Лекция 6: Тестирование и дефекты

Юрий Литвинов
y.litvinov@spbu.ru

02.04.2026

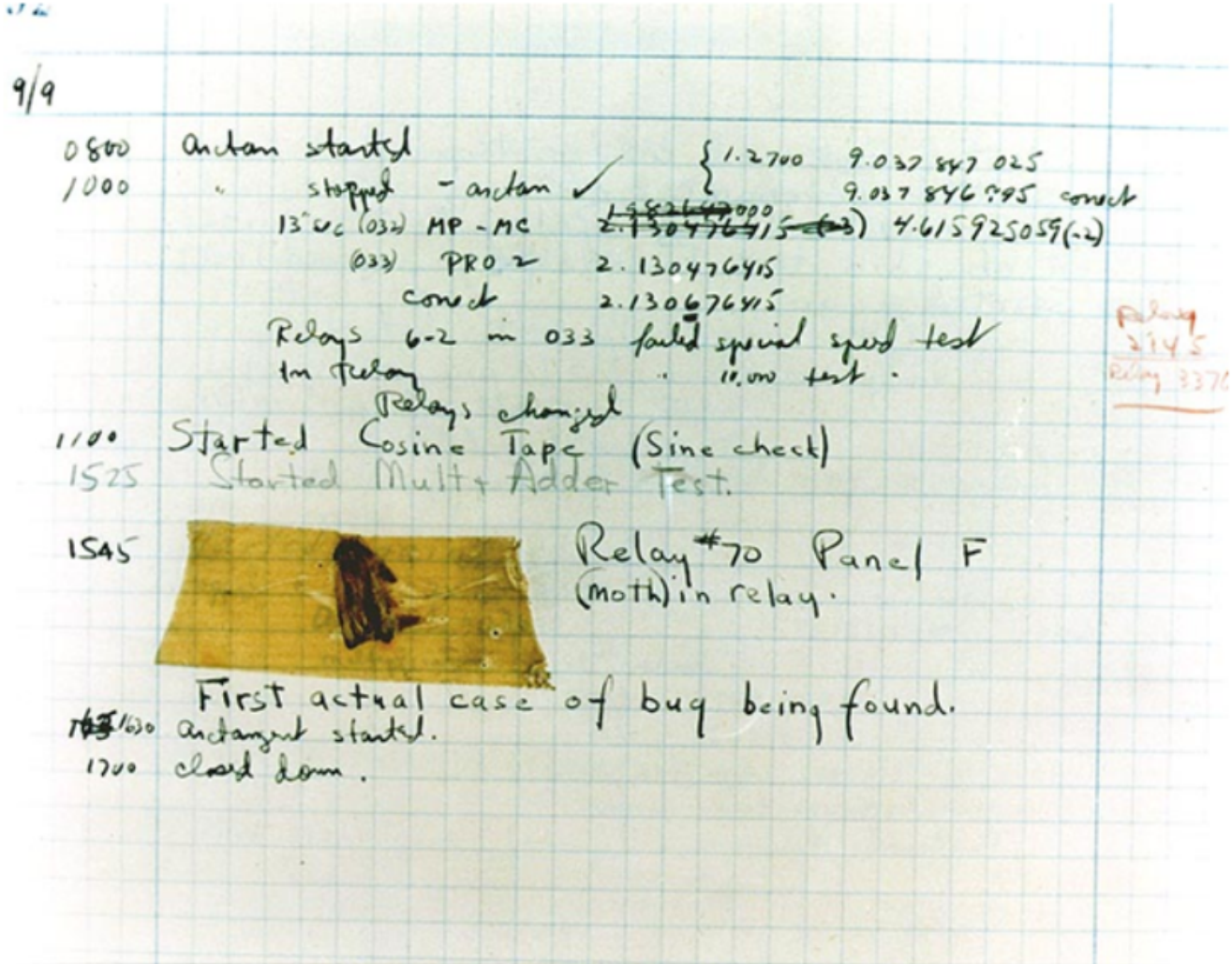
Тестирование

- Любая программа содержит ошибки
- Если программа не содержит ошибок, их содержит алгоритм, который реализует эта программа
- Если ни программа, ни алгоритм ошибок не содержат, такая программа даром никому не нужна

