

# Основы тестирования ПО

---

«Введение»



## В этом разделе:

---

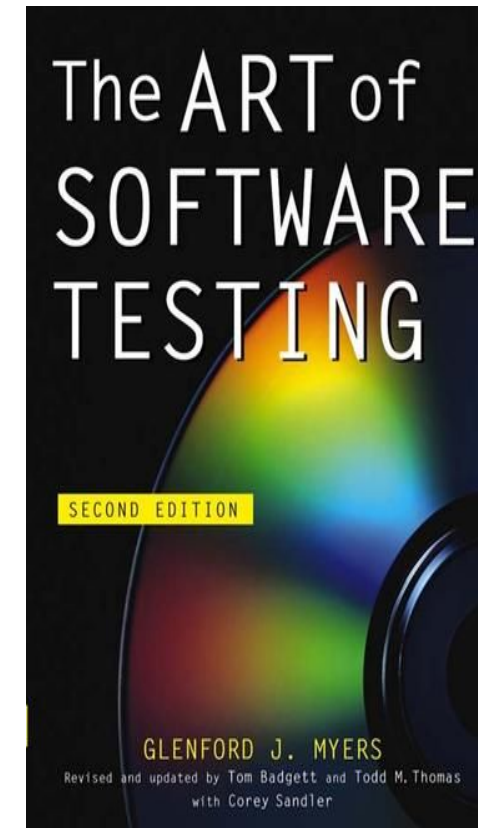
- История тестирования;
- Понять, какие качества делают тестировщиков хорошими;
- Почему тестирование необходимо;
- Базовая терминология.

Немного истории...

**60-е годы.**  
60-е годы –  
«исчерпывающее  
тестирование»



20 вложенных операторов if =>  
1'048'576 ветвей выполнения



## 70...80-е годы.

---

70-е годы – «поиск дефектов»

Почему?



80-е годы – «предупреждение дефектов»



80-е годы – «предупреждение  
дефектов»



90-е – 00-е годы – «обеспечение  
качества»



Современный этап –  
«гибкие методологии,  
тесная интеграция с  
разработкой,  
автоматизация»

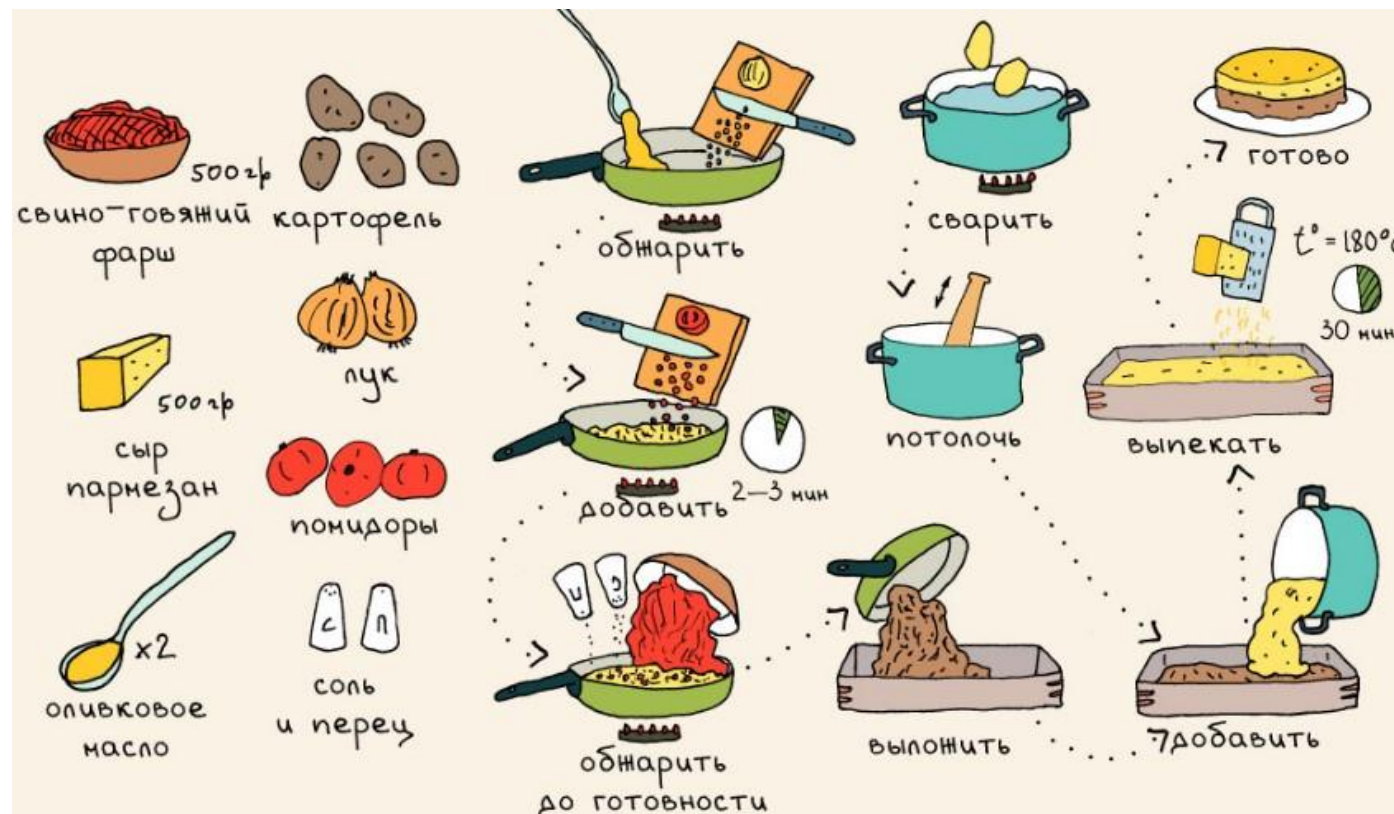




# Пара слов о методологиях

# Методология/модель/процесс разработки ПО

—  
это структура (набор правил), согласно которой построена разработка.



# Методология разработки ПО схематично:

Анализ требований

Спецификация (документ)

