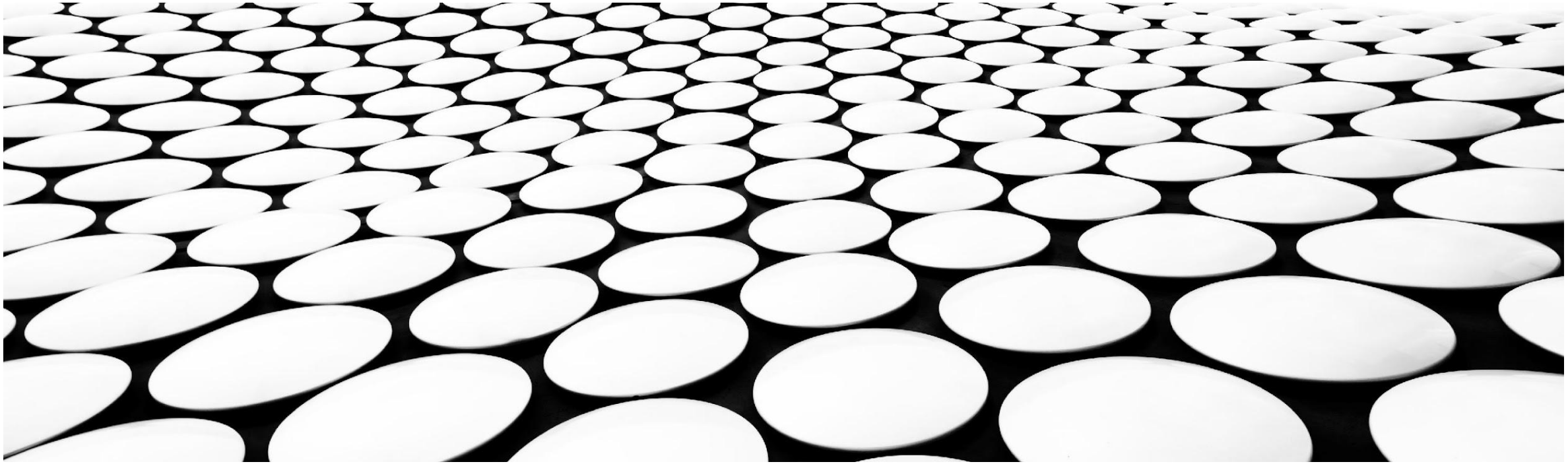
ТЕСТИРОВАНИЕ

EVGENIY SHVETSOV

2021

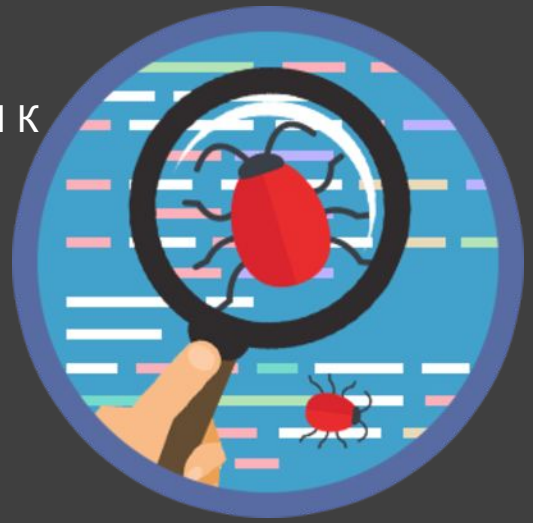


ТЕСТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

ПРОЦЕСС АНАЛИЗА ПРОГРАММНОГО СРЕДСТВА И СОПУТСТВУЮЩЕЙ ДОКУМЕНТАЦИИ С ЦЕЛЬЮ ВЫЯВЛЕНИЯ ДЕФЕКТОВ И ПОВЫШЕНИЯ КАЧЕСТВА ПРОДУКТА

КАЧЕСТВО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

ЭТО СОВОКУПНОСТЬ ХАРАКТЕРИСТИК ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ, ОТНОСЯЩИХСЯ К ЕГО СПОСОБНОСТИ УДОВЛЕТВОРЯТЬ УСТАНОВЛЕННЫЕ И ПРЕДПОЛАГАЕМЫЕ ПОТРЕБНОСТИ



ЭТАПЫ ТЕСТИРОВАНИЯ

- **TEST MANAGEMENT**
  - *АНАЛИЗ ПРОДУКТА*
  - *ВЫЯСНЕНИЕ ТРЕБОВАНИЙ*
- **TEST DESIGN**
  - *ВЫРАБОТКА СТРАТЕГИИ*
  - *ПЛАНИРОВАНИЕ ЭТАПОВ*
  - *ДОКУМЕНТИРОВАНИЕ*
- **TEST EXECUTION**
  - *РАЗРАБОТКА МОДЕЛИ ТЕСТИРОВАНИЯ*
  - *ВЫПОЛНЕНИЕ ТЕСТОРИВАНИЯ*
- **TEST ANALYSIS**
  - *АНАЛИЗ РЕЗУЛЬТАТОВ*
  - *ОБРАТНАЯ СВЯЗЬ*

Оценка качества

Верификация

Валидация

Оценка качества

Процесс оценки того, насколько система по итогам некоторого этапа ее разработки соответствует условиям, заданным в начале этапа

Процесс оценки того, насколько система соответствует требованиям по ее назначению

Приложение разрабатывается правильно

Разрабатывается правильное приложение

ДЕФЕКТ

Дефект *(баг)* – это ситуация, при которой в ходе тестирования выясняется, что фактический результат отличается от ожидаемого результата.