Ошибки

- Не несоответствие техническому заданию, а несоответствие ожиданиям
- Процесс тестирования субъективен

Баг



© https://education.nationalgeographic.org/resource/worlds-first-computer-bug/

Цель тестирования

- Тестирование — процесс поиска ошибок
  - Тест, не выявивший ошибку — впустую потраченное время
    - Не совсем правда, есть регрессионные тесты
- Тестирование не может доказать, что ошибок в программе нет
  - Субъективность ошибок
  - Огромное количество входных данных
    - Программа, складывающая два целых числа — сотни лет на полный тест
  - Огромное количество путей исполнения
    - 1979 год, около 20 строк кода, сто триллионов путей исполнений
- Формальная верификация

Виды тестирования

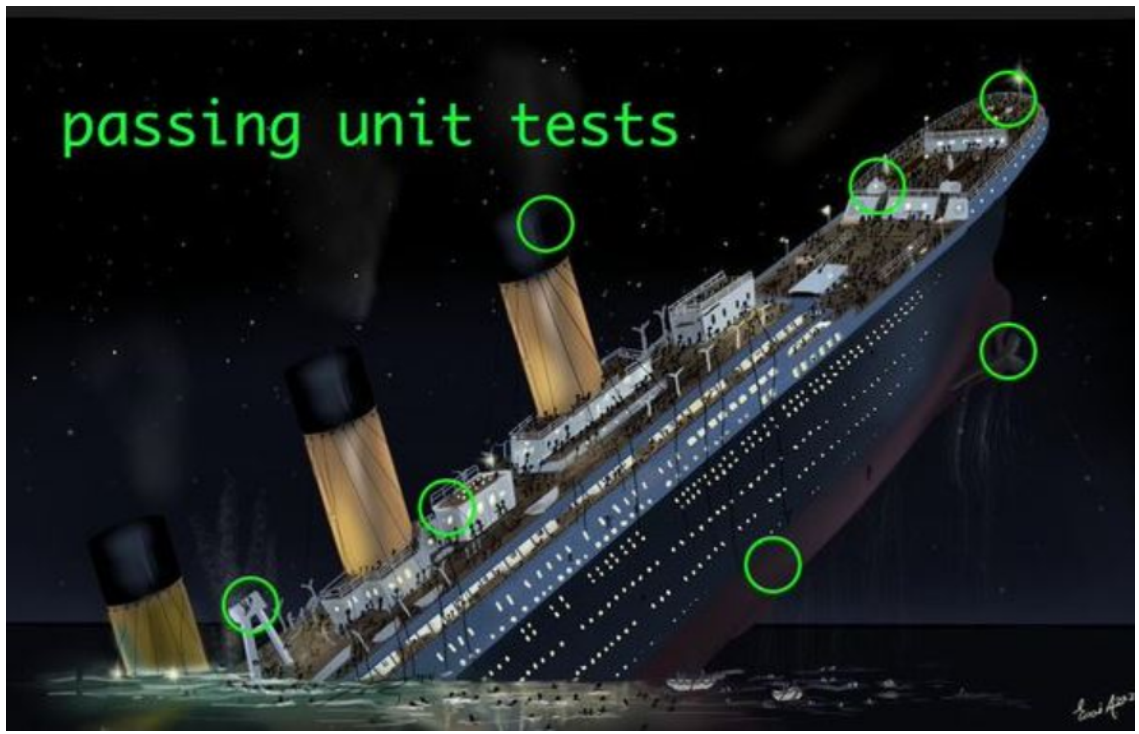
Классификация по тому, что тестируется

- Функциональное тестирование
- Тестирование производительности
  - Нагрузочное
  - Стресс-тестирование
  - Тестирование стабильности
  - Тестирование конфигурации
- Тестирование пользовательского интерфейса
- Тестирование удобства использования
- Тестирование безопасности
- Тестирование локализации
- Тестирование совместимости

