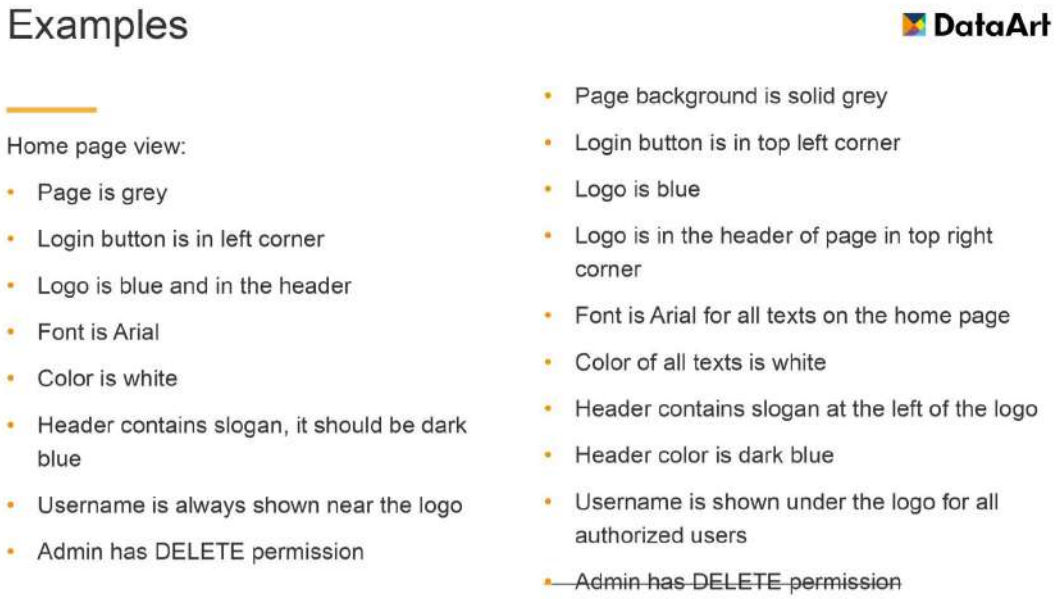
Когда мы проводим статическое тестирование, мы говорим о том, что мы улучшаем. Каждый раз, когда у нас есть дефект в требованиях, мы имеем в виду, что есть дефект в понимании и/или реализации. Человеческий фактор при разработке - это совсем не мелочь, это реальная проблема, когда есть спорный момент и вся команда переругалась, потому что никто так в результате и не решил как правильно. Чтобы этого избежать, нужно с друг другом общаться, дружить, обмениваться информацией и в целом улучшать все процессы и делать их более прозрачными.



1. **Есть много разных требований (бизнес требования, юзер требования) - это один документ или несколько?**

Оба варианта возможны. Можно объединить все требования в один документ, но тогда должны быть соблюдены границы. Например, раздел бизнес-требования, набор требований от пользователей (то каким они хотят видеть продукт), далее технические требования и детали реализации. У вас может быть один документ - некий discovery report, но тогда у вас должно быть разделение на главы и разделы.

2. **Привести пример избыточного требования.**



Есть набор требований к домашней странице, её внешнему виду. На скрине последние требование - у админа должны быть права на удаление, но каким образом это относится к внешнему виду? Чуть более, чем никак. Данное требование здесь избыточно, оно должно быть выделено в отдельный набор. Не нужно всё мешать в кучу,стоит всё описывать отдельно: как должно выглядеть (внешний вид, дизайн), что должно отображаться (набор данных), какой набор контролов (инструменты), что они должны делать (поведение). Если вы касаетесь какого-то аспекта и можете отделить его от остальных, то это подразумевает отдельный набор требований.

3. **Вопрос, как бизнес требования соотносятся с командой QA, девелоперами? Кроме запусков по срокам.**

В приёмочном тестировании проверяется система на соответствие неким формальным договоренностям. Например, у нас клиент - банк, и кроме основных функций, он хочет соответствовать стандартам безопасности. Когда мы говорим про бизнес требования и про пользовательские требования, мы говорим о высокоуровневом описании, что мы хотим получить от системы - человеческим языком. Когда мы говорим о технических требованиях, мы говорим на техническом языке, специфичными терминами, возможно с использованием особенностей языка реализации.

Это работа бизнес аналитика - собрать информацию от клиента и от пользователей, что они хотят, и перевести на технический язык.

4. **Кто принимает итоговое решение о внесение изменений в документацию? И кто может быть модератором в процессе ревью тестинга?**

В идеале все изменения в требованиях должны согласовываться с теми, по чьей инициативе они составлены. Например, вы как тестировщик находите, что одно требование противоречит другому. Вы спрашиваете аналитика, который написал эти требования, почему так случилось. Аналитик поднимает историю, видит, что было несколько встреч с представителем заказчика, и один представитель сказал, что хочет, чтобы данная фича работала таким образом, а другой - другим. И так вышло, что при составлении требований это не учлось. Аналитик не может принять решение об изменениях в требованиях, он идет к заказчику, чтобы уточнить этот момент. В данном случае финальное решение за заказчиком.

