WILL YOUR TESTS STAND THE TEST OF TIME?
PATTERNS FOR AUTOMATED UI TESTS

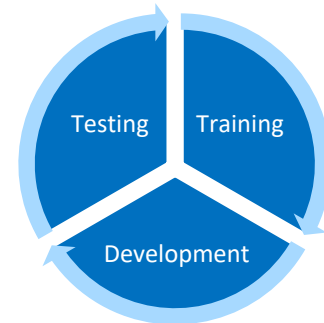ALEX SCHLADEBECK   @ALEX_SCHL
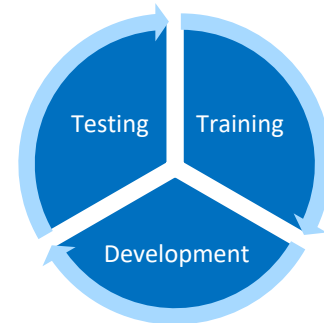
BREDEX

INDIVIDUELLE SOFTWARE

INTRODUCTIONS

- I am...
  - A tester, head of testing, a product owner
  - A geek ☺

- Bredex is
  - A software development company
  - With a strong focus on quality
    - Own open source tool
    - Multi-tool strategy
    - Agile testing
    - Scalable test automation

BREDEX

INTRODUCTIONS

- I am…
  - A tester, head of testing, a product owner
  - A geek ☺

- Bredex is … hiring ☺

TODAY'S JOURNEY

- The bad reputation of UI testing
  - Reasons and explanations
- Test structure for scalable UI tests
- Examples

BREDEX

Everyone hates UI tests

BREDEX

TO BOIL IT DOWN TO TWO POINTS…

- UI tests are often used to test things that shouldn't be tested via the UI

- And they are often written badly. Very badly…

Just because they're maybe a bit more difficult – and not the favourite domain of developers – doesn't mean that they are not potentially valid

BREDEX

</rant>

UI TESTING TOOLS ONLY THINK OF ONE THING...

Technically addressing UI components in an application

They give no structural support for test design

In fact, they help you to make a mess often...

BREDEX

"Don't use capture-replay. 'Tis the devil's work"

- Alex Schladebeck, repeatedly from ~2006 AD

BREDEX

WHAT USUALLY HAPPENS

Test Case

Missing technical and logical layers

Direct tool API calls to create workflow

input

click

check

Tool-API

- Redundancies
- Unreadable
- Not conducive to team work

BREDEX

THERE ARE WAYS OF AVOIDING THIS…

In software engineering, a software design **pattern** is a general reusable solution to a commonly occurring problem within a given context in software design.

It is not a finished design that can be transformed directly into source or machine code.

BREDEX

HOW TESTS SHOULD BE STRUCTURED

Test Case

Functional workflows

Functional atomic actions

Technical Keywords

Tool-API

BREDEX

HOW TESTS SHOULD BE STRUCTURED

Test Case

Functional workflows

Functional atomic actions

May be split into internals (private) and services (public)