Виды тестирования

Классификация по масштабности тестирования

- Модульное
- Интеграционное
- Системное
  - В т.ч. тестирование пользовательского интерфейса



Виды тестирования

По этапу жизненного цикла

- Смоук-тестирование
- Регрессионное тестирование
- Альфа-тестирование
- Бета-тестирование
- Релиз-кандидат

Виды тестирования

По знанию о системе

- Тестирование «чёрного ящика»
- Тестирование «белого ящика»
- Тестирование «серого ящика»
- Исследовательское тестирование

Тестирование требований

- Однозначность
  - слова «обычно», «как правило», «иногда», «необязательно» и т.п.
  - субъективные оценочные суждения: «удобно», «быстро», «гибко» и т.п.
- Атомарность (без предлогов и/или)
- Чёткий критерий приёмки (acceptance criteria)
- Отсутствие избыточных и противоречивых требований
- Мотивация каждой роли
- Не предполагает конкретного способа реализации

Критерии приёмки

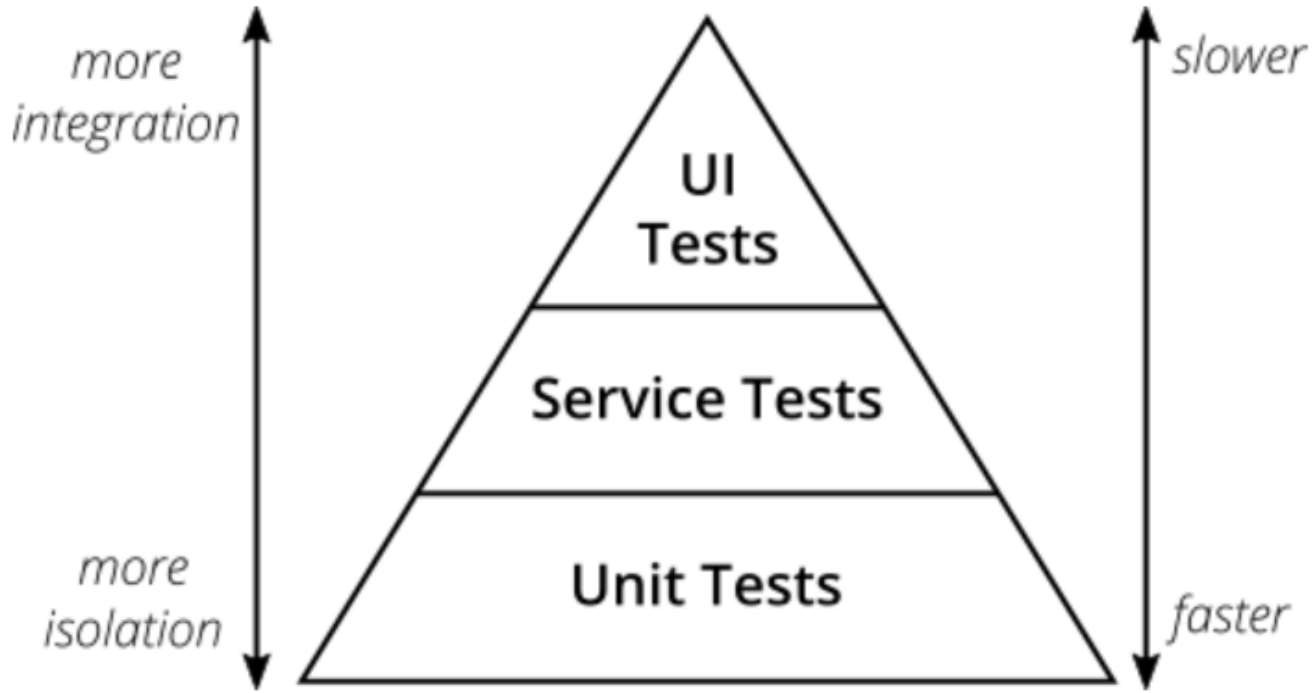
- Чёткие
- Контекст-действие
- Однозначно документирующие поведение системы
- Добросовестные

