# ТЕСТИРОВАНИЕ

## ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Лектор: Сухорукова Ирина Геннадьевна
кафедра программной инженерии
ауд.408 к.1
контакт в телеграмме – у старост

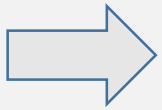
- Лекции – 16
- Лабораторные занятия ~ 12+
- Экзамен

# ТЕСТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

- ✓ Поиск ошибок
- ✓ Проверка соответствия ПО требованиям и здравому смыслу
- ✓ Оценка работоспособности ПО
- ✓ Способ контролировать качество ПО

**Тестирование программного обеспечения** — процесс анализа программного средства и сопутствующей документации с целью выявления дефектов и повышения качества продукта.

В глоссарии **ISTQB*** нет термина «тестирование ПО», который широко используется в русском языке. Там есть лишь термин «тестирование (**testing**)».



**Тестирование** (testing) — процесс, содержащий в себе все активности жизненного цикла, как динамические, так и статические, касающиеся планирования, подготовки и оценки программного продукта и связанных с этим результатов работ с целью определить, что они соответствуют описанным требованиям, показать, что они подходят для заявленных целей и для определения дефектов.

*** ISTQB** — International Software Testing Qualifications Board Glossary (совет по квалификациям в области тестирования программного обеспечения)

# ПРИЧИНЫ ОШИБОК В ПО

- ✓ Человеческий фактор
- ✓ Проблемы в описании требований к программному обеспечению
- ✓ Недостаток времени
- ✓ Недостаточно продуманная архитектура приложения
- ✓ Недостаточное знание бизнеса
- ✓ Нехватка профессиональных навыков и опыта
- ✓ Изменения «в последнюю минуту»

# ИСТОРИЯ ТЕСТИРОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

**50–60-е годы прошлого века**

Первые программные системы разрабатывались в рамках программ научных исследований или программ для нужд министерств обороны.

Тестирование таких продуктов проводилось **строго формализовано** с записью всех тестовых процедур, тестовых данных, полученных результатов.

**Тестирование выделялось в отдельный процесс**, который начинался после завершения кодирования, но при этом, как правило, выполнялось тем же персоналом.

Фактически тестирование **представляло собой** скорее **отладку программ** (*debugging*).

# ИСТОРИЯ ТЕСТИРОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

**В 1960-х** много внимания уделялось **«исчерпывающему» тестированию**, которое должно проводиться с использованием всех путей в коде или всех возможных входных данных.

Однако это невозможно:

- количество возможных входных данных очень велико;
- существует множество путей;
- сложно найти проблемы в архитектуре и спецификациях.

Итог: «**исчерпывающее» тестирование** было отклонено и **признано теоретически невозможным.**

# ИСТОРИЯ ТЕСТИРОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

**В 1970-х годах** фактически родились **две** фундаментальные идеи тестирования:

- ✓ тестирование сначала рассматривалось как **процесс доказательства работоспособности** программы в некоторых заданных условиях (*positive testing*), а затем:
- ✓ **как процесс доказательства неработоспособности** программы в некоторых заданных условиях (*negative testing*)

**Во второй половине 1970-х** тестирование представлялось как выполнение программы с намерением найти ошибки, а не доказать, что она работает.

- ✓ Успешный тест — это тест, который обнаруживает ранее неизвестные проблемы.

# ИСТОРИЯ ТЕСТИРОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

**В 80-х годах** произошло ключевое изменение места тестирования в разработке ПО:

- ✓ вместо одной из финальных стадий создания проекта **тестирование стало применяться на протяжении всего цикла разработки**, что позволило не только быстро обнаруживать и устранять проблемы, но даже предсказывать и предотвращать их появление.

В этот же период времени отмечено бурное **развитие и формализация методологий тестирования** и появление первых элементарных попыток автоматизировать тестирование.

В ходе тестирования надо проверить не только собранную программу, но и требования, код, архитектуру, сами тесты.