Или другая ситуация, мы приходим к аналитику, говорим, что у нас неполные требования, противоречащие или неоднозначные, какой-то принцип нарушен, нам непонятно, что делать. В этом случае аналитик может сам уточнить требования.

В роли модератора может выступать старший коллега в проекте или тимлид, который просит команду взглянуть на код, проект-менеджер, который просит что-то улучшить и пересмотреть подход, либо любой другой человек, ответственный за проект и процессы. В принципе, инициировать процесс ревью может любой участник команды, если у него есть весомый повод. Например, если рядовой тестировщиков нашел ошибку в документации, он идет к лиду, и это уже его задача - организовать процесс так, чтобы в него были вовлечены все заинтересованные и ответственные лица для принятия окончательного решения.

5. **Кто автор? Потому что есть впечатление, что это заказчик.**

И да, и нет. Заказчик может быть автором документации, требований и даже кода. Но как правило это человек, которые перерабатывает то, что сказал заказчик и преобразует в документацию и требования.

Как происходит процесс? К заказчику приходит аналитик, который говорит с ним про бизнес, про продукт, чего они хотят достигнуть, кто их клиенты, т.е. изучается среда, в которой должен работать конечный продукт, кто целевая аудитория. И в конце выкатывается конечный список требований. Да - это сказал заказчик, аналитик это не взял из головы. Но формализовал и описал в конечном итоге бизнес аналитик. В данном случае тот, кто написал, тот и автор.

6. **Откуда берется документация?**

Разные типы документации могут появляться в разные моменты времени и этапы разработки. В идеале, у нас есть список требований ещё до того, как мы начали разработку.

Например, у нас есть бизнес требования, пользовательские требования, мы пытаемся перевести их на технический язык, чтобы получить технические требования - это уже более поздний этап, когда аналитик может консультироваться с теми, кто будет заниматься реализацией, например, о том, какую терминологию использовать, какие будут инструменты и технологии.

Представим, мы уже реализуем код, и даже отправили его в продакшн. Например, мы написали интернет-магазин, пока здесь можно нажать всего две кнопки, но они уже есть. И есть инструкция к тому, как нажимать на эту кнопку, чтобы будет, если на нее нажать, объяснения по поводу того, что мы от программы ожидаем. Т.е. статическое тестирование и создание и сбор документации происходят на всех этапах. И всё зависит

от методологии, о которой подробнее будут рассказывать подробнее, например, Waterfall или Agile.

7. **Может ли одно требования попадаться в разных наборах требований?**

Статическое тестирование можно разбивать на уровни. Мы посмотрели на набор требований и говорим, что это бизнес реквайменты, это пользовательские реквайменты, это солюшен. Мы сделали 3 отдельных набора. Затем на основании этих наборов, мы проверяем, чтобы они не противоречили между собой.

Клиент может запросить в логотипе красный цвет, а пользователь скажет, что нравится синий, а солюшены говорят, что весь дизайн зелёный. И вы пытаетесь утрясти все эти требования, довести весь финальный набор до консистентного состояния, чтобы было некоторое однообразие.

На этапе сбора у вас одно и то же требование может быть описано в разных местах и разным языком. Но бизнес аналитик как раз и занимается тем, чтобы привести всё в порядок, чтобы всех всё устроило. Иногда бывают ситуации, что БА нет, и вы можете обращаться напрямую к заказчику за уточнениями, т.е. вы частично берете на себя роль аналитика.

8. **Когда мы находим дефекты в требованиях, вносим ли мы их в bug report или решаем неформальным образом?**

Необязательно, зависит от того, какой подход мы используем - это может быть и персональное неформальное обращение к автору. Вы можете сходить за уточнениями сразу к аналитику, и он при личном разговоре объясняет, как правильно интерпретировать информацию и обещает через неделю поправить и выложить новую версию документации. Может быть официальное письмо в техподдержку.

Очень часто процесс разработки происходит не только на одной стороне. Например, из компании, в которой вы работаете, наняты только тестировщики, а разработчики из другой компании. И для связи с ними нужно посылать запрос, тикет, письмо - тогда вам приходит формальный ответ.

Хотелось бы сказать, что лучше, конечно, документировать все коммуникации, чтобы это не забывалось со временем и потом не приходилось к этому возвращаться вновь.

9. **Порядок тестирования требований.**

Сначала анализируются высокоуровневые требования, выясняется, чего хочет бизнес, целевая аудитория и пользователи. Потом происходит перевод на технический язык, пишутся, анализируются и тестируются технические требования.

Если у нас есть несколько отдельных областей и есть несколько наборов требований, в этом случае, стоит смотреть в первую очередь на те требования, которые будут реализованы ранее и у которых будет более высокий приоритет. Если разработка идет параллельно, то и требования нужно будет смотреть параллельно. Если линейно, то можно в порядке того, как идет разработка.