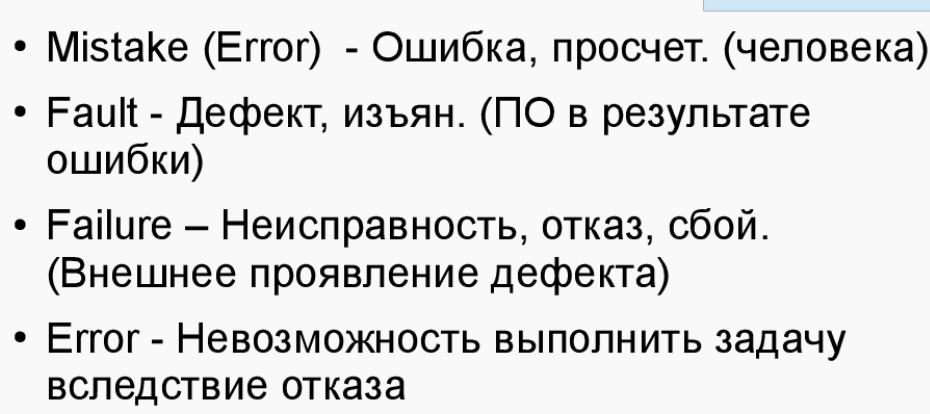
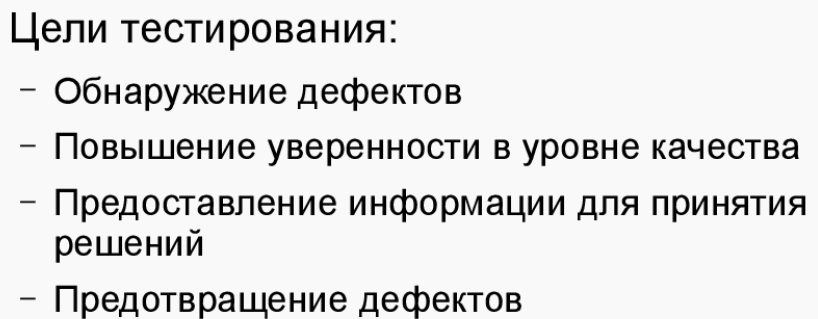
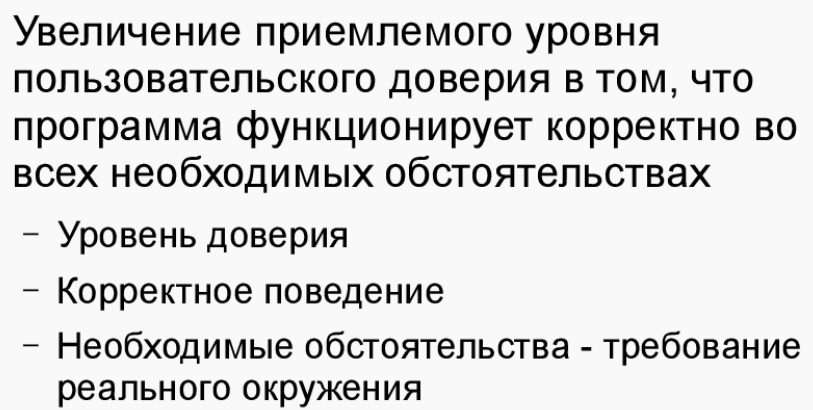
**Понятие тестирования ПО. Основные определения.**

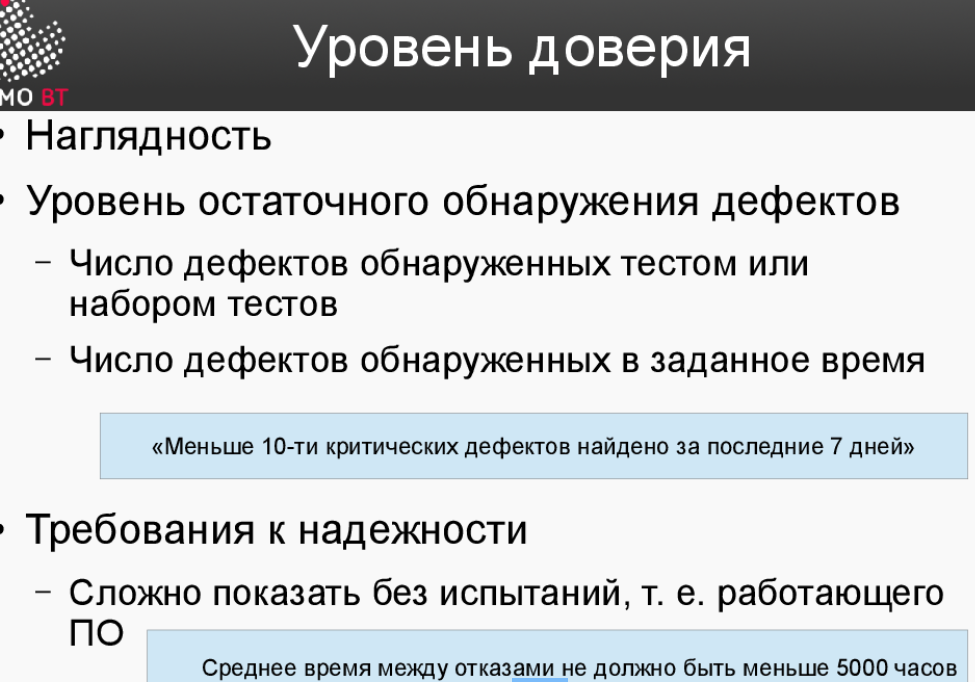
**Тести́рование програ́ммного обеспе́че́ния** — процесс исследования, испытания программного продукта, имеющий своей целью проверку соответствия между реальным поведением программы и её ожидаемым поведением на конечном наборе тестов, выбранных определённым образом.