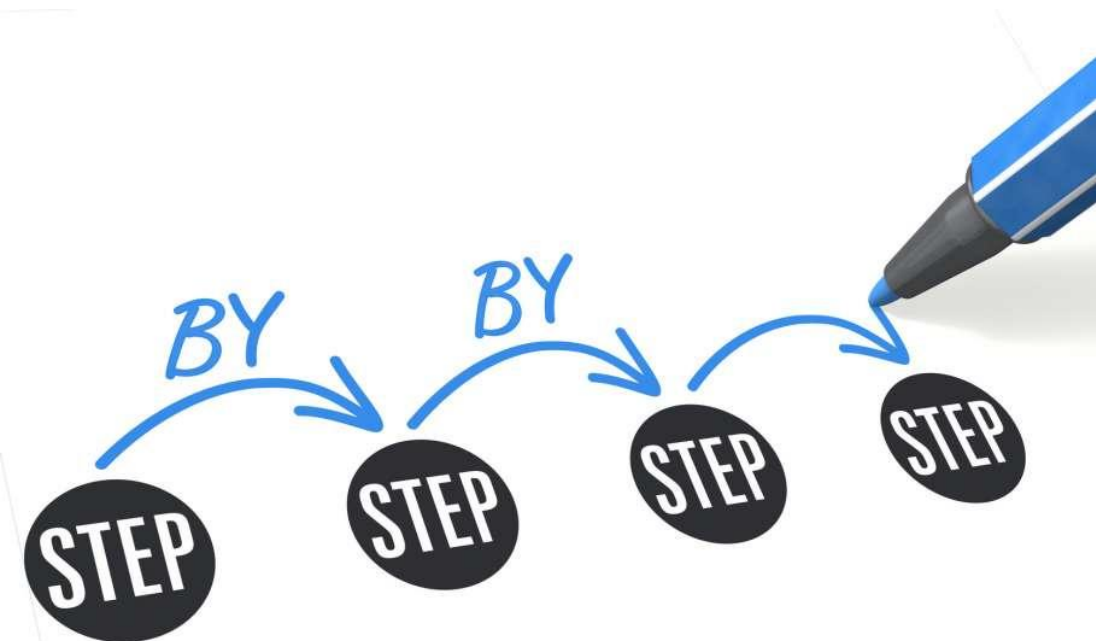
Проектирование ПО

Программирование

Тестирование

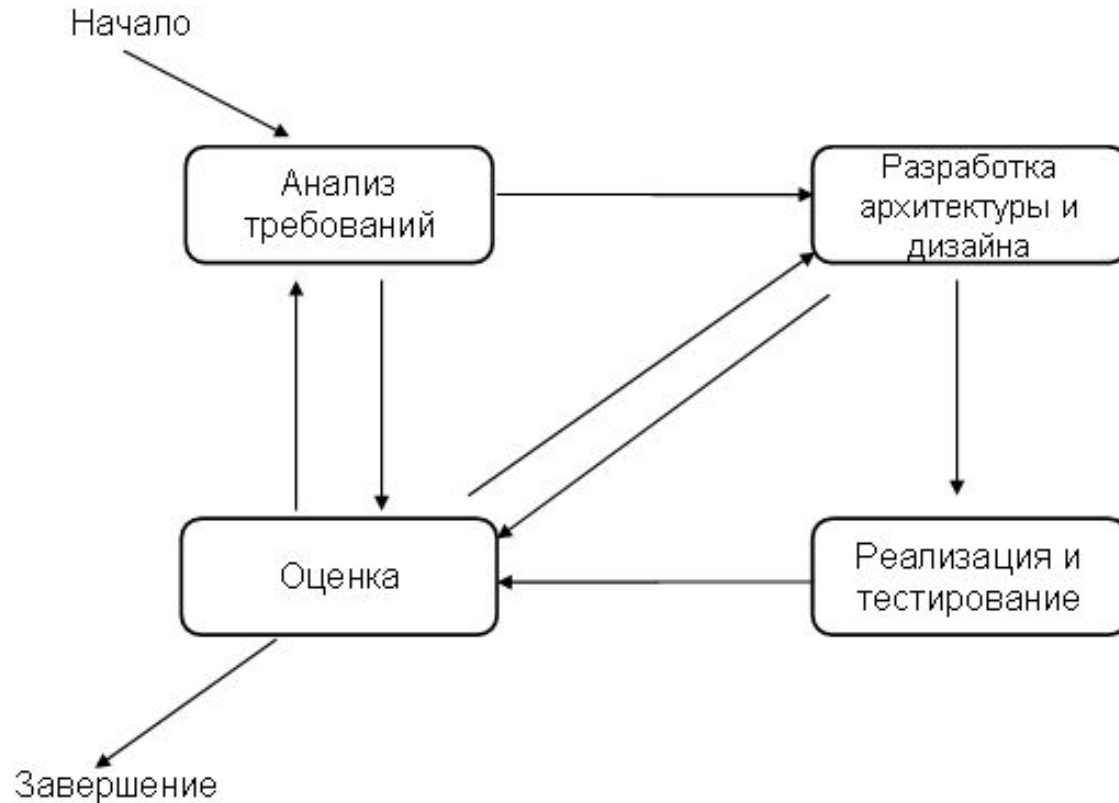
Внедрение (установка)

Сопровождение

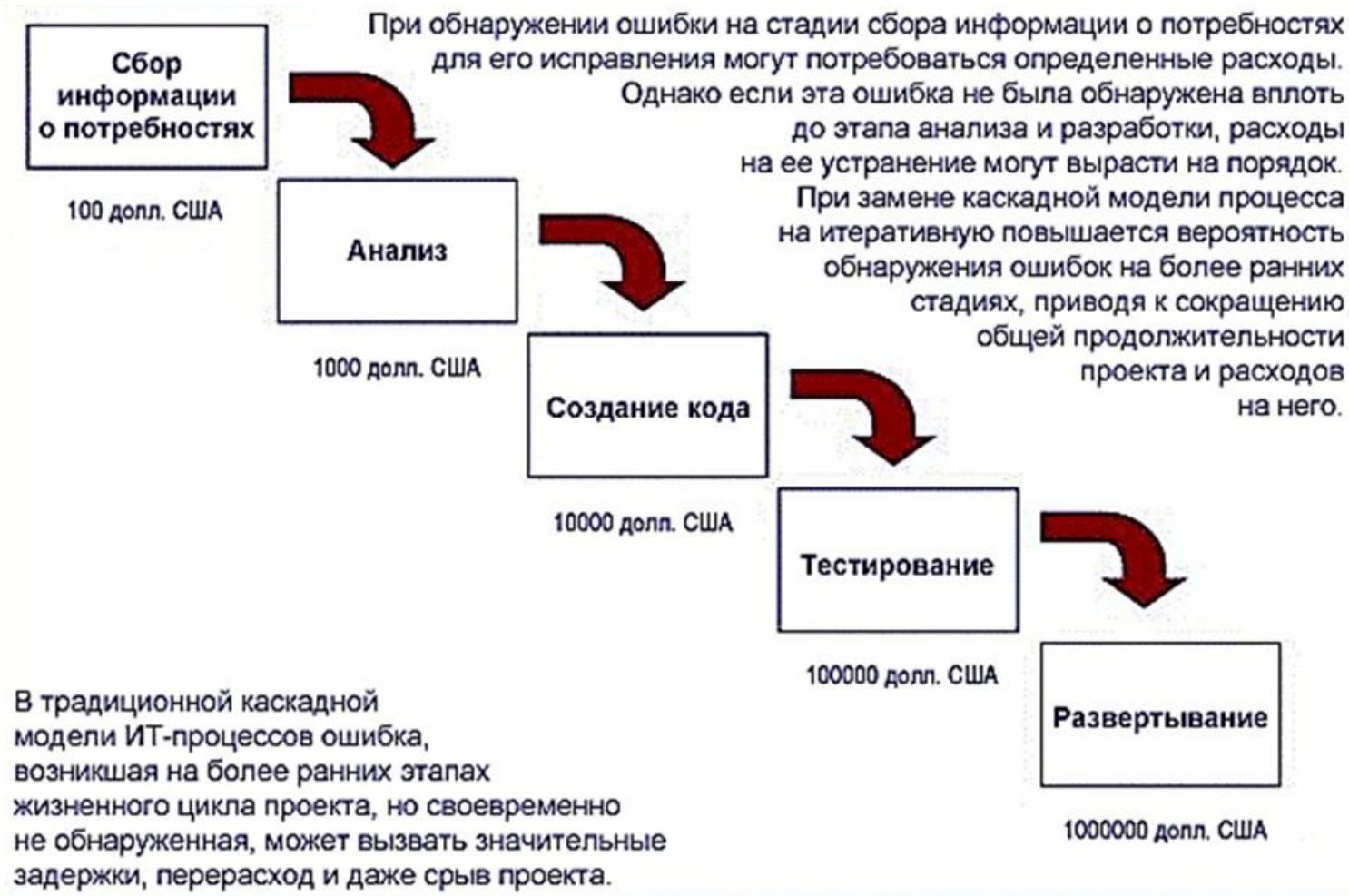


# Классические методология разработки ПО

водопадная и итерационная:



# Каскадный процесс



**Agile Manifesto** разработан и принят 11-13 февраля 2001 года на лыжном курорте The Lodge at Snowbird в горах Юты.

Манифест подписали представители следующих методологий:

- Extreme programming
- Scrum
- DSDM
- Adaptive Software Development
- Crystal Clear
- Feature-Driven Development
- Pragmatic Programming.

**Содержит 4 основные идеи и 12 принципов.  
Не содержит практических советов!**

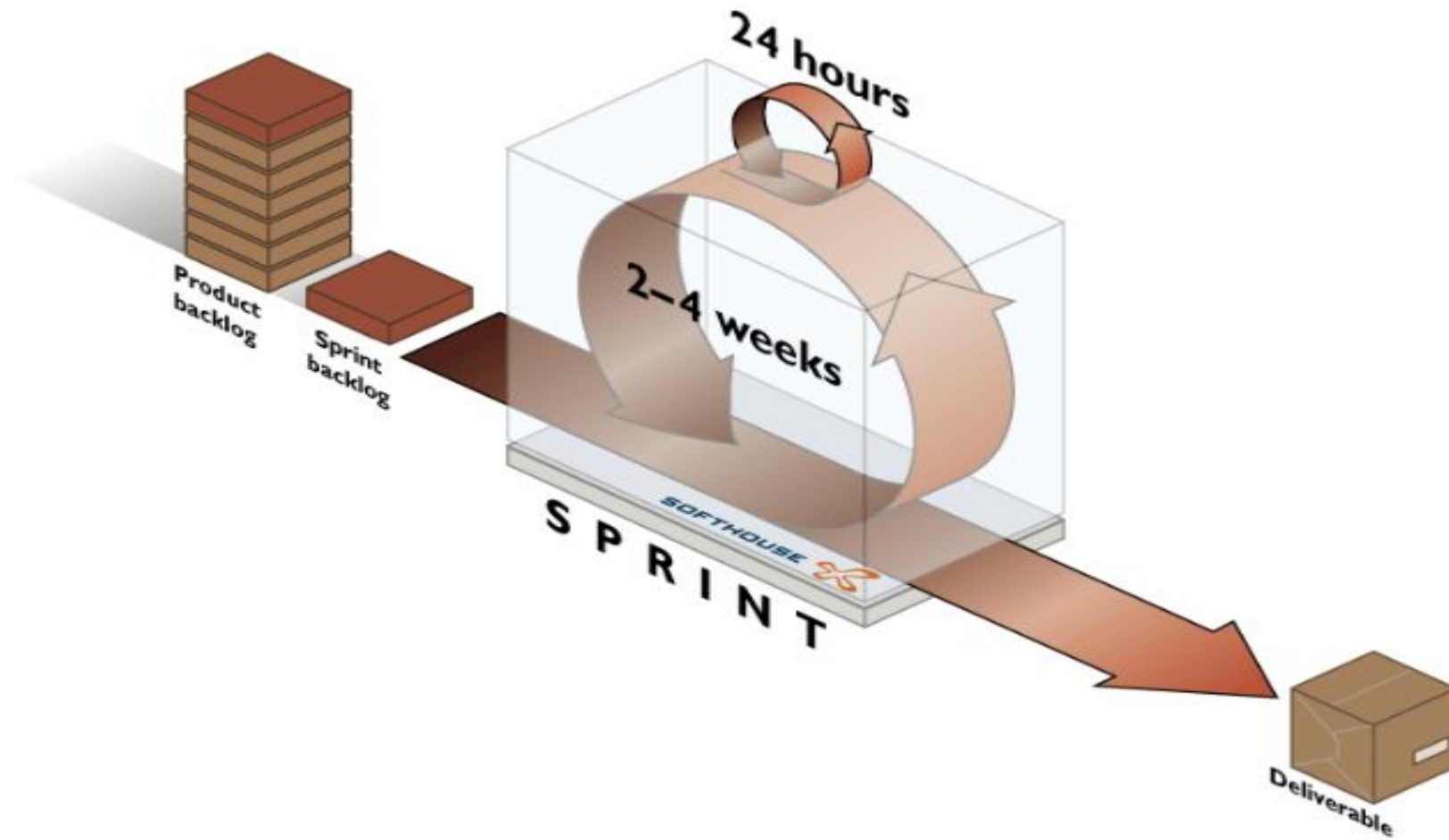
## Идеи:

- **Личности и их взаимодействия** важнее, чем процессы и инструменты;
- **Работающее программное** обеспечение важнее, чем полная документация;
- **Сотрудничество с заказчиком** важнее, чем контрактные обязательства;
- **Реакция на изменения** важнее, чем следование плану.

