Тестирование

Даниил Ефремов

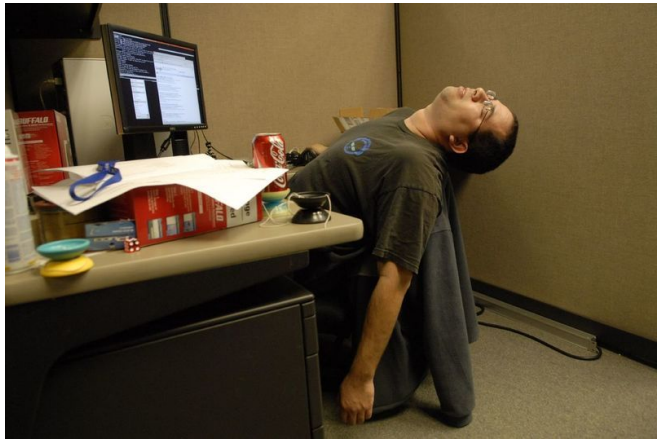
Senior Software Engineer @ Xored,
Выпускник ФИТ НГУ, к.э.н.

О чем пойдет речь

- Определимся с понятиями, целью и видами тестирования
- Для чего и для кого тестирование
- Тестирование в общем процессе разработке
- Советы разработчику как правильно писать и использовать тесты
- Инструменты для различных видов тестирования

Что есть тестирование ПО?

Тестирование — это способ убедить себя и заказчика, что разработка ПО завершена и можно (и не стыдно ;) просить деньги за хорошо выполненную работу.

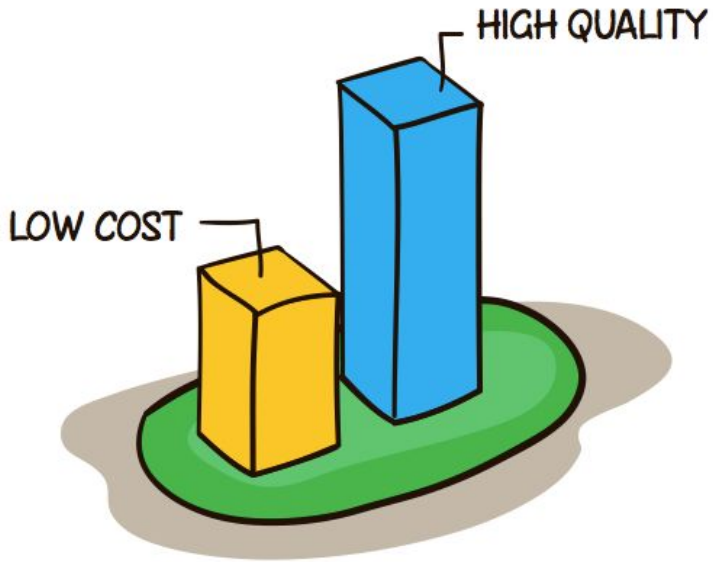


Понятие ошибки/дефекта

Дефект — неожиданное поведение программы, когда фактический результат отличается от ожидаемого.

Цель тестирования

Цель тестирования — повысить качество разрабатываемого ПО и сократить время разработки.



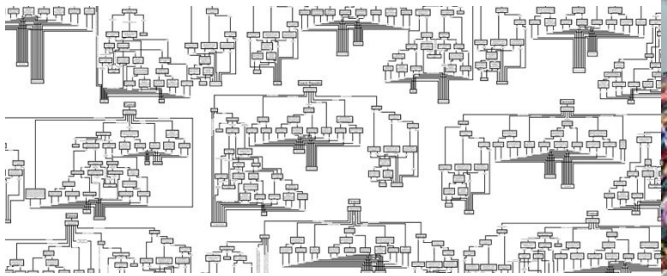
Кому нужно и кто должен тестировать

- Разработчик/QA-инженер

- Заказчик ПО (demo)
- Пользователь ПО (vk.com)