Errors
- Ошибки использования приложения пользователем

Bugs
- Ошибки разработчиков

Failures
- Сбои в работе приложения

It has been just so in all of my inventions. The first step is an intuition, and comes with a burst, then difficulties arise — this thing gives out and it is then that «Bugs» — as such little faults and difficulties are called — show themselves and months of intense watching, study and labor are requisite before commercial success or failure is certainly reached.

Thomas Alva Edison, 1878

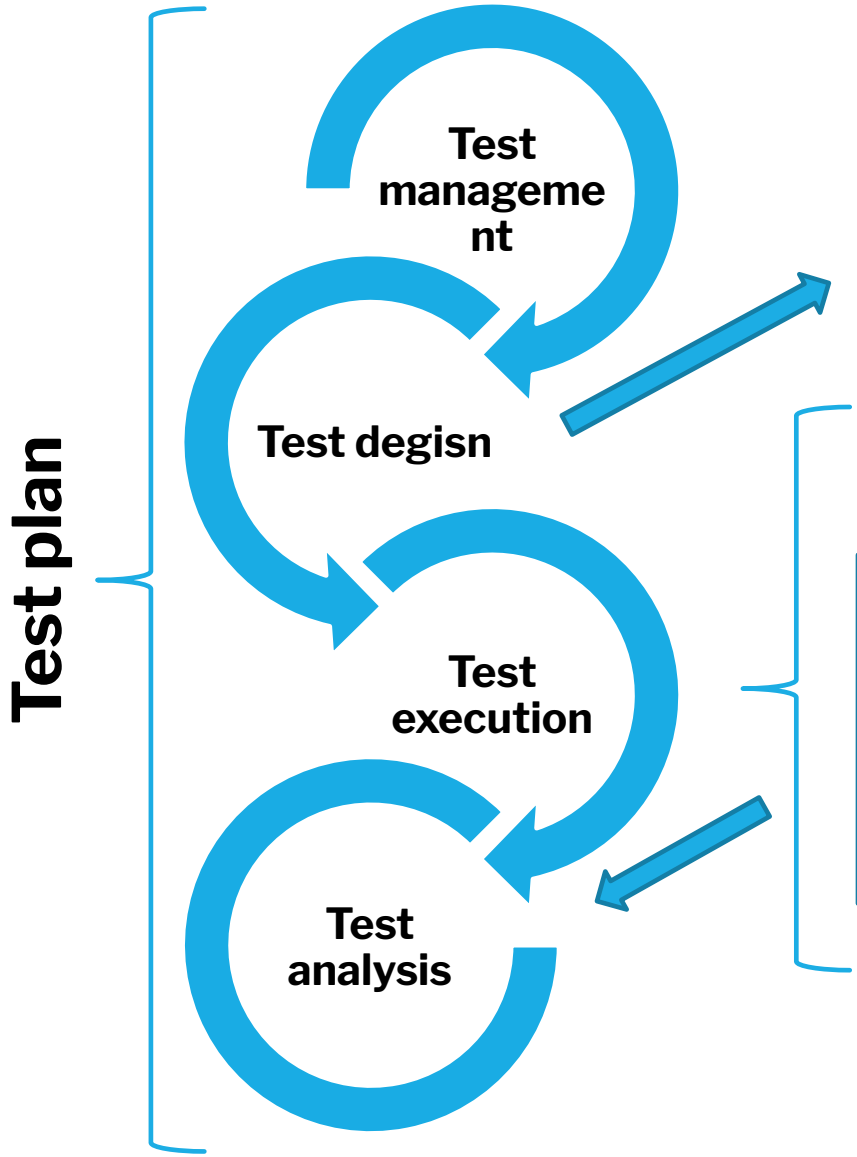
Разновидности багов:

- **Борбаг** (стабильная ошибка) – стабильная легкообнаруживаемая ошибка;
- **Гейзенбаг** (плавающая ошибка, глюк) – периодически исчезающая или меняющая свои свойства ошибка;
- **Мандельбаг** – ошибка со сложным и хаотическим поведением;
- **Шрёдинбаг** – ошибка, проявляющаяся только после ее обнаружения, приводящая к краху системы;
- **Гиндельбаг** – ошибка, приводящая к полному краху системы, часто без возможности восстановления;
- **Багсон Хиггса** – предсказанная математически либо по косвенным признакам ошибка, воспроизвести которую в реальной системе практически невозможно;

* редкоиспользуемая классификация в русскоязычной литературе

Первое упоминание бага в контексте программного обеспечения относится к записи Грейс Хоппер в техническом журнале Harvard Mark II от 9 сентября 1947 – моль, застрявшая между контактами реле, и вызвавшая нетипичное поведение вычислительной машины.