## Принципы:

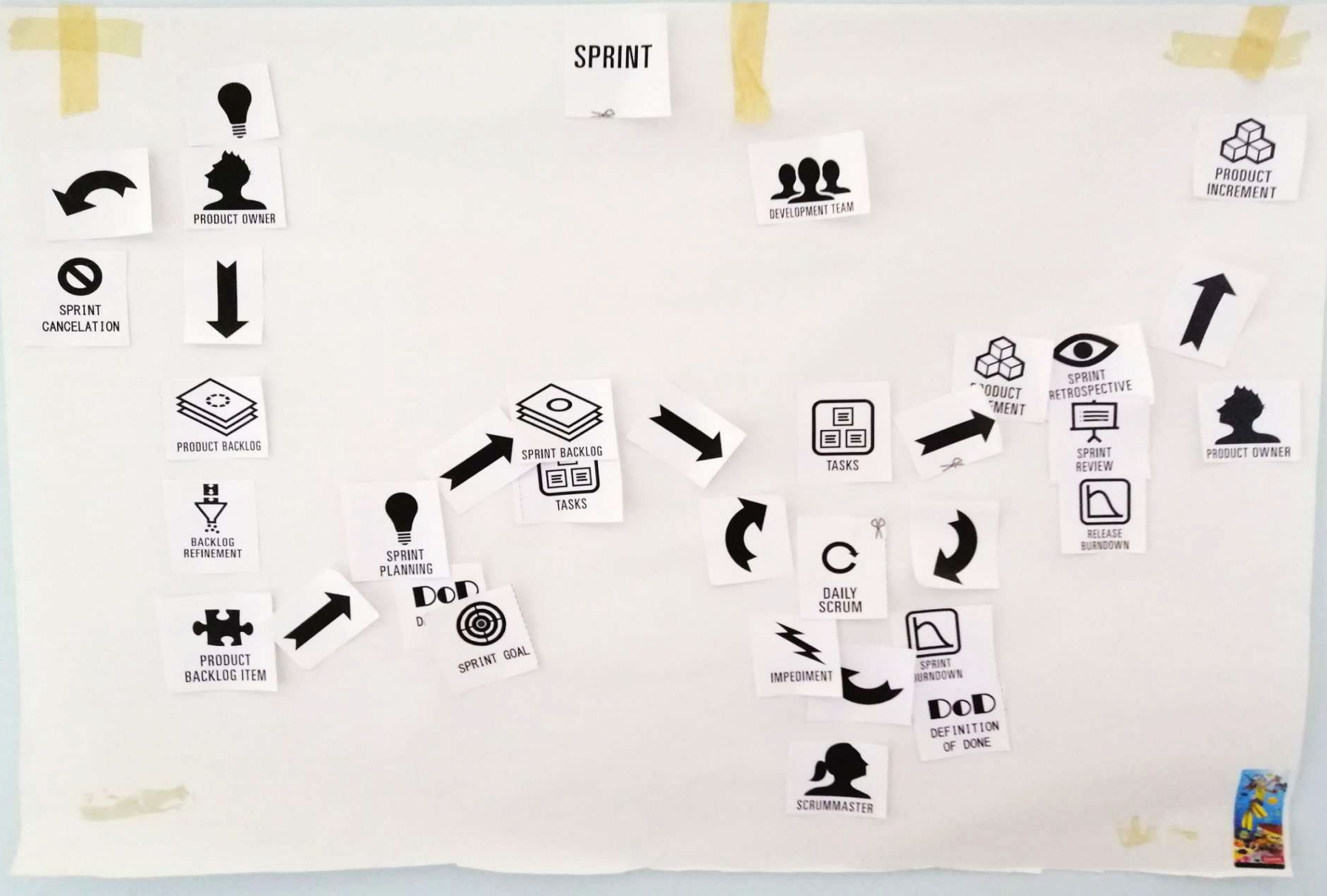
- удовлетворение клиента за счёт ранней и бесперебойной поставки ценного ПО;
- приветствие изменений требований, даже в конце разработки;
- частая поставка рабочего ПО (каждый месяц или неделю или ещё чаще);
- тесное, ежедневное общение заказчик <-> разработчики;
- проектом занимаются мотивированные личности, которые обеспечены нужными условиями работы, поддержкой и доверием;
- рекомендуемый метод передачи информации — личный разговор;
- работающее ПО — лучший измеритель прогресса;
- спонсоры, разработчики и пользователи должны иметь возможность поддерживать постоянный темп на неопределенный срок;
- постоянное внимание на улучшение технического мастерства и удобный дизайн;
- простота — искусство НЕ делать лишней работы;
- лучшие технические требования, дизайн и архитектура получаются у самоорганизованной команды;
- постоянная адаптация к изменяющимся обстоятельствам.

# Agile методологии (Scrum, Kanban... и множество других).





# SPRINT

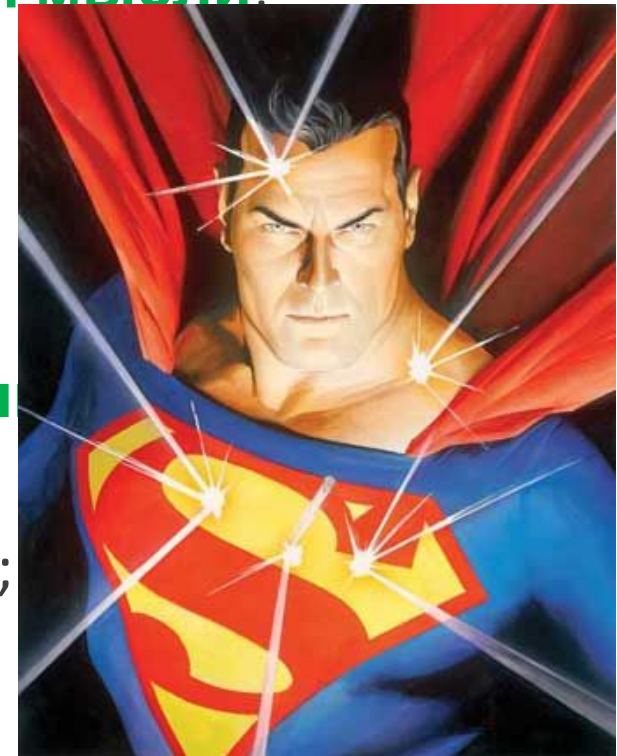


# Психология тестирования

# Психология тестирования.

Психологические навыки и особенности тестировщика таковы:

- **Хорошие коммуникативные навыки;**
- **Способность ясно, быстро, чётко выразить свои мысли;**
- Исполнительность;
- Ответственность;
- Терпение, усидчивость, внимательность к деталям, наблюдательность;
- Гибкое мышление, **хорошая способность к обучению;**
- Хорошее абстрактное и аналитическое мышление;
- Способность ставить нестандартные эксперименты;
- Склонность к исследовательской деятельности..



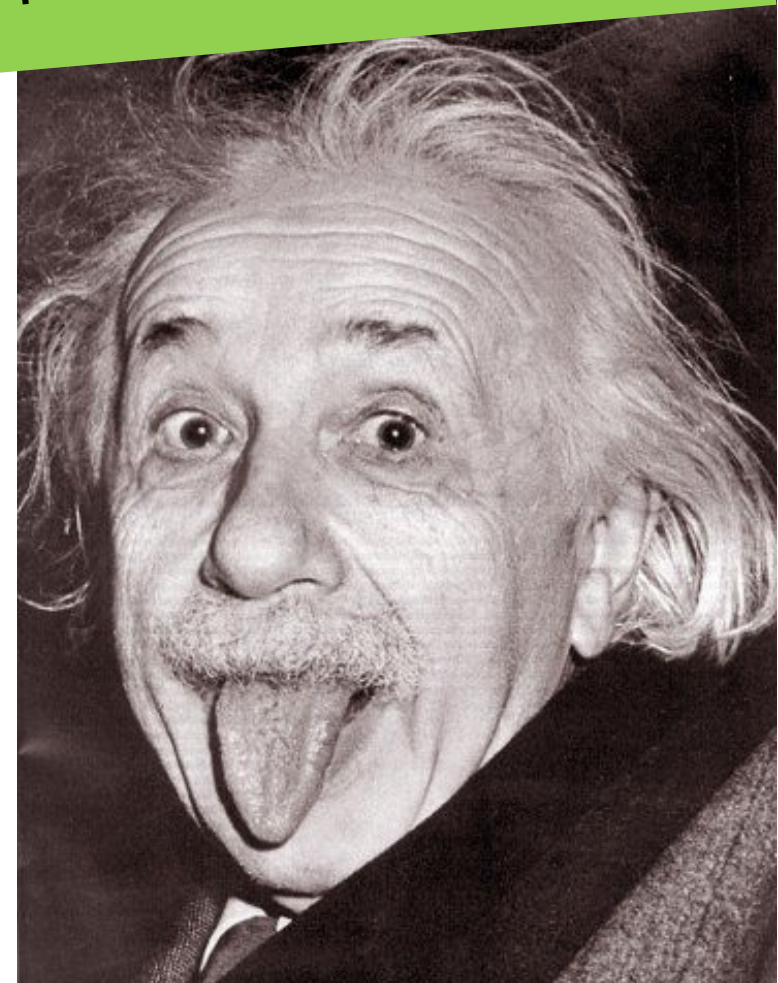
# Какие знания нужны:

## English

### Технические навыки:

- Общение с СУБД: **SQL**.
- Основы **Web** (Client-Server, HTTP, HTML)
- Администрирование ОС: Windows, Sun Solaris, HP-UX, Free-BSD, Linux.
- Сетевое администрирование: TCP/IP, IPX/SPX, NetBIOS.
- Программирование: C/C++/C#, Java, PHP, Object Pascal, Visual Basic, JavaScript, HTML, .NET.