# ИСТОРИЯ ТЕСТИРОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

**В 90-х годах** произошёл переход от тестирования как такового к более всеобъемлющему процессу, который называется **«обеспечение качества (quality assurance)»**.

- ✓ В понятие «тестирование» стали включать планирование, проектирование, создание, поддержку и выполнение тест-кейсов и тестовых окружений.
- ✓ Начинают появляться различные программные инструменты для поддержки процесса тестирования: более продвинутые среды для автоматизации с возможностью создания скриптов и генерации отчетов, системы управления тестами, ПО для проведения нагрузочного тестирования.

# ИСТОРИЯ ТЕСТИРОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

**С нулевых годов** появляются гибкие методологии разработки и такие подходы, как «разработка под управлением тестированием (*test-driven development, TDD*)».

- ✓ У тестировщиков появляются специализированные технологии и инструменты, а сам процесс тестирования интегрирован в цикл разработки программного обеспечения.
- ✓ Популярны идеи о том, что во главу процесса тестирования следует ставить не соответствие программы требованиям, а её способность предоставить конечному пользователю возможность эффективно решать свои задачи.

# ТЕСТИРОВАНИЕМ ЗАНИМАЮТСЯ

- ✓ заказчики
- ✓ разработчики
- ✓ тестировщики
- ✓ бизнес аналитики
- ✓ команда технической поддержки продукта
- ✓ дизайнеры
- ✓ конечные пользователи

# Кто такой тестировщик

В ЕКСД* РБ выделены 2 должности :

- ✓ специалист по тестированию программного обеспечения (**QA- инженер)**;
- ✓ тестировщик программного обеспечения (**QA- тестировщик)**.

| | |
|---|---|
| **QA-Тестировщик** — это специалист, который находит ошибки (баги) в работе программного обеспечения (ПО) во время его тестирования, чтобы повысить качество продукта.<br><br>В его обязанности входит:<br>1. Проверяют соответствие ПО заданным требованиям<br>2. Фиксируют ошибки<br>3. Создают тестовые планы. | **QA-инженер** — это специалист, который тестирует и контролирует качество продукта на всех этапах его создания, который дает рекомендации по улучшению продукта и повышению качества разработки. Обязанности:<br>1. Изучают продукт<br>2. Вносят замечания на начальных этапах тестирования<br>3. Определяют, какие нужно провести тесты<br>4. Ставят сроки начала тестирования<br>5. Координируют взаимодействие тестировщиков и разработчиков<br>6. Следят, чтобы обо всех ошибках, которые обнаружились на разных этапах разработки и тестирования, узнали нужные люди<br>7. Контролируют сроки решения проблем<br>8. Ставят приоритеты в работе. |

** ЕКСД – единый классификационный справочник должностей*

*Результатом работы тестировщика является **счастье** (удовлетворение) **конечного пользователя**. Причем "счастье" не в глобальном его значении, а та его часть, которая связана с качеством вашего продукта.*

Роман Савин
Книга: тестирование dot com или Пособие по жестокому обращению с багами в интернет-стартапах

# ТЕСТИРОВАНИЕ

---

# КОНТРОЛЬ И ОБЕСПЕЧЕНИЕ КАЧЕСТВА

**Обеспечение качества (Quality Assurance - QA)** - это совокупность мероприятий, охватывающих все технологические этапы разработки, выпуска и эксплуатации программного обеспечения информационных систем, предпринимаемых на разных стадиях жизненного цикла ПО, для обеспечения требуемого уровня качества выпускаемого продукта.

**Контроль качества (Quality Control - QC)** - это совокупность действий, проводимых над продуктом в процессе разработки, для получения информации о его актуальном состоянии в разрезах: "готовность продукта к выпуску", "соответствие зафиксированным требованиям", "соответствие заявленному уровню качества продукта".

**Тестирование программного обеспечения (Software Testing)** - это одна из техник контроля качества, включающая в себя активности по планированию работ (Test Management), проектированию тестов (Test Design), выполнению тестирования (Test Execution) и анализу полученных результатов (Test Analysis).