Артефакты тестирования



Actors

Test analytic

- What to test?

Test designer

- How to test?

Bug report

- Описание некорректной ситуации, фактический результат
- Шаги воспроизведения или причины
- Ожидаемый результат

Test artifacts

Traceability matrix

- Матрица соответствия требований и тест кейсов

Test cases

- Набор сценариев, описывающих тестирования сущности

Check list

- Спикок проверяемых сущностей

Defects

Errors

- Ошибки использования приложения пользователем

Bugs

- Ошибки разработчиков

Failures

- Сбои в работе приложения

ГРАДАЦИЯ ДЕФЕКТОВ

Серьезность (severity) – влияние дефекта на работоспособность приложения

- **Blocker** – блокирующая (S1)
- **Critical** – критическая (S2)
- **Major** – значительная (S3)
- **Minor** – незначительная (S4)
- **Trivial** – тривиальная (S5)

Приоритет (priority) – очередность устранения дефекта

- **High** – высокий (P1)
- **Medium** – средний (P2)
- **Low** – низкий (P3)

Определяется тестировщиком

Может влиять на выбор категории

Определяется бизнес или тех. менеджером

Defects

Errors

- Ошибки использования приложения пользователем

Bugs

- Ошибки разработчиков

Failures

- Сбои в работе приложения

ПИРАМИДА ТЕСТИРОВАНИЯ

Unit-тесты

- Проверяют поведение одной функции или метода

Компонентные тесты

- Проверяют поведение низкоуровнего компонента (объект, библиотека и т.д.)

Интеграционные тесты

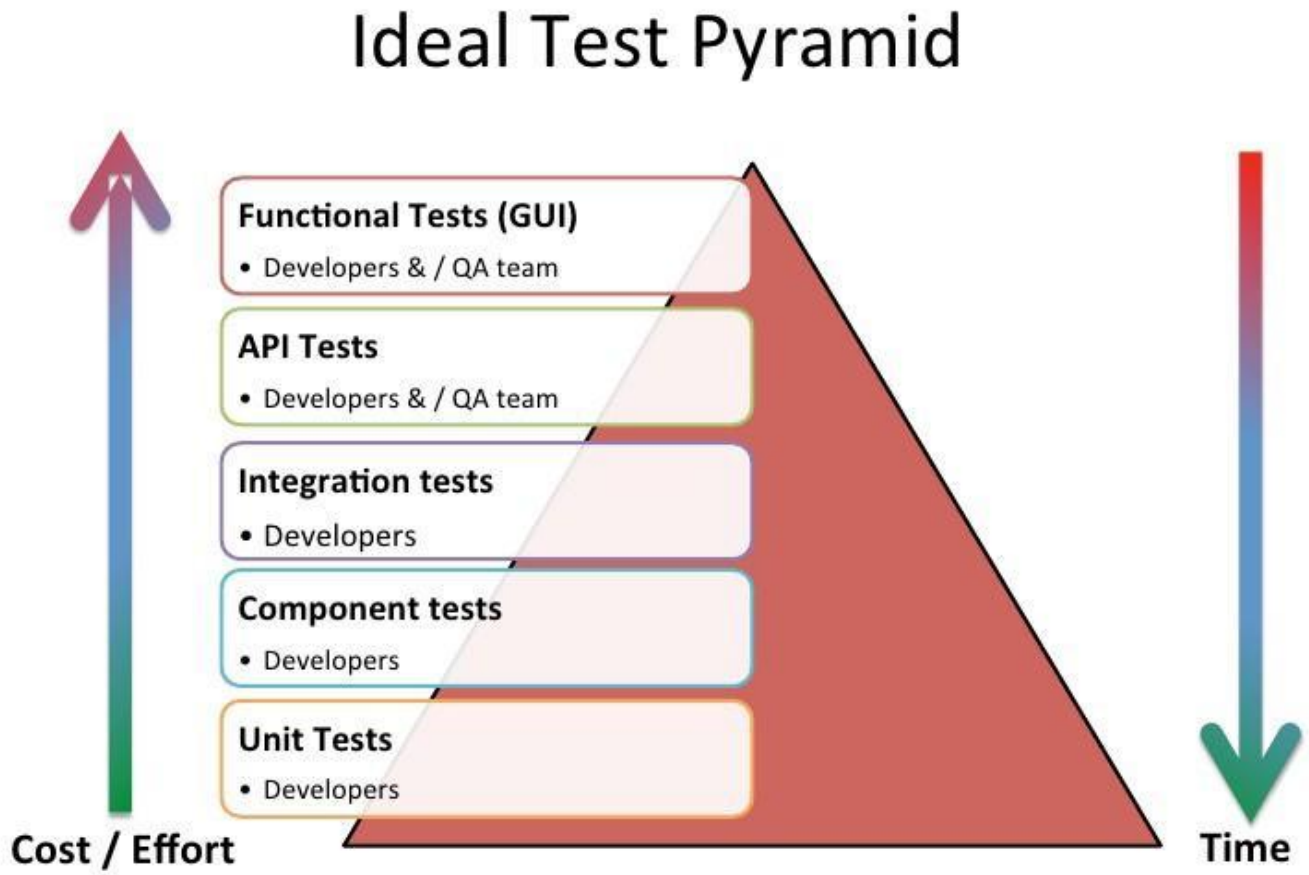
- Проверяют взаимодействие между компонентами

