6 TRAITS OF A SUCCESSFUL TEST AUTOMATION ARCHITECTURE

Erdem YILDIRIM
About Me

2000 – Software Developer
2006 – Discovered Test Automation & Decided to be a Test Engineer
2010 – Test Lead at STM (Defence Technologies and Engineering)
2013 – Test Lead at Innova (12 PRJs, 20 TEs)

https://www.linkedin.com/in/yildirim-erdem
AGENDA

- When/What to Automate?
- Design Principles/Patterns
- Locator Strategy
- Methodology
- Framework/Language Selection
- OOP Principles
Netflix

5500 unit test, 500 E2E test scenarios are automated

Problem
flaky tests

Effect
give up test automation

Solution
stability pipeline testing ...
What is The Point?

EFFICIENCY
EXPECTED
REPETITIVE
INTELLIGENCE
AUTOMATION

Expected Action

Machines are great for running straightforward tests

Gives Confidence

COMPLEMENTARY

EXPLORATORY

Unexpected Scenario

Humans are great for tests requiring intelligence and experience

Finds new defects
WHAT IS YOUR AUTOMATION ARCHITECTURE SCREAMING
the greatest enemy of knowledge is not ignorance, it is illusion of knowledge

Stephen Hawking
PREVENT/SOLVE AUTOMATION ISSUES

HIGH MAINTENANCE COST

LACK OF TEST AUTOMATION EXPERIENCE

STALLED AUTOMATION

UNREALISTIC EXPECTATIONS
Which Test Levels To Be Automated
Deciding Automation Levels Scenario Depth

Horizontal E2E (UI & Integration Layer) Login / Add to Chart / Checkout

Vertical E2E Implementing tests on all level of the pyramid

Exploratory Test
UI Tests
Integration Tests
Unit Tests
Test Data Preparation

CI – Continuous Integration
2Ws of A Test Automation

OK, that's an interesting story.
## Factors To Decide When & What To Automate

<table>
<thead>
<tr>
<th>Category</th>
<th>Factor</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SUT-related factors</strong></td>
<td>Maturity of SUT</td>
<td>39</td>
</tr>
<tr>
<td></td>
<td>Other SUT aspects</td>
<td>6</td>
</tr>
<tr>
<td><strong>Test-related factors</strong></td>
<td>Need for regression testing</td>
<td>44</td>
</tr>
<tr>
<td></td>
<td>Test type</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>Test reuse/repeatability</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>Test importance</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>Test oracle</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>Test stability</td>
<td>7</td>
</tr>
<tr>
<td><strong>Test-tool-related factors</strong></td>
<td>Automation (test) tool</td>
<td>8</td>
</tr>
<tr>
<td><strong>Human and organizational factors</strong></td>
<td>Skills level of testers</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>Other hum. and org. factors</td>
<td>11</td>
</tr>
<tr>
<td><strong>Cross-cutting and other factors</strong></td>
<td>Economic factors</td>
<td>43</td>
</tr>
<tr>
<td></td>
<td>Automatability of testing</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>Development process</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>Other factors</td>
<td>25</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td>344</td>
</tr>
</tbody>
</table>
SUSTAINABLE automation solutions
DESIGN

2nd trait
KISS
Keep It Simple Stupid

YAGNI
You Aren’t Going to Need It

DRY
Don’t Repeat Yourself
Layers in Your Automation Code

- Tests
- Selenium/GEB/etc.
- Browser (Web App)
The Framework should also be divided up into layers of functionality.
Page Object Model (POM)

Helps to hide the page implementation; the test is no longer allowed to directly interact with a given page, but instead uses a framework of classes and methods to accomplish the same goal. This pattern abstracts the business logic/implementation details.

Advantages
• Busines Point of View
• Testing Behavior
• DRY
• Modular & Reusable
• Clear Intentions

Disadvantages
• Complexity (Wrong Usage)
public class LoginSpec extends PageSpec {
    def "Login tests with user information"() {
        given: "Home Page"
        def loginPage = to LoginPage

        when: "user information is entered"
        loginPage.loginOl(userName, password)

        then: "login to the system, open main page"
            at resultPage

            where:
                userName | password | resultPage     | aciklama
                "buro1@innova.com.tr" | "123456" | MemberHomePage | "positive-test"
                "wrong@mail.com"  | "123"    | LoginPage      | "negative-test"
                ""           | ""       | LoginPage      | "exceptional-test"
    }
}
Simplicity is the ultimate sophistication

Leonardo da Vinci
3rd trait

LOCATOR STRATEGY
LOCATORS

Your locator strategy will determine your tests’ destiny as flaky or robust.

Good: If you know the right strategy you can apply it.

Bad: It is also dependent to AUT code/developers not only to you.