# В чем отличия QA от тестирования

- **QA** — это забота о качестве в виде превентирования появления багов
- **Тестирование** — это забота о качестве в виде обнаружения багов до того, как их найдут пользователи.

Пример о воспитании из книги Савина

Общее в QA и тестировании заключается в том, что они призваны улучшить ПО, различие между ними — в том, что:

- QA призвано улучшить ПО через улучшение процесса разработки ПО;
- тестирование — через обнаружение багов.

# КАЧЕСТВО СИСТЕМЫ

Это степень удовлетворения системой заявленных и подразумеваемых потребностей различных заинтересованных сторон, которая позволяет, таким образом, оценить достоинства.

Эти заявленные и подразумеваемые потребности представлены в международных стандартах серии SQuaRE посредством моделей качества, которые представляют качество продукта в виде разбивки на классы характеристик.

Стандарт качества ПО ISO/IEC 25010:2011

НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ГОСТ Р ИСО/МЭК 25010— 2015**

**Информационные технологии**

# СИСТЕМНАЯ И ПРОГРАММНАЯ ИНЖЕНЕРИЯ

## Требования и оценка качества систем и программного обеспечения (SQuaRE). Модели качества систем и программных продуктов

**ISO/IEC 25010:2011**
**Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models**
**(IDT)**

## Содержание


1 Область применения .....
2 Соответствие.....
3 Основы модели качества .....
3.1 Модели качества.....
3.2 Модель качества при использовании.....
3.3 Модель качества продукта.....
3.4 Цели моделей качества.....
3.5 Применение модели качества .....
3.6 Качество с точки зрения различных заинтересованных
3.7 Взаимосвязь моделей .....
4 Термины и определения.....
4.1 Термины к модели качества при использовании .....
4.2 Термины к модели качества продукта.....
4.3 Общие определения.....
4.4 Термины и определения из ИСО/МЭК 25000 .....
Приложение А (справочное) Сравнение с моделью качества
Приложение В (справочное) Пример отображения функциона
Приложение С (справочное) Использование модели качества
Библиография.....

# МОДЕЛИ КАЧЕСТВА

Модели качества представляют качество продукта в виде характеристик.

К настоящему времени в серии SQuaRE имеются три модели качества:

- ✓ Модель качества при использовании
- ✓ Модель качества продукта
- ✓ Модель качества данных, определенная в ИСО/МЭК 25012

Совместное использование моделей качества дает основание считать, что учтены все характеристики качества.

# МОДЕЛЬ КАЧЕСТВА ПРИ ИСПОЛЬЗОВАНИИ

- ✓ Эффективность (результативность)
- ✓ Производительность
- ✓ Удовлетворенность (полноценность, доверие, удовольствие, комфорт)
- ✓ Свобода от риска (смягчение отрицательных последствий):
  - экономического риска;
  - риска для здоровья и безопасности;
  - экологического риска.
- ✓ Покрытие контекста (полнота контекста, гибкость)

# Определение характеристик моделей качества подробно дано в ГОСТ Р ИСО /МЭК 25010—2015

Например:

4.1.1 **эффективность, результативность** (effectiveness): Точность и полнота, с которой пользователи достигают определенных целей (ИСО 9241-11).

4.1.2 **эффективность, производительность** (efficiency): Связь точности и полноты достижения пользователями целей с израсходованными ресурсами (ИСО 9241-11).

Примечание - Соответствующие ресурсы могут включать в себя время выполнения задачи (человеческие ресурсы), материалы или финансовые затраты на использование.

4.1.3 **удовлетворенность** (satisfaction): Способность продукта или системы удовлетворить требованиям пользователя в заданном контексте использования.

4.1.3.1 **полноценность** (usefulness): Степень удовлетворенности пользователя достижением прагматических целей, включая результаты использования и последствия использования.

4.1.3.2 **доверие** (trust): Степень уверенности пользователя или другого заинтересованного лица в том, что продукт или система будут выполнять свои функции так, как это предполагалось.