Какие сложности возникают при тестировании

- Среда для воспроизведения
- Предметная область
- Унаследованный запутанный код



Пирамида тестирования

Модульные тесты

- 1 сек
- 1 тест - 1 asset
- SOLID
- Переиспользование инициализации (Fixtures)
- Заглушки (Mock) – хорошо
- PowerMock – плохо

Как выбрать кейсы для тестирования

- Прямой кейс (на основе требований)
- Граничные условия
- Значение из диапазона
- Ошибки пользователя (негативные сценарии)
- Следить за покрытием

Инструменты для модульного тестирования

- Запуск (maven, JUnit, UnitTest++, QTestLib)
- Изоляции компонент (Заглушки)
- Автоматизации запуска (maven, Jenkins)

Пример плохих и хороших модульных тестов

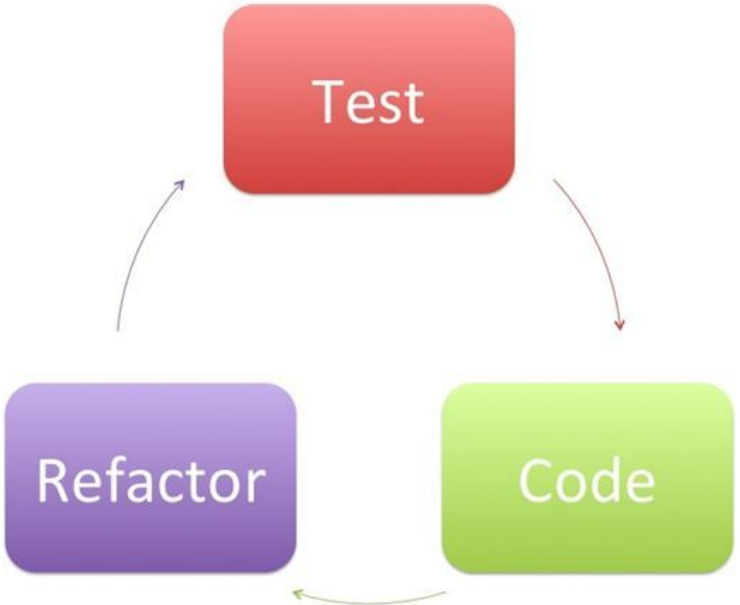
```java
/**
 * Модульные тесты сервиса AddService
 */
public class AddServiceTest {
    AddService service;

    void testAdd() {
        Assert.assertEquals(4, service.add(2, 2));
    }

    void testAddBad() {
        Assert.assertEquals(2 + 2, service.add(2, 2));
        Assert.assertEquals(3, service.add(1, 2));
        Assert.assertEquals(5, service.add(1, 4));
    }

    void testAddFirstNull() {
        try {
            service.add(null, 2);
            Assert.fail("Первый аргумент обязательный");
        }catch (IllegalArgumentException e) {
            Assert.assertEquals("Первый аргумент должен быть задан", e.getLocalizedMessage());
        }
    }

    void testAddSecondNull() {
        try {
            service.add(3, null);
            Assert.fail("Второй аргумент обязательный");
        }catch (IllegalArgumentException e) {
            Assert.assertEquals("Второй аргумент должен быть задан", e.getLocalizedMessage());
        }
    }
}
```

```java
public class AddService {

    /**
     * @param param1 - Первый аргумент, обязательный
     * @param param2 - Второй аргумент, обязательный
     * @return сумма param1 и param2
     */
    int add(Integer param1, Integer param2) {
        Preconditions.checkNotNull("Первый аргумент должен быть задан", param1);
        Preconditions.checkNotNull("Первый аргумент должен быть задан", param2);
        return param1 + param2;
    }
}
```

