**Test-driven development (TDD)** - Разработка через тестирование (англ. test-driven development, TDD) — является передовой техникой разработки программного обеспечения, основывающеюся на повторении кратких циклов разработки: изначально пишется тест, который покрывает желаемое изменение, а далее пишется сам код, позволяющий пройти данный тест.

Иными словами, разработка через тестирование это набор юнит-тестов, которые можно запустить на любом этапе, чтобы убедиться, что программное обеспечение по-прежнему работает.

**Behavior driven development (BDD)** — это методология разработки программного обеспечения, являющаяся ответвлением от методологии разработки через тестирование.

**Фре́ймворк** — программная платформа, определяющая структуру программной системы; программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта.

**Рефакторинг кода** – это процесс улучшения кода программы, не затрагивающий ее внешнего поведения (превращения «грязного» кода в «чистый»)  
  
**Информационная система** – система, предназначенная для хранения, поиска и обработки информации

**Биллинг** — комплекс процессов и решений на предприятиях связи, ответственных за сбор информации об использовании телекоммуникационных услуг, их тарификацию, выставление счетов абонентам, обработку платежей.

**GUI (Graphical User Interface) –** графический пользовательский интерфейс

**API (Application Programming Interface) –** интерфейс программирования приложений

**CLI (Command Line Interface) –** интерфейс командной строки

**B2C (Business-To-Consumer)** – продажи для потребителя (личного пользования)

**B2B (Business-To-Business)** – продажа для бизнеса

**Псевдокод** – это искусственный язык, комбинирующий в себе конструкции реального языка программирования с описанием на русском языке

**LOE (Level of Effort) –** время на прохождение тест-кейса

**Test run –** количество прогонов

**Original Estimate –** сколько времени ушло на прохождение

**Due data –** к какому сроку должен быть исправлен

**Assignee –** на кого назначили исправить (уполномоченный)

**SLA –** время фиксации, время устранения дефекта

**Test Cycle –** цикл исполнения тест-кейсов на билде (однозначно определенном срезе системы)  
  
-------------------------------------------------------------------------------------------------------------------------------  
**1) Netcracker (1993г.-…)**– компания, специализирующаяся на создании , внедрении и сопровождении решений OSS/BSS систем для операторов связи, крупных предприятий и гос.учреждений.

**2) OSS/BSS (operational support system/business support system)** – общее наименование для категории прикладного ПО, которые используют операторы связи, энергетические компании и другие организации, регулярно взаимодействующие с клиентами.  
Решения класса *OSS/BSS* отвечают за:  
1) Управление инфраструктурой и ресурсами (за правильную работу сетевого оборудования: сети, подсети, АТС, коммутаторы)   
2) Взаимодействие с абонентами – биллинговая система (учет и контроль услуг и тарифов, выставление счетов)

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

**3) Waterfall (каскадная)** – модель разработки ПО, в котором процесс разработки выглядит как поток (водопад), последовательно проходящий фазы:

1) Initialization -> 2) Analysis -> 3) Design -> 4) Build -> 5) System Test -> 6) External Test -> 7) Production -> 8) Support

Преимущества:

1) Понятная структура работы (как для специалиста, так и для новичка)

2) Детальная продуманная документация

3) Известны конечные сроки и стоимость

4) Задачи ясны и не меняются

5) Этапы разработки полностью разделены

Недостатки:

1) Отсутствие гибкости (требования не могут меняться в процессе разработки)

2) Тратится большой объем ресурсов и времени

3) Невозможность «перепрыгнуть» через этап разработки

4) Отсутствие фидбека от заказчика

5) Невозможность что-либо изменить, вернуться назад

**4) Agile (гибкая, эволюционная модель)** – семейство процессов разработки, которое основано на Agile Manifesto – ценности и принципы, которыми руководствуется команда.

Преимущества:

1) Быстрый релиз пробной версии

2) Постоянный фидбек от заказчика

3) Возможность вносить изменения на любом этапе

4) Минимизация рисков

5) Быстро решаются появляющиеся проблемы

Недостатки:

1) Проект может сильно растянуться по времени и бюджету

2) Требует большой вовлеченности

3) Постоянные корректировки могут сказаться на качестве ПО

4) Пренебрежение документацией

5) Новые требования могут противоречить старому функционалу

**5) Agile Manifesto** – ценности и принципы, лежащие в основе *Agile*:

*1) Individuals and interactions over processes and tools.*

*2) Working software over comprehensive documentation.*

*3) Customer collaboration over contract negotiation*

*4) Responding to change over following a plan.*

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

**Баг-трекинговые системы** – прикладные программы, созданные с целью помочь учитывать и контролировать баги и неполадки

**Milestone планирование** (диаграммы Ганта) – момент, с которым связано завершение какого-либо ключевого мероприятия, подписание важных документов или любые другие значительные действия, предусмотренные планом проекта. Сдвиг вехи приводит к сдвигу всего проекта.