Тестирование архитектуры

- Аккуратность декомпозиции
  - Dependency Inversion
  - Слои
  - Микросервисы
    - Опасайтесь «микросервисного монолита»
- Простота
- Наблюдаемость
  - Возможность развернуть окружение
  - Быстрая обратная связь
  - Логирование
    - Поддержанное инструментами, например ElasticSearch и Kibana
  - Трассируемость
  - Метрики

Тестирование и реализация

Пирамида тестирования



© https://martinfowler.com/articles/practical-test-pyramid.html

Тестовые сценарии, свойства

- Идентификатор и название
- Предварительные шаги
- Тестовые шаги
  - Действие — ожидаемый результат
- Итоговый ожидаемый результат
- Шаги по восстановлению окружения
- Конфигурация
- Тэги
- Зависимости

Тесты, терминология

- Сьюты
- Проекты
- Тестовые прогоны
  - Статус
  - Результаты по каждому шагу
- Тест-планы
  - Отобранные тестовые сценарии
  - Календарные сроки
  - Ответственные

Пример тестового сценария

Что делаем	**Что происходит**
Вводим *adder* и жмём на *Enter*	Экран мигает, внизу появляется знак вопроса
Нажимаем 2	За знаком вопроса появляется цифра 2
Нажимаем *Enter*	В следующей строке появляется знак вопроса
Нажимаем 3	За вторым знаком вопроса появляется цифра 3
Нажимаем *Enter*	В третьей строке появляется 5, несколькими строками ниже — ещё один знак вопроса

Выявленные проблемы

- Нет названия программы на экране, может, мы запустили не то
- Нет никаких инструкций, пользователь без идей, что делать
- Непонятно, как выйти

Позитивный сценарий

Ввод	**Ожидаемый результат**	**Замечания**
99 + 99	198	Пара наибольших допустимых чисел
-99 + -99	-198	Отрицательные числа, почему нет?
99 + -14	85	Большое первое число может влиять на интерпретацию второго
-38 + 99	61	Отрицательное плюс положительное
56 + 99	155	Большое второе число может повлиять на интерпретацию первого
9 + 9	18	Два наибольших числа из одной цифры
0 + 0	0	Программы часто не работают на нулях
0 + 23	23	0 — подозрительная штука, его надо проверить и как первое слагаемое,
-78 + 0	-78	и как второе

Негативные сценарии

Ввод	**Замечания**
100 + 100	Поведение сразу за диапазоном допустимых значений
Enter + *Enter*	Что будет, если данные не вводить вообще
123456 + 0	Введём побольше цифр
1.2 + 5	Вещественные числа, пользователь может решить, что так можно
A + b	Недопустимые символы, что будет?
Ctrl-A, Ctrl-D, F1, Esc	Управляющие клавиши часто источник проблем в консольных программах

Ещё больше тестов!

- Внутреннее хранение данных — двузначные числа могут хранить в **byte**
  - 99 + 99, этот случай покрыли
- Кодовая страница ввода: символы ’/’, ’0’, ’9’ и ’:’
  - Программист может напутать со строгостью неравенства при проверке
  - Не надо вводить A + b, достаточно граничные символы

Библиотеки модульного тестирования

- JUnit со товарищи (NUnit, pytest и т.п.)
  - Инициализация SUT
  - Выполнение действия
  - Проверка результатов
- Библиотеки матчеров: Hamcrest и т.п.
  - Assert.That(f(), Is.EqualTo(1))
  - NUnit умеет «из коробки»
- Библиотеки тестирования, основанного на свойствах: QuickCheck, FsCheck
- Библиотеки символьного исполнения

