

## Основы программной инженерии (ПИ)

### Тестирование ПО

#### План лекции:

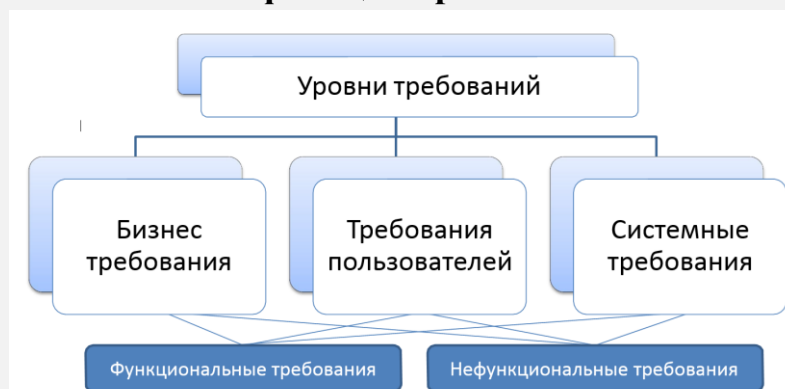
- тестирование ПО: основные понятия и определения;
- классификация тестирования;
- виды тестирования;
- цели, задачи и принципы тестирования;
- ручное тестирование;
- примеры.

#### На прошлых лекциях:

#### 1. Управление требованиями

**Требование** – это утверждение, которое идентифицирует эксплуатационные, функциональные параметры, характеристики или ограничения проектирования продукта или процесса, которое *однозначно, проверяемо и измеримо*.

#### Классификация требований



#### Требование –

- ✓ *условие или возможность, необходимые пользователю для решения его задач или достижения цели (1)*
- ✓ *условие или возможность, которым должна отвечать или которыми должна обладать система или ее компонента, чтобы удовлетворить контракт, стандарт, спецификацию или иной формальный документ (2)*

#### Цели разработки требований

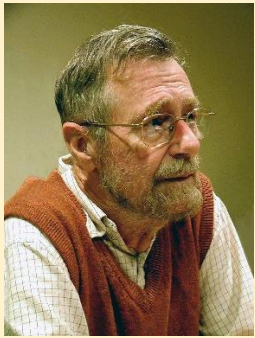
- ✓ обеспечение наиболее полного и точного отражения условий или возможностей, необходимых заказчику для решения его проблем и достижения бизнес-целей;
- ✓ снижение затрат на разработку, обслуживание и поддержку сложного программного обеспечения.

✓ <i>документированное представление условия или возможности, указанное в (1) или (2)</i>	
<p><b>Требования делятся на:</b></p> <ul style="list-style-type: none"> <li>• <b>функциональные</b> (то, что система позволяет сделать, желаемая функциональность);</li> <li>• <b>нефункциональные</b> (требования к оборудованию, операционной системе и т.п.).</li> </ul>	

## Формализация функциональных требований

<p><b>Диаграмма вариантов использования (англ. use-case diagram) –</b></p> <p>диаграмма, описывающая, какой функционал разрабатываемой программной системы доступен каждой группе пользователей.</p>
<p><b>Диаграмма вариантов использования = Диаграмма прецедентов</b></p>
 <p>- А ты строишь диаграммы при проектировании?</p> <p>- Да, как видишь</p>
<p><b>Диаграммы вариантов использования</b></p> <ul style="list-style-type: none"> <li>✓ показывают взаимодействия между <b>вариантами использования</b> и <b>действующими лицами</b>, отражая функциональные требования к системе с точки зрения <b>пользователя</b>.</li> <li>✓ являются исходной концептуальной моделью системы в процессе ее проектирования и разработки.</li> </ul>

## 2. Тестирование программного обеспечения



**Эдсгер Вибе Дейкстра** (11 мая 1930, — 6 августа 2002, Нидерланды) — нидерландский ученый.

Один из тех людей, с именем которых связано превращение программирования из шаманства в науку; один из разработчиков концепции структурного программирования, исследователь формальной верификации и распределенных вычислений. Тьюринговский лауреат (1972).

<https://habr.com/ru/post/303712/>

**«Тестирование программ можно использовать для того, чтобы показать наличие ошибок, и никогда — для того чтобы показать их отсутствие!»**

## 3. Тестирование программного обеспечения – основные понятия и определения

**Тестирование программного обеспечения (Software Testing)** –

проверка соответствия между реальным и ожидаемым поведением программы, осуществляемая на конечном наборе тестов, выбранном определенным образом.