Technical Keywords

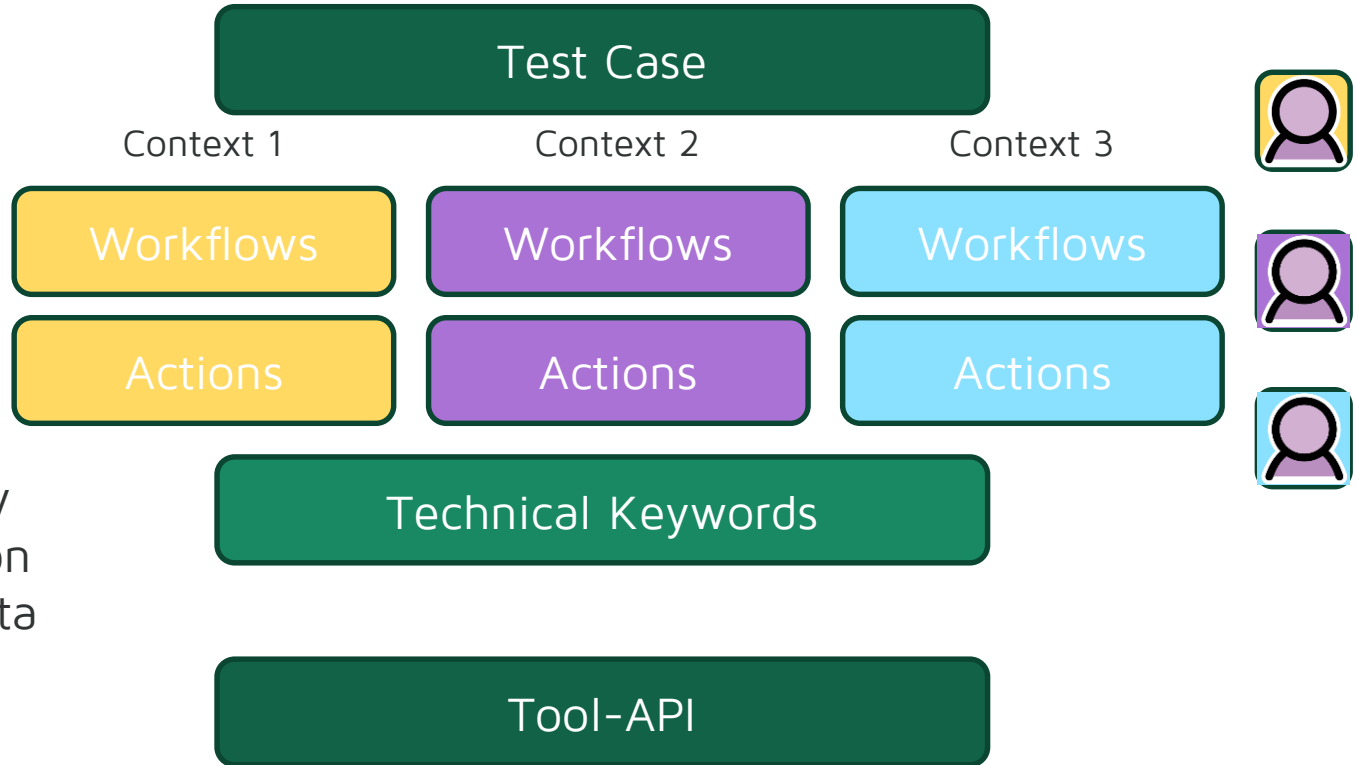Tool-API

BREDEX

HOW TESTS SHOULD BE STRUCTURED

EXAMPLE: GOOD LAYERING

Test Case	TC CUS 001: Create new customer - Check customer does not exist [CUSTOMER_KEY] - Create new customer [CUSTOMER_KEY] - Check customer exists [CUSTOMER_KEY]
Functional workflow	Create new customer [CUSTOMER_KEY] - Open new customer dialog - Fill out customer form [CUSTOMER_KEY] - Click Save
Functional workflow	Fill out customer form [CUSTOMER_KEY] - Enter customer name [NAME] - Enter customer street address [ADDRESS] - Enter postcode [POSTCODE]
Functional atomic action	Enter customer name [NAME] - Replace text [TEXT]
Technical keyword	Replace text [TEXT] - Click - Select all - Send keys

EXAMPLE: SPLITTING INTO CONTEXTS

Test Case

Context 1	Context 2	Context 3
Workflows	Workflows	Workflows
Actions	Actions	Actions

Technical Keywords

Context: functionally connected application area, e.g. Master Data

Tool-API

BREDEX

YOU MAY KNOW SIMILAR APPROACHES

- Page Object Pattern
  - Services and Implementation separated → more layers
  - Focus on page → focus on functional context
  - Assertions → checks
  - Strong user focus
- Gojko's 3 layers → more layers

BREDEX

https://github.com/SeleniumHQ/selenium/wiki/PageObjects
https://gojko.net/2010/04/13/how-to-implement-ui-testing-without-shooting-yourself-in-the-foot-2/

BAD EXAMPLE 1

```java
public void deleteCustomer() {
  driver.findElement(By.id("openCustomer")).click();
  WebElement customer = driver.findElement(By.xpath("//table[@id='table']/tbody"));
  List<WebElement> customerList = customer.findElements(By.tagName("tr"));
  for (WebElement actualCustomer : customerList) {
    if (actualCustomer.equals("MyCustomer")) {
      driver.findElement(By.id("delete")).click();
    }
  }
}
```

BREDEX

- No abstraction levels
- Redundant object location
- Technical operations at highest level
- Hard coded data

REAL LIFE BAD EXAMPLE 1

```java
public void login() {
    WebElement usernameTextField = driver.findElement(By.name("username"));
    WebElement passwordTextField = driver.findElement(By.name("password"));

    usernameTextField.sendKeys("...");
    passwordTextField.sendKeys("...");

    WebElement loginButton = driver.findElement(By.className("login-button"));
    loginButton.click();

    WebElement uploadLink = driver.findElement(By.linkText("Datei-Upload"));
    uploadLink.click();

    WebElement fileInput = driver.findElement(By.name("UPLOAD_FILENAME"));
    fileInput.sendKeys("...");

    WebElement uploadButton = driver.findElement(By.name("UPLOAD_BUTTON"));
    uploadButton.click();
}
```

BREDEX

LAYERING EXAMPLES

```java
@Test
public void CUST_TC_01_01_03__CreateCustomer() throws InterruptedException {
  CustomerAdministration.checkCustomerExists("My", "Customer", "Street", "9", "38100", "Brunswick", true, false, false);
  Thread.sleep(5000);
  CustomerAdministration.createCustomer("My", "Customer", "Street", "9", "38100", "Brunswick");
  Thread.sleep(5000);
  CustomerAdministration.checkCustomerExists("My", "Customer", "Street", "9", "38100", "Brunswick", true, false, false);
}
```