Библиотеки тестовых заглушек

- Mockito, Moq и т.п.

```
LinkedList mockedList = mock(LinkedList.class);
// or even simpler with Mockito 4.10.0+
// LinkedList mockedList = mock();

// stubbing appears before the actual execution
when(mockedList.get(0)).thenReturn("first");

// the following prints "first"
System.out.println(mockedList.get(0));

// the following prints "null" because get(999) was not stubbed
System.out.println(mockedList.get(999));
```

Тестирование интерфейсов

- Пользовательские интерфейсы
  - Selenium: клик по координатам, запросы к DOM
  - White (Windows Accessibility API и т.п.)
- Программные интерфейсы
  - Postman, Swagger вручную, модульные тесты с запросами
  - Фаззеры, сканеры безопасности

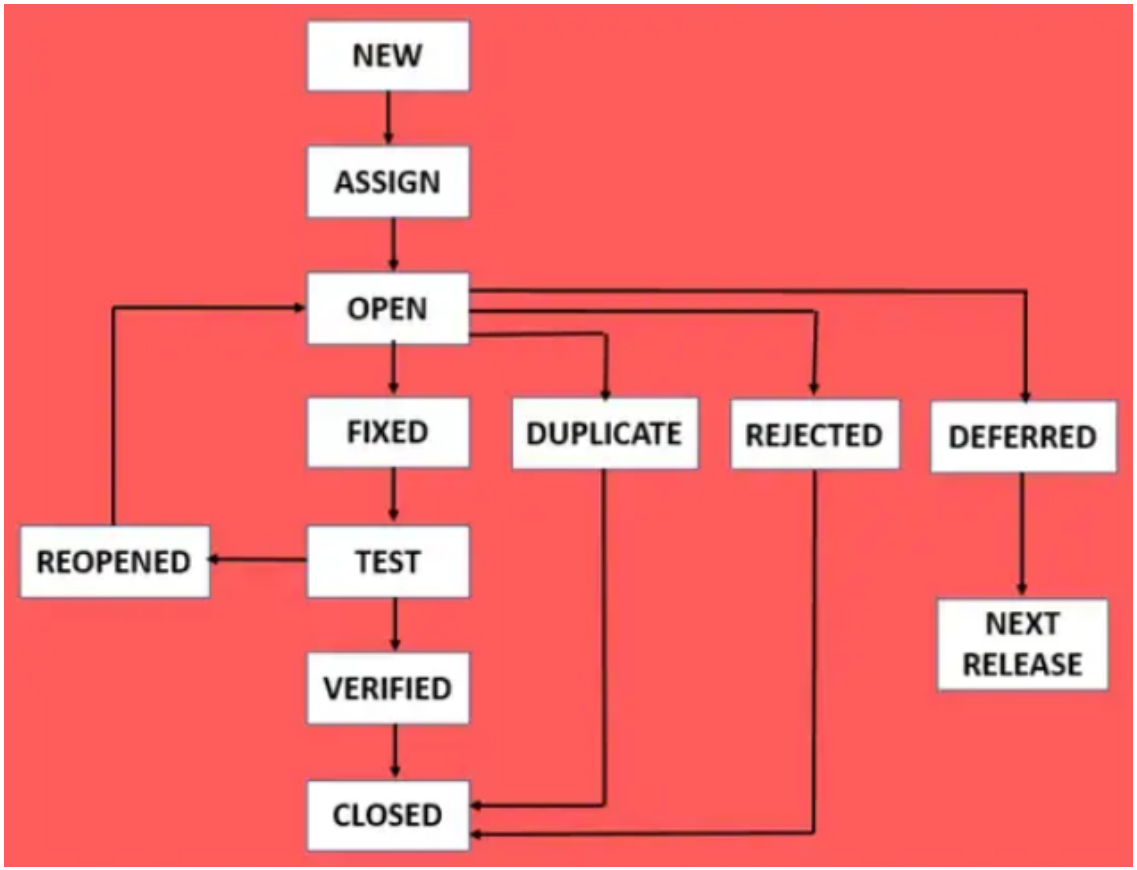
Системы управления тестированием

- Электронные таблицы, документы и т.п.
- TestRail
- Test IT, Xray, Kiwi TCMS, Sitechko
- TestY