**[IEEE Guide to Software Engineering Body of Knowledge, SWEBOK, 2004]**

**Верификация (Verification)** – это процесс оценки системы или её компонентов с целью определения удовлетворяют ли результаты текущего этапа разработки условиям, сформированным в начале этого этапа [IEEE]. Т.е. выполняются ли наши цели, сроки, задачи по разработке проекта, определенные в начале текущей фазы.

**Валидация (Validation)** – это определение соответствия разрабатываемого ПО ожиданиям и потребностям пользователя, требованиям к системе [BS7925-1].

**План Тестирования (Test Plan)** – это документ, описывающий весь объем работ по тестированию, начиная с описания объекта, стратегии, расписания, критериев начала и окончания тестирования, до необходимого в процессе работы оборудования, специальных знаний, а также оценки рисков с вариантами их разрешения.

**Тест дизайн (Test Design)** – это этап процесса тестирования ПО, на котором проектируются и создаются тестовые случаи (тест кейсы), в соответствии с определёнными ранее критериями качества и целями тестирования.

**Тестовый случай (Test Case)** – это артефакт (побочный продукт, созданный в процессе тестирования ПО), описывающий совокупность шагов, конкретных условий и параметров, необходимых для проверки реализации тестируемой функции или её части.

***Баг/Дефект Репорт (Bug Report)*** – это документ, описывающий ситуацию или последовательность действий, приведшую к некорректной работе объекта тестирования, с указанием причин и ожидаемого результата.

***Тестовое Покрытие (Test Coverage)*** – это одна из метрик оценки качества тестирования, представляющая из себя плотность покрытия тестами требований либо исполняемого кода.


***Детализация Тест Кейсов (Test Case Specification)*** – это уровень детализации описания тестовых шагов и требуемого результата, при котором обеспечивается разумное соотношение времени прохождения к тестовому покрытию

***Время Прохождения Тест Кейса (Test Case Pass Time)*** – это время от начала прохождения шагов тест кейса до получения результата теста.

#### 4. Понятие дефекта

Error, defect, failures, bug	
1843 г. – первое упоминание <i>ошибки</i> в аналитическом движении Чарльза Бэббиджа.	1878 г. – Том Эдисон, первое слово « <i>bug</i> » в письме.
	
<p><b>Error</b> (ошибка) – логическая или другая ошибка, которая может привести к возникновению дефекта.</p> <p><b>Defect</b> (дефект) – различие между ожидаемым и фактическим результатом (программа или система не делает то, что должна).</p> <p><b>Failure</b> (отказ) – сбой, к которому может привести дефект.</p> <p><b>Bug</b> (баг) – ошибка в программе или системе, которая выдает неожиданный или неправильный результат.</p>	

**Пример:**

	<b>Основные функции:</b> <ul style="list-style-type: none"><li>– вскипятить воду;</li><li>– налить воду в чашку;</li><li>– вместимость воды;</li><li>– перемещение воды в пространстве.</li></ul>			
<b>Тестирование свойства:</b> <i>вскипятить воду</i>				
<b>Шаги проверки:</b> <ul style="list-style-type: none"><li>– налить воду в чайник;</li><li>– поставить на подставку;</li><li>– включить;</li><li>– подождать;</li><li>– визуально оценить целостность чайника.</li></ul>		<b>Ожидаемый результат:</b> <ul style="list-style-type: none"><li>– вода вскипела;</li><li>– чайник целый.</li></ul>		 <b>OK</b>
<b>Тестирование свойства:</b> <i>налить воду в чашку</i>				
<b>Шаги проверки:</b> <ul style="list-style-type: none"><li>– налить воду в чайник;</li><li>– налить воду в чашку.</li></ul>		<b>Ожидаемый результат:</b> <ul style="list-style-type: none"><li>– чайник целый;</li><li>– количество воды в чайнике уменьшилось.</li></ul>		 <b>OK</b>
<b>Тестирование свойства:</b> <i>вместимость воды</i>				
<b>Шаги проверки:</b> <ul style="list-style-type: none"><li>– налить воду в чайник;</li><li>– визуально оценить целостность чайника.</li></ul>		<b>Ожидаемый результат:</b> <ul style="list-style-type: none"><li>– чайник целый;</li><li>– количество воды в чайнике не изменилось.</li></ul>		 <b>OK</b>
<b>Тестирование свойства:</b> <i>перемещение воды в пространстве</i>				
<b>Шаги проверки:</b> <ul style="list-style-type: none"><li>– налить воду в чайник;</li><li>– поднять чайник;</li><li>– перенести чайник;</li><li>– визуально оценить целостность чайника.</li></ul>		<b>Ожидаемый результат:</b> <ul style="list-style-type: none"><li>– чайник целый;</li><li>– количество воды в чайнике не изменилось.</li></ul>		 <b>OK</b>