Сервисные тесты

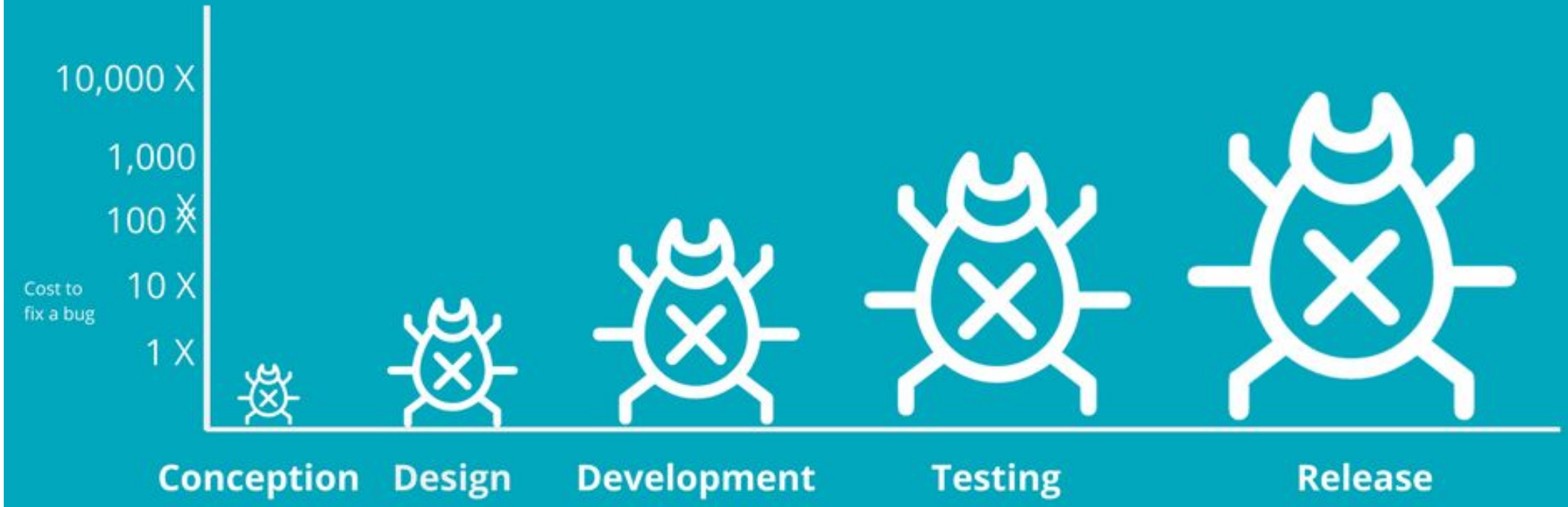
- Проверяют доступность веб-сервисов

Функциональные тесты

- Проверяют реакцию системы на действия пользователей



Resolving bugs early and often reduces associated costs

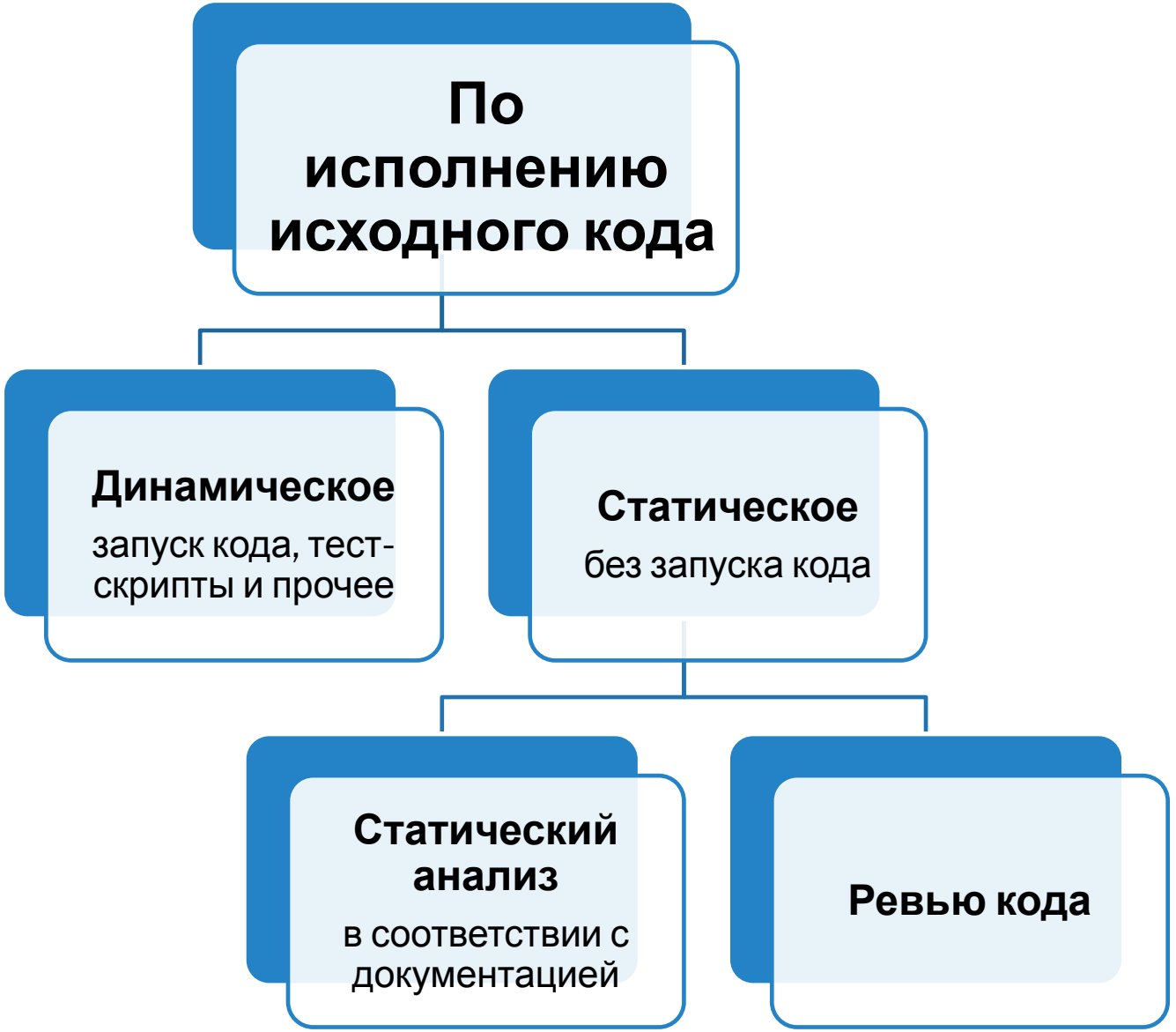


RAYGUN

ТЕСТИРОВАНИЕ ПО ЦЕЛЯМ

ТЕСТИРОВАНИЕ ПО ЗНАНИЮ СИСТЕМЫ

ТЕСТИРОВАНИЕ ПО ИСПОЛНЕНИЮ ИСХОДНОГО КОДА



ТЕСТИРОВАНИЕ ПО ПОЗИТИВНОСТИ СЦЕНАРИЯ

ТЕСТИРОВАНИЕ ПО СТЕПЕНИ АВТОМАТИЗАЦИИ

По степени автоматизации

Ручное
без инструментов автоматизации

Достоинства:
- пользовательский фитбек;
- дешевизна в краткосрочной перспективе;
- тестирование в реальном времени;
- возможность применения гибкого исследовательского тестирования;

Недостатки:
- человеческий фактор;
- трудоемкость итераций;
- сложность или невозможность нагрузочного тестирования;

Полуавтоматизированное
автоматизация применяется только для определенных целей, является слиянием ручного и автоматизированного тестирования

Автоматизированное
средства автоматизации применяются на всех уровнях тестирования

Достоинства:
- отсутствие человеческого фактора;
- скорость выполнения;
- автоматическое формирование отчетов;
- фоновое выполнение;
- проведение нагрузочного тестирования;

Недостатки:
- требует высокой квалификации;
- высокая стоимость;
- зависимость от изменений в системе