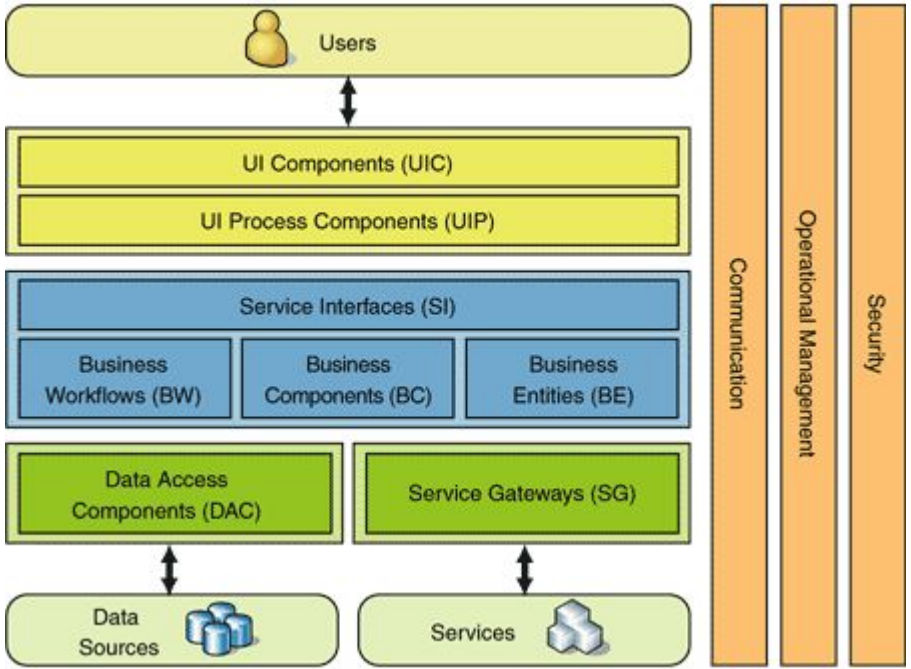
Параметризация модульных тестов

```
public class LocaleUtilsTest extends Assert {

  @DataProvider
  public Object[][] parseLocaleData() {
    return new Object[][]{
      {null, null},
      {"", LocaleUtils.ROOT_LOCALE},
      {"en", Locale.ENGLISH},
      {"en_US", Locale.US},
      {"en_GB", Locale.UK},
      {"ru", new Locale("ru")},
      {"ru_RU_some_variant", new Locale("ru", "RU", "some_variant")},
    };
  }

  @Test(dataProvider = "parseLocaleData")
  public void testParseLocale(String locale, Locale expected) {
    final Locale actual = LocaleUtils.parseLocale(locale);
    assertEquals(actual, expected);
  }
}
```

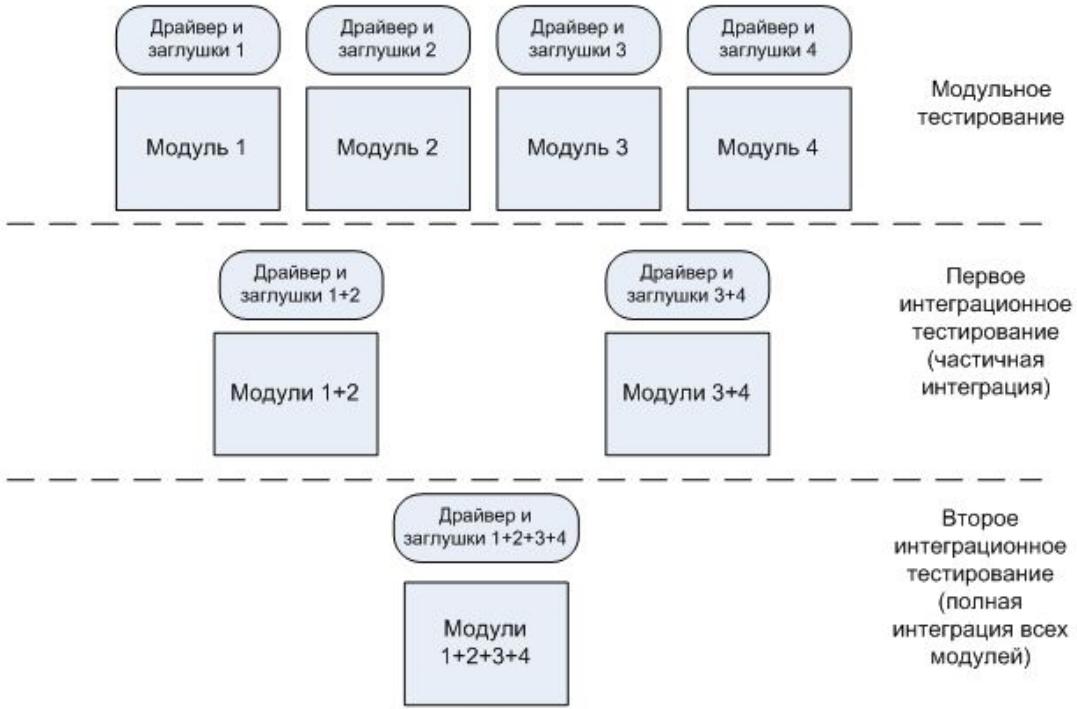
Test Driven Development (TDD)



