HEISENBUG 08.12.2017 Moscow

# How to Build Your Own Test Automation Framework?

Dmitry Buzdin

# Пару Слов о Докладчике

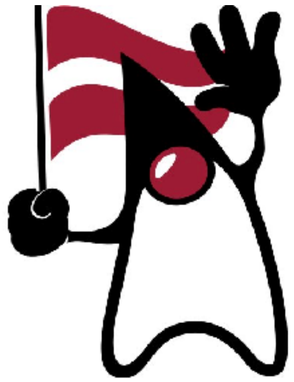

http://rigadevdays.lv

**Из Юрмалы**

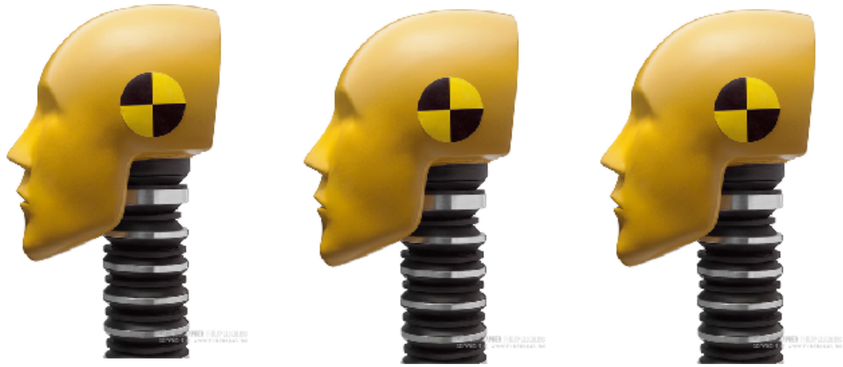
Dmitry Buzdin

@buzdin

dmitry@buzdin.lv



http://jug.lv



https://www.meetup.com/Riga-Test-Automation-Club/

2005 tieto


Keyword-driven
Cross-platform
Multi-language
Distributed

...

2013 ERICSSON

1000+ users
Scenario-based
Distributed

...

Мои прелестные фреймворки!

2016 FREELANCE

Cucumber
Selenium
Cloud and Microservices

Самый первый фреймворк, который я написал в 2005 году
до сих пор используют…

С годами я понял, что ничего не меняется
Технологии приходят и уходят, но базовые принципы остаются

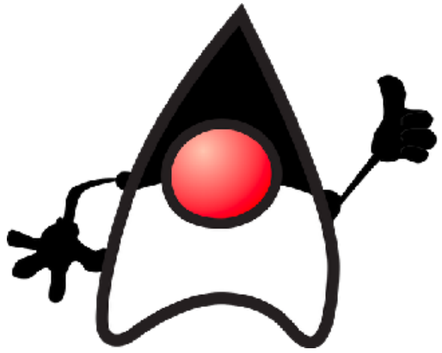
# О чем это я...

- Что-же такое тестовый фреймворк?
- Из каких компонентов он состоит?
- Что следует принять во внимание при построении фреймворка?
- Как сделать большой и долгоиграющий фреймворк?

# Мы не будем сегодня говорить о коммерческих тулах*

* но тема тоже к ним относится

# Большинство примеров будет на языке Java*



* не потому что он самый-самый, но и поэтому тоже

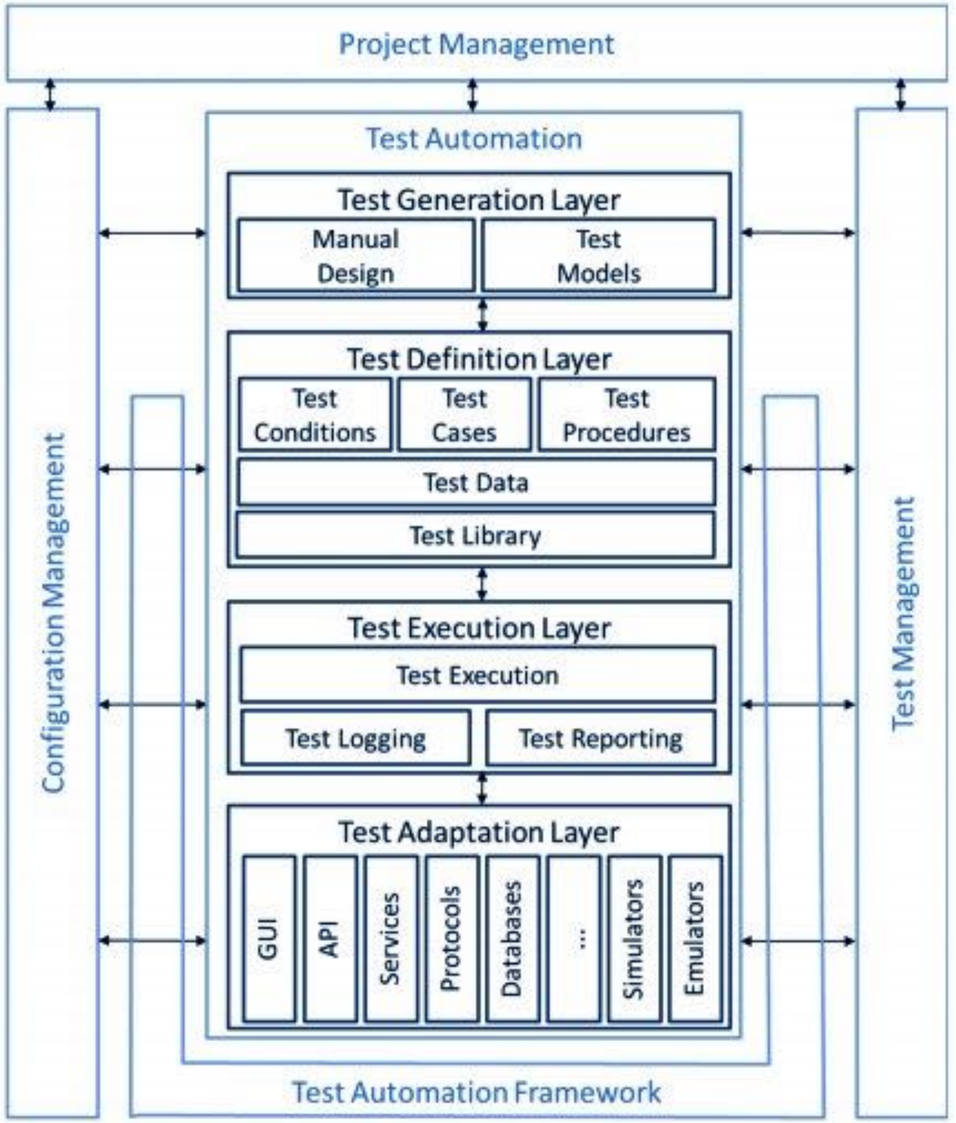
# Что-же такое тестовый фреймворк?

# Как разные люди видят фреймворк?


# Существуют некие архитектурные принципы!

# ISTQB Generic Test Automation Architecture

**По крайней мере один нашелся**

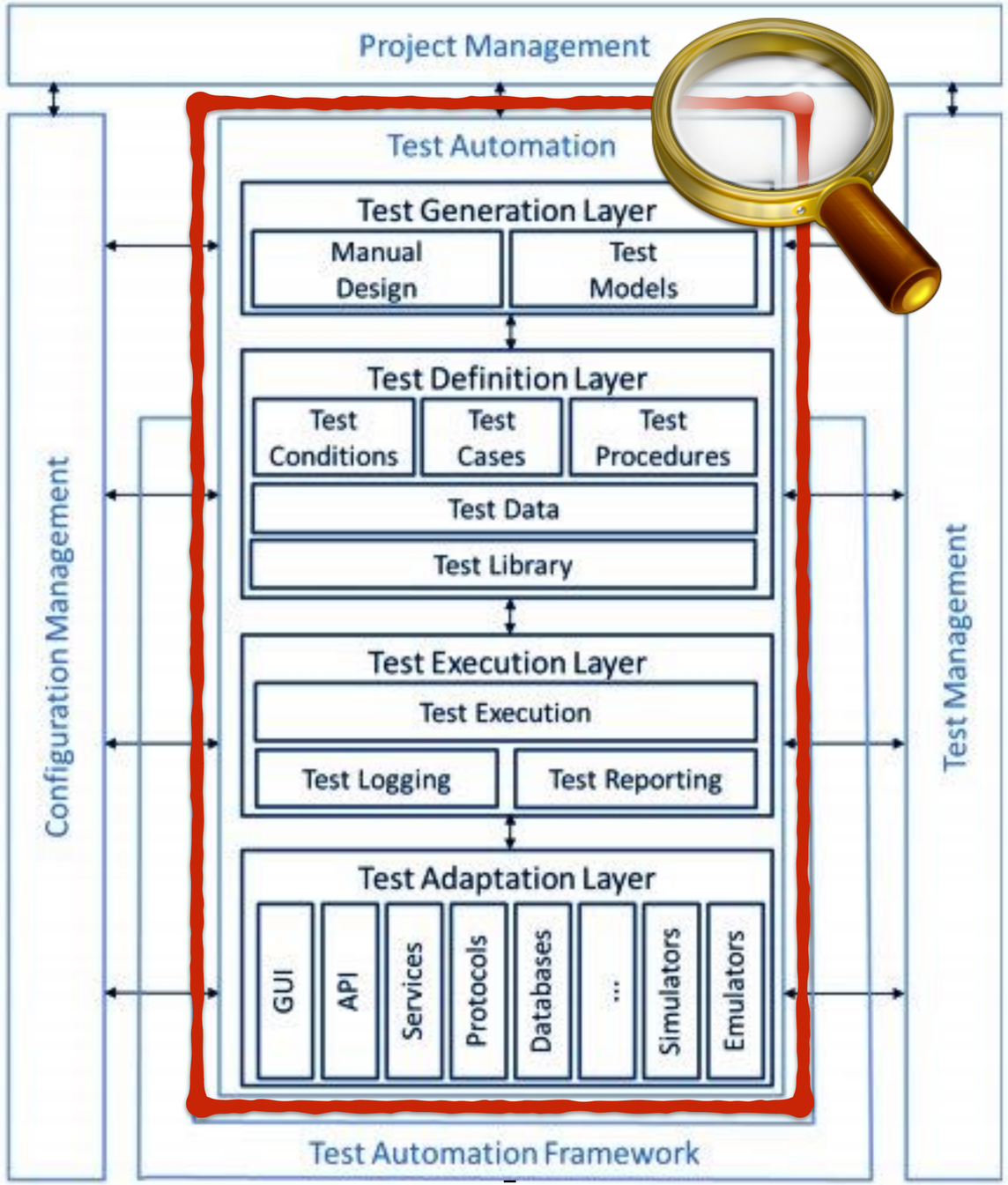



ISTQB International Software Testing Qualifications Board

https://www.istqb.org/downloads/category/48-advanced-level-test-automation-engineer-documents.html

“В теории нет разницы между практикой и теорией. На практике разница есть.”