Critical: Communication & Collaboration, mentoring dev teams.
CSS selectors
ID
Xpath
Custom Locators
Class
Name
Link Text
Partial Link Text
Tag Name
Dynamic Locators
LOCATORS
in order

ID
Custom Locators
Class
Name
CSS selectors
Xpath
Link Text
Partial Link Text
Tag Name
Dynamic Locators
Methodology
E2E Test Automation

High business logic coverage,
Testing from the user perspective,
Prevents production incidents,
Gives the confidence that everything is OK

Low code coverage,
Slow execution,

Eg: Login / Add to Chart / Checkout / Pay
Continuous Testing

To develop software within shorter delivery cycles; Agile, DevOps and Continuous Delivery approaches are on the forefront.

Continuous Delivery supports Continuous Testing which is a testing strategy that consists of a large number of automated unit and acceptance tests but a small number of automated end-to-end tests.

Continuous Testing claims that E2E tests are inefficient and they should be limited.

It is a means of using Vertical E2E Test Automation effectively but avoiding Horizontal E2E Test Automation
TDD – Test Driven Development

START HERE

WRITE A TEST

TEST FAILS

WRITE CODE TO MAKE TEST PASS

CODE PASSES TEST

CLEAN CODE

WORKABLE CODE HOW TO IMPROVE?

REFACTOR

RED

GREEN

Test Driven Development
WHY TDD

Simplest way to achieve both good quality code and good test coverage.
BDD – Behaviour Driven Development

Developing unambiguous features in collaboration with BA, QA, Dev

Writing features’ definition and also the behaviours driving the design process.

Preventing defects rather than finding defects at early stages.
FORM YOUR CONTINUOUS DELIVERY STRATEGY
BEST TOOL

The Illusion of Using one Test Automation Tool for All Projects

NO SUCH TOOL

Depends on the project dynamics
CONSIDER
Technologies used in SUT development code, 3rd party components

LOOK FOR
Efficiency
(produce more with less effort)
Just a Few Examples on Personel Experience

GEB, Spock, Groovy What bring us?
the power of Selenium WebDriver,
the elegance of jQuery content selection,
the robustness of Page Object modelling,
the expressiveness of the Groovy language,
the velocity of Spock style BDD
CODECEPTJS

**Backend agnostic**; the power of Selenium WebDriver, webdriverio, protractor, nightmarejs, appium, puppitateer Facilitated one language for all above frameworks, Scenario driven highly readable,

Built in **POM** support,

Support for both Spock style BDD and Cucumber **BDD Fast** to develop and learn.
CUCUMBER
Readable BDD for whole stakeholders,
Ideal for automating backend, middleware applications.
Increase the communication / collaboration in team.
Resolve the issue of developing the wrong/misunderstood requirement.
public class LoginSpec extends PageSpec {
    def "Login tests with user information"() {
        given: "Home Page"
        def loginPage = to LoginPage

        when: "user information is entered"
        loginPage.loginOl(userName, password)

        then: "login to the system, open main page"
        at resultPage

        where:
        userName | password | resultPage | aciklama
        "burol@innova.com.tr" | "123456" | MemberHomePage | "positive-test"
        "wrong@mail.com" | "123" | LoginPage | "negative-test"
        "" | "" | LoginPage | "exceptional-test"
    }
}

All (BDD, automation code) in one file;
very concise/readable code,
very fast and fun to code
Cucumber BDD

Feature: Login functionality
As a customer
In order to use the application
I want to login with email and password

Scenario: Logging in with invalid credentials
Given I am at the Account/Login page
When I fill the account email textbox with value 'incorrect@email.com'
And I fill the password textbox with value 'incorrectpassword'
And I click the login button
Then a text 'Can't login! Wrong email or password.' should appear in the validation error

Scenario: Logging in with valid credentials
Given I am at the Account/Login page
When I fill the account email textbox with value 'myname@myemail.com'
And I fill the password textbox with value 'mypassword'
And I click the login button
Then I should be at the home page

public class StepDefinitions
{
    [Given("I am at the (.+) page")]
    public void GivenAtPage(string page)
    {
        WebBrowser.Url = WebBrowser.GetUrl(page);
    }

    [When("I fill the (.+) (.+) with value "'(\+)')")]
    public void WhenFillValue(string label, string role, string value)
    {
        WebBrowser.FindElement(label, role).SendKeys(value);
    }

    [When("I click the (.+) (.+)")]
    public void WhenPress(string label, string role)
    {
        WebBrowser.FindElement(label, role).Click();
    }

    [Then("a text '(\+) should appear in the (.+) (.+)" )]
    public void ThenTextShouldAppear(string text, string label, string role)
    {
        Assert.AreEqual(
            text,
            WebBrowser.FindElement(label, role).Text);
    }

    [Then("I should be at the (.+) page")]
    public void ThenShouldBeAtPage(string expectedPage)
    {
        Assert.AreEqual(
            WebBrowser.GetUrl(expectedPage),
            WebBrowser.Url);
    }

    [AfterScenario]
    public void Destroy()
    {
        WebBrowser.Current.Quit();
    }
}

Two separate files; one for Scenarios, one for code readable for non-developers
OOP Principles