Тестирование определяет архитектуру



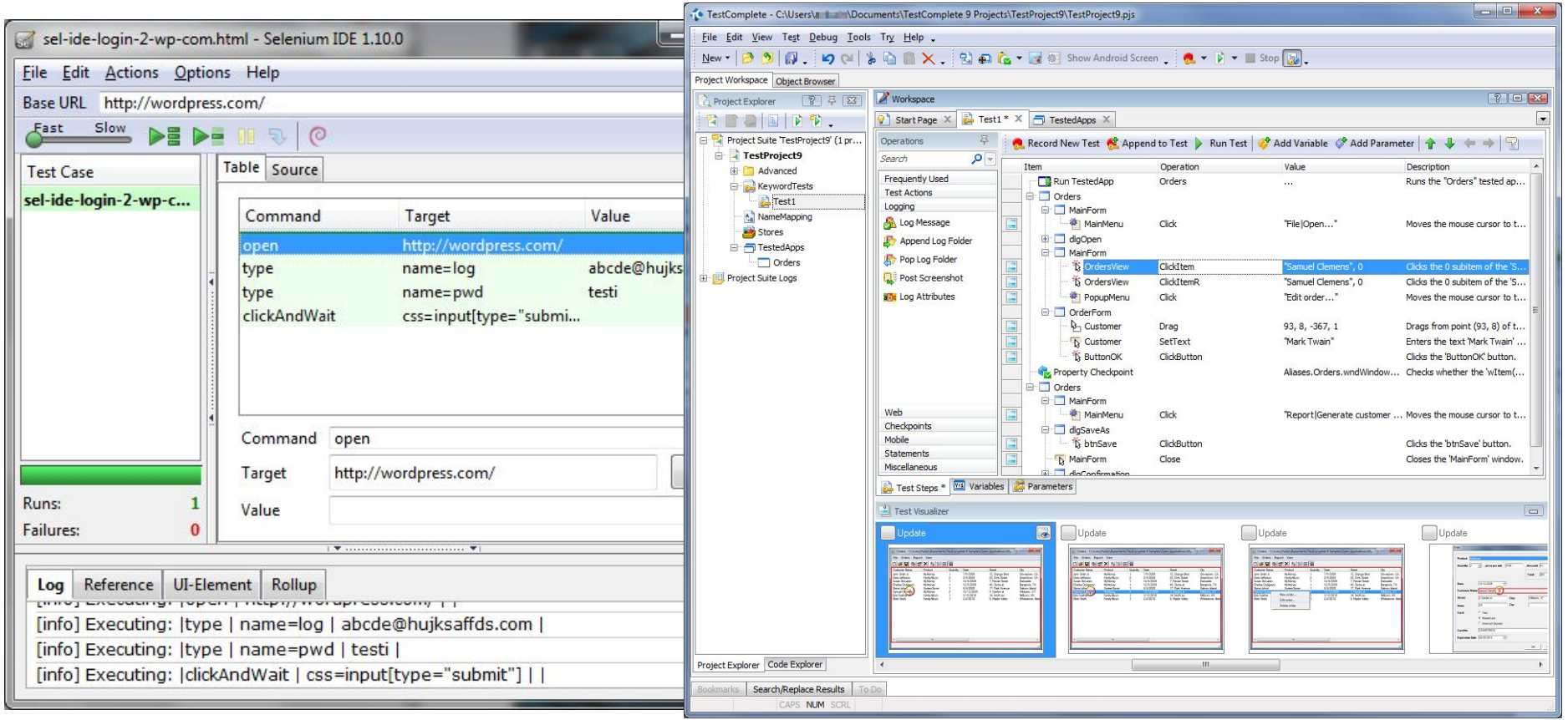
Интеграционные тесты



Системное тестирование

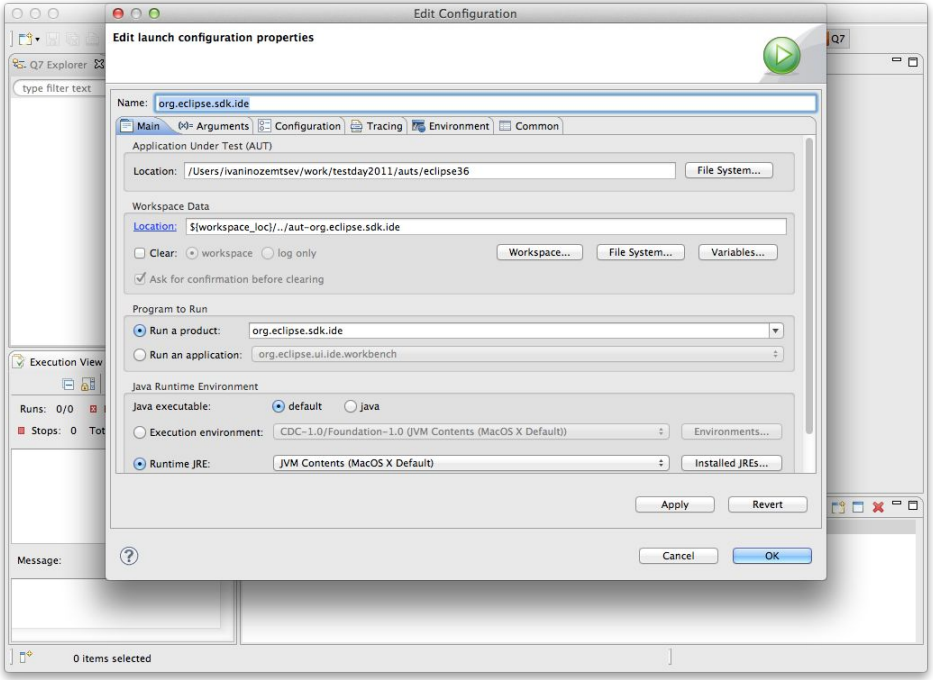
При системном тестировании тестируется интегрированная система целиком и проверяется её соответствие исходным требованиям.

Инструменты для UI тестов Selenium, TestComplete



RCPTT(Q7) by Xored

Средство автоматизации тестирования UI для приложений на основе Eclipse.