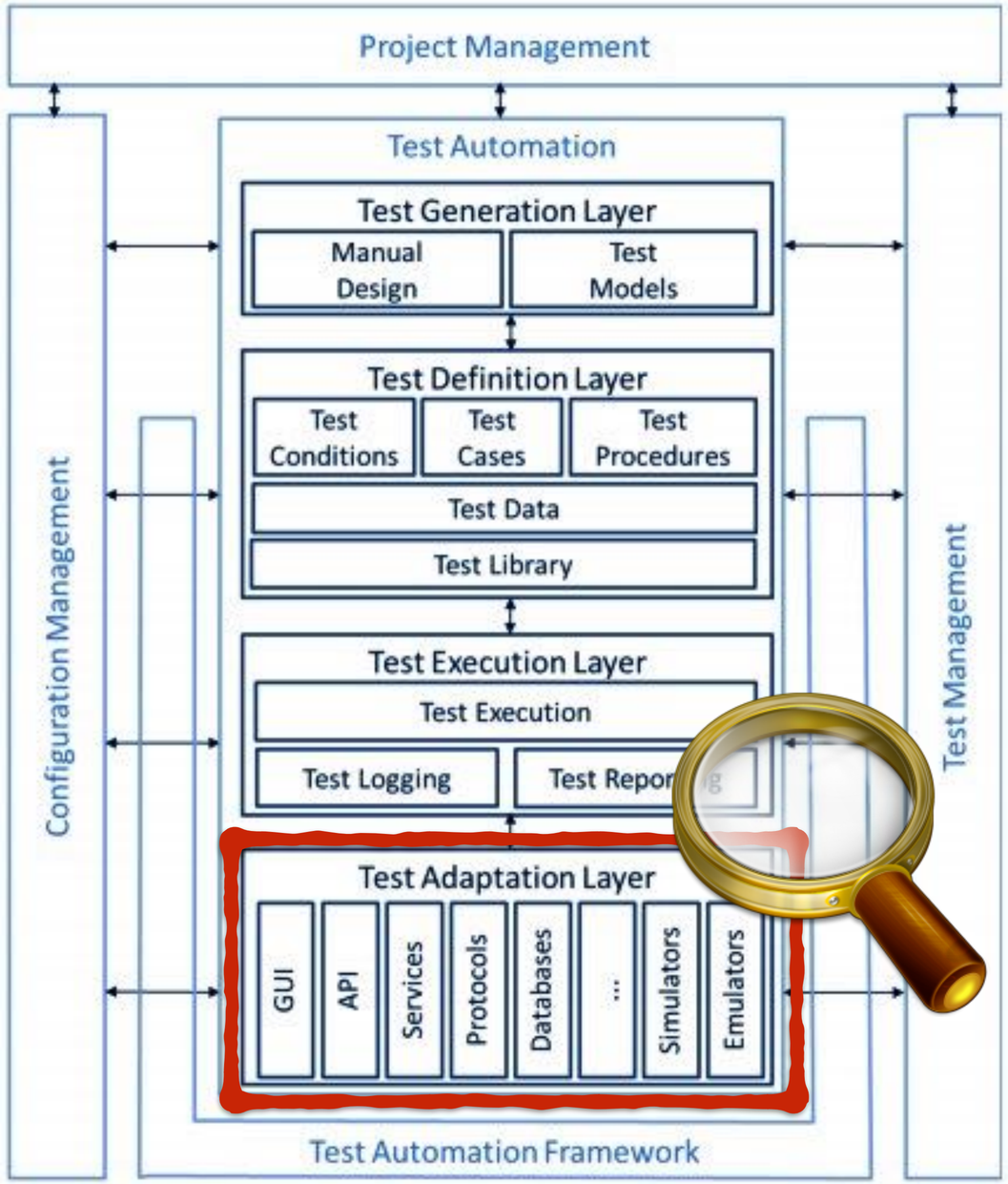
Давайте посмотрим, что-же нам впаривают...




**Опустим детали типа ДевОпс, тестового и проектного менеджмента**

**Нам интересно само построение фреймворка**

# Test Adaptation Layer




GUI
API
Services
Protocols
Databases
Simulators
Emulators

# “Test Adaptation” Слой

- Взаимодействие с тестируемой системой
- Абстракция над GUI технологиями:
  Web, Mobile, Desktop, Image Recognition
- Абстракция над протоколами общения:
  HTTP, AMQP, SSH, JDBC и т.д.

# Тестовые Адаптеры

- Просты в использовании
- Конфигурабельные
- Независимые
- Можно заменить

# Пример Адаптера

Передаем параметры

Контроль над ресурсами

Простой АПИ
Оригинальный JMX АПИ - ужасен!

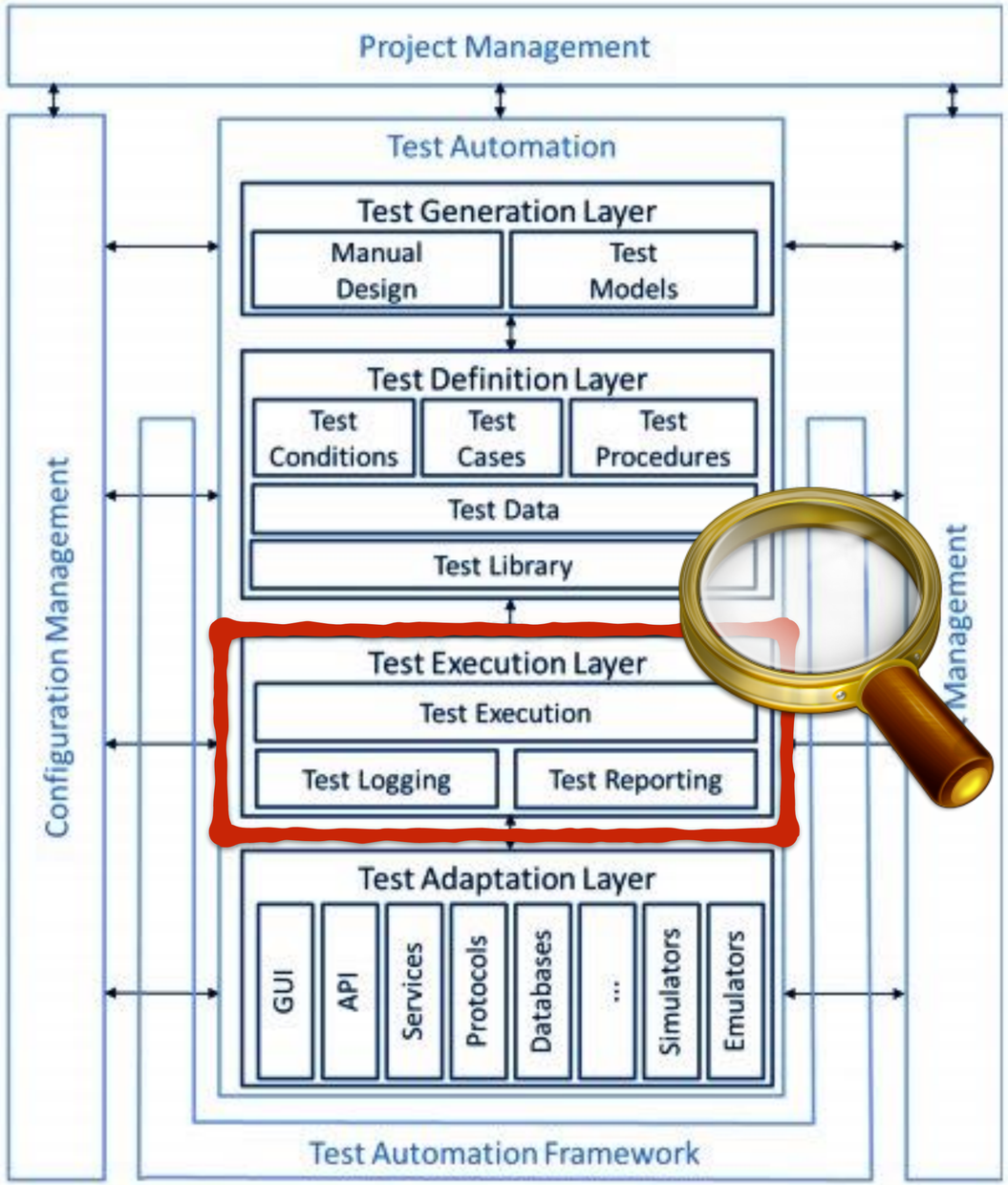
```
JmxAdapter adapter = JmxAdapter.builder()
        .withHost("http://localhost")
        .withPort(5000)
        .withContext("context")
        .build();

JmxSession session = adapter.connect();
Long memoryUsed = adapter.readMemoryUsed(session);
System.out.println("Used memory is : " + memoryUsed);
```