Атрибуты отчёта об ошибке

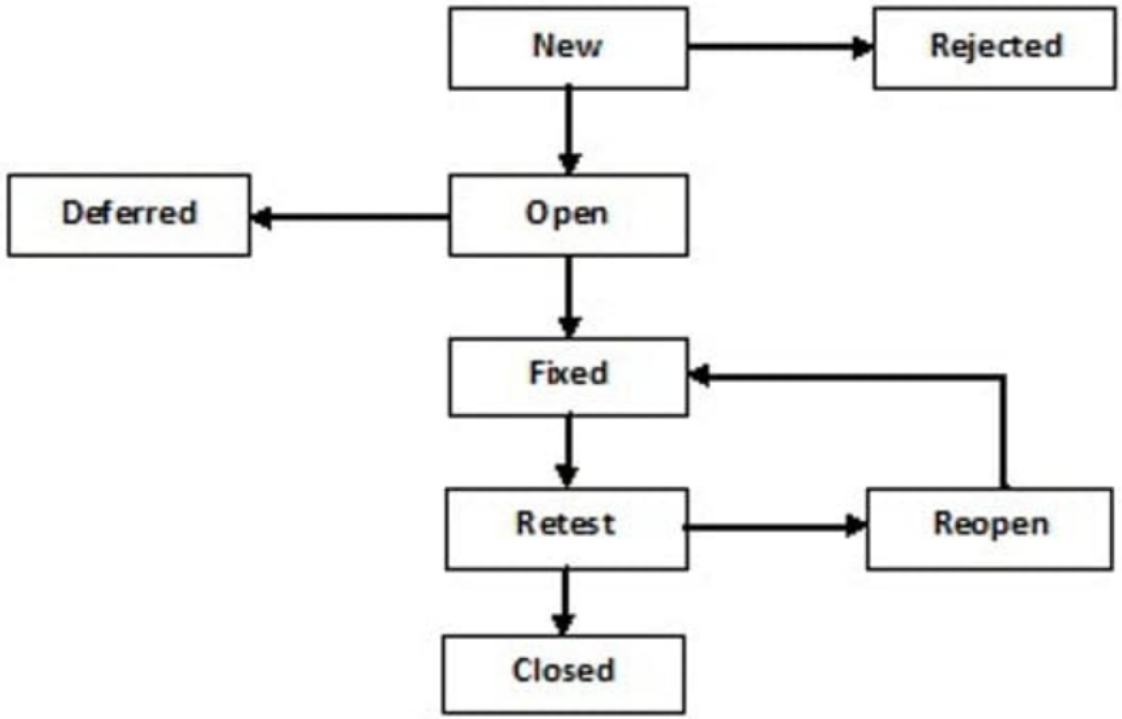
- Уникальный идентификатор
- Заголовок
- Описание дефекта
  - Контекст — версия, конфигурация и т.п.
  - Шаги воспроизведения — минимальны и воспроизводимы
  - Ожидаемый и полученный результаты
  - Дополнительная информация
- Серьёзность (блокер, высокая, средняя, низкая, тривиальная)
- Приоритет
- Статус
- Тип (ошибка, улучшение)
- Автор, ответственный за исправление, ответственный за проверку

Жизненный цикл ошибки



© https://www.softwaretestingmaterial.com/bug-life-cycle/

Ещё пример



© https://www.softwaretestinghelp.com/bug-life-cycle/

Системы отслеживания ошибок

- Jira
- GitHub Issues, Gitlab и т.п.
- Yandex Tracker
- Microsoft Team Foundation Server
- Redmine
- JetBrains Youtrack
- Bugzilla
- OpenProject