## 5. Разработка требований и тестовые артефакты

Артефакты — это некоторые побочные продукты, возникающие в процессе проектирования, разработки, тестирования программного обеспечения.

<i>Разработка требований</i>	<i>Требования</i> – совокупность утверждений относительно атрибутов, свойств или качеств разрабатываемого программного обеспечения.
	<i>Спецификация</i> – законченное описание поведение программы, которую требуется разработать.
	<i>Функциональные требования</i> – требуемые характеристики системы (функциональность).
	<i>Нефункциональные требования</i> – требования, которые не влияют на основную функциональность системы.

<i>Тестовые артефакты</i>	<i>Тестовый случай (тест, test case)</i> – совокупность шагов, конкретных условий и параметров, необходимых для проверки реализации тестируемой программы, функции.
	<i>Ошибка (дефект, bug)</i> – отклонение фактического результата от ожидаемого.
	<i>Отчет об ошибке (bug report)</i> – это документ, описывающий ситуацию, которая привела к обнаружению ошибки, фактический и ожидаемый результат.

### Что такое тестирование? Что такое ПО?

<i>Тестирование ПО</i>	<i>ПО</i> – совокупность программ системы обработки информации, соответствующая документация и данные, относящиеся к функционированию системы.
	<i>Тестирование ПО</i> – проверка и оценка соответствия между реальным и ожидаемым поведением программы.

## 6. Цели и задачи тестирования

<i>Цели тестирования</i>	• убедиться, что <b>ПО</b> отвечает заявленным требованиям.
	• выявить ситуации, в которых поведение программы является неправильным, нежелательным или не соответствующим требованиям.

<i>Задачи тестирования</i>	• убедиться, что <b>ПО</b> отвечает заявленным требованиям.
	• выявить ситуации, в которых поведение программы является неправильным, нежелательным или не соответствующим требованиям.
	• предотвратить как можно больше дефектов
	• проверить, что известные дефекты устранены
	• проверить, что при устранении известных дефектов, не было внесены новые дефекты
	• информировать всех заинтересованных лиц о качестве системы.

## 7. Цикл тестирования:

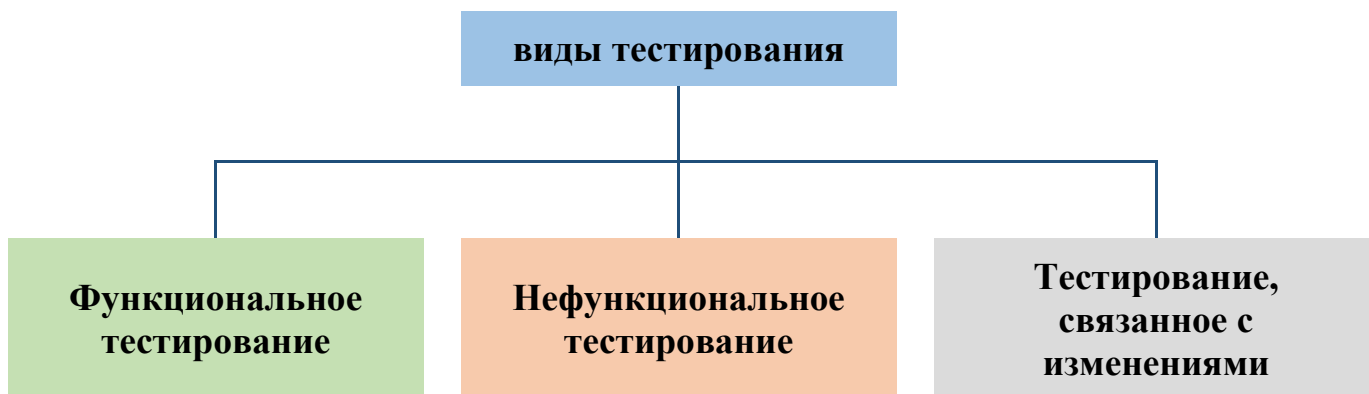


<b>Принципы тестирования</b>	1. Тестирование демонстрирует наличие дефектов.
	2. Исчерпывающее тестирование недостижимо.
	3. Раннее тестирование.
	4. Скопление дефектов.
	5. Парадокс пестицида.
	6. Тестирование зависит от контекста.
	7. Заблуждение, что ошибки отсутствуют.