УРОВНИ ТЕСТИРОВАНИЯ ПО ПРОЦЕССУ ДОСТАВКИ АРТЕФАКТОВ

Unit testing

- Проверяется функциональность модулей и компонентов

Интерграционное тестирование

- Проверяется взаимодействие между компонентами

Системное тестирование

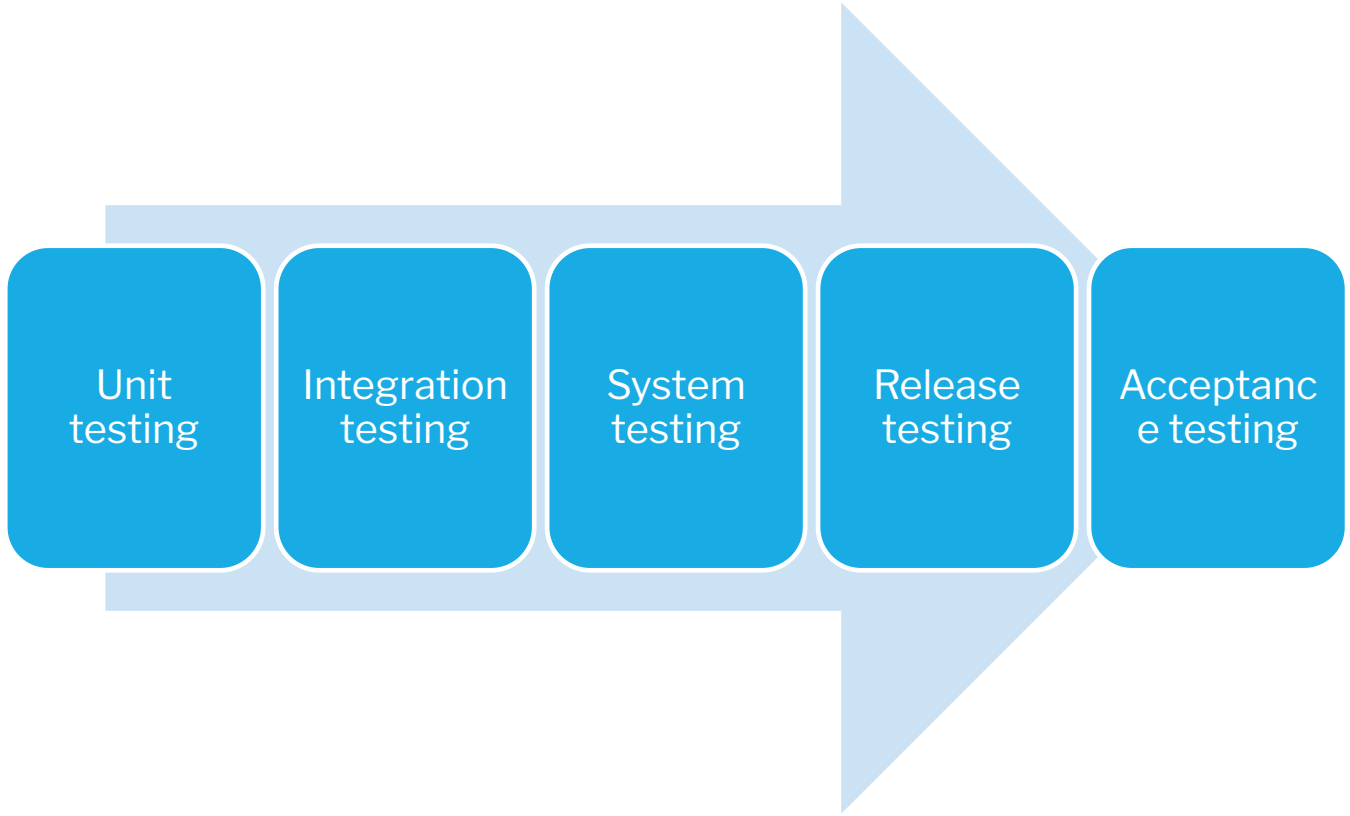
- Проверяется функциональность, а также аспекты окружения и ресурсов

Операционное тестирование

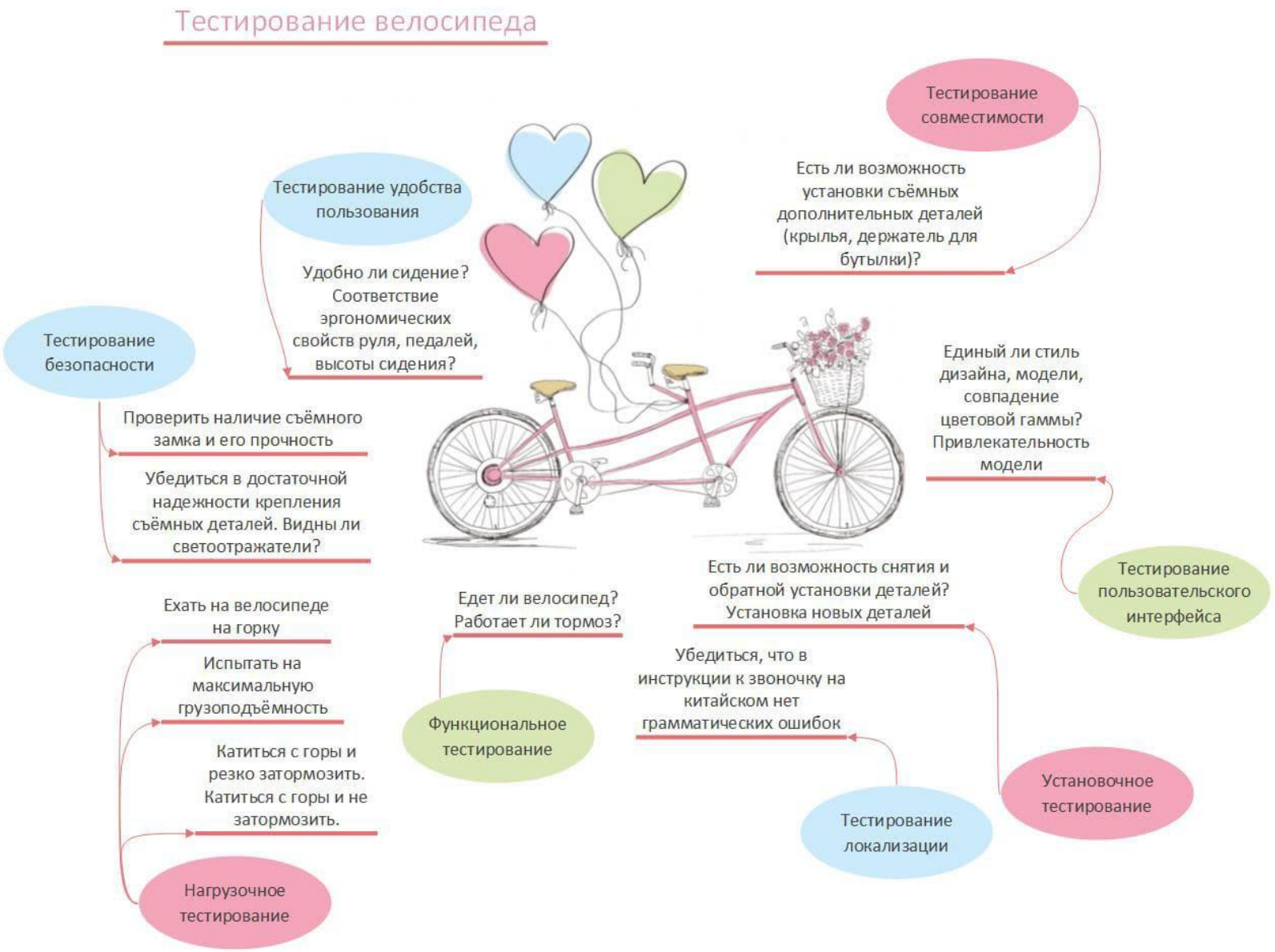
- Проверяет соответствие бизнес-модели, а также работу в среде приемки