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

**1) Обеспечение качества (Software Quality):**

**1) Обеспечение качества (Software Quality)** – способность характеристик ПО удовлетворять установленные и предполагаемые потребности

***2) … [ISO]*** *- совокупность характеристик программного обеспечения, относящихся к его способности удовлетворять установленные и предполагаемые потребности.*

***3) … [IEEE]*** *- cтепень, в которой программное обеспечение обладает требуемой комбинацией свойств.*

**2) Обеспечение качества (Quality Assurance)** – совокупность мероприятий, охватывающих все этапы разработки, выпуска и эксплуатаций ПО, предпринимаемых на разных стадиях жизненного цикла ПО для обеспечения качества продукта

**Quality Assurance -> Quality Control -> Software Testing - > Verification -> Validation**

**3) Quality Control** – совокупность действий, проводимых для получения информации об актуальном состоянии объекта тестирования

Портативность

Удобство использования

Удобство сопровождения

Надежность

Эффективность

Функциональность

Удобство установки,

Совместимость

Анализируемость,

Изменяемость

Удобство изучения,

Понятность

Восстанавливаемость,

Устойчивость к отказам

Эффективность использования,

Эффективность по времени

Функциональная исправность,

Соответствие стандартам

**4) Software Testing** – техника контроля качества, включающая в себя

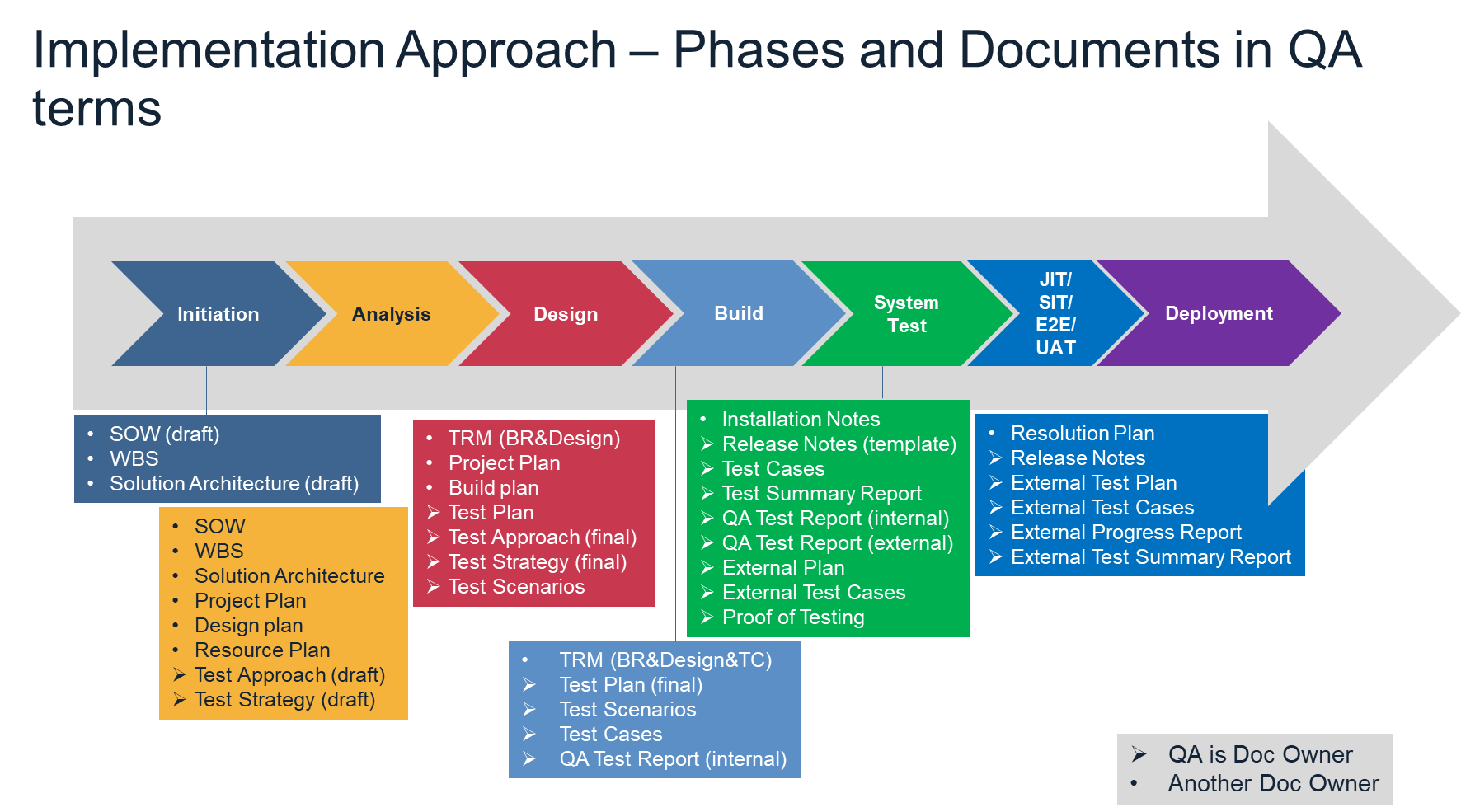
1. **Test Management**
2. **Test Design**
3. **Test Execution**
4. **Test Analysis**

**Вопросы входного контроля**

**1) Цели тестирования**

1) Цели тестирования – это улучшение качества ПО; нахождение дефектов до того, как их найдут пользователи, исправление дефектов и их перепроверка

**2) Стадии разработки ПО. Основные этапы разработки.**

****



**3) Модели разработки ПО**

**Waterfall (каскадная)** – модель разработки ПО, в котором процесс разработки выглядит как поток (водопад), последовательно проходящий фазы:

1) Initialization -> 2) Analysis -> 3) Design -> 4) Build -> 5) System Test -> 6) External Test -> 7) Production -> 8) Support

Преимущества:

1) Понятная структура работы (как для специалиста, так и для новичка)

2) Детальная продуманная документация

3) Известны конечные сроки и стоимость

4) Задачи ясны и не меняются

5) Этапы разработки полностью разделены

Недостатки:

1) Отсутствие гибкости (требования не могут меняться в процессе разработки)

2) Тратится большой объем ресурсов и времени

3) Невозможность «перепрыгнуть» через этап разработки

4) Отсутствие фидбека от заказчика

5) Невозможность что-либо изменить, вернуться назад

**Agile (гибкая, эволюционная модель)** – семейство процессов разработки, которое основано на Agile Manifesto – ценности и принципы, которыми руководствуется команда.

Преимущества:

1) Быстрый релиз пробной версии

2) Постоянный фидбек от заказчика

3) Возможность вносить изменения на любом этапе

4) Минимизация рисков

5) Быстро решаются появляющиеся проблемы

Недостатки:

1) Проект может сильно растянуться по времени и бюджету

2) Требует большой вовлеченности

3) Постоянные корректировки могут сказаться на качестве ПО

4) Пренебрежение документацией

5) Новые требования могут противоречить старому функционалу

**Agile Manifesto** – ценности и принципы, лежащие в основе *Agile*:

*1) Individuals and interactions over processes and tools.*

*2) Working software over comprehensive documentation.*

*3) Customer collaboration over contract negotiation*

*4) Responding to change over following a plan.*

**4) Проектная команда. Роли в команде, для чего нужны?**

**Project manager (PM)** – это специалист, чьей задачей является управление проекта в целом: проектирование, планирование, расстановка приоритетов, контроль (verification, validation), оперативное решение появляющихся проблем

**QA Lead** – контролирует результаты работы своей команды, пишет тест план, тест стратегию, распределяет задачи

* *QA analyst –* проверяет не противоречат ли требования дизайну, совпадают ли они, выполнимы ли они, находит баги в документации
* *QA engineer –* исполнение тест-кейсов, оформление bug reports

**BA Analysis** – выявление, уточнение требований заказчика, составление проектной документации, встреча с бизнес-клиентами и заказчиками

**Technical Manager** – координирует работу с разработчиками ПО, находится в коммуникации с техническими командами

**Sales Manager** – менеджер по продажам, который отвечает за первичное общение с клиентом: от первого контакта и предложения услуг компании до заключения сделки

**5) Стоимость исправления программной ошибки на каждой из стадии разработки?**

**6) В чем состоит тестирование на этапе планирования?**

**Общее планирование и анализ требований** – *Тестирование требований* – объективно необходимо как минимум для того, чтобы иметь ответ на такие вопросы, как: что нам предстоит тестировать; как много будет работы; какие есть сложности; всё ли необходимое у нас есть и т.п. Как правило, получить ответы на эти вопросы невозможно без анализа требований, т.к. именно требования являются первичным источником ответов.

**Уточнение критериев приёмки** позволяет сформулировать или уточнить метрики и признаки возможности или необходимости *начала тестирования (entry criteria)*, *приостановки (suspension criteria)* и *возобновления (resumption criteria) тестирования* и *прекращения тестирования (exit criteria).*

**Уточнение стратегии тестирования** представляет собой ещё одно обращение к планированию, но уже на локальном уровне: *рассматриваются и уточняются те части стратегии тестирования (test strategy)*, которые актуальны для текущей итерации.

**7) В чем состоит тестирование на этапе проектирования?**

То же самое, что в планировании, только гораздо лучше формализованы и подробнее описаны

**8) Что такое тестирование «белого ящика»?**

White box testing – метод тестирования, в рамках которого у тестировщика есть доступ к внутренней структуре и коду приложения, а также есть достаточно знаний для понимания увиденного.

**9) Что такое тестирование «черного ящика»?**

Black box testing – метод тестирования, в рамках которого у тестировщика либо нет доступа к внутренней структуре и коду приложения, либо недостаточно знаний для их понимания, либо он сознательно не обращается к этим данным в процессе тестирования; тестирование основывается на предполагаемых действиях (pattern`ах) пользователей

**10) Интеграционное тестирование**

Integration testing - тестирование, которое направлено на проверку взаимодействия между несколькими частями (модулями, компонентами) приложения (каждая из которых, в свою очередь, проверена отдельно на стадии модульного тестирования).

**11) Восходящее и нисходящее тестирование**

Восходящее (Bottom up) – ПО собирается и тестируется снизу вверх. Сначала тестируются изолированно модули самого нижнего уровня, а затем тестируются модули, непосредственно вызывающие их, которые тестируются неавтономно, а вместе с уже проверенными модулями. И так пока не будет достигнута вершина. В последнюю очередь тестируется ПО в целом.

Нисходящее (Top down) – ПО собирается и тестируется сверху вниз. Изолированно тестируется только головной модуль. Затем с ним соединяются один за другим модули, непосредственно вызываемые им и тестируется полученная комбинация. Так до тех пор, пока не будут проверены все модули.