4.1.3.3 **удовольствие** (pleasure): Степень удовольствия пользователя от удовлетворения персональных требований.

4.1.3.4 **комфорт** (comfort): Степень удовлетворенности пользователя физическим комфортом.

И т.д. …….

# МОДЕЛЬ КАЧЕСТВА ПРОДУКТА

- ✓ Функциональная пригодность:
  - функциональная полнота;
  - функциональная корректность;
  - функциональная целесообразность.
- ✓ Уровень производительности:
  - временные характеристики;
  - использование ресурсов;
  - потенциальные возможности.
- ✓ Совместимость:
  - сосуществование;
  - интероперабельность.
- ✓ Переносимость:
  - адаптируемость;
  - устанавливаемость;
  - взаимозаменяемость.
- ✓ Удобство использования:
  - определимость пригодности;
  - изучаемость;
  - управляемость;
  - защищенность от ошибки пользователя;
  - эстетика пользовательского интерфейса
  - доступность.
- ✓ Надежность:
  - завершенность;
  - готовность;
  - отказоустойчивость;
  - восстанавливаемость.
- ✓ Защищенность:
  - конфиденциальность;
  - неподдельность;
  - отслеживаемость;
  - подлинность.

# Показатели надёжности ПО

На практике различают несколько типов показателей надёжности ПО:

- ✓ количество ошибок перед отладкой и после неё;
- ✓ наработка часов на отказ;
- ✓ интенсивность отказов;
- ✓ вероятность безотказного действия в течение заданного отрезка времени.

Показатели оцениваются как количественные, качественные и порядковые. Наиболее ценную информацию дают количественные. Их получают путем непосредственных наблюдений и обработки результатов испытаний систем.

# Тестирование требований

# Как оценить качество требований?

Требования считаются качественными, если они **отражают потребности заинтересованных сторон и предоставляют достаточно информации для создания программного обеспечения**, которое удовлетворяет целям продукта или проекта.

На любом проекте **некачественные требования приводят к нежелательным последствиям**, таким как:

- Множество раундов разъяснений и уточнений;
- Сложности в планировании;
- Частые переделки;
- Превышение времени и бюджета;
- Сложности в поддержке требований;
- Несчастная команда проекта;
- Недовольный клиент.

# Какие существуют критерии качества?

Существует масса списков критериев качества требований, составленных различными авторами, и "заточенных" под разные методологии разработки. Вот лишь некоторые из них:

**По BABOK:**

Atomic (Атомарное)
Complete (Полное)
Consistent (Непротиворечивое)
Concise (Краткое)
Feasible (Выполнимое)
Unambiguous (Однозначное)
Testable (Тестируемое)
Prioritized (Приоритизированное)
Understandable (Понятное)

**BABOK** – руководство по своду знаний по бизнес-анализу

**По Вигерсу:**

Correct (Верное)
Feasible (Выполнимое)
Necessary (Необходимое)
Prioritized (Приоритизированное)
Unambiguous (Однозначное)
Verifiable (Тестируемое)

**INVEST (Bill Wake):**

Independent (Независимое)
Negotiable (Обсуждаемое)
Valuable (Полезное)
Estimable (Поддающееся оценке)
Small (Компактное)
Testable (Тестируемое)

**Requirements Management Using IBM Rational RequisitePro*:**

Unambiguous (Однозначное)
Testable (verifiable) (Тестируемое)
Clear (Четкое)
Correct (Верное)
Understandable (Понятное)
Feasible (Выполнимое)
Independent (Независимое)
Atomic (Атомарное)
Necessary (Необходимое)
Implementation-free (abstract) (Абстрактное)
Consistent (Согласующееся)
Nonredundant (Не избыточное)
Complete (Полное)

И Х М Н О Г О

* Инструмент IBM Rational RequisitePro предназначен для организации работы аналитиков и автоматизации их деятельности в области управления требованиями.