# Page Object* это тоже адаптер

Как и клиентская библиотека REST API

* наверное самый популярный шаблон проектирования тестов на Selenium

# Test Execution Layer



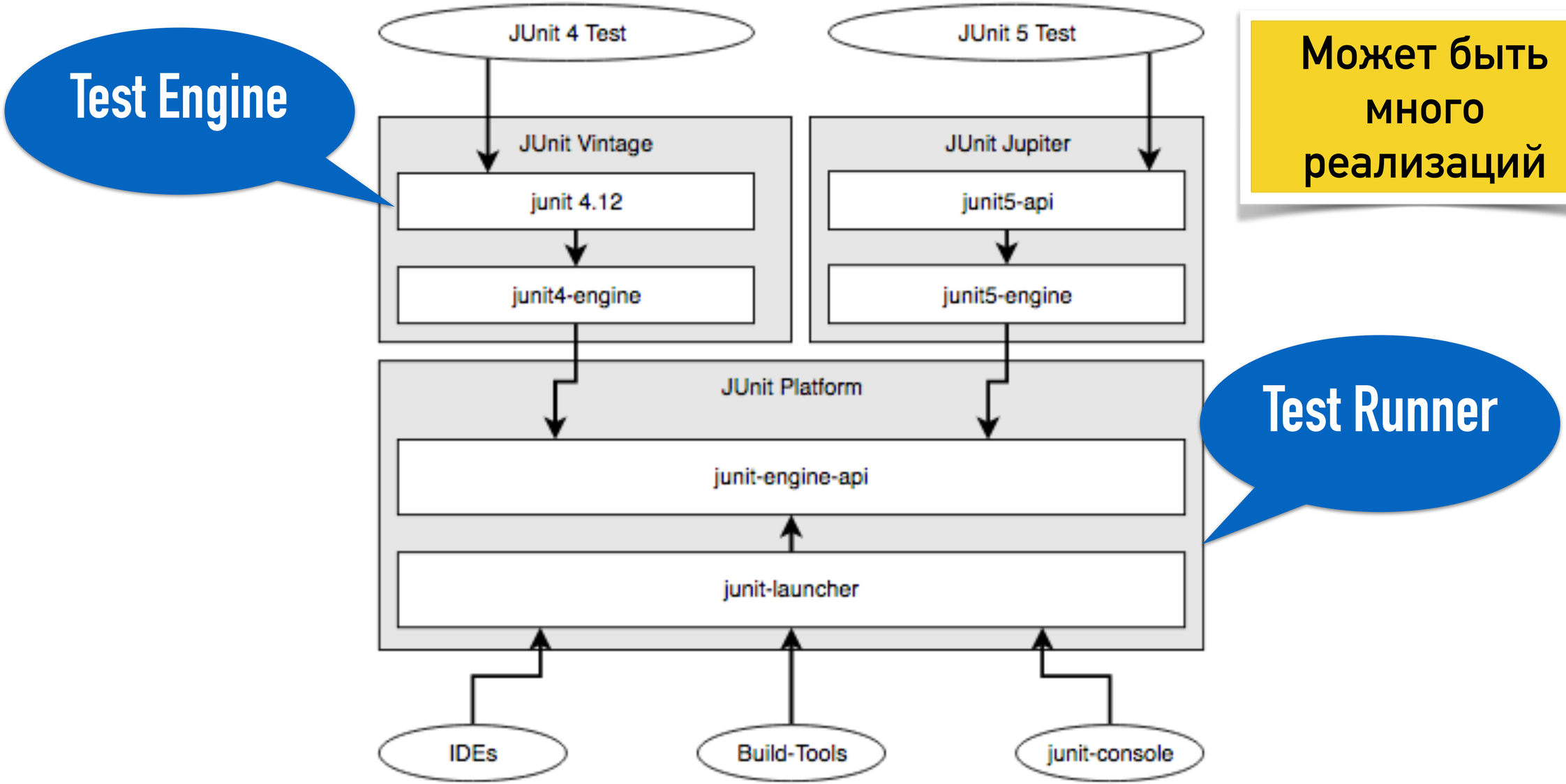

Test Execution
Test Logging
Test Reporting

# “Test Execution” Слой

Отвечает за:

- Нахождение тестов
- Запуск тестов
- Сбор отчетности и логов
- Интеграцию с внешними тулами

# На примере архитектуры JUnit 5




Может быть много реализаций

# Ответственность Test Engine

- Предоставляет API для описания тестов
- Находит и запускает тесты
- Определяет точки расширения

# Ответственность Test Runner

- Интеграция с IDE, Build и другими тулами
- Интеграция с тестовыми движками
- Запуск тестов в движках согласно процедуре исполнения

Если эти две компонента независимые, значит-ли что можно запускать тесты на удаленной машине?

# Зачем Запускать Удаленно?

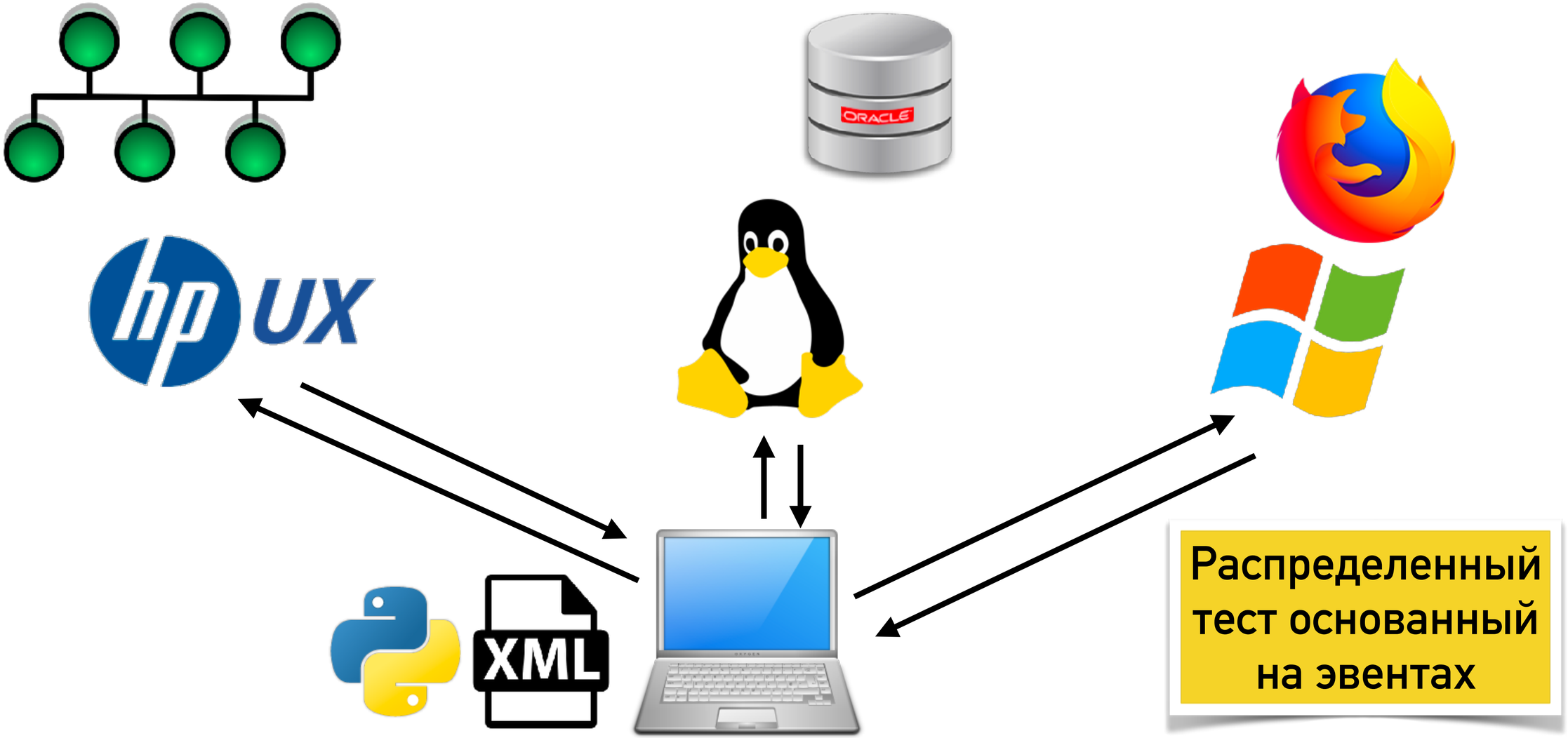
- Запуск из IDE в другом процессе, чтобы не убить редактор
- Платформо-зависимые тесты
- Распределенные тестовые сценарии

# На примере STAF

- Тестовый фреймворк от IBM
- Открытый код
- Кросс-платформенный
- Много-языковой
- Распределенный

http://staf.sourceforge.net/

# Тест “из жизни”



# Software Testing Automation Framework (STAF)

Похоже Source Forge не выключают именно из-за этого тестового фреймворка

```
--------------------------------------------------------------------------------

Version 0.10: 02/16/1998

  + Initial release

--------------------------------------------------------------------------------
```

# Test Logging

- Оставляем историю тестов для дальнейшего анализа
- Предоставляем связку логов с действиями теста
- Хорошее логирование экономит время, деньги и здоровье!

# Не самый лучший вариант…

```
System.out.println(
  “HTTP Result for ” +
  req.getUrl() +
  “ is ” +
  resp.getStatus()
);
```

- Конкатенация строк :(
- Нет уровней детализации
- Медленное исполнение
- Нет форматирования

# Стандартный Подход

```
private static final Logger logger =
  LoggerFactory.getLogger(“NAME”);

logger.info(“Hello, logger!”);
logger.error(“Error!”);
logger.debug(“…”);
```


https://logback.qos.ch/

Самая популярная библиотека логов в Java

# Пример использования меток MDC

```
try {
    MDC.put(“test-id“, “TMS-1234”);
    logger.info(“Starting test”);
    logger.debug(“Stopping test”);
} finally {
    MDC.clear();
}
```

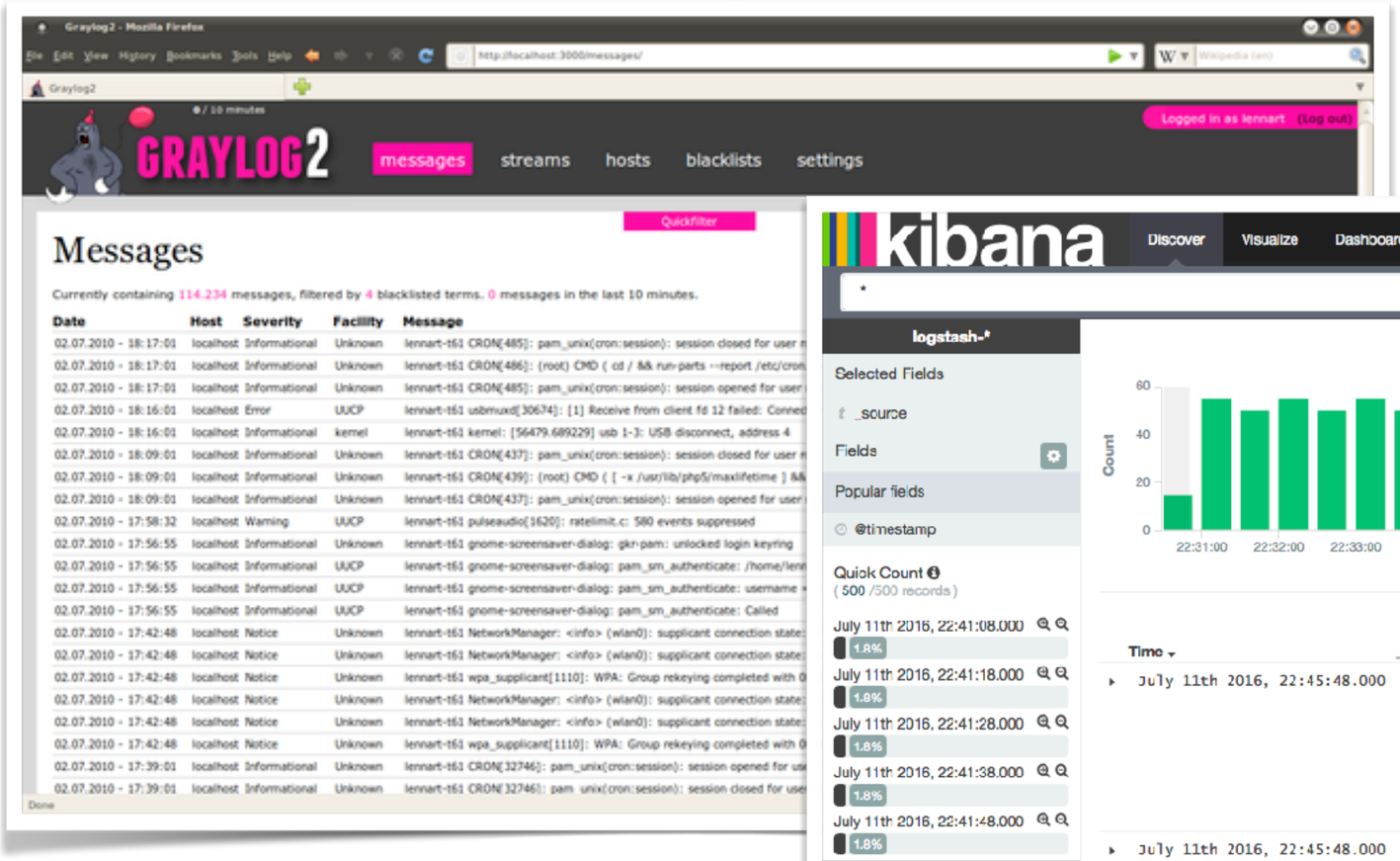
Можно добавлять
- Название теста
- Название сьюта
- Идентификатор запуска