6th Trait
4 PILLARS OF OOP

Examples on How to Use Them in Automation Code

Encapsulation
Inheritance
Polymorphism
Abstraction
ENCAPSULATION

Hiding data and binding that data to public methods that other classes can utilize to access to data to protect the data integrity and consistency. Accessing data from the getter/setter methods.

class Account
{
    private double balance;
    public setBalance(double balance)
    {
        if(balance > 0)     // Validating input data in order to maintain data integrity
            this.balance = balance;
    }
}

public static void main()
{
    Account theGeekyAsian = new Account(); //Creating an object of class Account
    theGeekyAsian.setBalance(1000); //Balance set successful
}
INHERITANCE

Extend data and behaviours. Inheritance is a way to reuse code. The class which is inherited from, is called the base class, and the class which inherits the code from the base class is called a derived class. A derived class can use all the functions which are defined in the base class, making the code reusable.

class Shape
{
    private int height;
    private int width;

    public void setWidth(int width)
    {
        this.width = width;
    }

    public void setHeight(int height)
    {
        this.height = height;
    }
}

class Rectangle extends Shape
{
    public int getArea()
    {
        return height * width;  //Rectangle's getArea method inherited the height and width from Shape
    }
}
POLYMORPHISM

The ability to take **multiple forms**. The name polymorphism defines itself. Poly means many, and morph means forms. It is a process of a method of a class taking different/many forms.

**Static Polymorphism – Known as Method Overloading**

**Dynamic Polymorphism – Known as Method Overriding**

class Numbers
{
    public int sum(int numberA, int numberB)
    {
        return numberA + numberB;
    }

    public double sum(double numberA, double numberB)  //Here, method overloading takes place,
    {
        return numberA + numberB;  //by providing another feature to the same method 'sum'
    }
}

class Test
{
    public static void main(String[] args)
    {
        Numbers objNumber = new Numbers();
        System.out.println("Sum of two integer numbers is: " + objNumber.sum(2,3));
        System.out.println("Sum of two double numbers is: " + objNumber.sum(2.5,3.2));
    }
}
ABSTRACTION

In abstraction, we provide only essential/necessary features (methods) of an entity to the outside world (the user) while hiding other irrelevant details, in order to reduce operational complexity at the user-end.

Add to Chart / Checkout / Pay

class Purchase
{
    public void addToChart(){}
    public void checkout(){}
    public void pay() //pay is accessible by the user to pay
    {
        /* Something happens here */
        what_happens_when_he_pay(someValue);
    }
    private void checkPaypal(){}
    private void checkCreditCard(){}

    private void what_happens_when_he_pays(String someValue)
    {
        /* Something happens here */
    }
    private void sendPaymentConfirmationMail(){}
}
References

Test Automation Patterns: Issues and Solutions, Dorothy Graham, Seretta Gamba


Testistanbul 2016 - Keynote: "Why Automated Verification Matters" by Kristian Karl


Selenium Design Patterns and Best Practices, Dima Kovalenko

Behavior Driven Development Webapps mit Groovy Spock und Geb

https://blog.testlodge.com/tdd-vs-bdd/

https://www.itexico.com/blog/software-development-kiss-yagni-dry-3-principles-to-simplify-your-life Tight

Khanh Le Common Design Principles and Design Patterns in Automation Testing

https://medium.freecodecamp.org/test-driven-development-what-it-is-and-what-it-is-not-43fa68ca02a2

https://blog.jbrains.ca/permalink/the-worlds-shortest-article-on-behavior-driven-development-revisited

https://blog.jbrains.ca/permalink/how-test-driven-development-works-and-more

Selenium Design Patterns, Liraz Shay

http://thegeekyasian.com/4-pillars-of-oop/

https://www.it.uu.se/edu/course/homepage/ood/ht12/overview/patterns/lecture6.pdf

https://alvinalexander.com/java/java-factory-pattern-example

https://mentornate.com/blog/bdd-using-web-accessibility-sauce-spectr flows-net-cucumber-examples/

http://www.diogonunes.com/blog/four-testing-quadrants/

https://www.youtube.com/watch?v=jpfm7sS72Y

Applying the Pillars of Object-Oriented Programming to Test Automation - with Angie Jones

https://www.youtube.com/watch?v=jpfm7sS72Y

Creating A Test Automation Framework Architecture With Selenium (Step-By-Step), John Sonmez, Bulldog Mindset.

Common design principles and design patterns in automation testing, KMSTechnolog
Image References

Slidemodel.com,
pexels.com
(Re)Building an engineering culture: DevOps at Target, Heather Mickman, Ross Clanton
https://tenor.com/view/peepshow-interesting-story-gif-4852643
Andreas von der Heydt 12 Minutes To Create A Mind Changing Presentation
Damon Nofar 5 Ways To Surprise Your Audience (and keep their attention)
QUESTIONS?

Erdem YILDIRIM

https://www.linkedin.com/in/yildirim-erdem