Если вызываемый модуль не существует, то для имитации функций программируется «заглушки»

**12) Суть статического и динамического тестирования**

*Cтатическое тестирование* – производится без запуска программного кода.

*Динамическое тестирование* – тестирование с запуском кода на исполнение.

**13) Что такое регрессионное тестирование? Для чего оно применятся?**

*Регрессионное тестирование* – это процесс, при котором тестируются сначала новые функциональности, а затем старые тест-кейсы

**Проводится для того, чтобы:**

1) Убедиться, что выявленная ошибка исправлена программистом и больше не проявляется

2) Убедиться, что исправляя одну часть программы, программист не испортил другую

**14) Когда применяется регрессионное тестирование?**

После того, как программист починил дефект.

**15) Тесты для регрессионного тестирования**

Некоторое время после начала тестирования ПО формируется библиотека регрессионных тестов. Это полный набор тестов, охватывающих всю программу и выполняющийся каждый раз, когда программисты сдают ее очередную рабочую версию. Лучше всего, если они автоматизированы.

Периодически необходимо пересматривать эту библиотеку и удалять тесты, которые не снизят качество работы:

* Удалять тесты, которые эквивалентны другим тестам библиотеки
* Комбинировать тесты
* Автоматизировать тесты
* Выделить часть тестов для периодического тестирования

**16) Приемочное тестирование. Суть? Для чего необходимо?**

Acceptance testing – формализованное тестирование, направленное на проверку ПО с точки зрения конечного пользователя/заказчика и вынесения вердикта о том, принимает ли заказчик работу.

**17) Как можно «измерить» эффективность тестов?**

Можно использовать метрику эффективности тестов:

* **Эффективность теста (тестового набора)** = Количество обнаруженных ошибок / Количество тестов в тестовом наборе

**18) Характеристики хорошего теста**

1) Отсутствие ссылок на другие тесты

2) Существует обоснованная вероятность выявления тестом ошибки

3) Не должен быть слишком простым или слишком сложным

4) Четкие формулировки

**19) Стадии тестирования**

**System Test:**

**Installation Notes || Smoke testing || Component test case execution || E2E Testing|| Final Regression test**

**External Test:**

**SIT (System Integration Test) || UAT (User Acceptance Test) || E2E testing**

**20) Виды тестирования**

|  |  |  |
| --- | --- | --- |
| **Функциональное** | **Нефункциональное** | **Связанное с изменениями** |
| Тестирование безопасности *(Security and Access Control testing)* | Тестирование производительности *(Performance testing)* | Дымовое тестирование *(Smoke testing)* |
| Тестирование взаимодействия *(Interoperability testing)* | Стрессовое тестирование *(Stress testing)* | Тестирование сборки *(Build Verification test)* |
| *Component, Integration, E2E testing* | Объемное тестирование *(Volume testing)* | Санитарное тестирование *(Sanity test)* |
|  | Тестирование надежности *(Stability/Reliability testing)* | Регрессионное тестирование *(Regression testing)* |
|  | Тестирование установки *(Installation testing)* |  |
|  | Тестирование удобства пользования *(Usability testing)* |  |

**21) Понятие классов эквивалентности. Привести пример**

**Класс эквивалентности –** это набор данных, обрабатываемых одинаковым образом и приводящих к одинаковому результату. Протестировав один тест-кейс классов эквивалентности, «как будто» протестировали все – *эквивалентные тесты*

**Пример классов эквивалентности:**

Например, если программа должна принимать числа от 1 до 99, существует, как минимум, 4 класса эквивалентности

1) Ввод [1; 99]

2) Любое число {- ∞; 0]