Первоначально важно хотя бы "Умение излагать мысли и замечания на родном языке", а потом уже "Английский для тестировщиков"!



# Почему тестирование необходимо?

# Почему тестирование необходимо?

**Бизнес:** «Пользователи склонны пользоваться качественными продуктами (даже если они дороже)»



# Почему тестирование необходимо?

Пользователи: «лучше не  
рисковать личными  
данными, деньгами и т.п.»



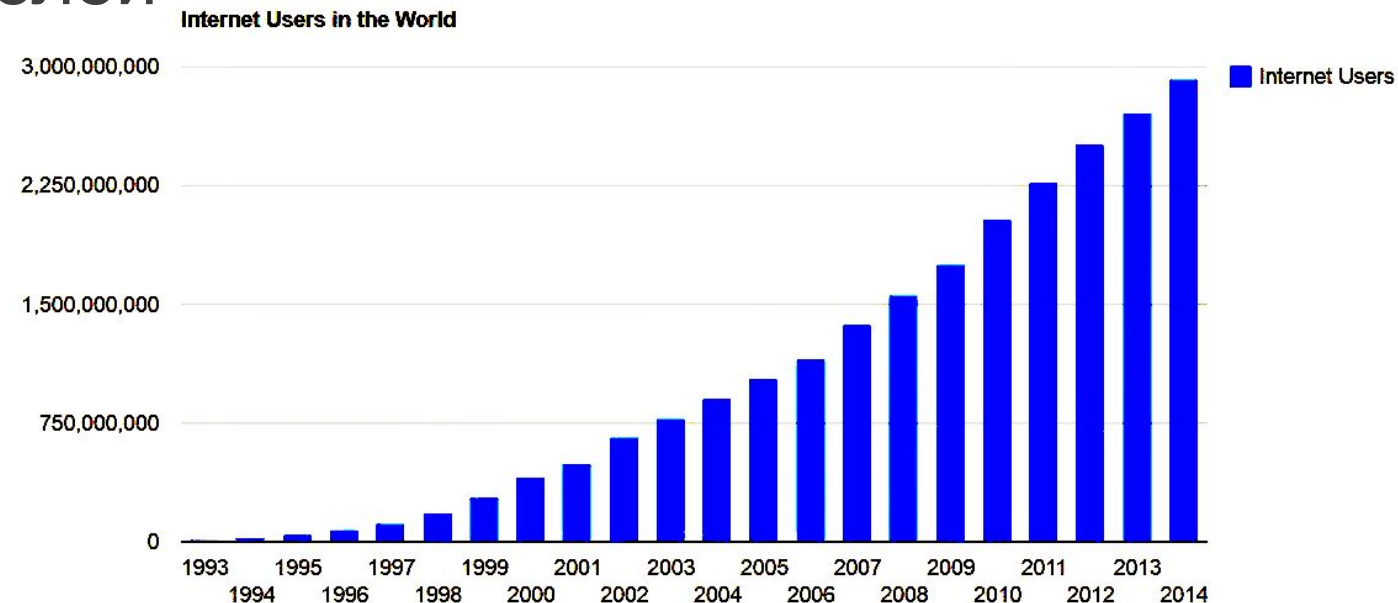




# Почему тестирование необходимо?

Почему тестирование становится все более важным?

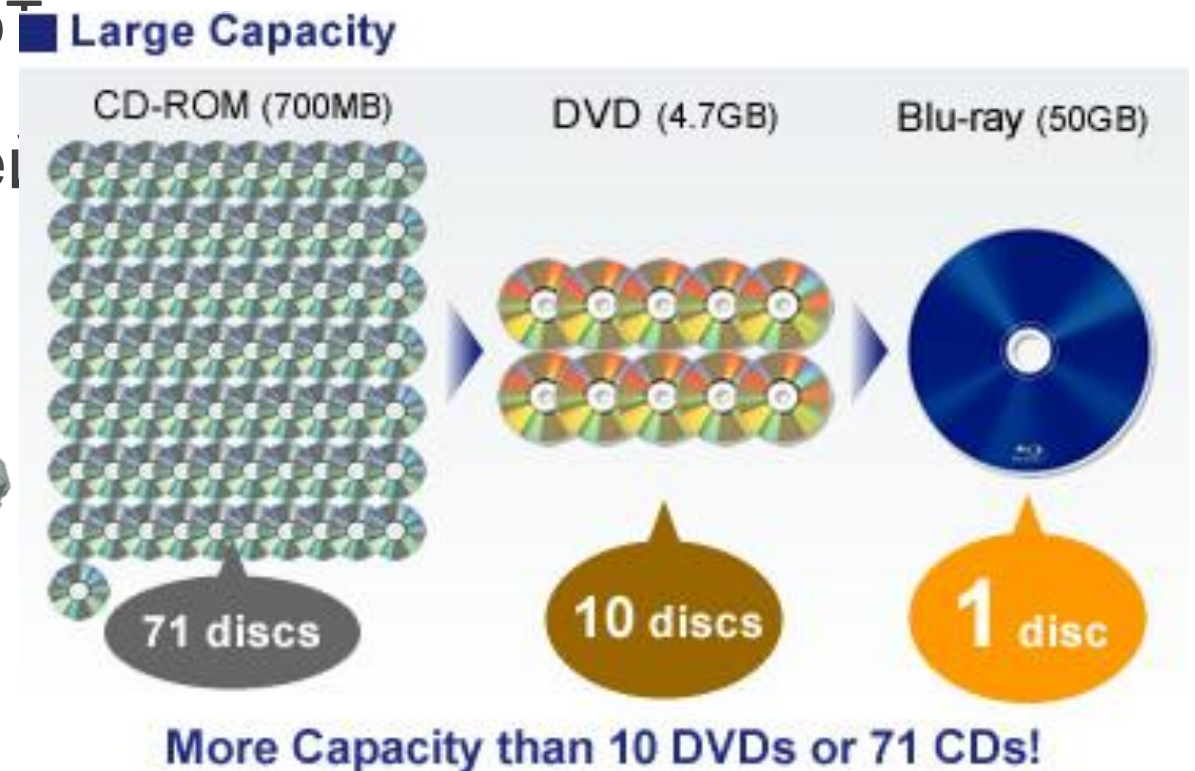
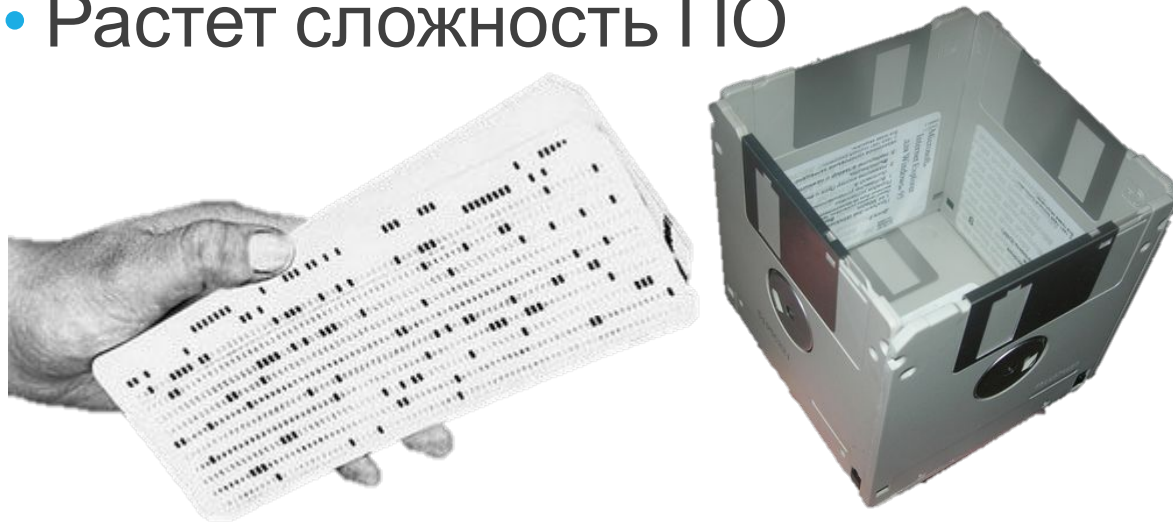
- Растет количество устройств – IoT
- Растет количество пользователей



# Почему тестирование необходимо?

Почему тестирование становится все более важным?

- Растет количество устройств – IoT
- Растет количество пользователей
- Растет сложность ПО



# Почему тестирование необходимо?

Почему тестирование становится все более важным?

- Растет количество устройств – IoT
- Растет количество пользователей
- Растет сложность ПО
- Скорость выхода на рынок является ключевым конкурентным преимуществом сегодня



# Почему тестирование необходимо?

Все: «Мы не хотим  
рисковать!»