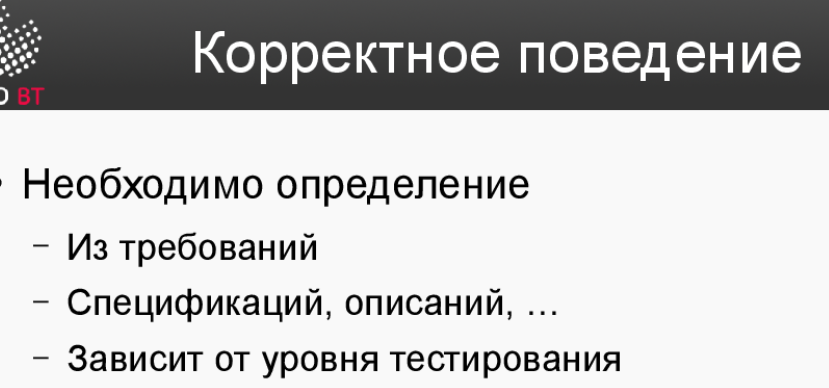


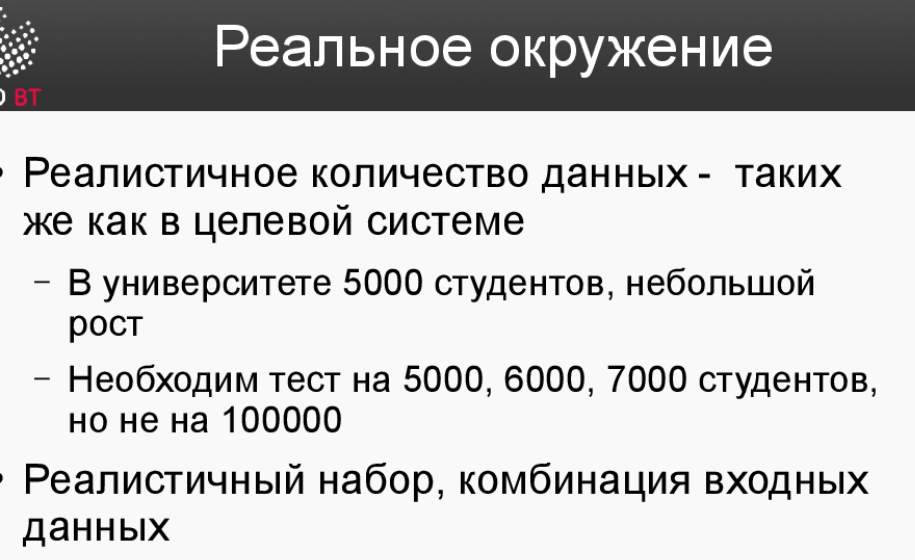
**Цели тестирования. Классификация тестов.**