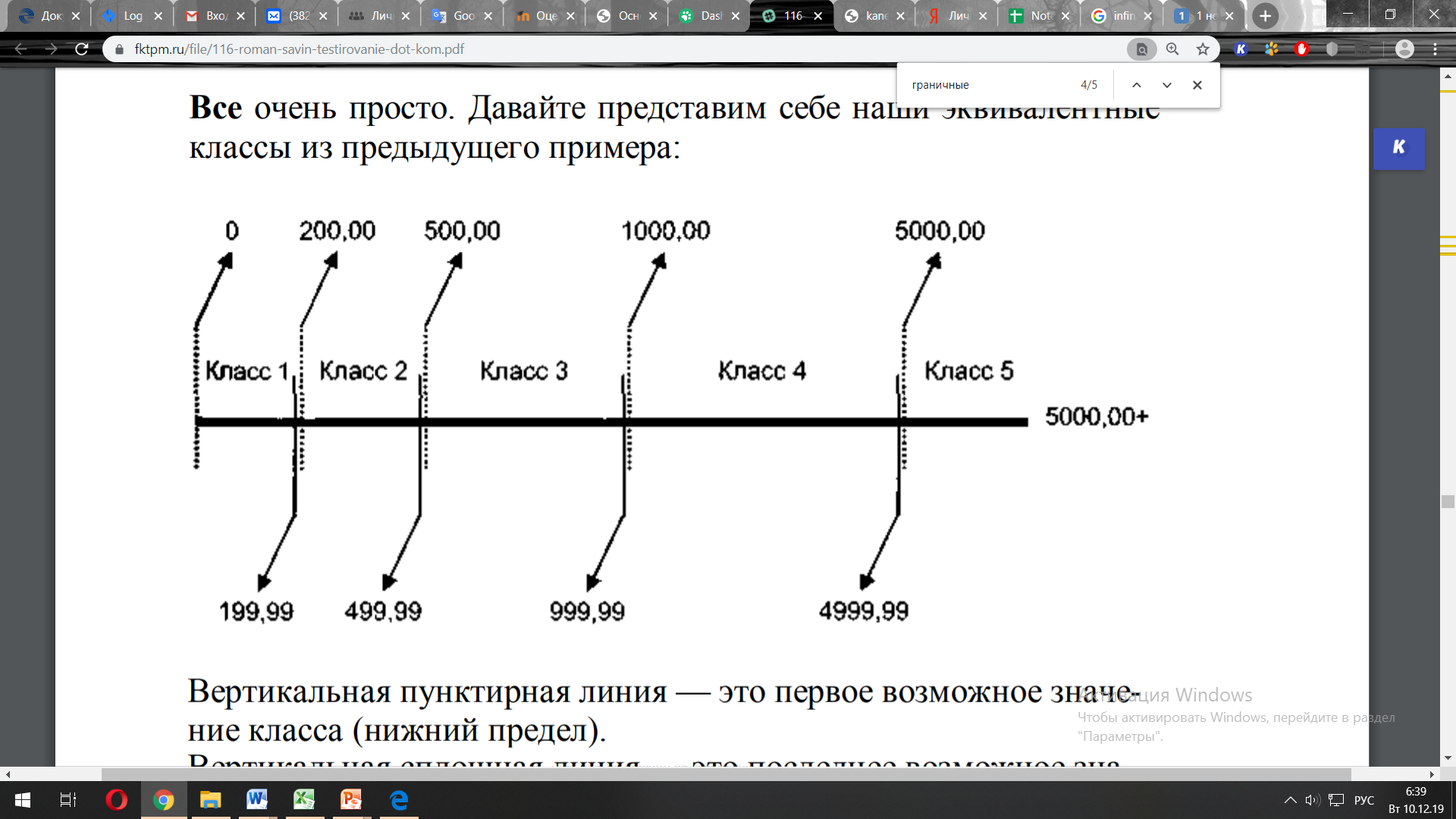
3) Любое число [100; +∞}

4) Ввод нечислового значения

**22) Граничные условия. Привести пример**

**Граничное условие** – значение, находящееся на границе классов эквивалентности

Предположим, что наш книготорговый веб-сайт запускает новую кампанию "Больше тратишь — больше скидка".



**Как тестировать (Например, класс 2):**

1) Нижний предел данного класса 2 (200)

2) Верхний предел данного класса 2 (499,99)

3) Любое значение внутри данного класса 2 (315)

4) Верхний предел класса 1, предшествующего данному классу 2 (199,99)

5) Нижний предел класса 3, следующего за данным классом 2 (500)

**23) Понятие качества и надежности ПО**

**Качество ПО –** степень, в которой ПО обладает требуемой комбинацией свойств (соответствие фактического результата ожидаемому)

**Надежность ПО –** способность ПО безотказно выполнять определенные функции при заданных условиях в течение определенного времени.   
То есть чем меньше сбоев, тем надежнее ПО и чем быстрее ПО возвращается в нормальное состояние, тем оно надежнее.

**24) Что такое программная ошибка?**

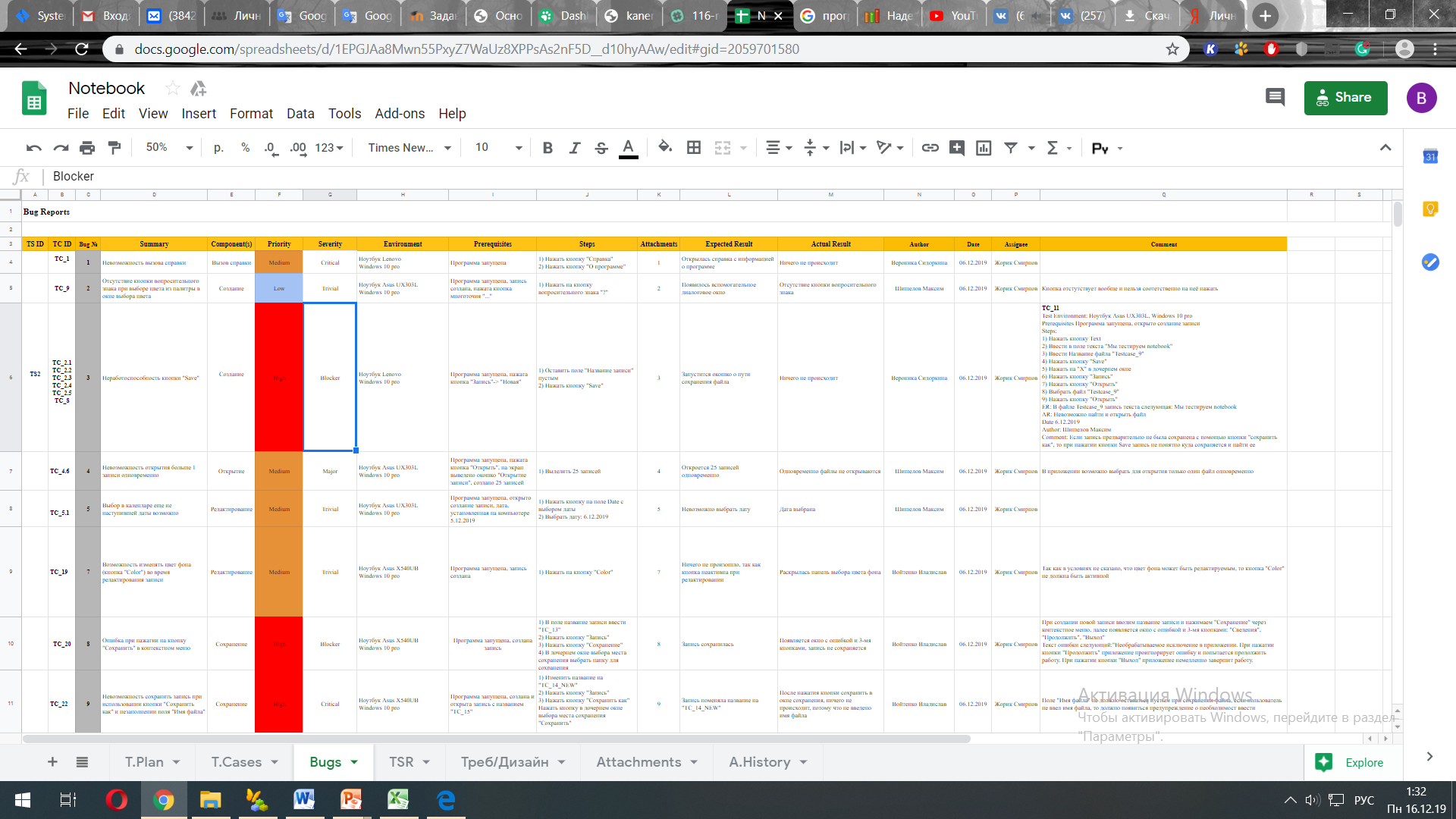
**Программная ошибка (Баг) –** если ПО не делает то, чего от него ожидает юзер