Выведет

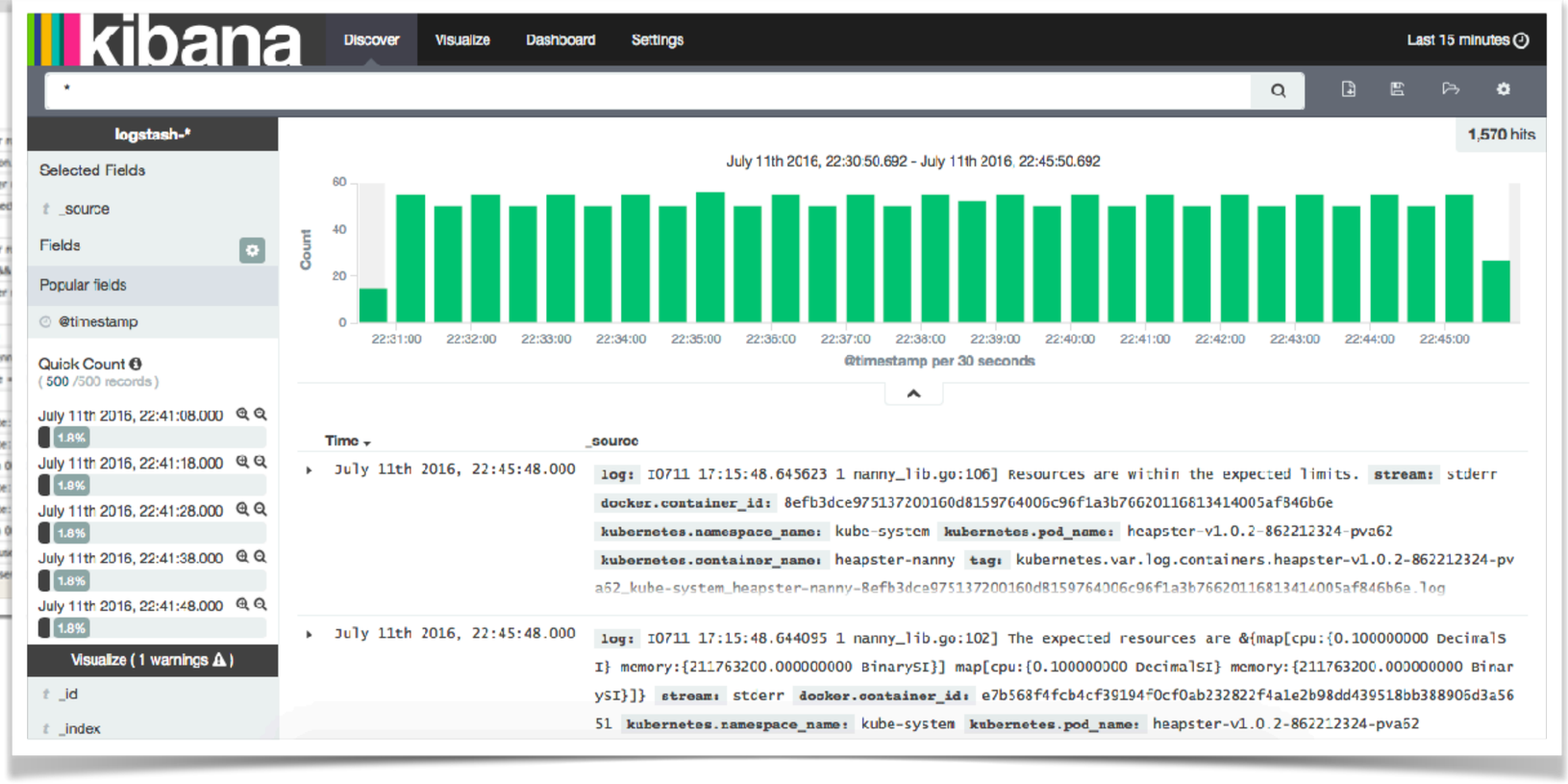
Будет легко найти

```
2017-12-06 13:03:15.198  INFO  [TMS-1234] : Starting test
2017-12-06 13:03:17.245  DEBUG [TMS-1234] : Stopping test
```

# Log Management Tools




https://www.graylog.org/




http://www.elastic.co/products/kibana

# Зачем Туда?

- Логи не будут утеряны (никто не сотрет)
- Легко доступны
- Мгновенный поиск по логам (ElasticSearch)

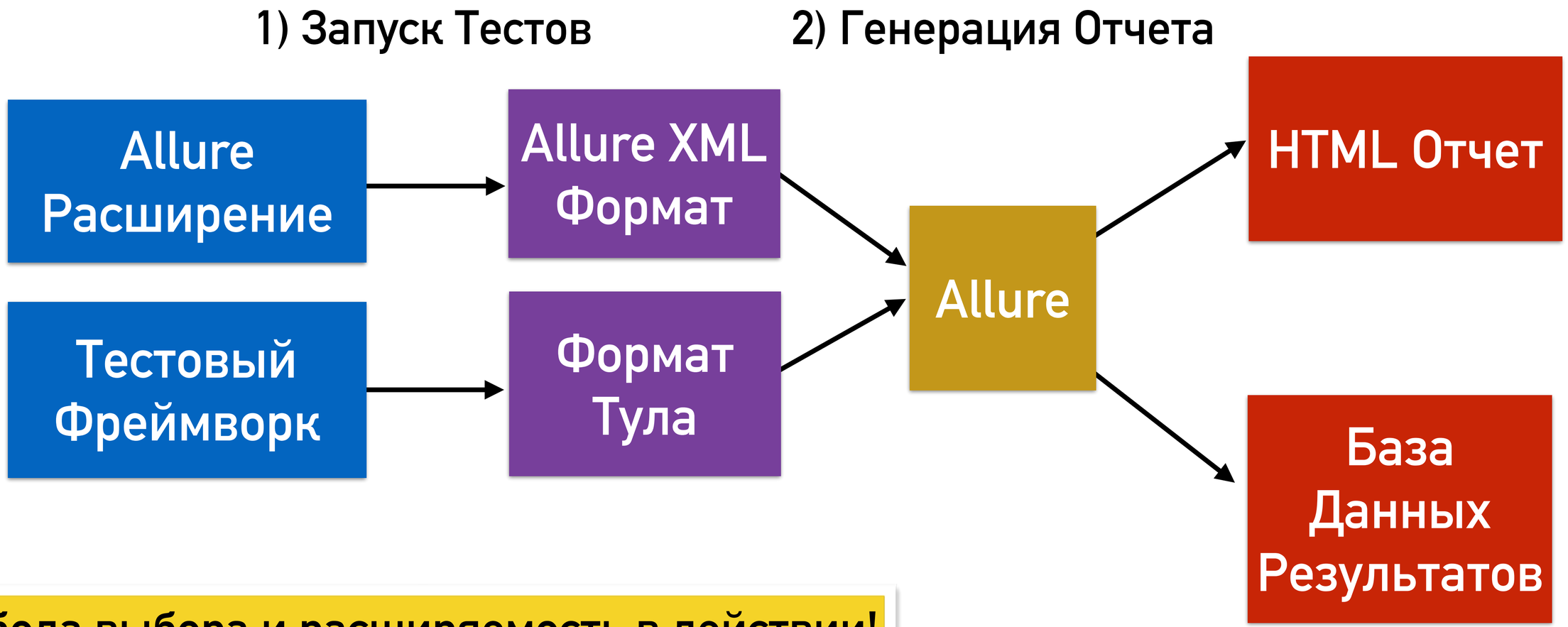
# Test Reporting

- Результаты тестов хотят видеть разные люди
- Как люди отвечающие за продукт, так и технари
- Хорошая отчетность должна помогать обоим категориям специалистов!