Приемочное тестирование

- Формальная проверка соответствия системы требованиям



ПРИМЕР ТЕСТИРОВАНИЯ ВЕЛОСИПЕДА



ЖИЗНЕННЫЙ ЦИКЛ ДЕФЕКТА

МЕТОДЫ ТЕСТИРОВАНИЯ

Динамическое тестирование – метод, направленный на проверку функционала программы с помощью запуска кода.

Статическое тестирование – метод тестирования без запуска кода приложения.

Исследовательское тестирование – тестирование без использования спецификаций и сценариев. Каждый следующий тест базируется на результате предыдущего.

СРАВНЕНИЕ МЕТОДОВ ТЕСТИРОВАНИЯ

Динамическое тестирование

Плюсы

- Тщательность изучения
- Структурированность
- Поиск сложных дефектов
- Может быть автоматизировано

Минусы

- Сложность механизма
- Дорогостоящий процесс
- Обычно следует после процесса разработки

Статическое тестирование

Плюсы

- Происходит на ранних этапах разработки
- Может привести у улучшению функционала
- Высокоуровневый анализ приложения
- Невысокая стоимость

Минусы

- Трудно автоматизировать
- Длительный процесс
- Зависит от компетенции инженера

Исследовательское тестирование

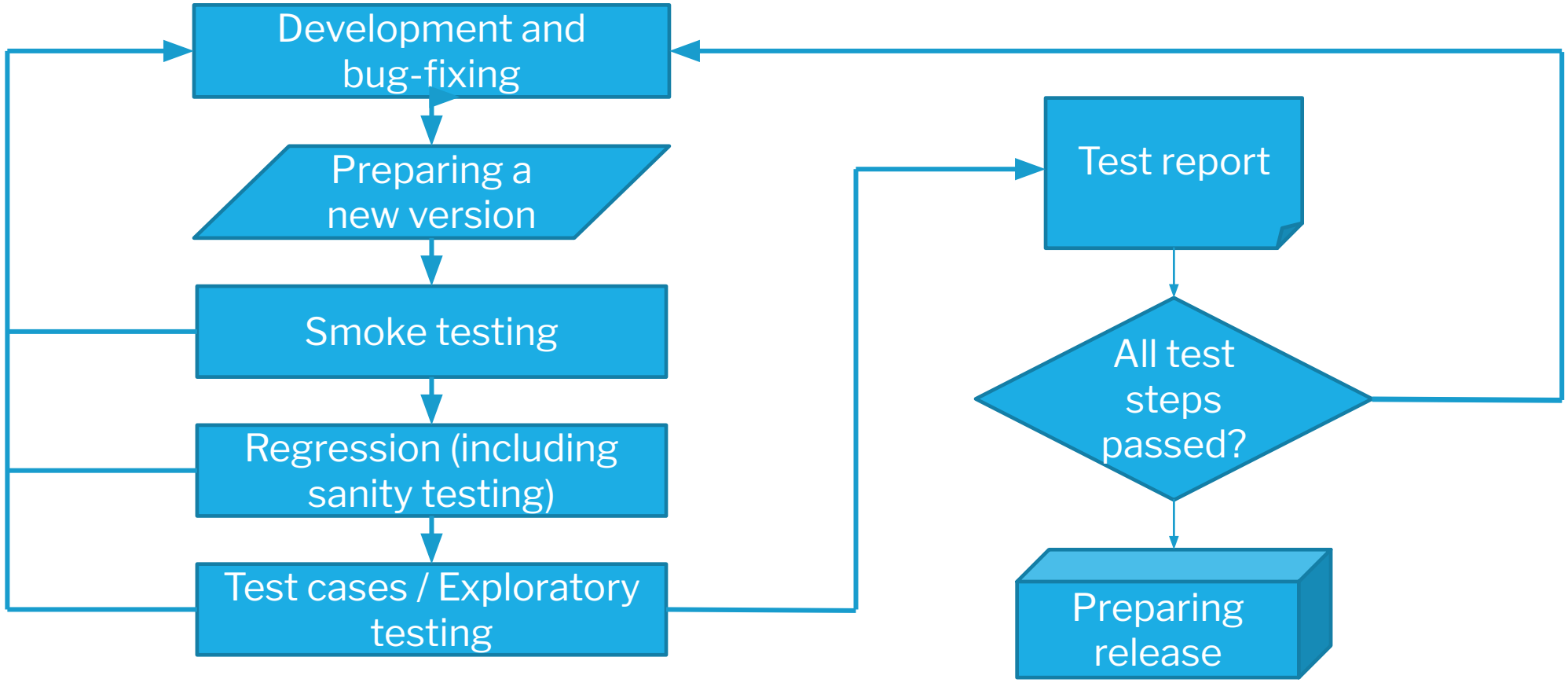
Плюсы

- Гибкость
- Фокусировка на важных аспектах приложения
- Скорость
- Ранний старт (даже без требований)
- Может дополнять сценарный подход

Минусы

- Детали критичны
- Глубокое знание проекта
- Высокая квалификация тестировщиков

ОБЩИЙ АЛГОРИТМ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ



* с фокусом на тестирование

ПРИНЦИПЫ ТЕСТИРОВАНИЯ

ПОДХОДЫ К ИНТЕГРАЦИОННОМУ ТЕСТИРОВАНИЮ

Bottom Up Integration

- Сначала тестируются низкоуровневые модули и функции, затем собирается и тестируется следующий уровень.

- Требует готовности большинства компонентов
- Применяется для оценки степени готовности продукта
- Стадия тестирования или готовности продукта хорошо накладывается на процентную ось

Top Down Integration

- Сначала тестируются высокоуровневые модули, низкоуровневые модули закрываются заглушками. По мере продвижения заглушки заменяются реальными имплементациями.

- Применение заглушек позволяет отвязать тестирование от временной оси

Big Bang Integration

- Модули всех уровней собираются в законченный компонент и производится тестирование.

- Позволяет значительно сэкономить время на тестирование
- Требует глубокой проработки тест-кейсов
- Может привести к лавинообразному появлению багов

АВАРИИ, ВЫЗВАННЫЕ ОШИБКАМИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Дата	Компания	Описание, причина	Последствия