**25) Что такое баг – репорт (отчет об ошибке)? Для чего он нужен?**

**Bug report –** документ, описывающий найденный дефект, приоритизирующий его и содействующий его устранению.

**26) Из каких разделов состоит отчет об ошибке?**

Bug report состоит из TS ID, TC ID, Summary, Components, Priority, Severity, Environment, Prerequisites, Steps, Attachments, Expected Result, Actual Result, Author, Date, Assignee, Comment



**27) Что такое приоритет ошибки? Какие могут быть приоритеты?**

**Priority –   
1)** это показатель важности дефекта для бизнес-компании (Коммерческий аспект)  
**2)** это показатель того, как быстро должен быть исправлен дефект

**Blocker, Critical, Major, Medium, Low**

**28) Что такое степень важности?**

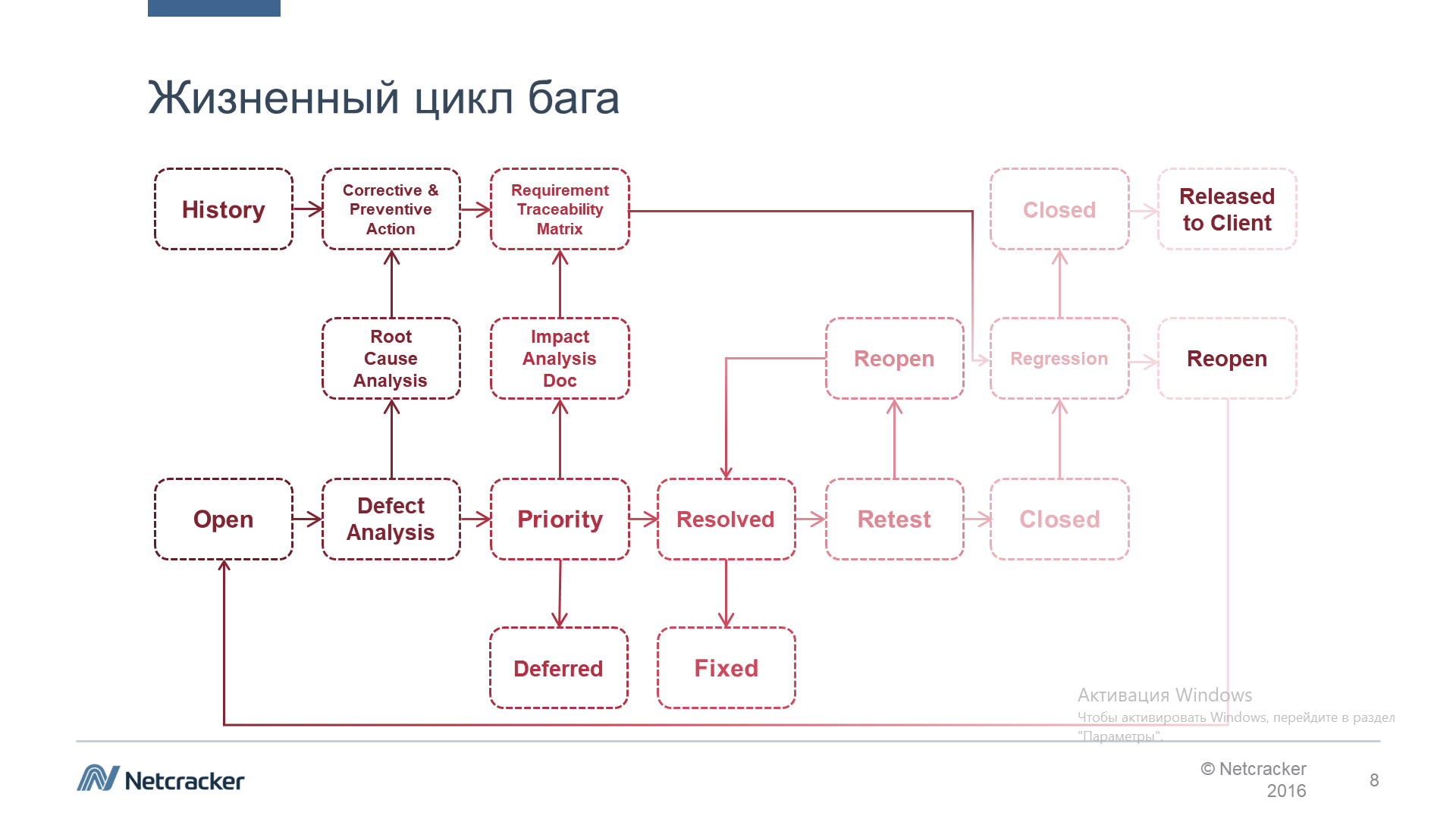
**Severity –   
1)** степень воздействия дефекта на ПО, исходя из принадлежности к определенной технической категории (Технический аспект)  
2) как сильно влияет дефект на бизнес-процесс, степень ущерба проекту  
3) приоритет на внешнем тестировании