Самый простой способ оценить требования – это проверить их на соответствие критериям качественного формулирования:

- ✓ **атомарность** – в одном требовании изложено только одно требование;
- ✓ **полнота** – достаточный уровень детализации;
- ✓ **согласованность** – непротиворечивость с другими требованиями и бизнес-правилами;
- ✓ **краткость** – отсутствие лишней информации в описании;
- ✓ **понятность** – представление в терминах, ясных для стейкхолдеров и в соответствии с глоссарием проекта;
- ✓ **однозначность** – возможность проверить, удовлетворяет ли решение исходную потребность;
- ✓ **реализуемость** – осуществимость в рамках проектных ограничений (время и деньги) с приемлемым уровнем риска;
- ✓ **тестируемость** – возможность проверить выполнение в конкретных количественных показателях, например, SLA * 99,99% вместо высокой доступности;
- ✓ **приоритезируемость** – возможность определить относительную важность и ценность.

# Предотвращение неопределенности

Качество требований определяет читатель, а не автор. Бизнес-аналитик может считать, что написанное им требование кристально-ясное, свободно от неоднозначностей и других проблем. Но если у читателя возникают вопросы, требование нуждается в дополнительном совершенствовании.

Одной из причин возникновения неопределенности в требованиях является то, что требования пишутся на **естественном языке** (natural language), в отличии от **формального языка** (formal language), в котором двусмысленность исключена.

**Формальный язык** – это язык, который характеризуется точными правилами построения выражений и их понимания. К формальным языкам относятся формулы, блок-схемы, алгоритмы и т.д.

**Естественный язык** — это язык, который используется в повседневной жизни прежде всего для общения, и имеет целый ряд особенностей, к примеру, слова могут иметь более одного значения, иметь синонимы или быть омонимами, а иногда значения отдельных слов и выражений зависят не только от них самих, но и от контекста, в котором они употребляются.

# Предотвращение неопределенности

Требования, изложенные неясным языком, не поддаются проверке, поэтому нужно избегать двусмысленных и субъективных терминов.

| Неоднозначные термины | Способы улучшения |
|---|---|
| Приемлемый, адекватный | Определите, что понимается под приемлемостью и как система это может оценить |
| И/или | Укажите точно, что имеется в виду — «и» или «или», чтобы не заставлять читателя гадать |
| Между, от X до Y | Укажите, входят ли конечные точки в диапазон |
| Зависит от | Определите природу зависимости. Обеспечивает ли другая система ввод данных в вашу систему, надо ли установить другое ПО до запуска вашей системы и зависит ли ваша система от другой при выполнении определенных расчетов или служб? |

| Неоднозначные термины | Способы улучшения |
| --- | --- |
| Эффективный | Определите, насколько эффективно система использует ресурсы, насколько быстро она выполняет определенные операции и как быстро пользователи с ее помощью могут выполнять определенные задачи |
| Быстрый, скорый, моментальный | Укажите минимальное приемлемое время, за которое система выполняет определенное действие |
| Гибкий, универсальный | Опишите способы адаптации системы в ответ на изменения условий работы, платформ или бизнес-потребностей |
| Улучшенный, лучший, более быстрый, превосходный, более качественный | Определите количественно, насколько лучше или быстрее должны стать показатели в определенной функциональной области или аспект качества |

| Неоднозначные термины | Способы улучшения |
|---|---|
| Устойчивый к сбоям | Определите, как система должны обрабатывать исключения и реагировать на неожиданные условия работы |
| Цельный, прозрачный, корректный | Что означает «цельный» или «корректный» для пользователя? Выразите ожидания пользователя, применяя характеристики продукта, которые можно наблюдать |
| Несколько, некоторые, много, немного, множественный | Укажите сколько или задайте минимальную и максимальную границы диапазона |
| Не следует | Старайтесь формулировать требования в позитивной форме, описывая, что именно система будет делать |
| Современный | Поясните этот термин для заинтересованного лица |
| Достаточный | Укажите, какая степень чего-либо свидетельствует о достаточности |
| Поддерживает, позволяет | Дайте точное определение, из каких действий системы состоит «выполнение» конкретной возможности |
| Дружественный, простой, легкий | Опишите системные характеристики, которые будут отвечать потребностям пользователей и его ожиданиям, касающимся легкости и простоты использования продукта |