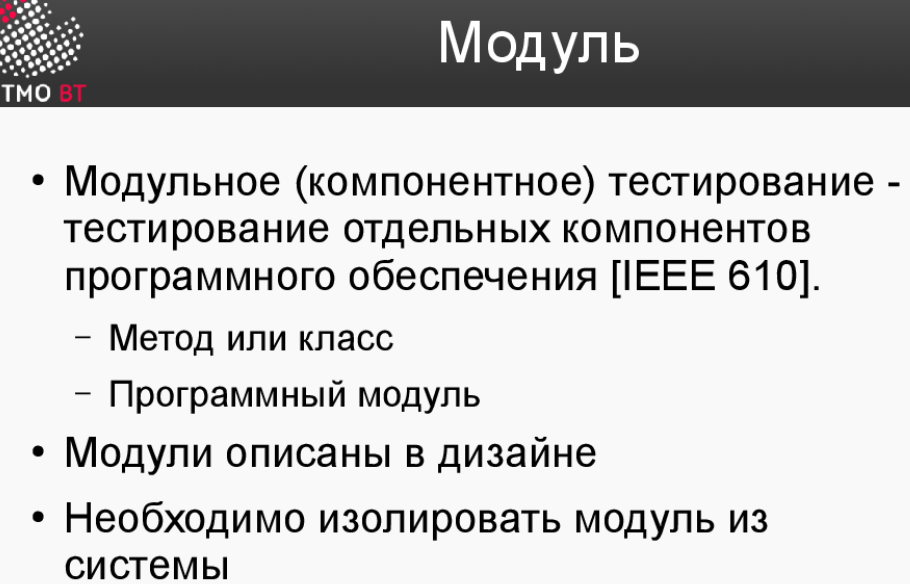


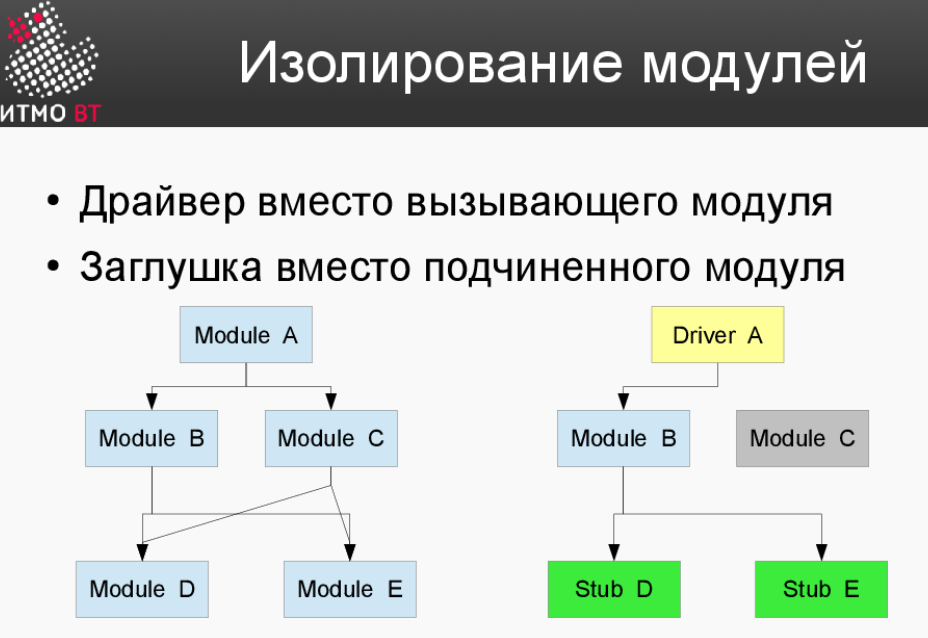


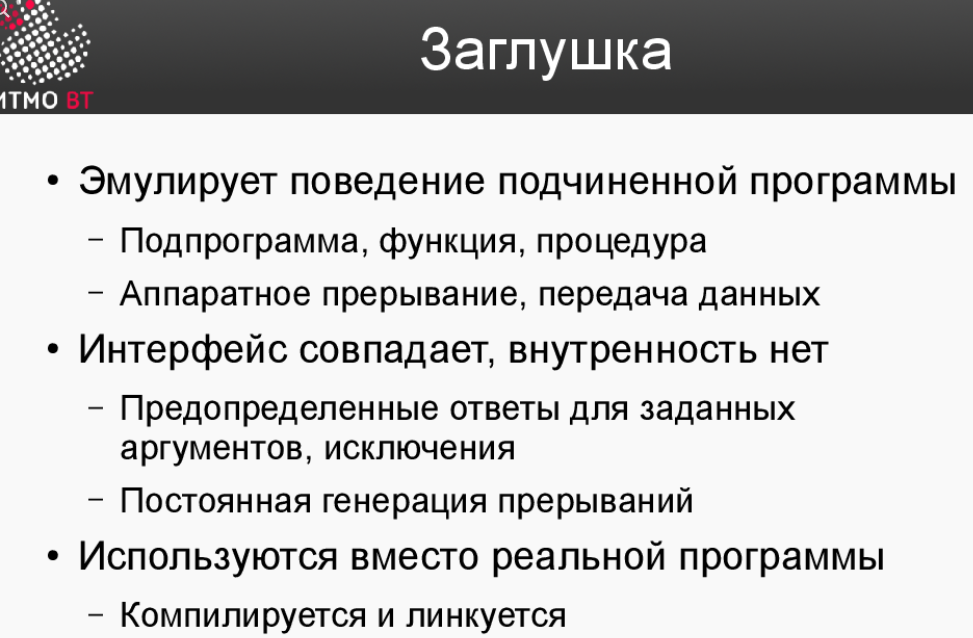


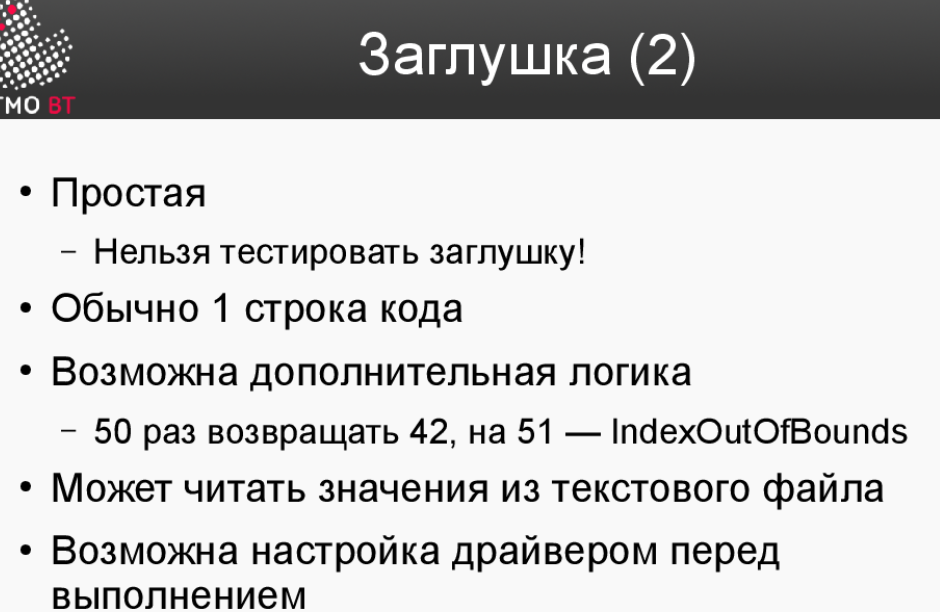


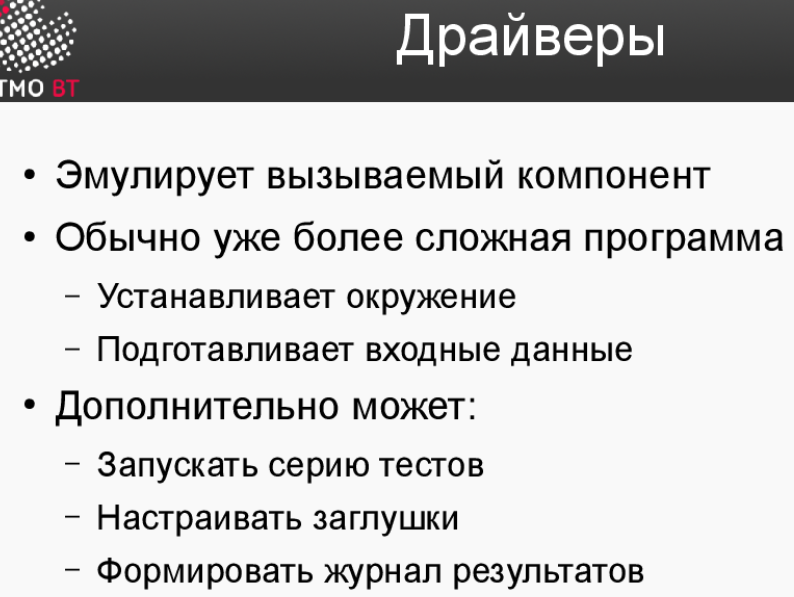
**Модульное тестирование. Понятие модуля.**