# Почему тестирование необходимо?

---

Причины дефектов программного обеспечения. Как возникают ошибки

- Никто **не совершенен!**
- Чем большее **давление** на нас оказывается, тем более мы склонны делать ошибки.
- В ИТ-разработке мы должны соблюдать **временные** сроки и **бюджет**.
- Требования определены нечетко или **плохо документированы**.
- Спецификации данных **не завершены**.
- **ПРЕДПОЛОЖЕНИЯ!**

# Почему тестирование необходимо?

---



Приведите примеры «багов ПО» из жизни

# Какую ошибку дешевле и проще исправить? Почему?

---



# То, что обнаружена как можно ранее.

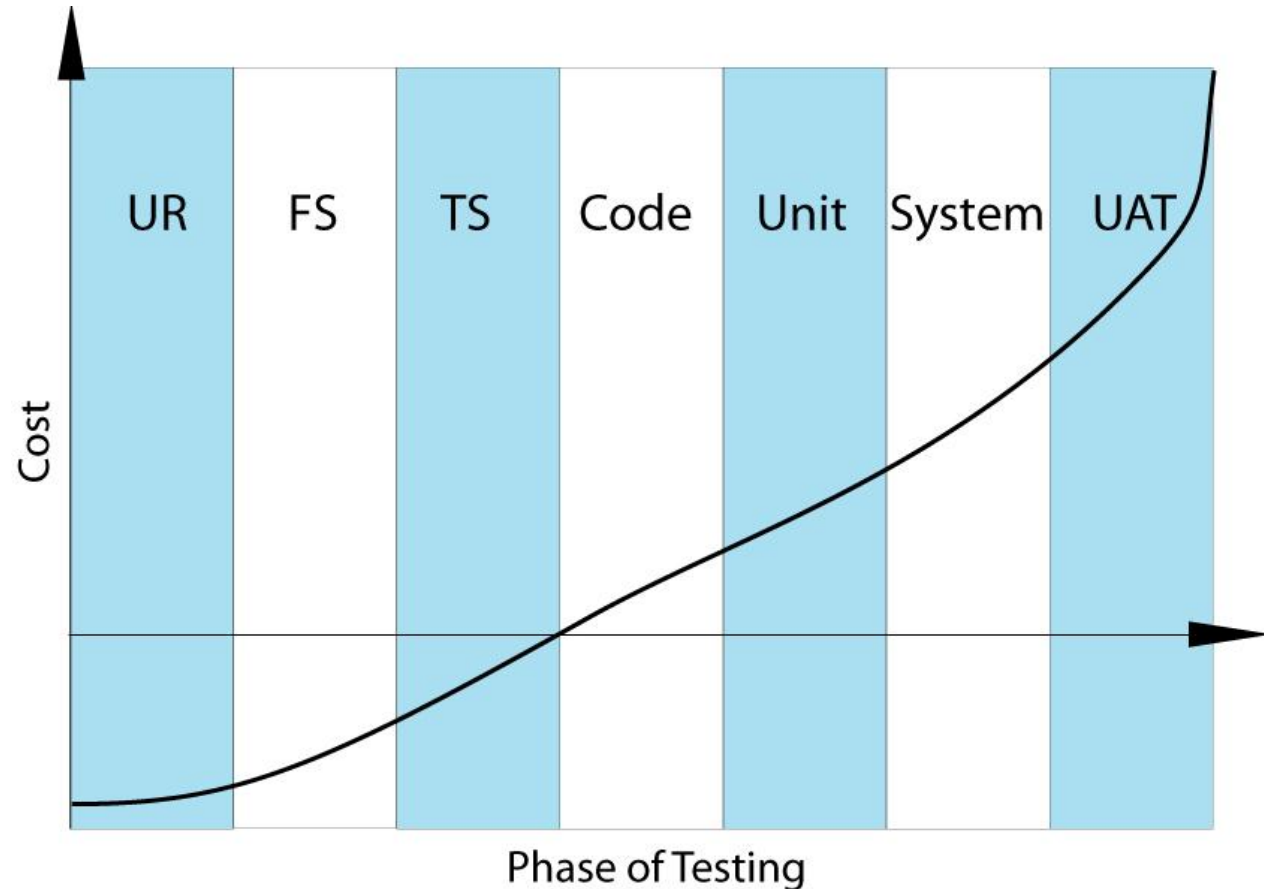
## Модель роста стоимости по SDLC (Software Development Life Cycle)

**UR** = User Requirements  
(Требования пользователей)

**FS** = Functional Specification  
(Функциональная спецификация)

**TS** = Technical Specification  
(Техническая спецификация)

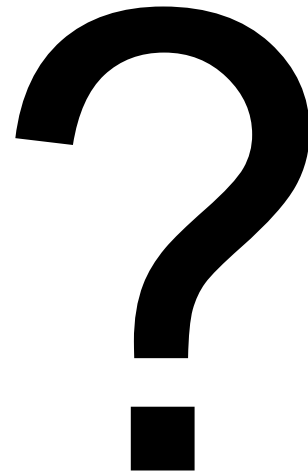
**UAT** = User Acceptance Testing  
(Пользовательское приемочное тестирование)





**Есть вопросы? Давайте обсудим!**

---



# Основы тестирования ПО

Семь принципов тестирования

# Семь принципов тестирования

---

## Принцип 1 – Тестирование демонстрирует наличие дефектов



Тестирование **может показать**, что **дефекты** в программном обеспечении есть, **но не может доказать**, что никаких дефектов нет.

**Тестирование снижает вероятность** того, что в программном обеспечении остались необнаруженные **дефекты**, но, даже если никаких дефектов не обнаружено, это **не доказательство** правильности работы программы.

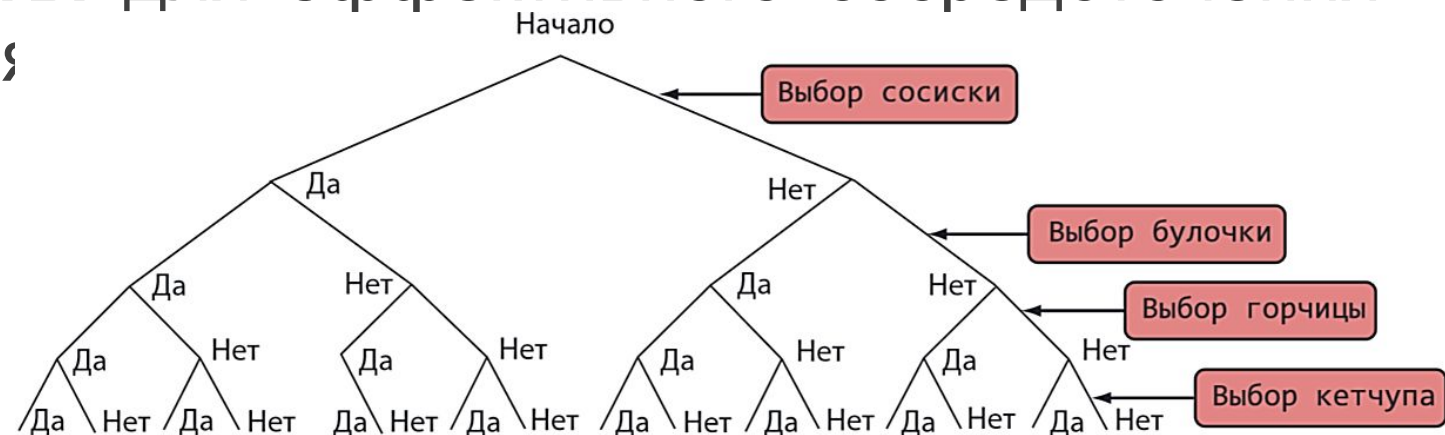
# Семь принципов тестирования

## Принцип 2 – Исчерпывающее тестирование невозможно

**Протестировать** абсолютно **все** (все комбинации входов и условий) **не** представляется **ВОЗМОЖНЫМ**, за исключением тривиальных случаев.



Вместо исчерпывающего тестирования, мы используем **риски и приоритеты** для эффективного сосредоточения усилий тестирования;



# Семь принципов тестирования

---

## Принцип 3 – Раннее тестирование



Тестовые активности должны начинаться **как можно раньше** в цикле разработки программного обеспечения или системы, и должны быть направлены на достижение определенных целей.

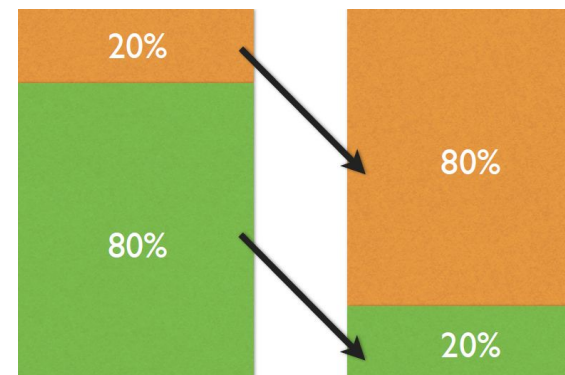
# Семь принципов тестирования

## Принцип 4 – Скопление дефектов



Небольшое количество модулей содержат **большинство дефектов**, выявленных в ходе тестирования, или демонстрируют наибольшее количество операционных сбоев.

Это еще одно проявление правила Парето 80/20 – 80% дефектов находятся в 20% функций.



# Семь принципов тестирования

## Принцип 5 – «**Парадокс пестицида**» (DDT paradox)



Если одни и те же тесты повторяются снова и снова, в конце концов с их помощью вы **перестанете находить дефекты**.