**29) Приоритет vs Степень важности?**

Priority коммерческий аспект vs. Severity технический аспект

|  |  |  |  |
| --- | --- | --- | --- |
|  | | **Severity** | |
| **Critical/Major** | **Normal/Low** |
| **Priority** | **Blocker/Critical**  **/Major** | Проблемы с безопасностью во введённом в эксплуатацию банковском ПО. | На корпоративном сайте повредилась картинка с фирменным логотипом. |
| **Normal/Low** | В самом начале разработки проекта обнаружена ситуация, при которой могут быть повреждены или вовсе утеряны пользовательские данные. | В руководстве пользователя обнаружено несколько опечаток, не влияющих на смысл текста. |

**30) Жизненный цикл ошибки?**



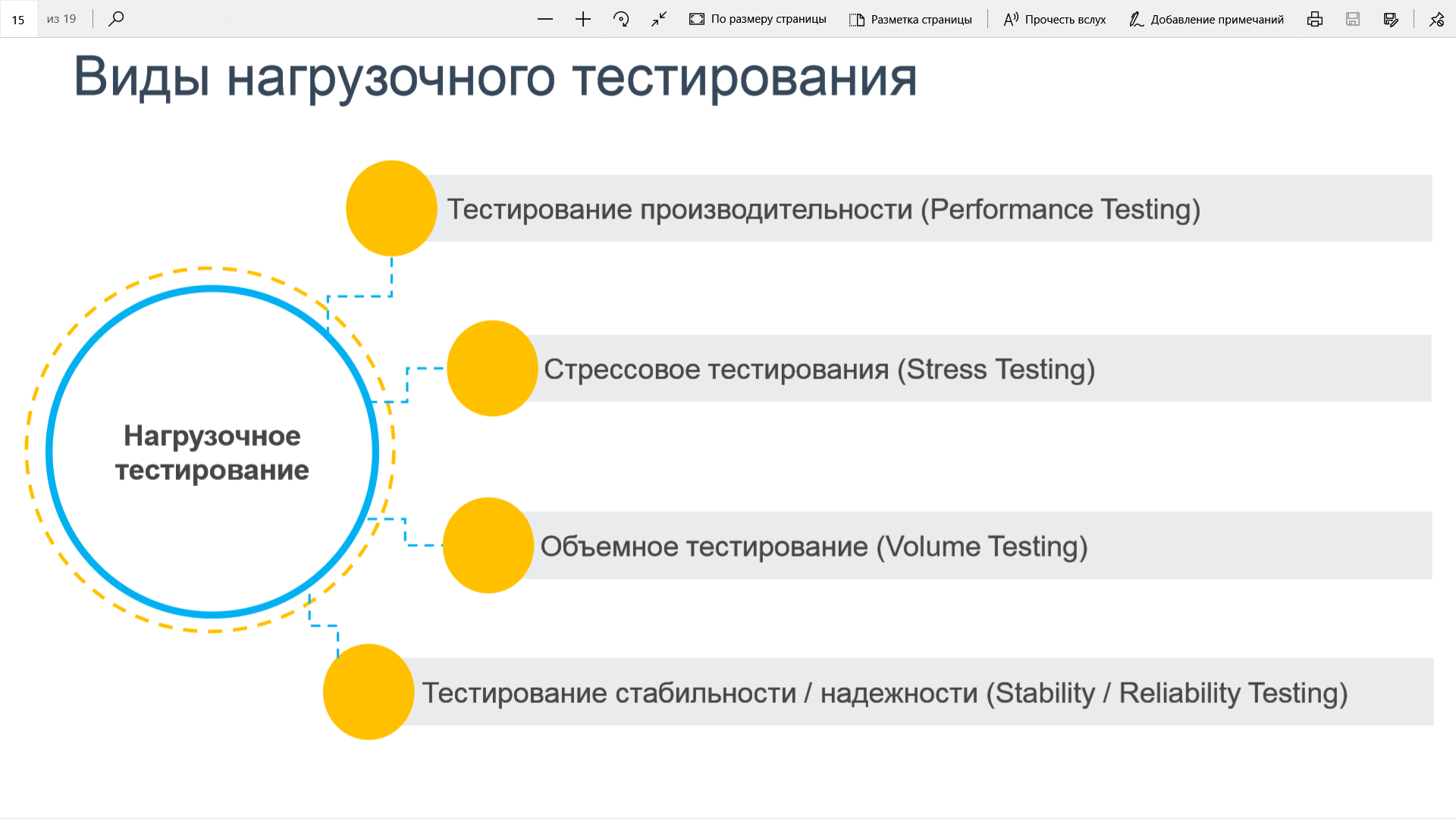
**31) Назначение баг-трекинговой системы (системы отслеживания ошибок)**