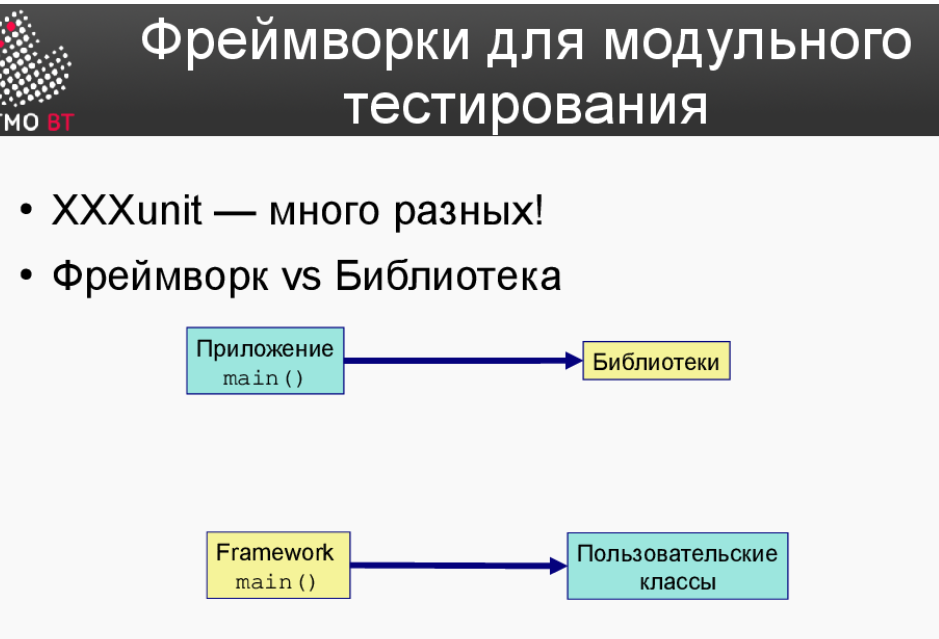




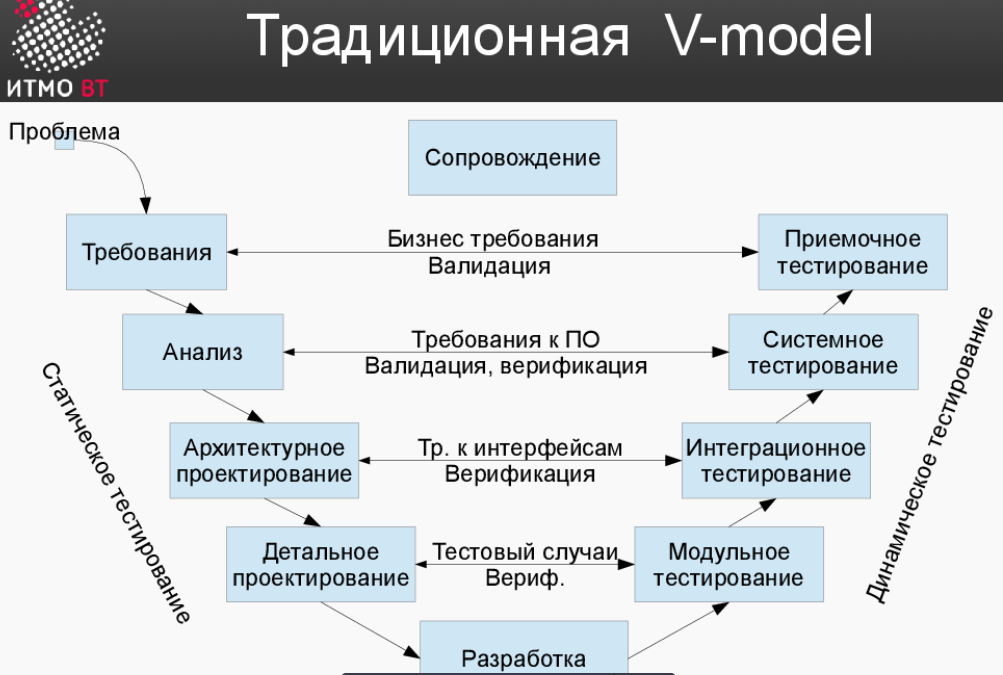


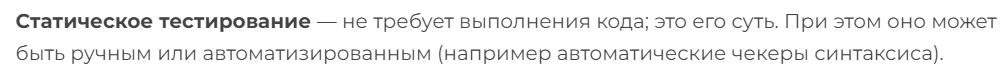






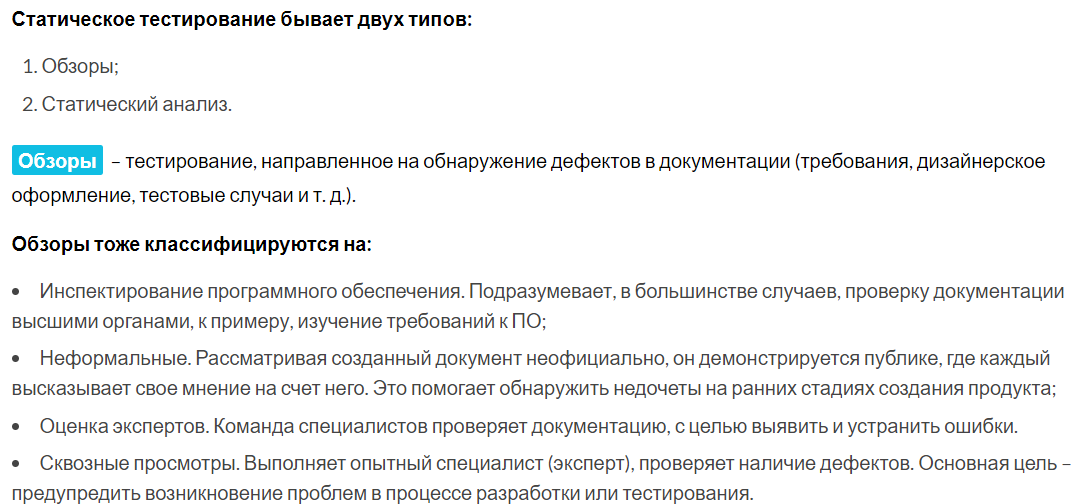
**V-образная модель. Статическое и динамическое тестирование.**

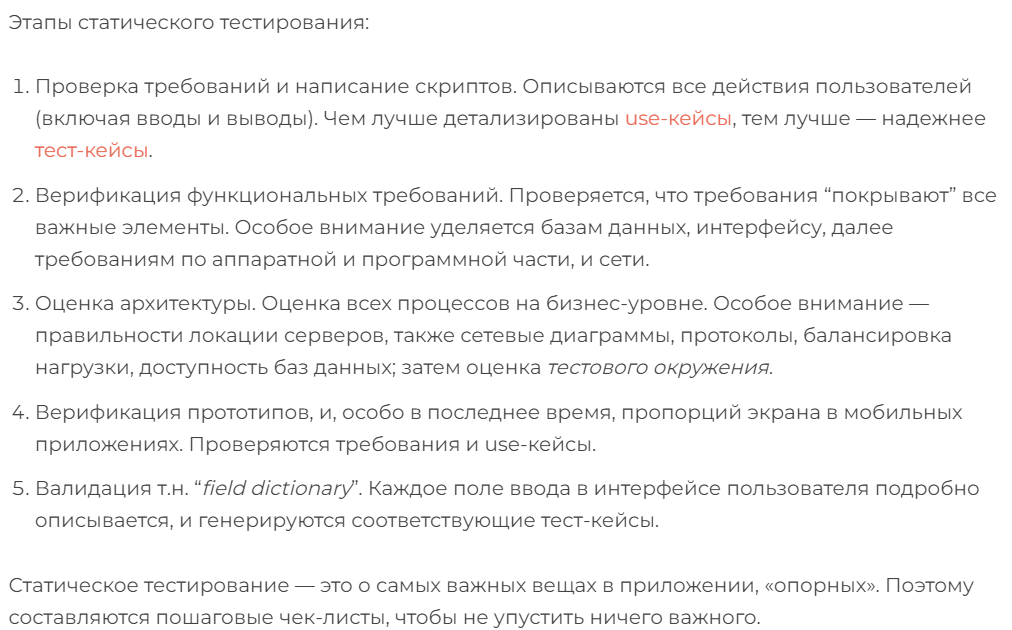


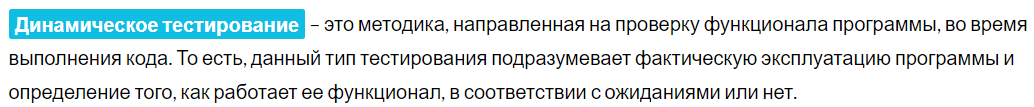


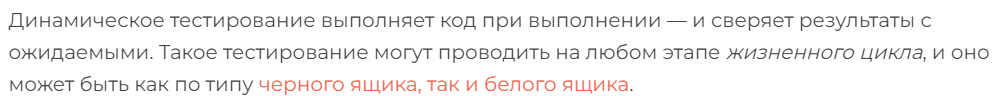
Начинаются на ранних этапах разработки.