# Чтобы обойти это, тестировщикам необходимо...

---

**Пересмотреть** и обновить существующие тестовые сценарии.

**Добавить** новые и отличные от предыдущих **тесты**, чтобы проверить различные части программного обеспечения.

Изучать **новые инструменты** и **методы** и изобретать новые способы тестирования

Провести **ротацию кадров** таким образом, чтобы один и тот же тестировщик не работал долгое время с одним и тем же разработчиком (функционалом, проектом)



# Семь принципов тестирования

## Принцип 6 – Тестирование зависит от контекста



6

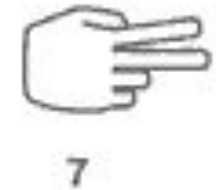
Тестирование **проводится по-разному** в различных контекстах. Контекст включает:

- тип продукта
- его цели
- связанные риски
- доступные инструменты
- ресурсы и время
- опыт команды и т.д.

# Семь принципов тестирования

---

## Принцип 7 – **Заблуждение об отсутствии ошибок**



Нахождение и исправление дефектов **не поможет**, если разработанная система **не удовлетворяет** **нуждам** и ожиданиям пользователей.

# Основы тестирования ПО

Еще Немного терминологии

# Тестирование программного обеспечения

(software testing) – процесс анализа программного средства и сопутствующей документации с целью выявления дефектов и повышения качества продукта.

**выявления дефектов и повышения качества продукта.**



# Немного терминологии

QA ≠ QC ≠ Testing



# Немного терминологии

---

## Верификация vs Валидация



# Немного терминологии

---

**План тестирования** (test plan): документ, описывающий цели, подходы, ресурсы и график запланированных тестовых активностей и определяющий:

- что тестировать;
- что не нужно тестировать;
- кто будет тестировать;
- где это нужно тестировать и на каком оборудовании;
- методы и подходы для проектирования тестов;
- критерии для начала и окончания тестирования, а также любые риски, требующие планирования на случай чрезвычайных обстоятельств.

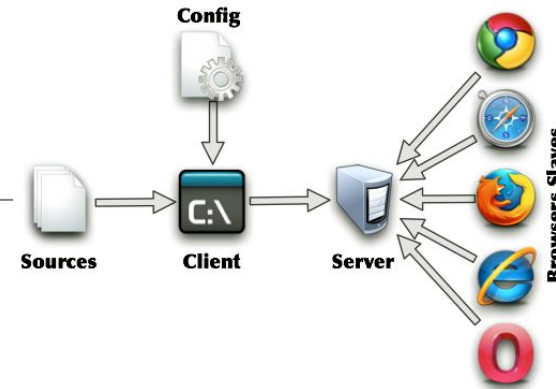
# Чек-лист (check-list) – набор идей тестов.

Почему мы не сразу приступаем к разработке тестов?



Приведите пример чек-листа из Вашей жизни





**Тест-кейс** (test case) – набор входных данных, условий выполнения и ожидаемых результатов, разработанный с целью проверки того или иного свойства или поведения программного средства.

Что случится, если у нас не будет:

- а) входных данных;
- б) условий выполнения;
- в) ожидаемых результатов;
- г) цели.

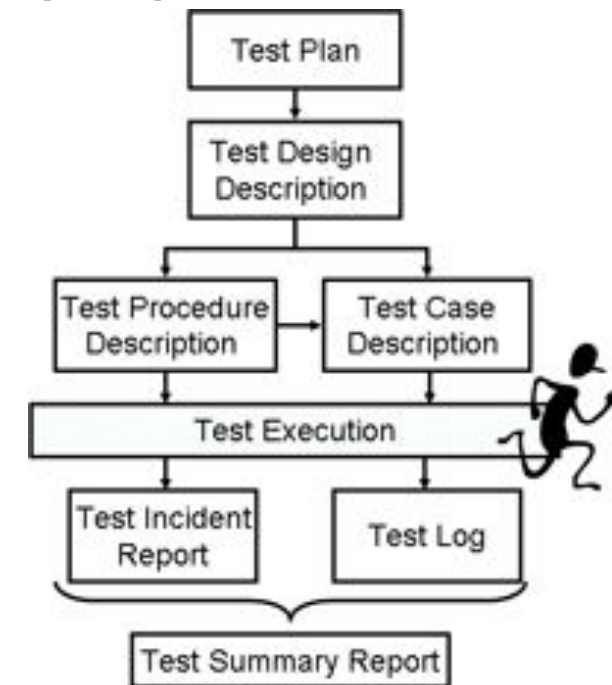
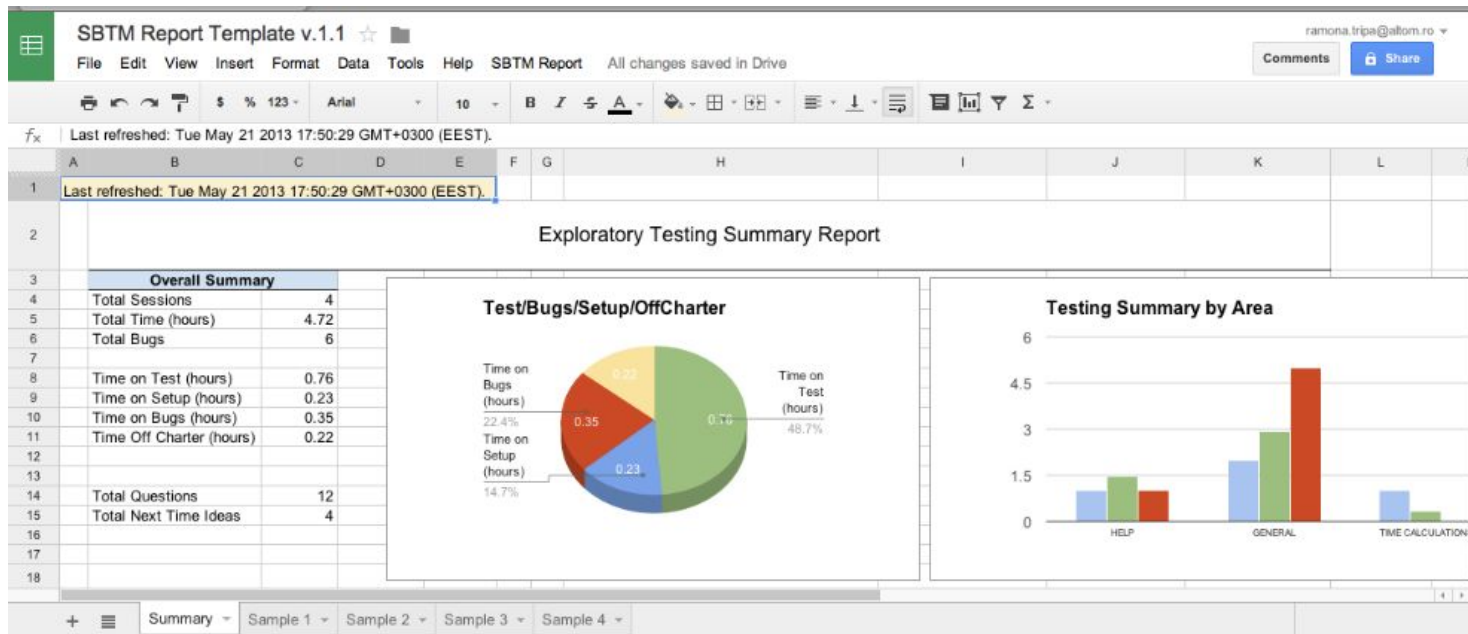
**Тестовый сценарий, тест-сьют**  
(test scenario, test-suite) – набор тест-кейсов, собранных в группу (последовательность) **для достижения некоторой цели.**

И снова: а в жизни бывают примеры тест-сьютов?



# Немного терминологии

**Отчет о тестировании** (test result report, TRR): Документ, подводящий итог проделанной работы в ходе тестирования, а также содержащий оценку состояния качества программы.



Source: [IEEE 829]

Все эти документы (и многие другие) называются

...

---



## Test Artefacts

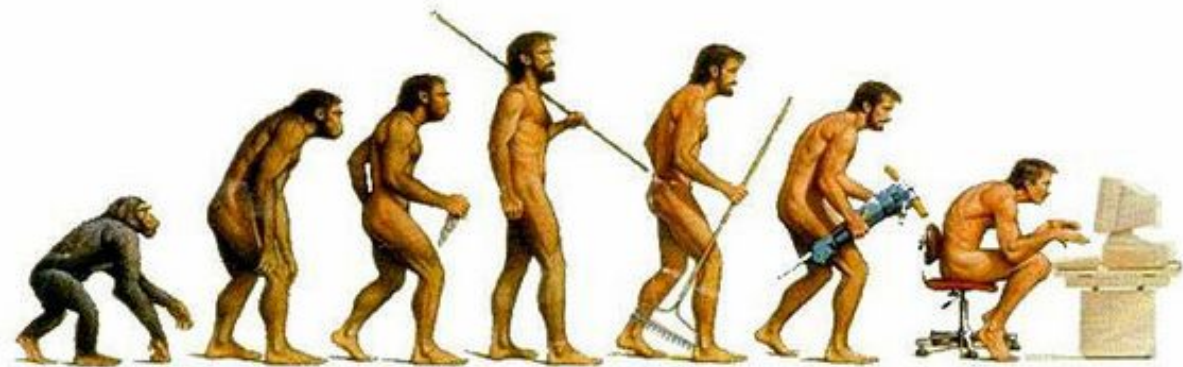
(Project artefacts, проектная документация)