1985-1987	**США, Канада – Аппарат лучевой терапии Therac-25**	Ошибки программного кода: ○ Использование одной переменной для различных целей; ○ Многозадачность без синхронизации; ○ Ошибки деления на ноль; Отсутствие системы механической блокировки излучателей.	Шесть передозировок радиацией, 2 фатальные.
15 января 1990	**США – Авария коммутаторов AT&A**	Один перезагрузившийся коммутатор вызвал каскадную бесконечную перезагрузку остальный коммутаторов системы. Причина: неверное расположение оператора break, вызывавшее повторную отправку сингала об ошибке, отсутствие соответствующих обработчиков. Программное обеспечение не тестировалось в окружении.	Последствия: 60 тысяч пользователей остались без связи на 9 часов.
Февраль 1991	**США – Комплекс противоракетный обороны «Patriot», размещенный в Саудовской Аравии**	Неверная разрядность при расчетах позиционирования привела к накоплению ошибок. Ошибка была обнаружена инженерами и исправлена до инцидента, но обновление комплекса прибыло на следующий день.	Пропуск атакующего ракетного удара по армейским казармам. Погибло 28 солдат.
1995	**США – Графический интерфейс Microsoft Bob для Windows 3.1**	Ошибка системы авторизации, позволявшая осуществить доступ к рабочему месту с неверным паролем пользователя. Причина: не в полной степени протестированный GUI.	Кто-нибудь вспомнит Microsoft Bob?
4 июня 1996	**ЕС – Авария ракеты-носителя Ariane-5**	Причина: программное обеспечение (унаследованное от Ariane-4) не было протестировано для новой аппаратной платформы.	4 спутника (500 млн долларов), 7 млрд долларов и 10 лет разработки.
1998-1999	**США – Mars Climate Orbiter**	Использование метрической системы на космическом аппарате, британской системы – в программном обеспечении на Земле. NASA перешло на СИ только в 2007 году.	125 млн долларов.
27 марта 2008	**Великобритания – Багажный и транспортный коллапс на 5-м терминале аэропорта Хитроу**	Программное обеспечение автоматизированной багажной линии тестировалось по идеальным сценариям, не предусматривая человеческого фактора	Потеря 42000 единиц багажа за 10 дней, отмена 5000 авиарейсов
1 августа 2012	**США – Биржевой брокер Knight Capital**	Из-за некорректного флага вместо production-кода активировался тестовый,проверявший ситуацию «покупать дороже, продавать дешевле», что значительно понизило цену акций, инициировав другие брокерские алгоритмы участвовать в сделках.	Урон составил 440 млн долларов за 45 минут и разорение компании.

СПАСИБО ЗА ВНИМАНИЕ

ТЕСТИРОВАНИЕ

EVGENIY SHVETSOV

2021