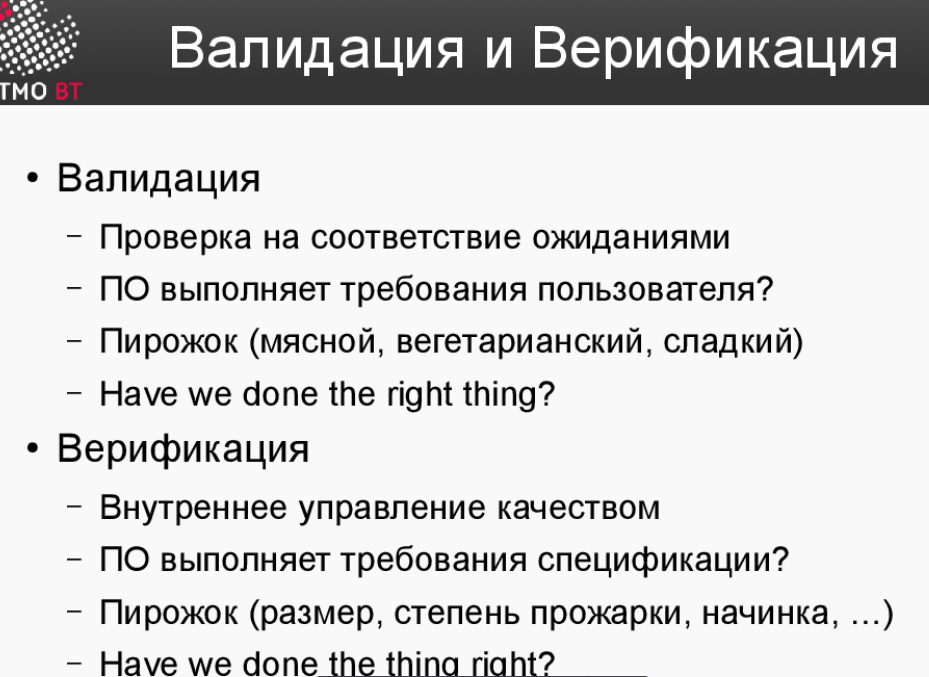


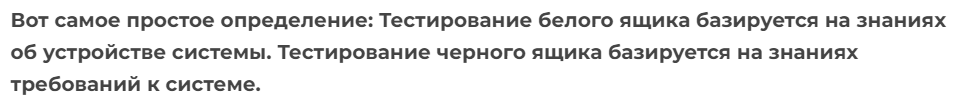




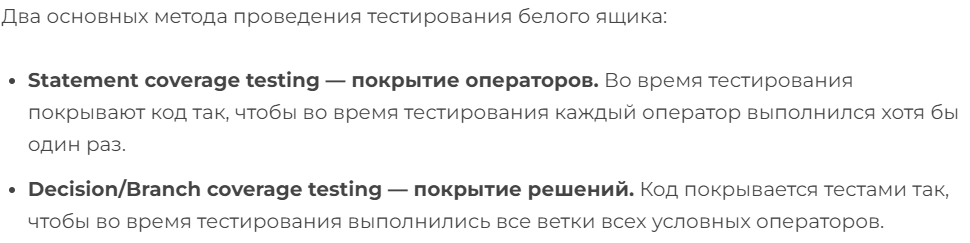


**Валидация и верификация. Тестирование методом "чёрного" и "белого" ящика.**

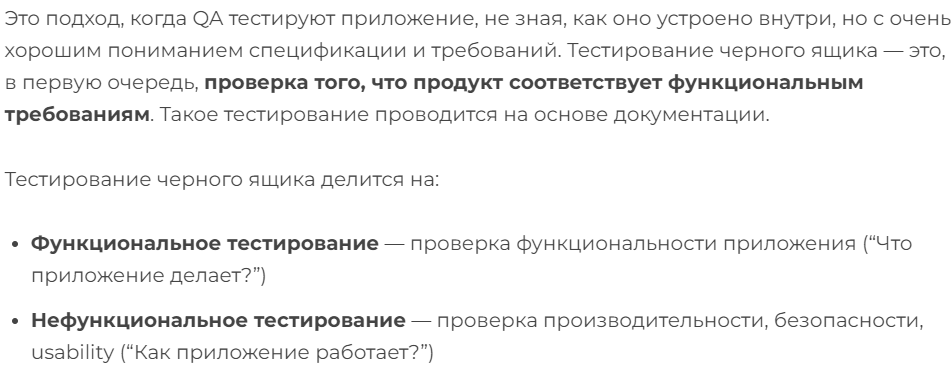




Часто тестирование белого ящика проводят сами разработчики.

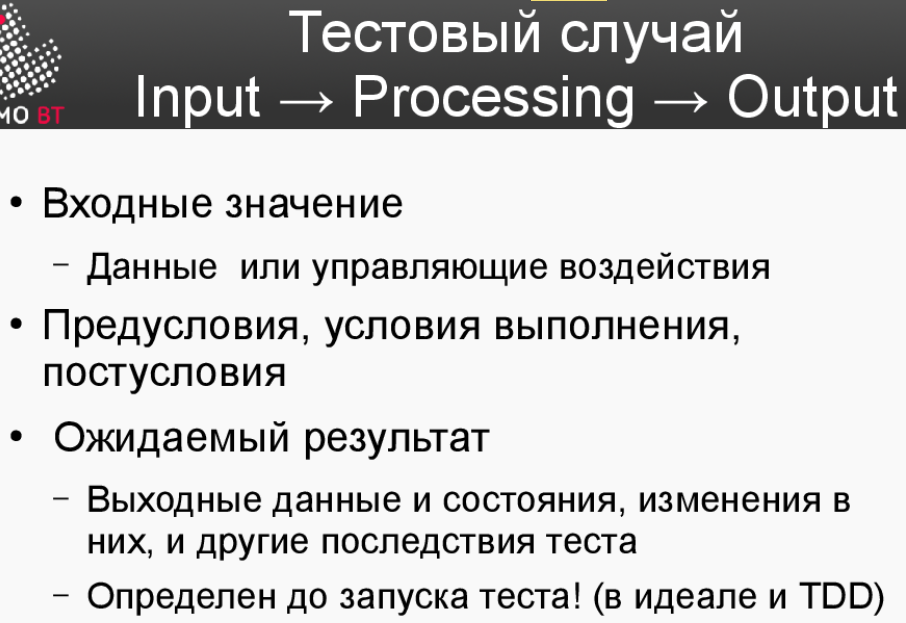


Тестирование черного ящика.



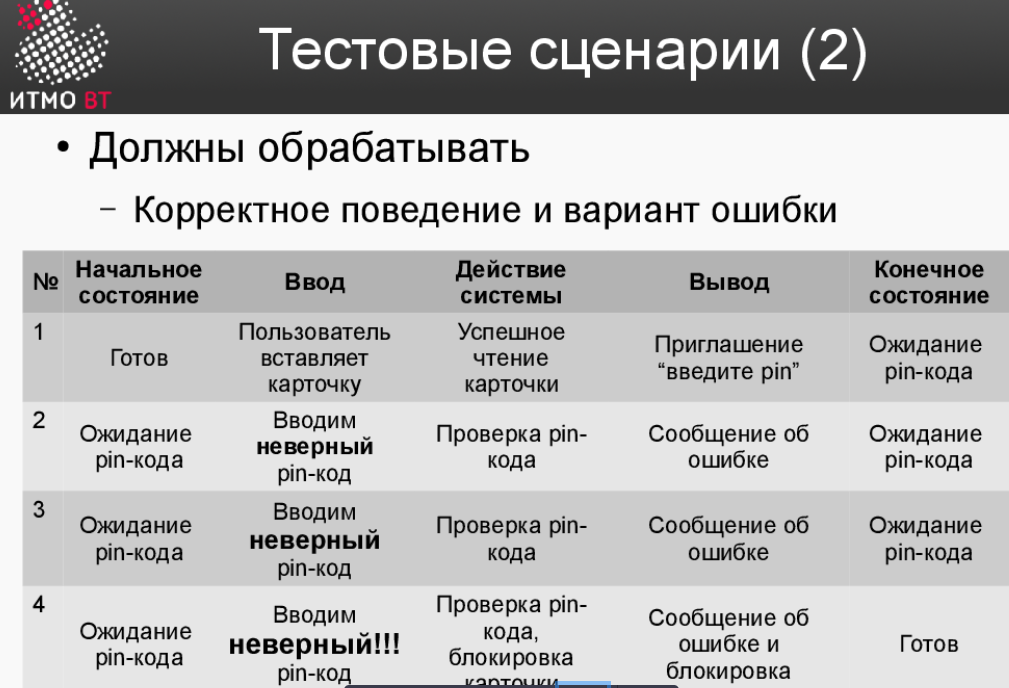
**Тестовый случай, тестовый сценарий и тестовое покрытие.**

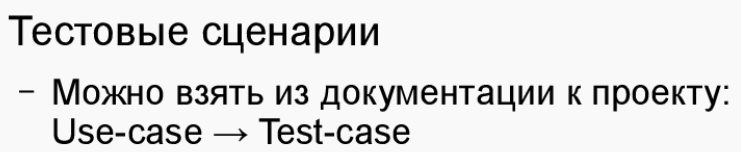
Тестовый случай — это артефакт, описывающий совокупность шагов, конкретных условий и параметров, необходимых для проверки реализации тестируемой функции или её части.