# Allure Framework 2.0

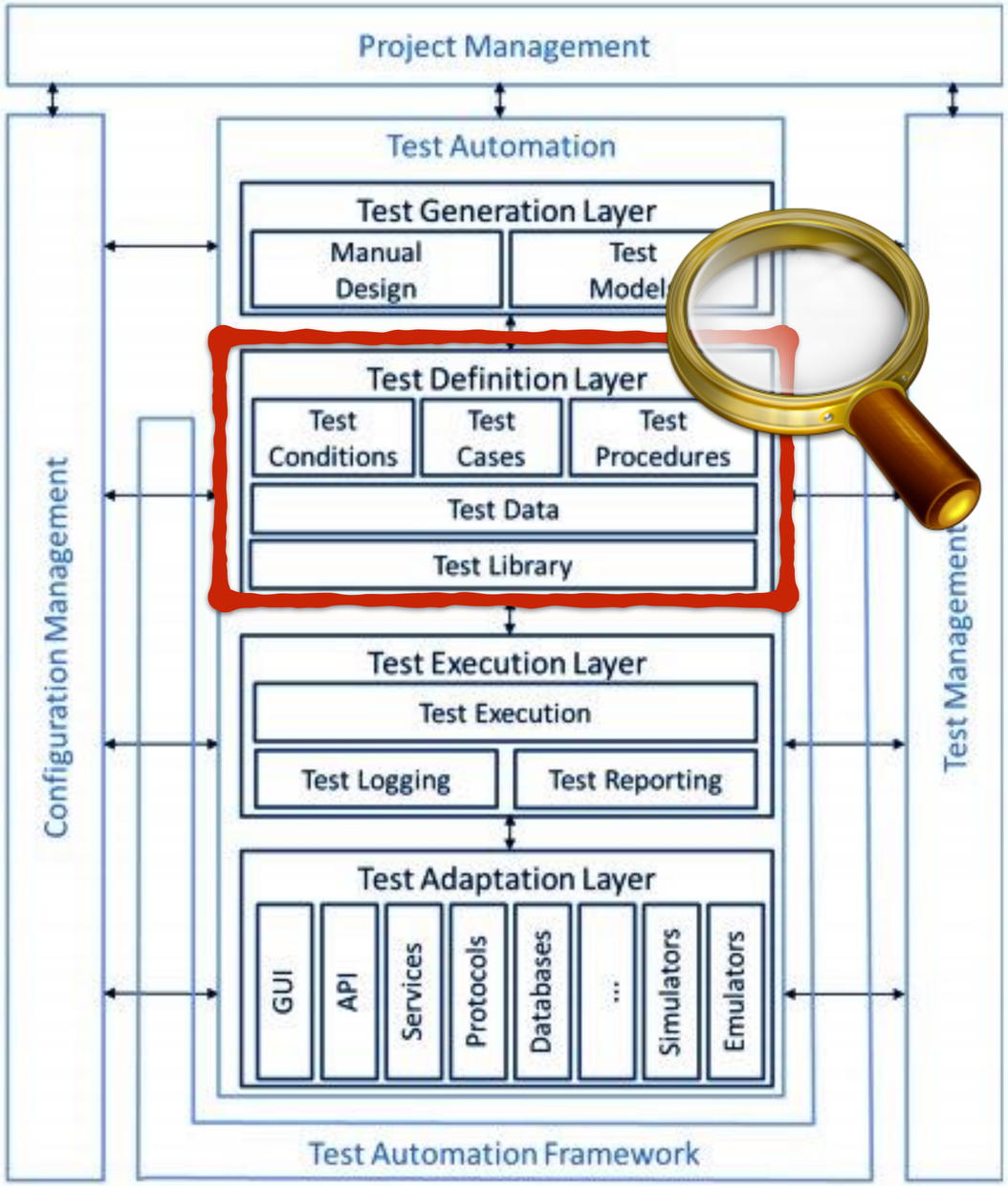

Хочешь отчеты?
Спроси меня как!

# Архитектура Allure




Свобода выбора и расширяемость в действии!

# Test Definition Layer




Test Library
Test Procedures
Test Cases
Test Conditions
Test Data

# Test Library

- Тестовая Библиотека - набор переиспользуемых средств для тестирования
- Дает АПИ к тестируемым функциям системы

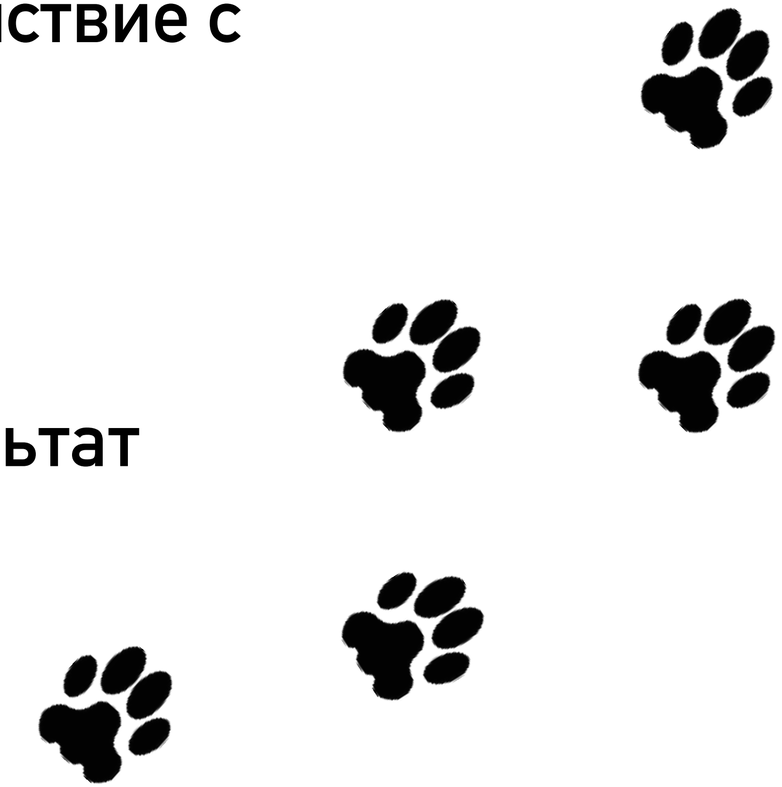
# Test Step

Тестовая Библиотека состоит из Тестовых Шагов

```
@Step
public void login(String username, String password) {
    // performs login
}

@Step
public void navigateToPage(String page) {
  // navigates browser to specified page
}
```

# Правильные Шаги

- Тестовый шаг - одно целостное действие с точки зрения пользователя
- Шаги не зависят друг от друга
- Имеют входные параметры и результат



# Test Conditions

Test Conditions == Assertions

Тестовые утверждения должны быть

- Простыми и читаемыми
- Выдавать понятные сообщения об ошибке
- Расширяемыми

# AssertJ Fluent Assertions

```
assertThat(list).contains("element");
assertThat(list).hasSize(9);
assertThat(list).containsExactlyInAnyOrder("one", "two", "three");
```

Эквивалентно

```
assertThat(list)
  .contains("element")
  .hasSize(9)
  .containsExactlyInAnyOrder("one", "two", “three");
```

**Одна из лучших библиотек утверждений для Java**

http://joel-costigliola.github.io/assertj/

# Custom Assertions

```
class CustomTaskAssert extends AbstractAssert<CustomTaskAssert, Task> {

  public static CustomTaskAssert assertThat(Task actual) {
    return new CustomTaskAssert(actual);
  }

  public CustomTaskAssert hasValidDuration() {
    if (actual.startTime > actual.endTime) {
      failWithMessage("Task %s can not finish (%s) before started (%s)",
          actual.name, actual.startTime, actual.endTime);
    }
    return this;
  }
}
```

Пишем свои

```
assertThat(task).hasValidDuration();
```

# Test Case

- Тестирует конкретный набор требований
- Содержит в себе
  - Условия исполнения (Preconditions)
  - Условия окончания исполнения (Postconditions)
  - Утверждения (Assertions)
  - Тестовые данные
  - Запуск тестовых шагов

# Описание Тестов в Ручном Тестировании

| No | Test Step | Expectation | Data |
|---|---|---|---|
| 1 | **User logs in** | Navigated to home page | username=“joe” |
| 2 | **User selects item** | Navigated to item page | item=“123” |
| 3 | **User clicks “Add to Cart”** | Shopping cart updated | |
| 4 | **User clicks “Go To Cart”** | Navigated to shopping cart | |
| 5 | **User clicks “Checkout”** | Checkout screen opens | |

**Легко читать**
**Можно дать коллегам, они разберутся**

# Некоторые Автоматические Тесты

```
namespace ExportToSushi.MainTests
{
    public partial class MyTests
    {
        [Test]
        public void TheExportedT
        {
            selenium.WaitForTex
            Verify.AreEqual("Home",
            Verify.AreEqual("SQL", selen
            Verify.AreEqual(".NET", selenium.
            Verify.IsTrue(selenium.IsElementPrese
            Verify.IsTrue(selenium.IsElementPresent(
            Verify.IsTrue(selenium.IsElementPresent("/
            Verify.IsTrue(selenium.IsElementPresent("c
            Verify.AreEqual("images/interface/aservice
            Verify.AreEqual("images/interface/aservice
            Verify.IsTrue(Regex.IsMatch(selenium.GetTe
            Verify.IsTrue(Regex.IsMatch(selenium.GetTe
            Verify.IsTrue(Regex.IsMatch(selenium.GetTe
            Verify.IsTrue(Regex.IsMatch(selenium.GetTe
            selenium.Click("//img[@alt='A service from
            selenium.WaitForPageToLoad("30000");
            Verify.IsTrue(selenium.IsElementPresent("/
            Verify.IsTrue(selenium.IsElementPresent("searchBox"));
        }
    }
}
```

```
import java.io.IOException;

public class AllInOne {

    public static void main(String[] args) throws IOException {

        System.out.println("====Running all-in-one====");
        Driver driver = new FirefoxDriver();
        get("http://www.joecolantonio.com/HpSupport.html");
        println("====Perform Assertion====");
        als("Joe's HP Support Matrix Tool for QTP, LoadRunner, "
        Test and Quality Center",driver.getTitle());
        Enter to continue====");
        Syste
        WebElemen
        tools.sendKey
        ement(By.id("tools"));

        WebElement version =
        version.sendKeys("BROWSER
        System.out.println("====Veri
        id("areas"));
        selecting Tool and Version====");

        System.out.println("====Test Done====
    }
```

А где же структура и все такое???

# Нормальный Test Case

```
@Test
public void checkoutTest() {
  login(“joe”, “password”);
  assertOnPage(“/home”);

  viewItem(123);
  assertOnPage(“/item.123”);

  addToCart();
  assertCartItems(1);

  goToCart();
  …
}
```

Test Case содержит:

- Тестовые шаги
- Утверждения
- Тестовые данные

# Test Procedures

Тестовая Процедура описывает какие тесты и в каком порядке запускать

Также может задавать дополнительные свойства:

- Уровень параллелизма исполнения
- Тестовые данные
- Настройки окружения

Test Procedure == Test Suite

# TestNG Test Suite

Рассмотрим на примере

```
<?xml version="1.0" encoding="UTF-8"?>
<suite name="Suite" parallel="false">
  <test name="Test" preserve-order="true">
    <classes>
      <class name="com.seleniummaster.testsuite.UserSettingTest">
      <methods>
          <include name="launchSiteAndLogin" />
          <include name="openUserSettingPage" />
          <include name="ChangeUserSettings" />
          <include name="Logout" />
          </methods>
      </class>
     </classes>
  </test> <!-- Test -->
</suite> <!-- Suite -->
```

# Тот Самый Класс

```
class UserSettingTest {
 Browser browser;
 @Test
 public void launchSiteAndLogin() {
  this.browser = openBrowser(“http://site.com”)
 }

 @Test
 public void openUserSettingPage() {
  this.browser.click(“#settings”);
 }

 @Test
 public void changeUserSettings() {
  this.browser.input(“#option”);
 }
}
```

Каждый “тест” оставляет после себя состояние

Последующий “тест“ ожидает, что броузер уже будет находится на нужной странице

# В Чем Проблема?

- Шаги завязаны друг на друга
- Что если тестовая логика усложнится, будем программировать на XML?
- Как это поддерживать?