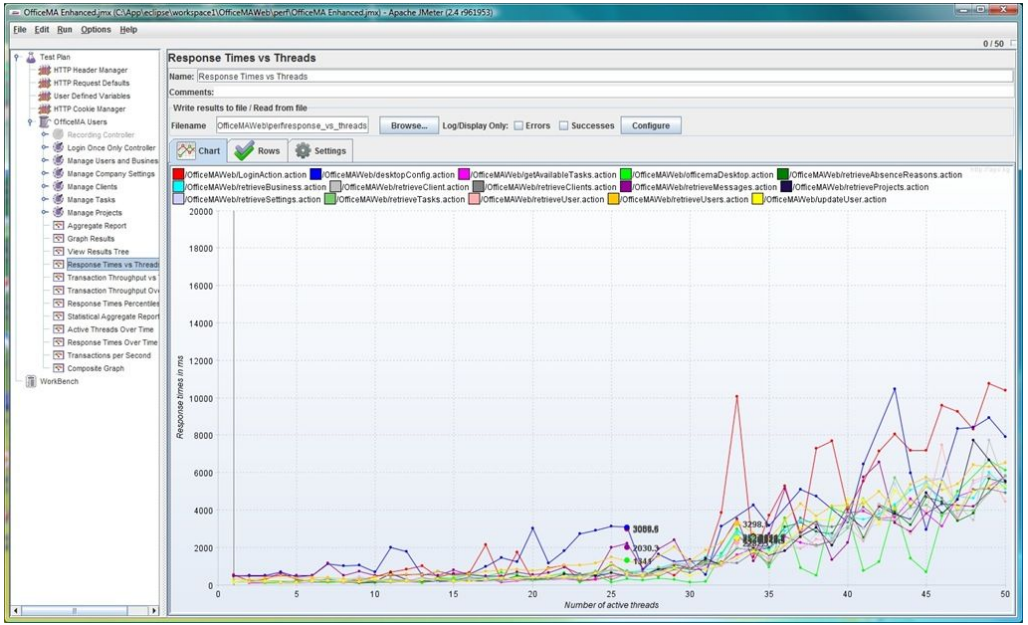


Нагрузочные тесты

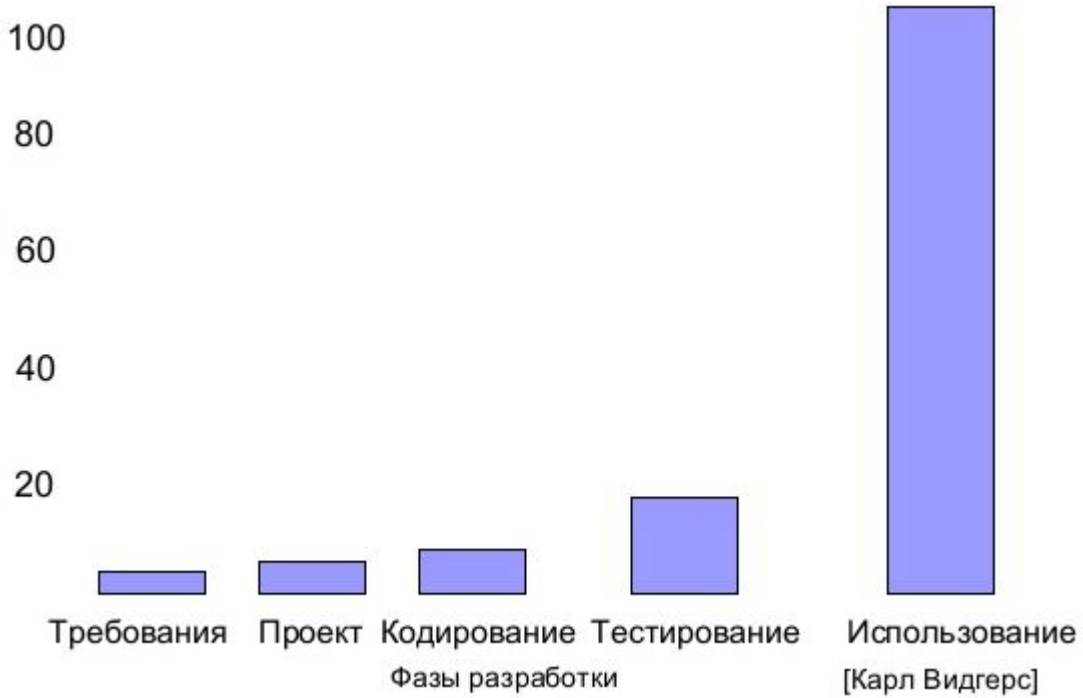
Обязательны при наличии нефункциональных требований, так как нефункциональные требования определяют архитектуру ПО.