# Почему тестирование необходимо? Немного терминологии

**Билд** («сборка», build) – очередная версия программы.

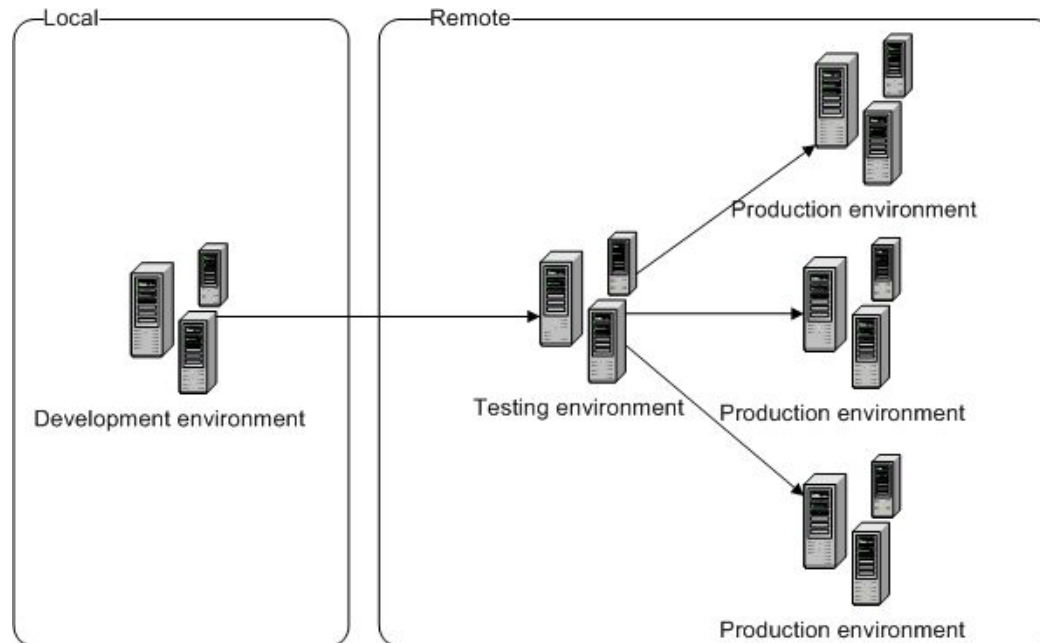
**Финальный билд** – часто называют Релизом (Release) – то, что уходит пользователям/заказчику.

**Ежедневная сборка (daily build)** – действия, в ходе которых система ежедневно (обычно ночью) компилируется и собирается целиком, так что целостная система доступна в любое время, включая все п



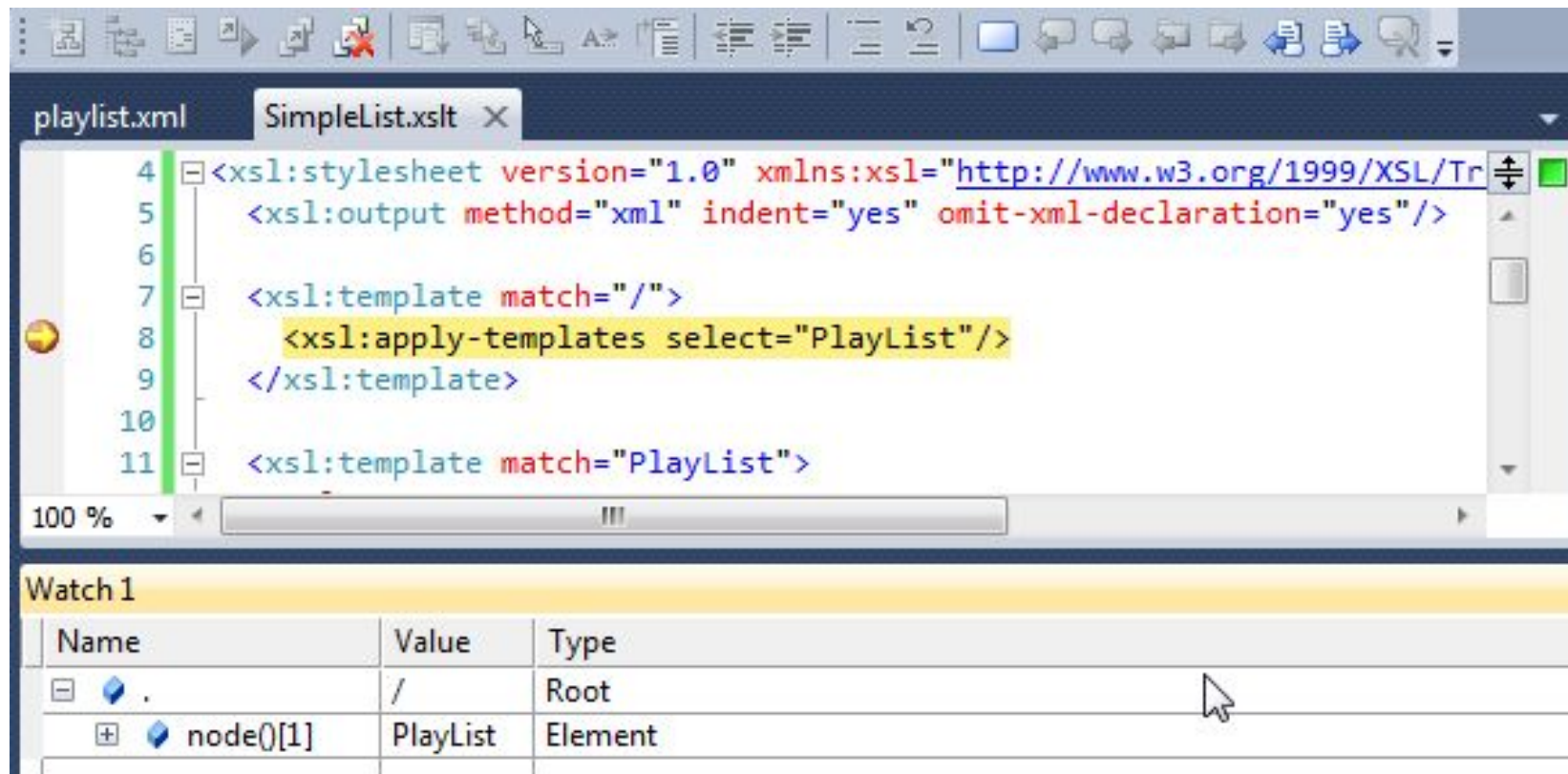
# Немного терминологии

**Тестовое окружение** (test environment): аппаратуру (по сути **компьютер/смартфон** и установленное на нем **ПО**) и инструментарий, необходимые для проведения теста (**которыми пользуется тестировщик**).



# Определение

**Отладка** (Debugging) – Процесс поиска, анализа и устранения причин отказов в программном обеспечении.



The screenshot shows an IDE with two tabs: 'playlist.xml' and 'SimpleList.xslt'. The 'SimpleList.xslt' tab is active, displaying XSLT code. A yellow highlight is on the line `<xsl:apply-templates select="PlayList"/>`. Below the code editor is a 'Watch 1' window with a table:

Name	Value	Type
.	/	Root
node()[1]	PlayList	Element

# Важно не путать!

---

Отладка и тестирование – разные активности!

- Тестирование может **показать сбой**, вызванные дефектами.
- Отладка это процесс разработки ПО, который **выявляет причину дефекта**, исправляет код и проверяет, что ошибка была исправлена корректно.

За каждый вид деятельности разная ответственность:

- **Тестировщики** тестируют.
- **Разработчики** занимаются отладкой.



# Немного терминологии

---

**ДЕФЕКТ** (DEFECT) = НЕДОЧЕТ (FAULT) = ПОМЕХА (**BUG**)  
= ПРОБЛЕМА (PROBLEM) = ISSUE

Изъян в компоненте или системе, который может привести компонент или систему к невозможности выполнить требуемую функцию, например неверный оператор или определение данных.

Дефект, обнаруженный во время выполнения отказам компонента или системы.

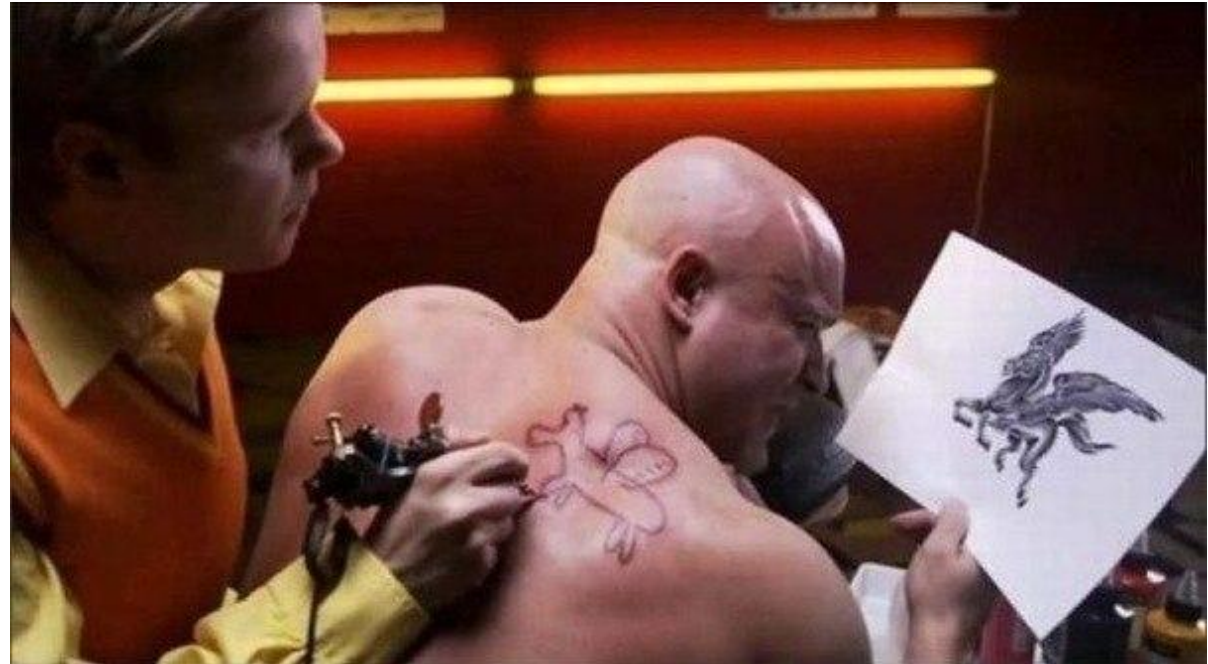


# Немного терминологии

---

## Ожидаемый результат (expected result)

– такое поведение программного средства, которое мы ожидаем в ответ на наши действия.