## 8. Классификация видов тестирования

Все виды тестирования программного обеспечения, в зависимости от преследуемых целей, можно условно разделить на следующие группы:

### Классификация видов тестирования



#### Функциональные виды тестирования:

**Функциональное тестирование** рассматривает заранее указанное поведение и основывается на анализе функциональных требований.

**Тестирование безопасности** проверяет безопасность системы, а также выполняется для анализа рисков, связанных с защитой приложения, атак хакеров, вирусов, несанкционированного доступа к конфиденциальным данным.

**Тестирование взаимодействия** оценивает способность приложения взаимодействовать с одним и более компонентами или системами.

### *Нефункциональные виды тестирования:*

**Тестирование производительности** – определение масштабируемости приложения под нагрузкой. Виды тестирования производительности:

- нагрузочное тестирование (Performance and Load Testing);
- стрессовое тестирование (Stress Testing);
- тестирование стабильности или надежности (Stability / Reliability Testing);
- объемное тестирование (Volume Testing).

**Тестирование установки** направленно на проверку успешной инсталляции и настройки, а также обновления или удаления программного обеспечения.

**Тестирование удобства пользования** – это метод тестирования, направленный на установление степени удобства использования, обучаемости, понятности и привлекательности для пользователей разрабатываемого продукта в контексте заданных условий. *[ISO 9126]*

**Тестирование на отказ и восстановление (Failover and Recovery Testing)** проверяет тестируемое ПО на способность противостоять и успешно восстанавливаться после возможных сбоев (ошибки программного обеспечения, отказы оборудования или проблемы связи).

**Конфигурационное тестирование (Configuration Testing)** направлено на проверку работы ПО при различных конфигурациях системы (заявленных платформах, поддерживаемых драйверах, при различных конфигурациях компьютеров и т.д.).

### *Тестирование, связанное с изменениями:*

**Дымовое тестирование** – минимальный набор тестов, проверяющих базовую функциональность (нового или исправленного ПО).

**Регрессионное тестирование** – это вид тестирования направленный на проверку изменений, сделанных в приложении или окружающей среде (починка дефекта, слияние кода, миграция на другую операционную систему, базу данных, веб сервер или сервер приложения), для подтверждения того факта, что существующая ранее функциональность работает как и прежде (используются тест кейсы, написанные на ранних стадиях разработки и тестирования).

**Тестирование сборки (Build Verification Test)** – это тестирование, направленное на определение соответствия, выпущенной версии, критериям качества для начала тестирования.

**Санитарное тестирование или проверка согласованности/исправности (Sanity Testing)** – это узконаправленное тестирование для доказательства того, что **конкретная функция** работает согласно заявленным в спецификации требованиям (подмножество регрессионного тестирования). Используется для определения работоспособности определенной части приложения после изменений произведенных в ней или окружающей среде. **Обычно выполняется вручную.**

## 9. Уровни тестирования программного обеспечения

### *Уровни тестирования программного обеспечения*

тестирование на разных уровнях производится на протяжении всего жизненного цикла разработки и сопровождения программного обеспечения. Уровень тестирования определяет то, над чем производятся тесты: над отдельным модулем, группой модулей или системой, в целом.

### *Компонентное или Модульное тестирование (Component Testing or Unit Testing)*

проверяет функциональность и ищет дефекты в отдельных частях приложения (модули программ, объекты, классы, функции и т.д.).

*Интеграционное тестирование (Integration Testing)* предназначено для проверки связи между компонентами, а также взаимодействия с различными частями системы (операционной системой, оборудованием либо связи между различными системами). *Подходы к интеграционному тестированию:*

- снизу вверх (Bottom Up Integration);
- сверху вниз (Top Down Integration);
- большой взрыв («Big Bang» Integration).