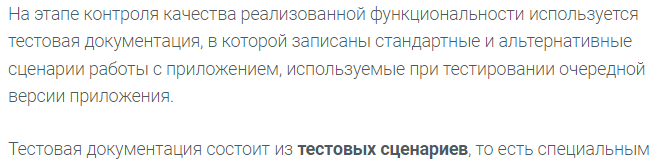


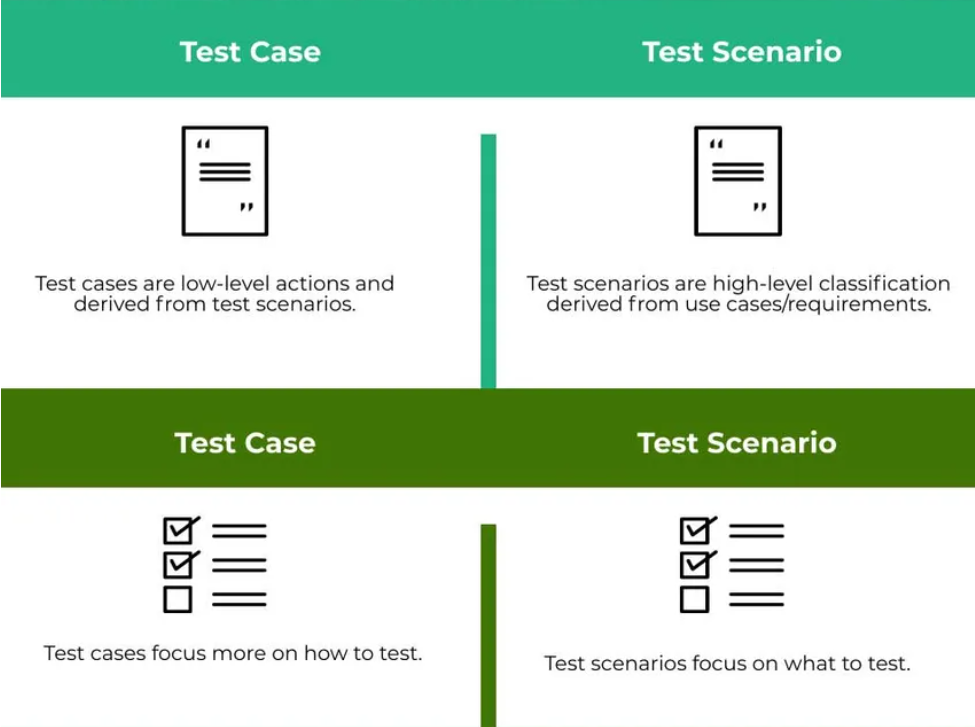
Повторяемый, автоматизируемый.

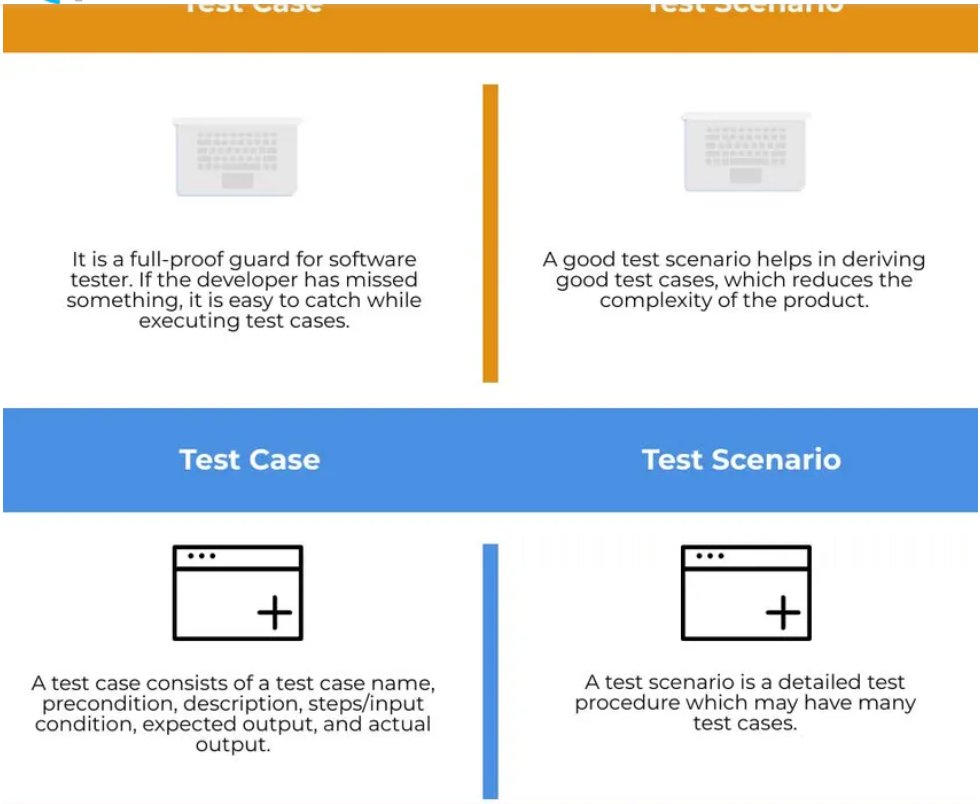




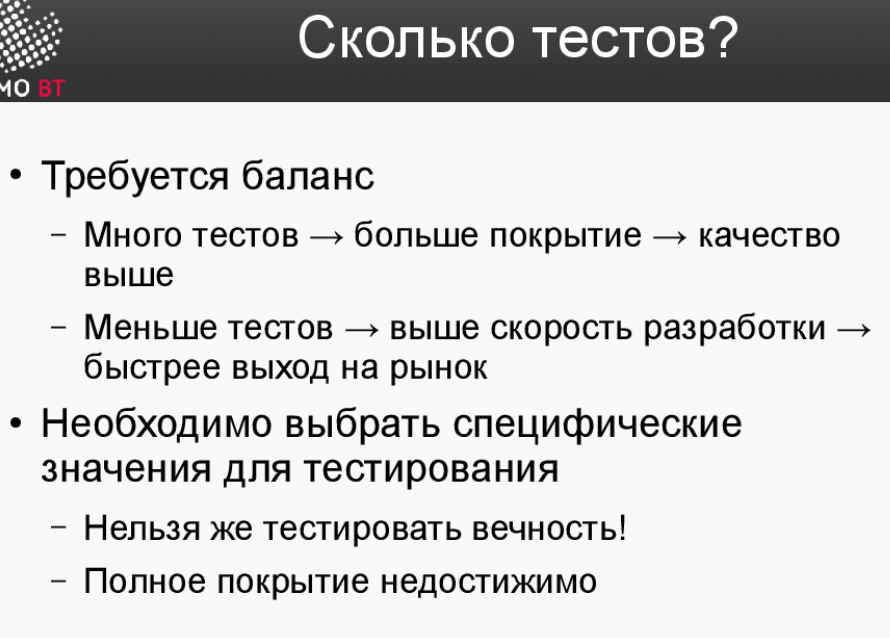








Тестовое Покрытие (Test Coverage) - это одна из метрик оценки качества тестирования, представляющая из себя плотность покрытия тестами требований либо исполняемого кода.

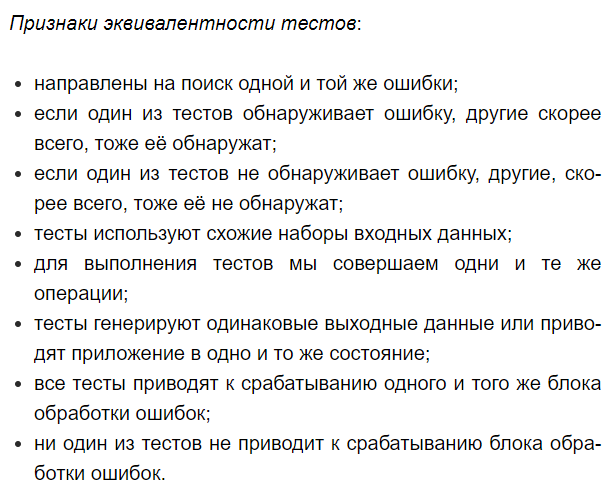


**Анализ эквивалентности.**

Техника для сокращения количества тестов, но при этом направленная на поддержание сохранения приемлемого тестового покрытия.