## Пограничные значения

*Отпуск длительностью до 5 дней не требует одобрения. Запросы на отпуск длительностью от 5 до 10 дней требуют одобрения непосредственного начальника. Отпуска длительностью 10 дней и более требуют одобрения директора.*

Лучше: *Отпуск длительностью 5 дней и меньше не требует одобрения. Запросы на отпуск длительностью более 5 и до 10 дней включительно требуют одобрения непосредственного начальника. Отпуска длительностью свыше 10 дней требуют одобрения директора.*

## Двойное отрицание

*Если пользователь не выбрал ни одного продукта, тогда система не позволяет сделать заказ.*

Лучше:

*Если пользователь выделил хотя бы один продукт, тогда система позволяет сделать заказ*

# Качества хороших требований

- ✓ Работоспособность сценариев
- ✓ Полнота описания
- ✓ Внимание обязательным пунктам
- ✓ Структурированность
- ✓ Наличие указаний на необратимость действий
- ✓ Указание на ожидаемый результат
- ✓ Описание последствий отсутствия действий пользователя
- ✓ Ясность изложения
- ✓ Логика и согласованность
- ✓ Последовательность изложения
- ✓ Орфография, синтаксис, пунктуация
- ✓ Описание настроек по умолчанию

# Оформление вопросов к требованиям

- ✓ Вопросы пишите короткими и простыми;
- ✓ Вопрос не должен содержать «или»;
- ✓ Формулируйте вопрос так, чтобы ответ на него был максимально коротким;
- ✓ Обдумывайте ответ, который можно получить;
- ✓ Предлагая улучшения, подкрепляйте их фактами и весомыми доводами;

**Пример требования:** Приложение должно быстро запускаться.

## «Плохие» вопросы

«Насколько быстро?»
*(Рискуете получить ответы в стиле «очень быстро», «максимально быстро»).*

«А если не получится быстро?»
*(Этим вы рискуете удивить или разозлить заказчика.)*

«Всегда?»
*(«Да, всегда». А вы ожидали другого ответа?)*

## «Хорошие» вопросы

«Каково максимально допустимое время запуска приложения?»

«Допускается ли фоновая загрузка отдельных компонентов приложения?»

«Что является критерием того, что приложение закончило запуск?»

**Пример требования:** Если дата события не указана, она выбирается автоматически.

## «Плохие» вопросы

«А если дату невозможно выбрать автоматически?»

*(Сам вопрос интересен, но без пояснения причин невозможности звучит как издёвка.)*

«А если у события нет даты?»

*(Автор вопроса, скорее всего, хотел уточнить, обязательно ли это поле. Но из самого требования видно, что обязательно.)*

## «Хорошие» вопросы

«Возможно, имелось в виду, что дата генерируется автоматически, а не выбирается?

Если «Да», то по какому алгоритму она генерируется?

Если «Нет», то из какого набора выбирается дата и как генерируется этот набор?»

# Ошибки при анализе требований

*Более полный список см. в книге С.Куликова «Тестирование ПО»*

. . .

**Критика текста или даже его автора**. Помните, что ваша задача — сделать требования лучше, а не показать их недостатки (или недостатки автора). Потому комментарии вида «плохое требование», «неужели вы не понимаете, как глупо это звучит», «надо переформулировать» неуместны и недопустимы.

**Категоричные заявления без обоснования**.
Заявления наподобие «это невозможно», «мы не будем этого делать», «это не нужно». Даже если вы понимаете, что требование бессмысленно или невыполнимо, эту мысль стоит сформулировать в корректной форме и дополнить вопросами, позволяющими автору документа самому принять окончательное решение. Например, «это не нужно» можно переформулировать так: «Мы сомневаемся в том, что данная функция будет востребована пользователями. Какова важность этого требования? Уверены ли вы в его необходимости?»