# Немного терминологии

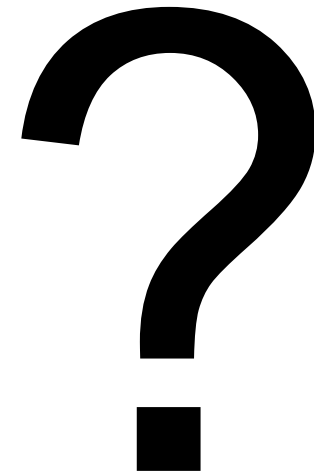
---

**Качество (Quality)** – Степень, с которой компонент, система или процесс соответствует зафиксированным требованиям и/или ожиданиям и нуждам пользователя или заказчика.



# Качество

---



Как мы определяем, что какая-то вещь, какая-то работа и т.д. могут быть названы «качественными», например – обувь?

# Некоторые простые рассуждения о качестве

---

- Если заказчик доволен продуктом – продукт качественный;
- Если продукт соответствует требованиям – продукт качественный;
- У качественного продукта всегда есть преимущества и нет серьёзных недостатков;
- Хорошее качество – низкий риск потерь (денег, времени, репутации...).

**Заказчик должен быть удовлетворен!**

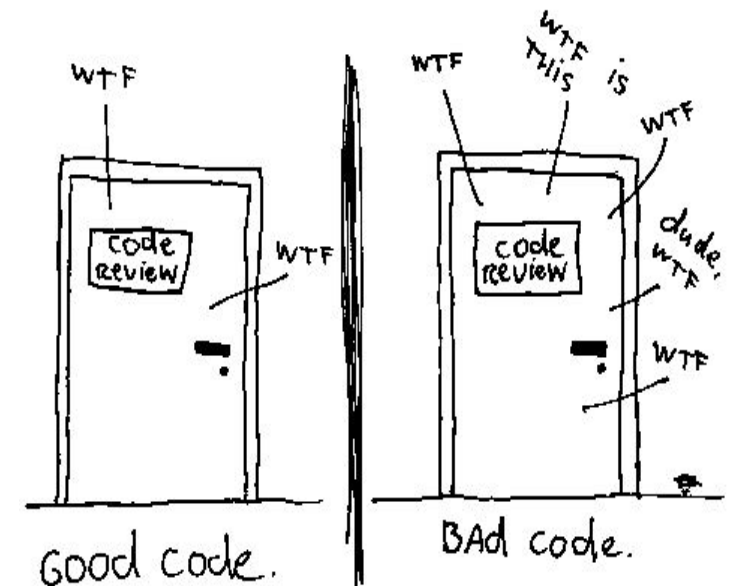
# Немного терминологии

**Метрика** (metric): Шкала измерений и метод, используемый для измерений [ISO 14598]

## Варианты метрик:

- Покрытие требований тестами – не менее 80%
- Плотность покрытия – не менее 3
- Закрито 100% известных критических дефектов, 90% дефектов средней критичности, 50% остальных дефектов.
- Общий показатель прохождения тестов – не менее некоторого значения:  
 $X = (\text{Passed}/\text{Executed}) * 100\%$

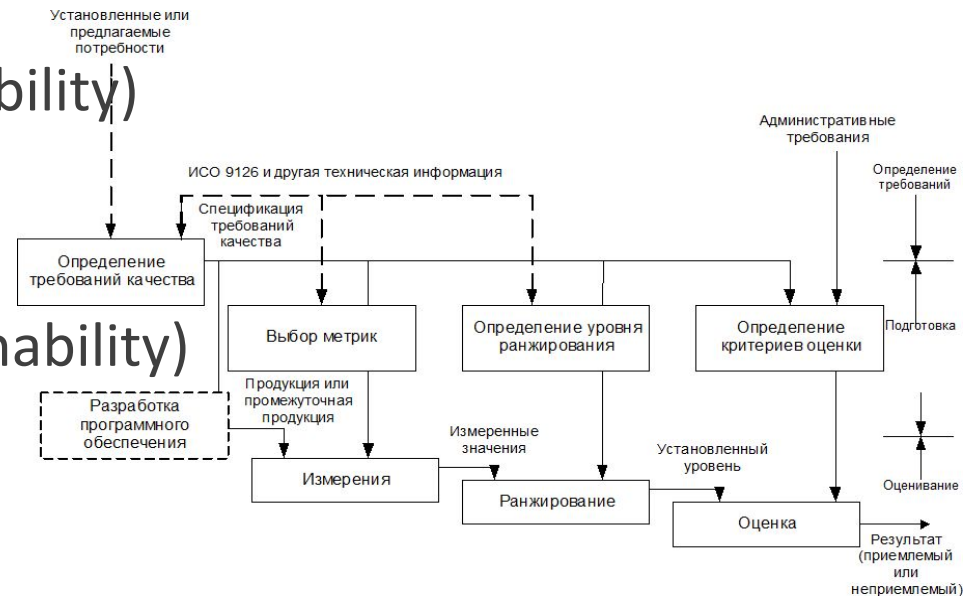
The ONLY VALID MEASUREMENT OF CODE QUALITY: WTFs/MINUTE




# Почему тестирование необходимо? Немного терминологии

На основании ISO/IEC 25010:2011 (пред. ISO/IEC 9126-1:2001) в понятие Качество ПО включено **ВОСЕМЬ** характеристик:

- Функциональная пригодность (Functional suitability)
- Производительность (Performance efficiency)
- Совместимость (Compatibility)
- Удобство использования (Usability)
- Надежность (Reliability)
- Безопасность (Security)
- Ремонтопригодность (Maintainability)
- Переносимость (Portability)



# Рекомендуемые ресурсы:

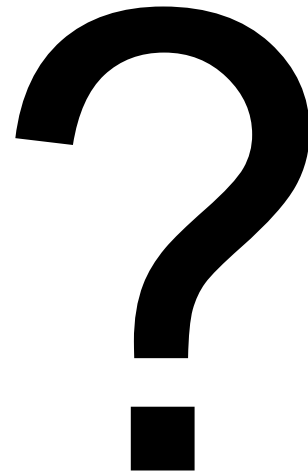
- <https://www.w3schools.com> – множество простой информации по целой серии технологий.
- [http://www.sql-ex.ru/learn\\_exercises.php](http://www.sql-ex.ru/learn_exercises.php) – множество практических заданий по SQL.
- <https://youtu.be/Z-a7MNStFQs> – простой полуторачасовой видеокурс по основам компьютерных сетей.
- <https://htmlacademy.ru> – серия бесплатных курсов по HTML / CSS / JS / PHP.
- <http://linux-user.ru> – ресурс для начинающих пользователей Linux.
- <http://software-testing.ru> – большой портал, на котором есть как профессиональные материалы, так и небольшие статьи, понятные и полезные начинающим.
-  **В ПОМОЩЬ!**





**Есть вопросы? Давайте обсудим!**


---



# Давайте поищем баги!

**Payment Details**  Remember

**CARD NUMBER**

Valid Card Numbe 

MM

**EXPIRY DATA**

YY

**CV CODE**

CV


**Final Payment** \$ €4200

**Day**

# Давайте поищем баги!

**Payment Details**  Remember

**CARD NUMBER**

Valid Card Numbe 

MM

**EXPIRY DATA**

YY

**CV CODE**

CV

Final Payment \$ €4200

Day

**Есть вопросы? Давайте обсудим!**

---



# Когда тестировщики отмечают свой праздник?

Photo # NH 96566-KN (Color) First Computer "Bug", 1947


9/2  
9/9

0800 Anctam started  
1000 " stopped - anctam ✓

13 <sup>00</sup> (032) MP-MC	1.98244000	1.2700	9.037 847 025
(033) PRO 2	2.130476415		9.037 846 995 correct
cond	2.130676415		4.615925059 (-2)

Relays 6-2 in 033 failed special speed test  
in relay " 11.00 test.

Relays changed  
1100 Started Cosine Tape (Sine check)  
1525 Started Multi-Adder Test.

1545  Relay #70 Panel F  
(moth) in relay.

First actual case of bug being found.

1630 Anctam started.  
1700 closed down.

Relay 2145  
Relay 3376

## Небольшое задание на дом:

---

Попытаться разобраться чем отличаются  
понятия:

«тестирование методом **черного ящика**»

и

«тестирования методом **белого ящика**»