Класс эквивалентности (equivalence class) — одно или несколько значений ввода, к которым программное обеспечение применяет одинаковую логику.

Техника анализа классов эквивалентности — это техника, при которой мы разделяем функционал (часто диапазон возможных вводимых значений) на группы эквивалентных по своему влиянию на систему значений. Такое разделение помогает убедиться в правильном функционировании целой системы — одного класса эквивалентности, проверив только один элемент этой группы. Эта техника заключается в разбиении всего набора тестов на классы эквивалентности с последующим сокращением числа тестов.



**Таблицы решений и таблицы переходов.**

Таблицы решений используются в системах со сложной логикой. Это взаимосвязь между множеством условий и действий, позволяет упорядочить определенный аспект системы для упрощения его тестирования.

**В таблицах решений** представлен набор условий, одновременное выполнение которых должно привести к определённому действию.

Таблица принятия решений, как правило, разделяется на 4 квадранта:

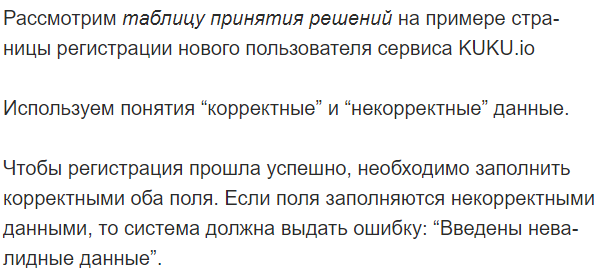
|  |  |
| --- | --- |
| Условия | Варианты выполнения действий |
| Действия | Необходимость действий |

Условия — список возможных условий.

Варианты выполнения действий — комбинация из выполнения и/или невыполнения условий этого списка.

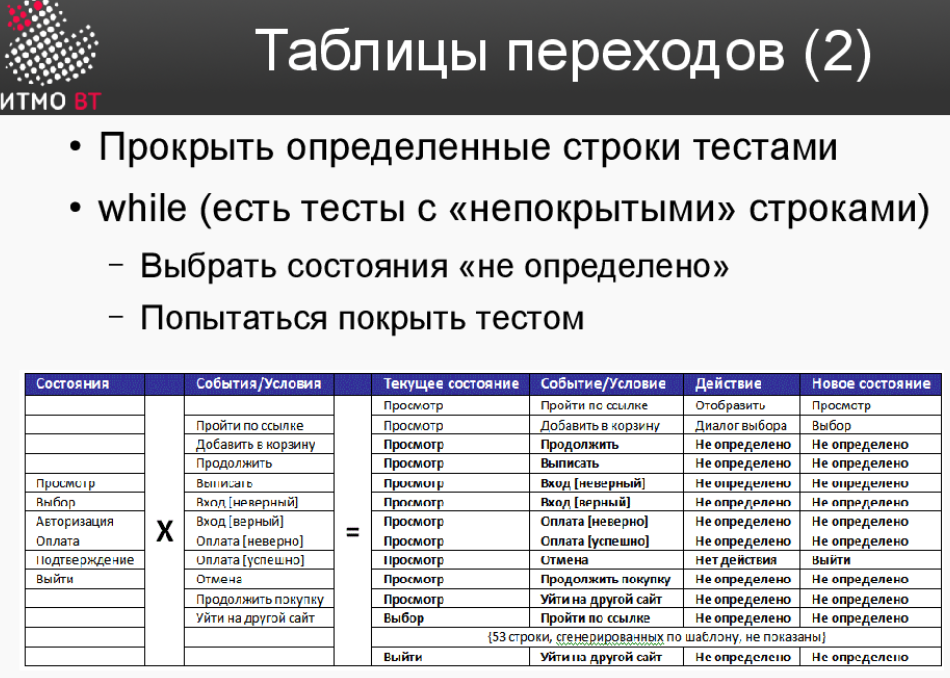
Действия — список возможных действий.

Необходимость действий — указание надо или не надо выполнять соответствующее действие для каждой из комбинаций условий.