**Bug Tracking system –** прикладное ПО, созданное с целью помочь разработчикам ПО учитывать и контролировать ошибки и дефекты (Bug reports), а также следить за их устранением

Bug Tracking system – инфраструктура, позволяющая:

* Создавать
* Хранить
* Модифицировать
* Просматривать

**32) Нагрузочное тестирование**



**Нагрузочное тестирование (Load Testing)** или **Тестирование производительности (Performance Testing) –** это автоматизированное тестирование, имитирующее работу определенного количества бизнес-пользователей на каком-либо общем ресурсе

***Цели нагрузочного тестирования:***

* Оценка производительности и работоспособности приложения на этапе разработки и передачи в эксплуатацию
* Оценка производительности и работоспособности приложения на этапе выпуска новых релизов, патч-сетов
* Оптимизация производительности приложения, включая настройки серверов и оптимизацию кода
* Подбор соответствующей для данного приложения аппаратной (программной) платформы и конфигурации сервера

**33) Стресс тестирование**

**Стрессовое тестирование** (**Stress Testing**) – один из видов тестирования ПО, которое позволяет проверить работоспособность системы в условиях, превышающих пределов нормального функционирования. Стрессом в данном контексте может быть повышение интенсивности выполнения операций до очень высоких значений или аварийное изменение конфигурации сервера.

***Цели стрессового тестирования:***

* Проверка работоспособности приложения в условиях запредельных нагрузок
* Проверка регенерации системы, то есть как быстро вернется в нормальное состояние
* Оценка деградации производительности

**Объемное тестирование (Volume testing) –** вид тестирования, исследование производительности приложения при обработке больших объемов данных

**Тестирование стабильности или надежности** (**Stability / Reliability Testing**) – вид тестирования, при котором проверяется отсутствие утечек памяти, перезапусков серверов под нагрузкой и другие аспекты, влияющие на стабильность/надежность работы

**34) Тестирование документации. Цель?**

**Тестирование документации –** начальная стадия процесса тестирования, которая заключается в выявлении ошибок в требованиях и дизайне

***Цели:***

1. Проверка корректности требований на однозначность, полноту, осуществимость, непротиворечимость
2. Уменьшение рисков несоответствия реализованного функционала, передачи некачественной документации

**35) Для чего нужна автоматизация тестирования?**

Автоматизация тестирования применяется для функционального и нагрузочного тестирования; человеческий труд заменяется работой машины с целью повысить производительность; создание набора автотестов, которые можно запустить

**36) Выбор тестов для автоматизации?**

* Полнофункциональное тестирование
* Регрессионное тестирование
* Smoke test
* System Volume Testing (SVT)
* Создание prerequisites
* Воспроизведение и ретест дефектов
* Помощь в ручном тестировании
* Приемочное тестирование, установочное

**37) Планирование тестирования. Задачи?**

Необходимо подготовить test environment, сервера для тестирования, компьютеры, рабочие места, тестовые данные, инструменты, автотесты, в каких циклах будет проходиться

Software testing это одна из техник контроля качества, включающая в себя следующие активности:



**38) Виды тестовой документации. Для чего необходима тестовая документация?**

1. **Test Strategy –** документ (word), в котором полностью описан весь объем работ по тестированию, включающий в себя:
2. **Test Plan –** документ (excel), в котором описаны объем, ресурсы и календарный график по всем QA activity
3. **Test Cases –** документ, состоящий из совокупности шагов, условий, данных, проверяющих работоспособность какой-либо функциональности
4. **Test Report –** сводный отчет о проведенном тестировании, который включает информацию о том, какие тесты были проведены, сколько Pass/Fail, обнаруженные дефекты, а также статистические данные (метрики)

**39) Тестовый план. Для чего нужен документ? Кем составляется? Из чего состоит?**

**Test Plan –** документ (excel), в котором описаны объем, ресурсы и календарный график по всем QA activity

***Включает в себя:***

*Test Phases, Test Stages, Start date, Finish date, Duration, Responsible*

*ID TC, Test Cases, Priority, LOE Preparation, LOE Execution, LOE Test data preparation, Test environment preparation, TOTAL LOE, All Cycles*

**40) Основные распространённые программные ошибки**