BREDEX

LAYERING EXAMPLES

```java
@Test
public void CUST_TC_01_01_01__CreateCustomer() {
  CustomerAdministrationData CUSTOMER = CustomerAdministrationData.CUST_TC_01_01_01;

  CustomerAdministration.checkCustomerExists(CUSTOMER, false);
  CustomerAdministration.createCustomer(CUSTOMER);
  CustomerAdministration.checkCustomerExists(CUSTOMER, true);

}
```

- Clear levels
- Data objects
- VAV pattern

BREDEX

A LOOK AT THE LOWER LAYERS

```java
//Internal workflow

private static void filloutCustomerForm(CustomerData data) {
    setName(data.name);
    setSurname(data.surname);
    setStreet(data.street);
    setPostcode(data.postcode);
    setCity(data.city);
}

//Public workflow (service)

public static void createCustomer(CustomerData data) {
    openMenuCustomer();
    openSubMenuCustomer();
    filloutCustomerForm(data);
    clickSave();
    MainView.toMainMenu();
}
}
```

BREDEX

A LOOK AT THE LOWER LAYERS

```java
@FindBy(id = "postcode")
private static WebElement postcode_cti;

@FindBy(id = "street")
private static WebElement street_cti;

@FindBy(id = "city")
private static WebElement city_cti;
```

```java
private static void setStreet(String street) {
  Reused.replaceText(street_cti, street);
}

private static void setPostcode(String postcode) {
  Reused.replaceText(postcode_cti, postcode);
}

private static void setCity(String city) {
  Reused.replaceText(city_cti, city);
}
```

BREDEX

DIFFERENT TOOLS AND DIFFERENT APIS

```java
// Selenium iterate through table and click on element
public static void selectValueFromTableSelenium(WebElement tableElement, String value) {
    String information = null;
    int rowIndex = 1;
    if (value != null) {
        List<WebElement> totalColumnCount = tableElement.findElements(By.tagName("tr"));
        for (WebElement colElement : totalColumnCount) {
            information = "";
            information += colElement.getText().trim();
            if (information.contains(value)) {
                Reused.clickLeftSingle(colElement.findElement(By.tagName("button")));
            }
            rowIndex++;
        }
    }
}
```

```java
//Jubula API select from table with some parameters set
private static void selectValueFromTable(String row, String value) {
    table.selectValueFromRow(row, Operator.equals, value, Operator.equals, 1, BinaryChoice.no, SearchType.absolute, InteractionMode.primary);
}
```

BREDEX

MORE PATTERNS: VAV

- Verify – Act – Verify
    - Because otherwise you just don't know

```
@Test
public void CUST_TC_01_01_01__CreateCustomer() {
    CustomerAdministrationData CUSTOMER = CustomerAdministrationData.CUST_TC_01_01_01;

    CustomerAdministration.checkCustomerExists(CUSTOMER, false);
    CustomerAdministration.createCustomer(CUSTOMER);
    CustomerAdministration.checkCustomerExists(CUSTOMER, true);

}
```

BREDEX

MORE PATTERNS: DATA OBJECTS

- One row per required data set
- Key and column are encapsulated in actions

```
                @Test
                public void CUST_TC_01_01_01__CreateCustomer() {
                  CustomerAdministrationData CUSTOMER = CustomerAdministrationData.CUST_TC_01_01_01;
@Test
public void CUST                                                                                    , false, false);
  CustomerAdmini   CustomerAdministration.checkCustomerExists(CUSTOMER, false);
  CustomerAdmini   CustomerAdministration.createCustomer(CUSTOMER);
  CustomerAdmini   CustomerAdministration.checkCustomerExists(CUSTOMER, true);                       , false, false);
}

                }
```

BREDEX

MORE PATTERNS: FORM PATTERN

- Fill in variable amounts of fields with one workflow

```
private static void filloutCustomerForm(CustomerData data) {
  setName(data.name);
  setSurname(data.surname);
  setStreet(data.street);
  setPostcode(data.postcode);
  setCity(data.city);
}
```

```
private static void setName(String name) {
  Reused.replaceText(name_cti, name);
}
```

```
//Technical keyword
public static void replaceText(WebElement element, String input) {
  fluentWait(element);
  if (input != null && element.isEnabled()) {
    element.clear();
    element.sendKeys(input);
  }
}
```

BREDEX

THERE ARE MORE...

- Responsibility assignment

- Fresh / shared setup

- ...

BREDEX

POST-EXAMPLE DISCLAIMER

This works all the time like perfect unicorn rainbow poop! *

* In the contexts we have used these things in so far, they have been very useful to us and have saved us a lot of time and hair-pulling out.

BREDEX

EXTRA DISCLAIMER

"These are patterns we know from software development!"

- people

*"Then train developers and testers to use them
    so that they write good UI tests"*

- me

BREDEX

THINGS TO TAKE AWAY

- You don't have to hate your UI tests
  - You just have to do them right

- Don't reinvent the wheel completely, but remember to expand and extend it
  - And re-evaluate for new contexts

- Use these patterns to get developers and testers of all flavours on board with UI automation

BREDEX