# Подрыв Устоев

TestNG Test Suite

TestNG Test Case

Test Suites

Test Cases

Test Steps

Нецелевое использование фреймворка!

# Почему так Произошло?

- TestNG слишком мягок с тобой
- TestNG не предоставляет API для шагов
- Тестер не знал теории

# От порядка тестов, или количества потоков в тестовой процедуре результат должен оставаться неизменным!

Одно из самых главных правил!

# Абстрактные и Конкретные Тесты

- Абстрактные тесты определяют шаблон сценария
- Конкретные совмещают шаблон с тестовыми данными
- Комбинация - шаблон + данные это независимый тест

# Пример Тестового Шаблона

```
@TestTemplate
public void abstractTest(Data data) {
}
```

Фреймворк должен поддерживать тестовые шаблоны!

Если у всех запусков теста с данными один идентификатор - это неправильно!
Передаем привет TestNG!

# Test Data

- Тестовые данные определяют входные параметры для шаблонов тестовых сценариев
- Тестовые данные могут приходить в разных форматах и из разных источников

# JUnit 5 Загрузка из CSV Файла

```
@ParameterizedTest
@CsvFileSource(resources = "/data/users.csv")
public void simpleTest(String user, String password, Integer id) {
    logger.info("{}:{}:{}", user, password, id);
}
```

| | Standard | Standard | Standard |
|---|---|---|---|
| 1 | user1 | password | 123 |
| 2 | user2 | password | 432 |

# В CSV Решении JUnit 5 не хватает

- Подгрузки значений по заголовку колонок
- Комбинации многих файлов данных (перемножения)
- Достаточного количества конверторов
- Работы с типизированными классами данных
- Шаблонизирования значений ячеек

# JUnit 5 Custom Data Provider

Можно легко реализовать свой провайдер данных и нехватающие функции

```
@ParameterizedTest
@MethodSource("custom")
public void simpleTest(String username, String password, Integer id) {
  logger.info("{}:{}:{}", user, password, id);
}

static Object[][] custom() {
  return new Object[][]{{"user", "password", 1}};
}
```

И это не единственный АПИ

# “Другие" Данные

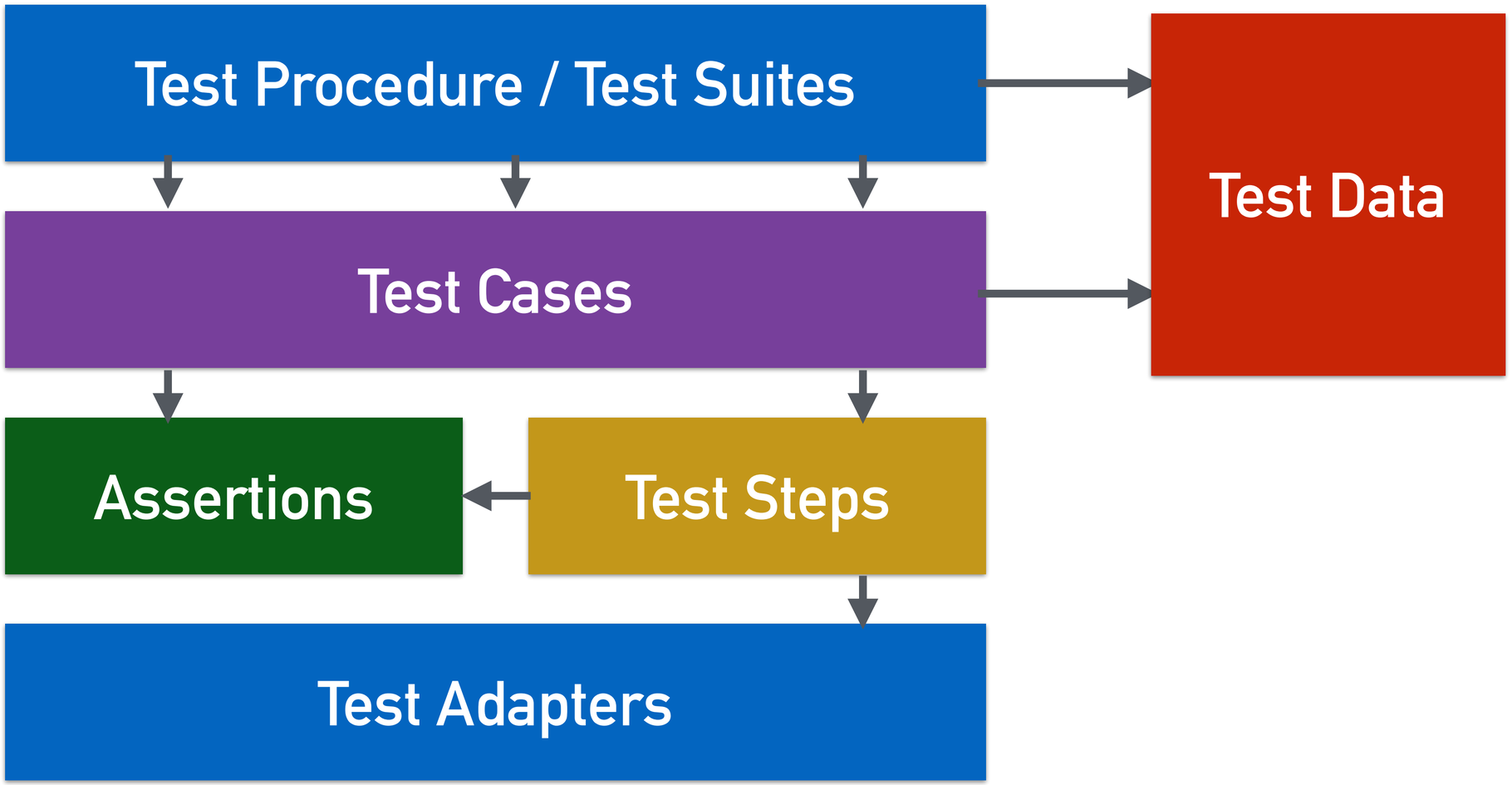
- Генерация данных на ходу в случае долгоиграющего, либо комбинаторного теста
- Получение данных из внешней системы
  - Синхронизация с эмулятором
  - Распределение данных между параллельными тестами


- Нет хорошей отчетности
- Отсутствует нормальная работа с данными
- Но это не значит, что он плохой…
- Он дает вам возможность написать свои расширения!
- И не мешает жить так, как хочется, как делают другие популярные фреймворки

Он просто написан правильно с точки зрения структуры!

# Блоки Построения Тестов

Годится для большинства фреймворков



# Пример

```
@Test
public void saveAccountLedger() {
  AccountLedger ac = new AccountLedger(“1234391”, …);
  String accountLedgerId = myTestSteps.saveAccountLedger(ac);
  assertThat(accountLedgerId).IsNotNull();
  myTestSteps.validateAccountLedgerExists(accountLedgerId);
}
```

**Тестовый шаг это посредник между тестом и Page Object**

```
class MyTestSteps {
 @Step
 public String saveAccountLedger(AccountLedger al) {
  LedgerPageObject page = navigator.goToPage(“/ledgers”)
  page.saveNewRecord(al);
  String notification = page.checkNotification();
  return extractAcountLedgerIdFromMessage(notification);
 }
}
```

# Есть еще кое-что в тестовом фреймворке!

# Язык Тестов

Можно писать тесты на

- Gherkin
- Java/Groovy/Kotlin
- Excel
- HTML
- Markdown

# А разве фреймворк не тянет за собой язык?

# Тест на JUnit 5

```
@Test
public void ensureAllFieldsAreDisabledForUser() {
  goToPage(“/login“);
  login(“janedoe”, “password”);
  assertPage(“/home”);
  logout();
}
```

# Язык Тестов Gherkin

```
1  Feature: Login Page
2
3      As a user I want to be able to login and out of the system.
4
5      Background:
6          Given I am at the login page
7
8      Scenario: Login as a user
9          When I login with user 'janedoe' and password 'password'
           Then I should see the home page
           And I can logout
```