*Системное тестирование (System Testing)* – это тестирование проверка как функциональных, так и нефункциональных требований в системе в целом.

*Приемочное тестирование (Acceptance Testing)* – это формальный процесс тестирования, который проверяет соответствие системы требованиям и проводится с целью:

- определения удовлетворяет ли система приемочным критериям;
- вынесения решения заказчиком принимается приложение или нет.

Используется набор типичных тестовых случаев и сценариев, разработанный на основании требований к данному приложению.

### *Тестирование по признаку позитивности:*

*Позитивное тестирование* – это тестирование на тестовых данных или сценариях, которые соответствуют ожидаемому (штатному, нормальному) поведению системы согласно техническим требованиям и документации.

*Негативное тестирование* – это тестирование на тестовых данных или сценариях, которые соответствуют внештатному поведению тестируемой системы. Это могут быть исключительные ситуации (ошибки) или неверные данные.

**Пример. Позитивное тестирование:**

- умножить на калькуляторе целые числа 3 и 5;
- в игре посадить морковь на грядку для овощей;
- оплатить покупку действующей картой.

**Пример. Негативное тестирование:**

- умножить на калькуляторе число 3 на «груша» (значение «груша» не является валидным для калькулятора);
- в игре – посадить морковь на асфальте;
- оплатить покупку несуществующей картой.

Сценарии **позитивного тестирования** направлены на проверку работы системы с теми типами данных для которых, она разрабатывалась.

**Негативное тестирование** направлено на проверку устойчивости системы к различным воздействиям, валидации неверных данных, обработку исключительных ситуаций и т.п.

**Реакция продукта на тесты. Какой результат можно ожидать от позитивных и негативных тестов?**

**Позитивное тестирование** должно всегда давать результат в виде **отсутствия багов**.

**Негативное тестирование** могут дать 2 результата:

1. На данный ввод у объекта **есть ответ** в виде либо сообщения, либо определенного для данной ситуации действия.
2. Система **не знает**, как реагировать на введенные данные.

Существует **три реакции** на действия по вводу данных:

- **действие**: создание новой сущности, переход на новый шаг и т.п.
- **контроль**: сообщение с контролем, блокировка дальнейших действий и т.п.
- **отказ**: возникает исключение (Exception, 404-я ошибка и т.п).

Создание позитивных сценариев (тест-кейсов), как правило, предшествует созданию негативных.

## 10. Методы тестирования ПО

### *Тестирование по степени подготовленности к тестированию:*

*Тестирование по документации* – тестирование проводится по заранее подготовленным тестовым случаям.

*Интуитивное тестирование* – тестирование проводится без какой-либо подготовки, без цели и плана.

*Исследовательское тестирование* – тестирование с целью изучения проекта и документации на него, но без подготовленной заранее тестовой документации

### *Тестирование по степени автоматизации:*

*Ручное тестирование* – все тестирование проводится тестировщиком вручную без помощи скриптов.

*Полуавтоматизированное тестирование* – часть тестирования проводится вручную, а часть автоматизирована.

*Автоматизированное тестирование* – тестирование полностью автоматизировано.

### *Тестирование по знанию системы:*

*Тестирование черного ящика* – тестировщик не имеет доступа к коду.

*Тестирование белого ящика* – тестировщик имеет доступа к коду.

*Тестирование серого ящика* – тестировщик знает общую структуру приложения.

## Тест Дизайн (*Test Design*)

Роли, ответственные за тест дизайн

- Тест-аналитик – определяет «**ЧТО тестировать?**»
- Тест-дизайнер – определяет «**КАК тестировать?**»

## План тестирования

**Тест план (*Test Plan*)** – это документ, описывающий работы по тестированию, начиная с описания объекта, стратегии, расписания, критериев начала и окончания тестирования, до всего необходимого в процессе работы (оборудования, специальных знаний, оценки рисков, вариантов их разрешения).