Инструменты

- На базе интеграционных
- JMeter



Когда необходимо начинать тестирование



Эффективность тестирования

Методика устранения дефектов	Эффективность: Min.	Typical	Max.
Неформальные обзоры проекта	25%	35%	40%
Формальные инспекции проекта	45%	55%	65%
Неформальные обзоры кода	20%	25%	35%
Формальные инспекции кода	45%	60%	70%
Моделирование или прототипирование	35%	65%	80%
Самостоятельная проверка кода	20%	40%	60%
Блочное тестирование	15%	30%	50%
Тестирование новых функций (компонентов)	20%	30%	35%
Интеграционное тестирование	25%	35%	40%
Регрессивное тестирование	15%	25%	30%
Тестирование системы	25%	40%	55%
Ограниченное бета-тестирование (<10 организациях)	25%	35%	40%
Крупномасштабное бета-тестирование (>1000 организаций)	60%	75%	85%

Мотивация или бонусы от тестов

- Поощрение изменений
- Упрощение интеграции
- Документирование кода
- Положительно влияют на архитектуру приложения

Другие виды тестирования

- Смок тесты

- Тестирование Usability

- Инсталляционное тестирования

- Коридорное тестирование

Тестирование QA отделом

- QA готовит план тестирования на основе требований, и критически относится к самим требованиям
- Что передать в тестирование — сборку с указанием версии и release notes
- Указать какие модули дополнительно стоит проверить

Как репортить дефекты

Как незарегистрированный пользователь я должен иметь возможность зарегистрироваться #52

Open **DVEfremov** opened this issue 3 hours ago · 0 comments

DVEfremov commented 3 hours ago — Xored Educational member

Предусловия

- Запущенное веб приложение с произвольным напонением

Шаги воспроизведения

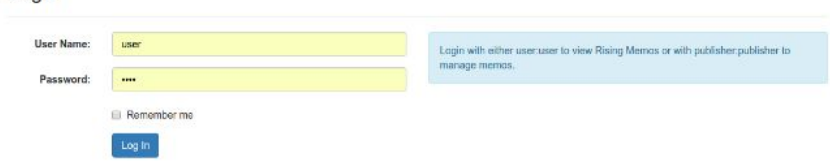
- Открываю домашную страницу

Ожидаемый результат

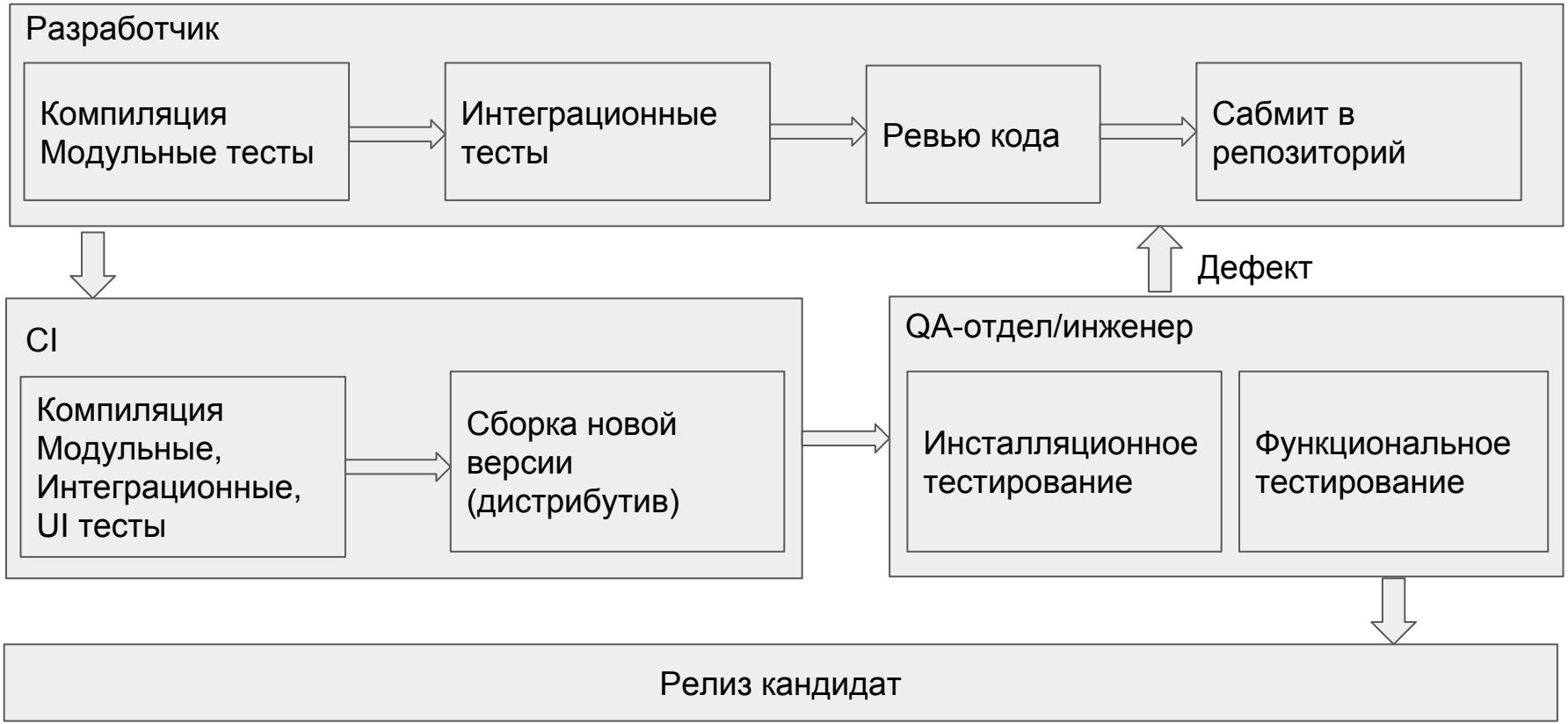
- Вижу ссылку которая ведет на форму регистрации