Test Step

Test Step

Test Case

Test Conditions

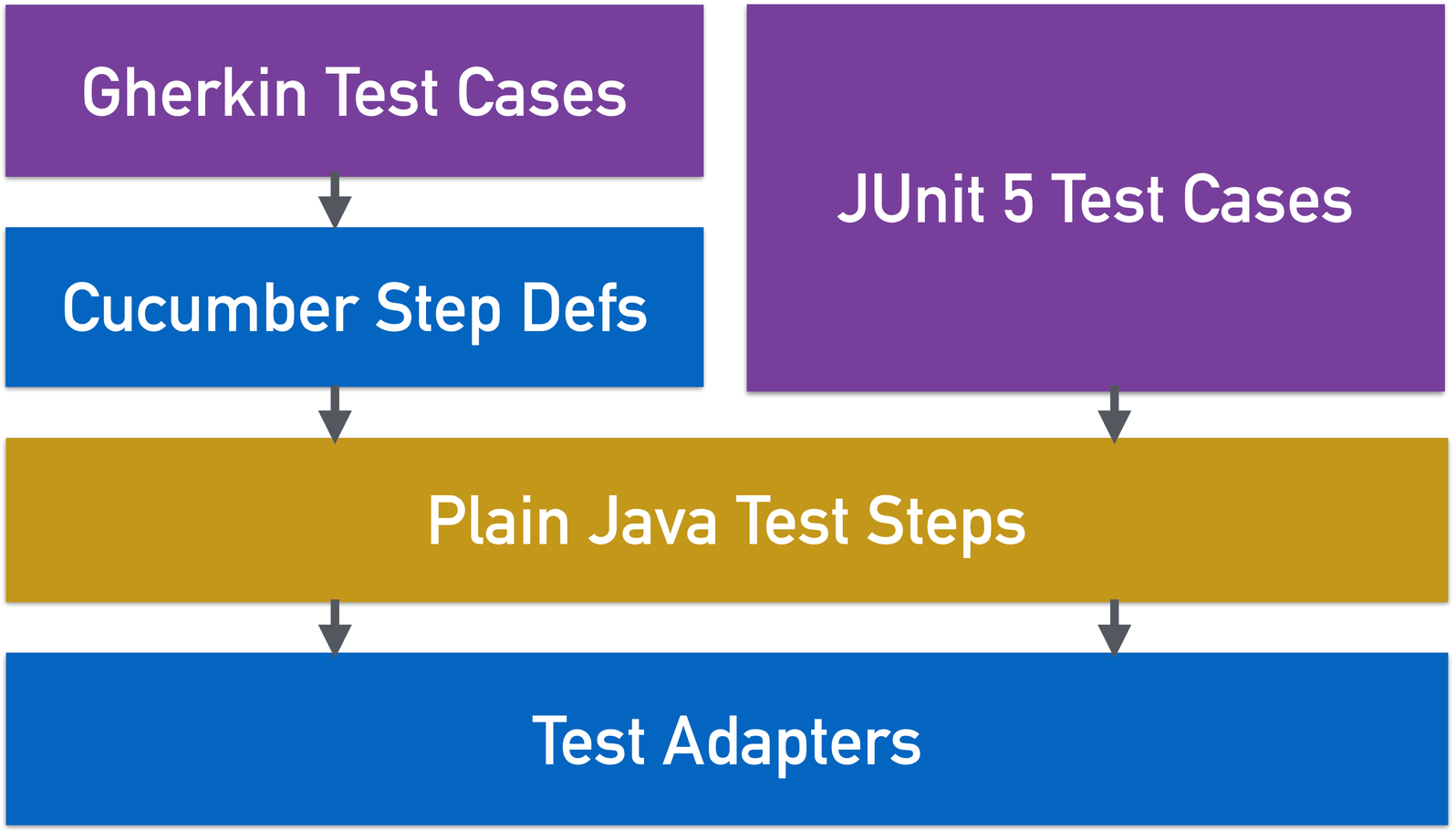
Эквивалентен предыдущему

# Описание Шага в Cucumber-Java

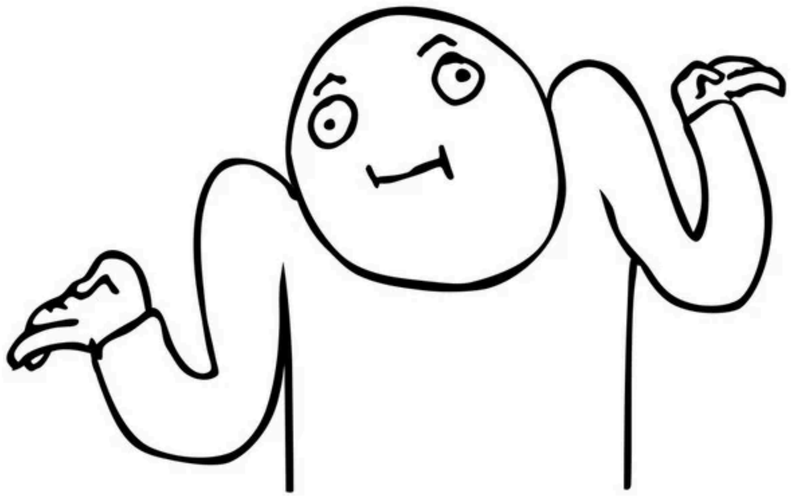
```
@Then(“^I login with user (.*) and password (.*)$")
public void login(String username, String password) {
  myTestSteps.login(username, password);
}
```

Переиспользуем те-же тестовые шаги и адаптеры

# Многоязычный Фреймворк

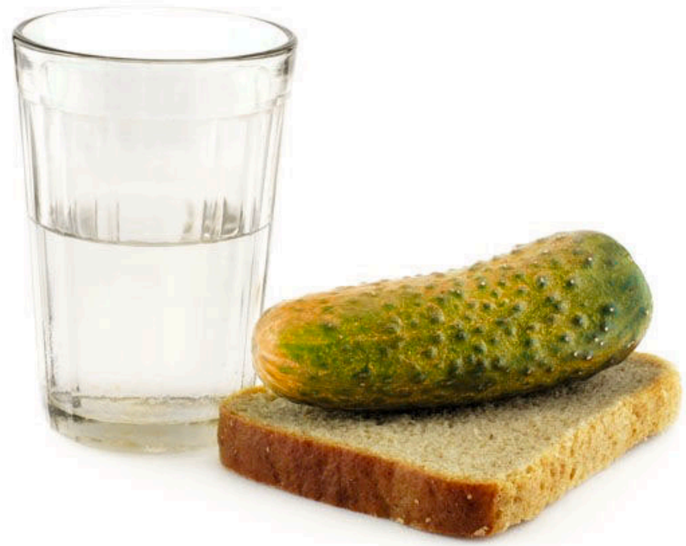


# А что, нельзя определиться?



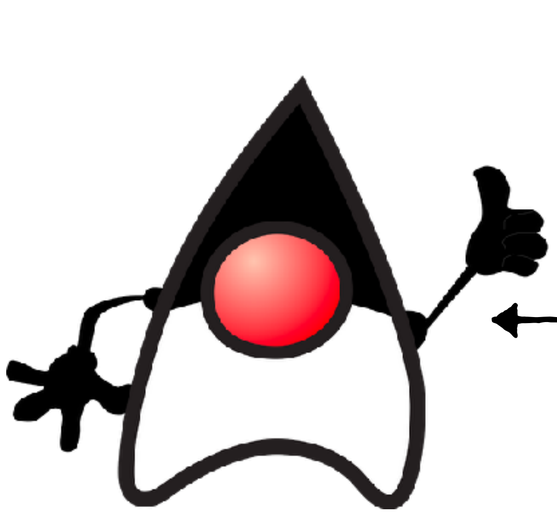
# “Огурец” – не диагноз!

| Плюсы | Минусы |
| --- | --- |
| Наглядно читаем | Стабилизация языка занимает время |
| Прост в освоении | Сложности с рефакторингом |
| Легко составлять новые тесты по примеру | Нетривиальные сценарии не ложатся на BDD модель |



Если ядро фреймворка
независимое от языка,
мы не ставим все на “красное”

# Простая Стратегия

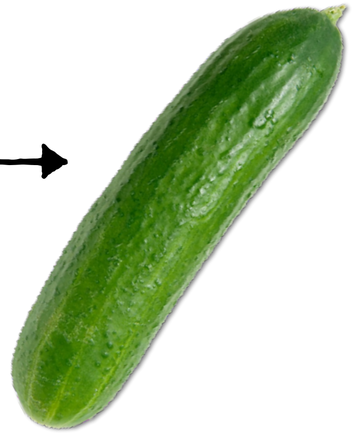


Java

Да!

Тест
Структурно
Сложный?

Не думаю!



Cucumber

# Что Такое “Сложный” Тест?

- Многопоточность
- Обработка событий
- Сложные манипуляции с данными

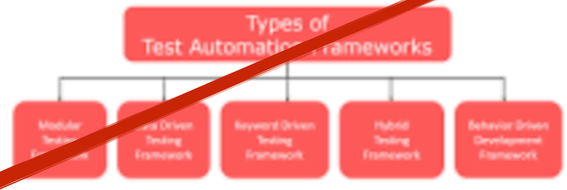
# Переходим к Выводам

# Какие бывают фреймворки?

**Types of Test Automation Frameworks:**

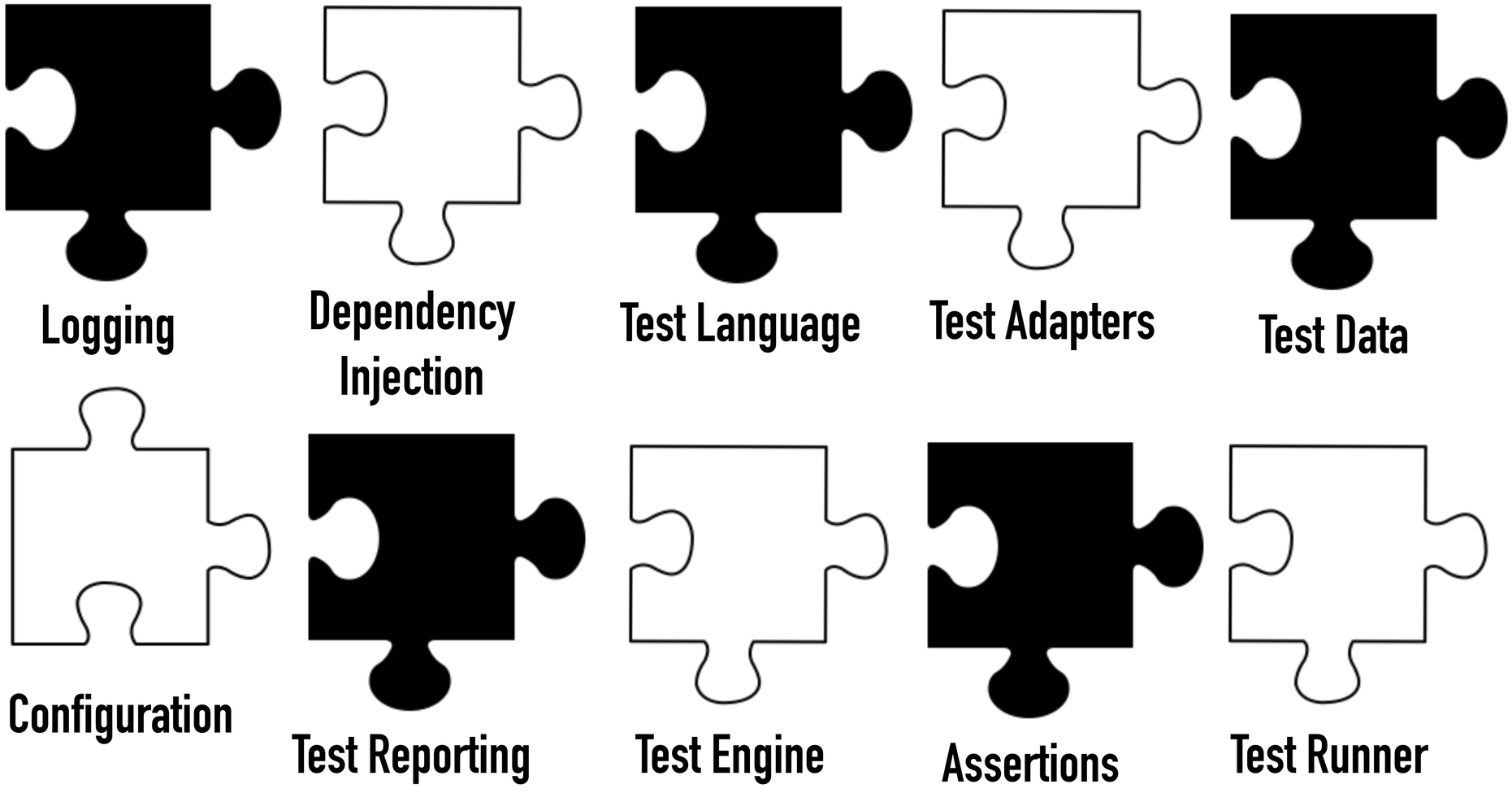
- Linear Scripting Framework.
- Modular Testing Framework.
- Data Driven Testing Framework.
- Keyword Driven Testing Framework.
- Hybrid Testing Framework.
- Behavior Driven Development Framework.