Каждая методология разработки ПО предлагают свои форматы оформления планов тестирования. Например:

- Test Plan Template RUP
- Test Plan Template IEEE 829

## Шаблон тест плана:

- Что надо тестировать?
  - описание объекта тестирования: системы, приложения, оборудования, ...
- Что будете тестировать?
  - список функций и описание объекта тестирования и его состав
- Как будете тестировать?
  - стратегия тестирования, а именно: виды тестирования и их применение по отношению к объекту тестирования
- Когда будете тестировать?
  - последовательность проведения работ: подготовка (Test Preparation), тестирование (Testing), анализ результатов (Test Result Analysis)
- Критерии начала тестирования:
  - готовность тестовой платформы (тестового стенда)
  - законченность разработки требуемого функционала
  - наличие всей необходимой документации
  - ...
- Критерии окончания тестирования:
  - результаты тестирования удовлетворяют критериям качества продукта:
    - требования к количеству открытых багов выполнены
    - выдержка определенного периода без изменения исходного кода приложения Code Freeze (CF)
    - выдержка определенного периода без открытия новых багов Zero Bug Bounce (ZBB)
  - ...
- Окружение тестируемой системы (описание программно-аппаратных средств)
- Необходимое для тестирования оборудование и программные средства (тестовый стенд и его конфигурация, программы для автоматизированного тестирования и т.д.)
- Риски и пути их разрешения

**Набор тест кейсов и тестов (Test Case & Test suite)** – это последовательность действий, по которой можно проверить соответствует ли тестируемая функция установленным требованиям.

**Тестовый случай (Test Case)** – это артефакт, описывающий совокупность шагов, конкретных условий и параметров, необходимых для проверки реализации тестируемой функции или её части.

#### Структура *Test Case*:

<b>Action</b> (Действие)	>	<b>Expected Result</b> (Ожидаемый результат)	>	<b>Test Result</b> (Результат тестирования) (passed/failed/blocked)
-----------------------------	---	---	---	---

Результат тест кейса: *passed / failed / blocked* (успешно / ошибка / заблокирован)

#### Пример:

Действие	Ожидаемый результат	Результат тестирования (passed / failed / blocked)
сложение на калькуляторе целых чисел 3 и 5	целое число, равное сумме чисел 3 и 5	passed (или успешно)

#### Виды тестовых случаев (по ожидаемому результату):

тест кейсы разделяются по ожидаемому результату на **позитивные** и **негативные**

##### Позитивный тест кейс

- использует только корректные данные
- проверяет, что приложение правильно выполнило вызываемую функцию.

##### Негативный тест кейс

- использует корректные данные и
- использует некорректные данные (минимум 1 некорректный параметр)
- проверяет, что приложение не выполняет вызываемую функцию (срабатывание валидатора).

**!** При автоматизированном тестировании актуально:  
Каждый тест кейс должен иметь 3 части:

<b><i>PreConditions</i></b> <b><i>(предусловия)</i></b>	Список действий, которые приводят систему к состоянию готовности для проведения основного тестирования.
<b><i>Test Case Description</i></b> <b><i>(описание действий)</i></b>	Список действий, переводящих систему из одного состояния в другое, для получения результата тестирования
<b><i>PostConditions</i></b> <b><i>(постусловия)</i></b>	Список действий, переводящих систему в первоначальное состояние (состояние до проведения теста – <i>initial state</i> )

**Примечание:** *постусловия* не является обязательной частью. Это правило хорошего тона: «*намусорил – убери за собой*» (актуально при автоматизированном тестировании, т.к. за один прогон база данных наполняется сотней и более некорректных документов).

## Пример позитивного тестирования

### Тест кейс 1:

Проверка отображения страницы		
Действие	Ожидаемый результат	Результат теста
Открыть страницу «Вход в систему»	<ul style="list-style-type: none"> <li>- окно «Вход в систему» открыто</li> <li>- название окна – Вход в систему</li> <li>- логотип компании отображается в левом верхнем углу</li> <li>- на форме 2 поля – Логин и Пароль</li> <li>- кнопка Вход доступна</li> <li>- ссылка «забыл пароль» – доступна</li> </ul>	...

## 11. Процесс тестирования

Автоматизация – ручное тестирование (или иное).

**Последовательность действий:**

1. Тестирование начинается после получения спецификаций на разрабатываемое ПО.
2. Подготовлен первый прототип. Необходимо провести **дымовое тестирование**, по результатам которого делается вывод о возможности дальнейшего тестирования.
3. В случае если «*smoke test failed!!!*», переходим к п.2 (приложение отправляется на доработку).
4. В случае если «*smoke test passed!!!*», переходим к следующему виду тестирования – **регрессионное тестирование** (Regression testing) и **санитарное тестирование** (Sanity testing).

**Последовательность выполнения тестирования:**

Дымовое > Регрессионное > все остальные виды тестирования
---