Фактический результат

- Вижу панель с указанием демонстрационных логинов и паролей, которой быть не должно

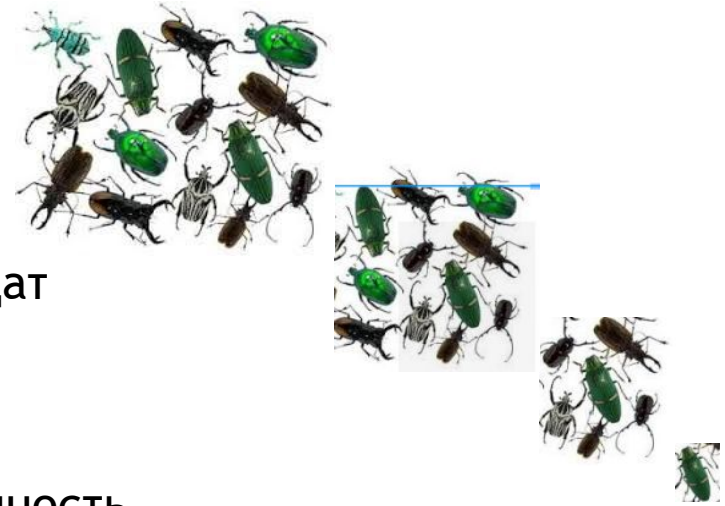


Описание процесса тестирования завершенной задачи



Стадии разработки программного обеспечения

- Пре-альфа
- Альфа
- Бета
- Релиз-кандидат
- Релиз
- [Пост-релиз]
- Общая доступность



Выводы

- Модульные + Интеграционные ~ 100% покрытие
- Тесты = Код
- Полная автоматизация
- Тесты -> рефакторинг -> архитектура
- Тестовая среда приближена к реальной

Литература

- XUnit test patterns : refactoring test code / Gerard Meszaros.
- Test-driven Development by Example / Kent Beck
- Рефакторинг. Улучшение существующего кода / Мартин Фаулер
- Совершенный код / Стив Макконнелл
- Разработка требований к программному обеспечению / Карлс Видгерс

Спасибо за внимание!

Вопросы?