Types of Test Automation Frameworks | Software Testing Material
www.softwaretestingmaterial.com/types-test-automation-frameworks/




Это все просто компоненты, которые могут быть совмещены в рамках архитектуры

# Собираем все вместе



# Успешные фреймворки всегда собираются из готовых ингредиентов

# Мой Последний Java Рецепт

| Компонент | Технология |
|---|---|
| Test Engine | JUnit 5 + extensions |
| Test Reporting | Allure 2 + extensions |
| Assertions | AssertJ + extensions |
| Configurations | OWNER |
| Test Language | Cucumber-JVM, Java |
| Logging | Logback + ELK |
| Test Adapters | HttpClient, Selenide, Wiremock |
| Dependency Injection | Guice |
| Data-Driven | JUnit 5 + extensions |

# Именно правильная структура определяет “долголетие” и “крутость” фреймворка

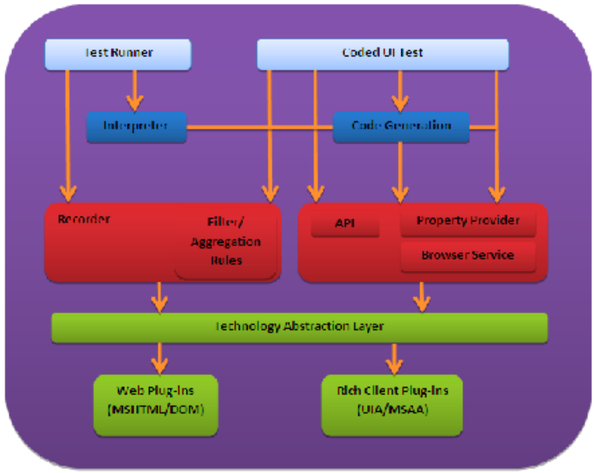
Хоть раз в жизни соберите свой фреймворк из готовых кусочков!

# Придерживайтесь Стандартов

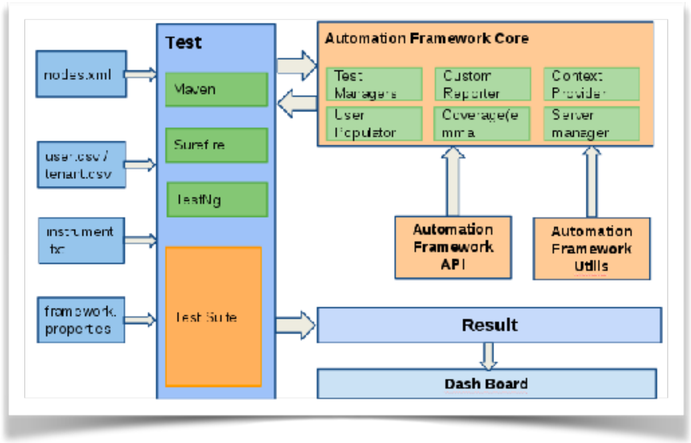
- Язык Gherkin https://cucumber.io/docs/reference#gherkin
- Протокол WebDriver https://www.w3.org/TR/webdriver/
- Платформа JUnit 5 http://junit.org/junit5/
- Формат отчетов Allure https://docs.qameta.io/allure/latest/

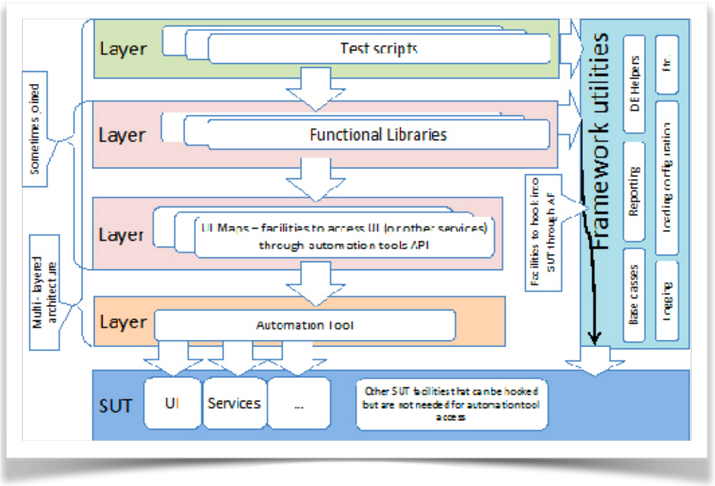
# Отдавайте предпочтение легко расширяемым инструментам!

# Посмотрите как делают другие



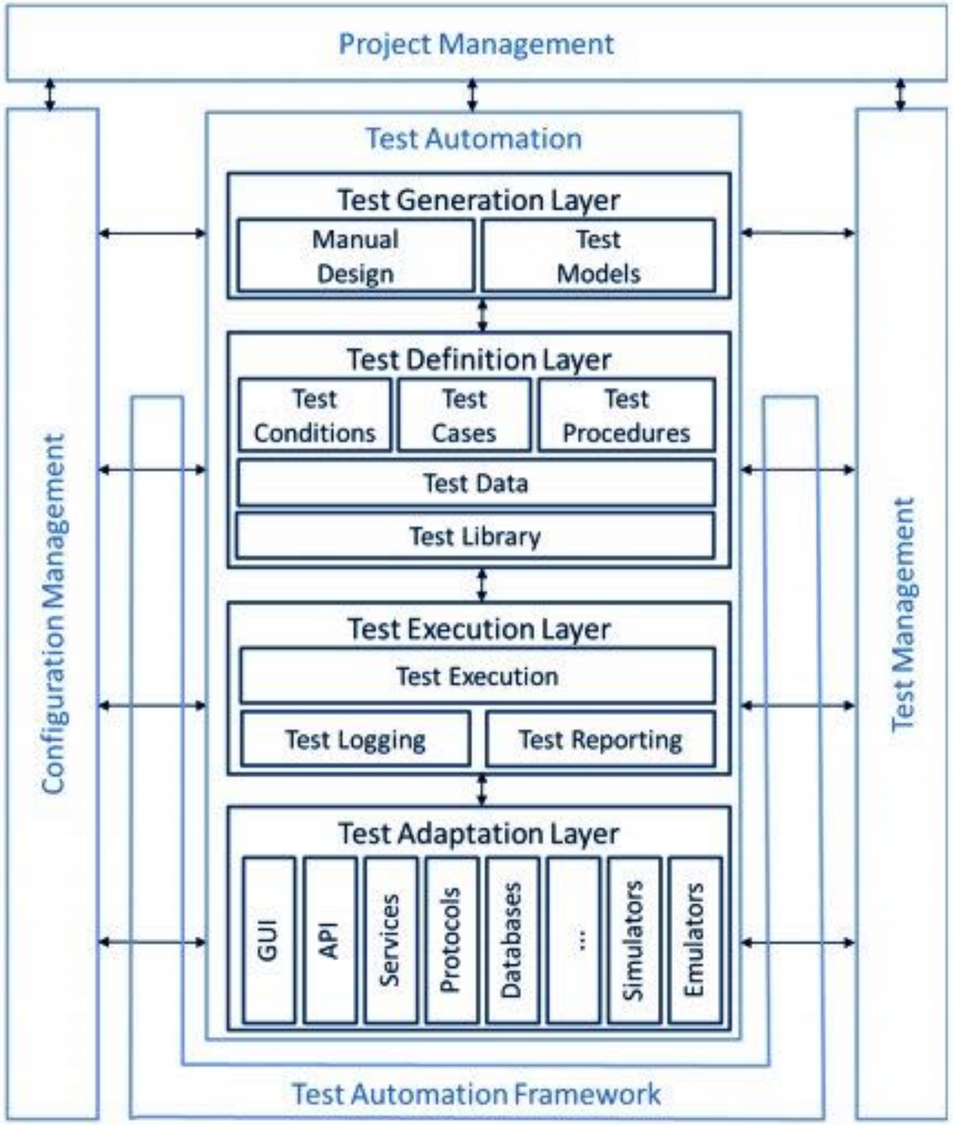
CUIT FRAMEWORK ARCHITECTURE



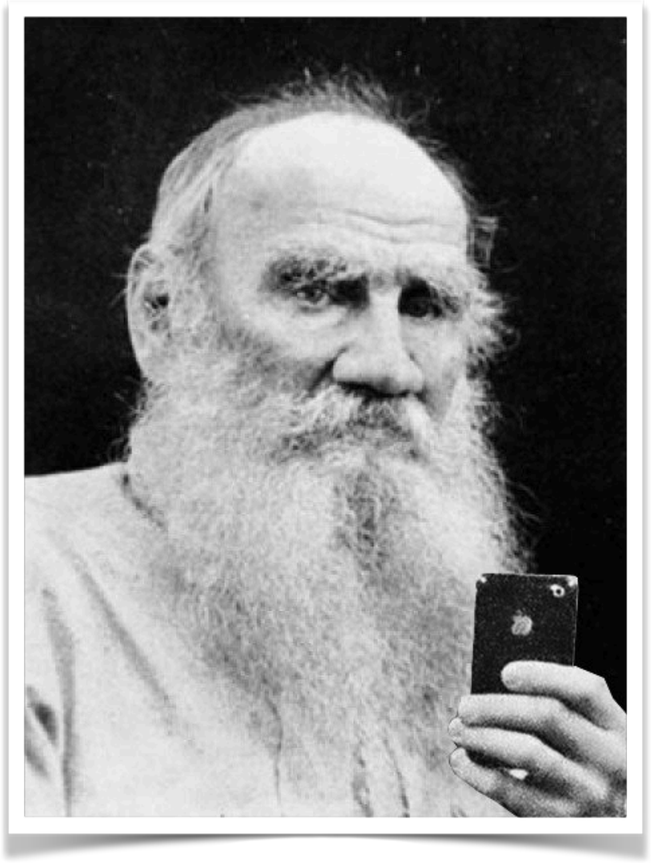

- Посмотрите на нишевые фреймворки
- Статьи за 2005 год не потеряли актуальность!

# Как показывает практика – практика все-же базируется на теории!




https://www.istqb.org/downloads/category/48-advanced-level-test-automation-engineer-documents.html

# Перефразируя классика



«Все успешные фреймворки похожи друг на друга, каждый неудачный фреймворк несчастлив по-своему.»

# Подключаем Адаптер к Фреймворку