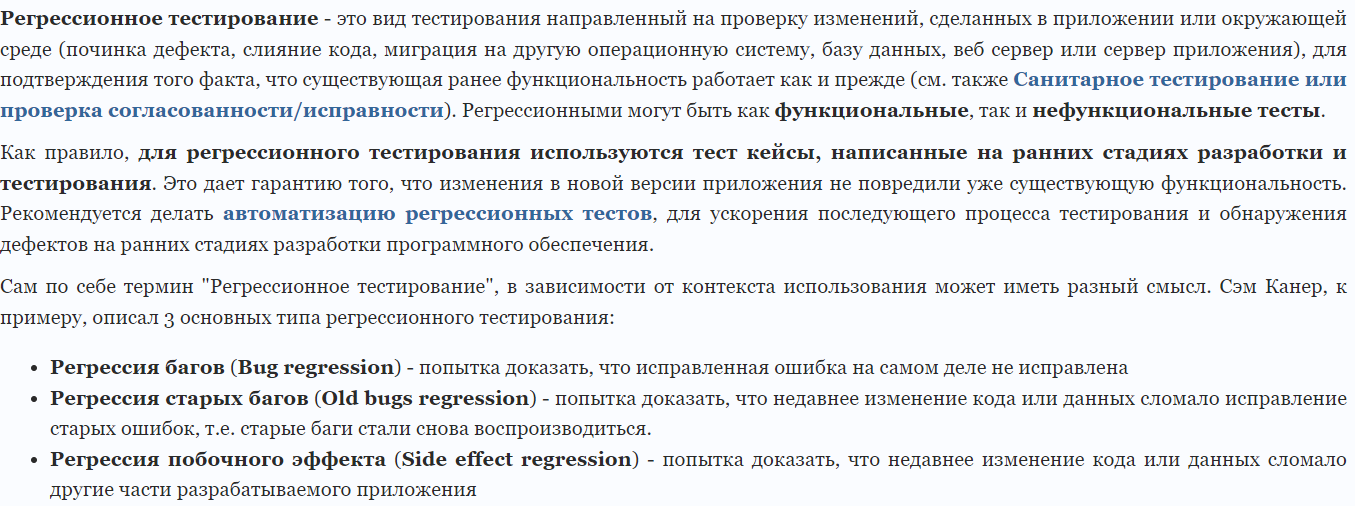




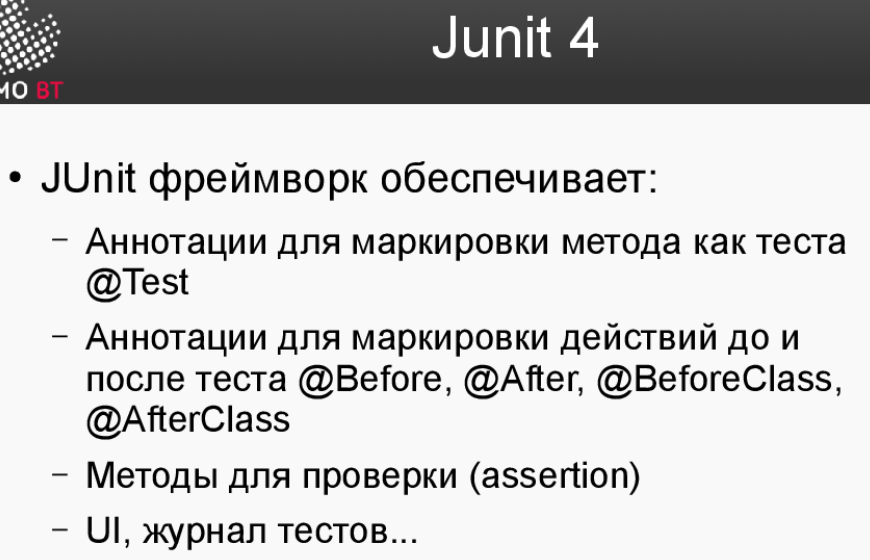
**Таблицы переходов** – используются в системах, аспекты которых можно представить в виде конечного автомата. (Это просто означает, что система может находиться в (конечном) числе разных состояний, а переходы из одного состояния в другое определяются правилами «машины».) Позволяет упорядочить определенный аспект системы для упрощения дальнейшего тестирования.

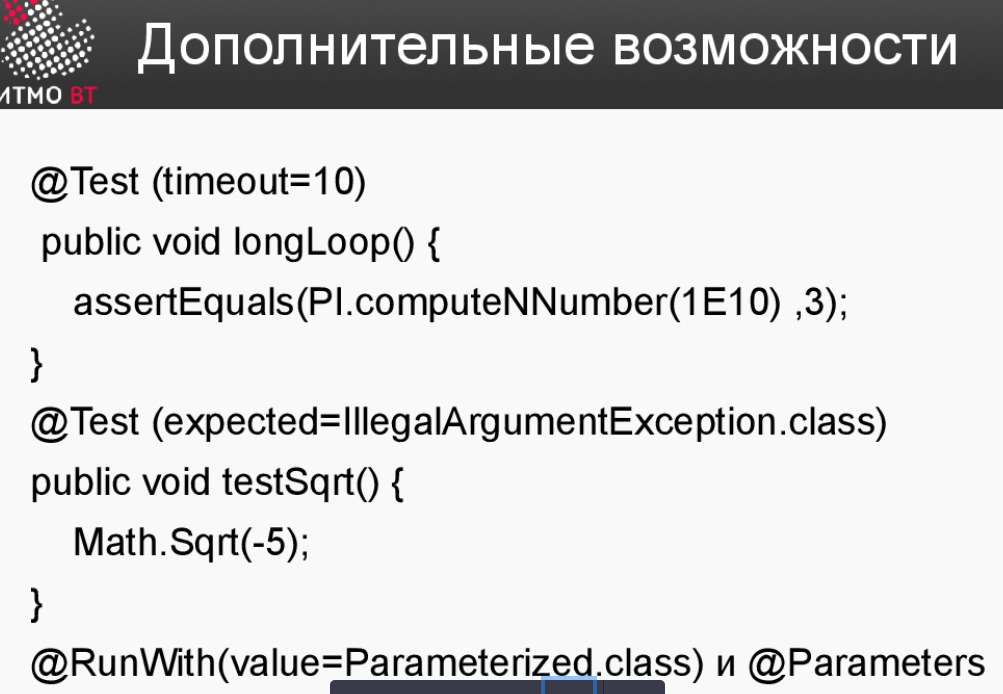


**Регрессионное тестирование.**

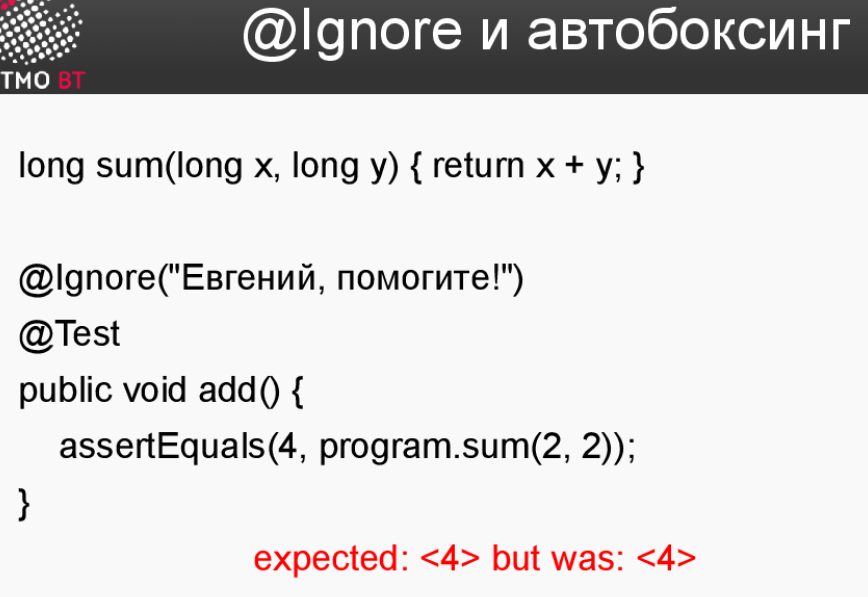


**Библиотека JUnit. Особенности API. Класс junit.framework.Assert.**

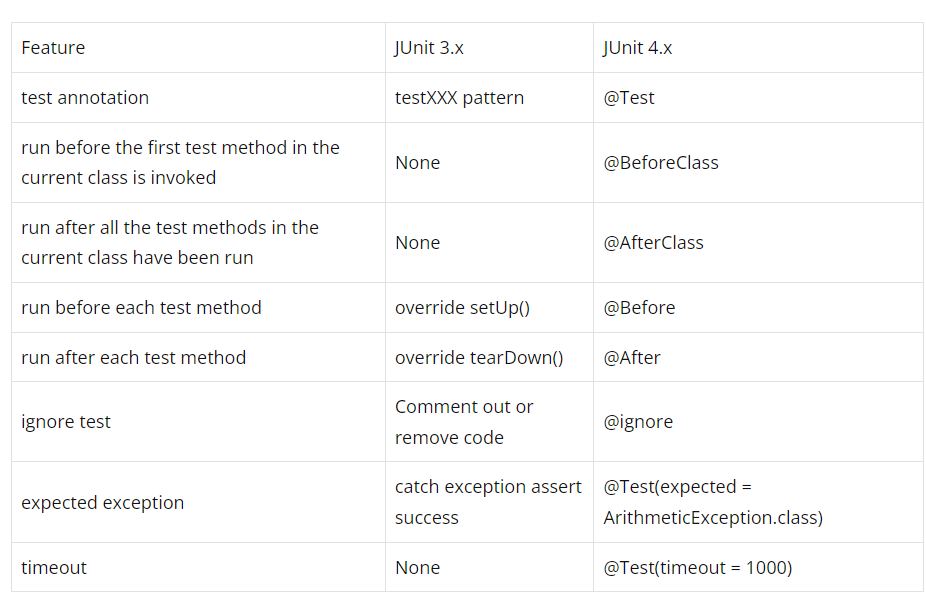




@Ignore – чтобы игнорировать тест



**Отличия JUnit 3 от JUnit 4.**



don't need to extend TestCase anymore.

@Test annotation replaces testSomeMethod() naming convention.