. . .

# Принципы тестирования

## Тестирование показывает наличие дефектов, а не их отсутствие

Очень сложно обнаружить нечто, относительно чего мы не знаем — ни «где оно», ни «как оно выглядит», ни даже «существует ли оно вообще». Так как не существует физической возможности проверить поведение сложного программного продукта во всех возможных ситуациях и условиях, тестирование не может гарантировать, что в той или иной ситуации, при стечении тех или иных обстоятельств дефект не возникнет. Тестирование может проверить наиболее вероятные, востребованные ситуации и обнаружить дефекты при их возникновении. Такие дефекты будут устранены, что ощутимо повысит качество продукта, но по-прежнему не гарантирует от возникновения проблем в оставшихся, не проверенных ситуациях и условиях

## Исчерпывающее тестирование невозможно

Даже для одного простого поля для ввода имени пользователя может существовать порядка $2.4^{32}$ позитивных проверок и бесконечное количество негативных проверок. Потому невозможно протестировать программный продукт полностью, «исчерпывающе».
Однако, из этого не следует, что тестирование как таковое не является эффективным. Вдумчивый анализ требований, учёт рисков, расстановка приоритетов, анализ предметной области, моделирование, работа с конечными пользователями, применение специальных техник тестирования — эти и многие другие подходы позволяют выявить те области или условия эксплуатации продукта, которые требуют особенно тщательной проверки.

# Принципы тестирования

## Тестирование тем эффективнее, чем раньше оно выполняется

Раннее тестирование помогает устранить или сократить дорогостоящие изменения. У данного принципа есть прекрасная аналогия из обычной повседневной жизни. Представьте, что вы собираетесь в поездку и продумываете список вещей, которые необходимо взять с собой. На стадии обдумывания добавить, изменить, удалить любой пункт в этом списке не стоит ничего. На стадии поездки по магазинам для закупки необходимого недоработки в списке уже могут привести к необходимости повторной поездки в магазин. На стадии отправки на место назначения недоработки в списке вещей явно приведут к ощутимой потере нервов, времени и денег. А если фатальный недостаток списка вещей выяснится только по прибытии, может так оказаться, что вся поездка потеряла смысл.

## Кластеризация дефектов

Дефекты как правило группируются в какой-то «проблемной» области приложения.
Группировка дефектов по какому-то явному признаку является хорошим поводом к продолжению исследования данной области программного продукта: скорее всего, именно здесь будет обнаружено ещё больше дефектов. Обнаружение подобных тенденций к кластеризации (и особенно поиск глобальной первопричины) часто требует от тестировщиков определённых знаний и опыта, но если такой «кластер» выявлен — это позволяет ощутимо минимизировать усилия и при этом существенно повысить качество приложения.

# Принципы тестирования

## Парадокс пестицида

Название данного принципа происходит от общеизвестного явления в сельском хозяйстве: если долго распылять один и тот же пестицид на посевы, у насекомых вскоре вырабатывается иммунитет, что делает пестицид неэффективным.
Проявляется в повторении одних и тех же (или просто однотипных) проверок снова и снова: со временем эти проверки перестанут обнаруживать новые дефекты. Чтобы преодолеть парадокс пестицида, необходимо регулярно пересматривать и обновлять тест-кейсы, разнообразить подходы к тестированию, применять различные техники тестирования, смотреть на ситуацию «свежим взглядом».

## Тестирование зависит от контекста

Набор характеристик программного продукта влияет на глубину тестирования, используемый набор техник и инструментов, принципы организации работы тестировщиков и т.д.

## Отсутствие дефектов — не самоцель

Программный продукт должен не только быть избавлен от дефектов настолько, насколько это возможно, но и удовлетворять требованиям заказчика и конечных пользователей.
Именно понимание контекста продукта и потребностей пользователей позволяет тестировщикам выбрать наилучшую стратегию и добиться наилучшего результата.