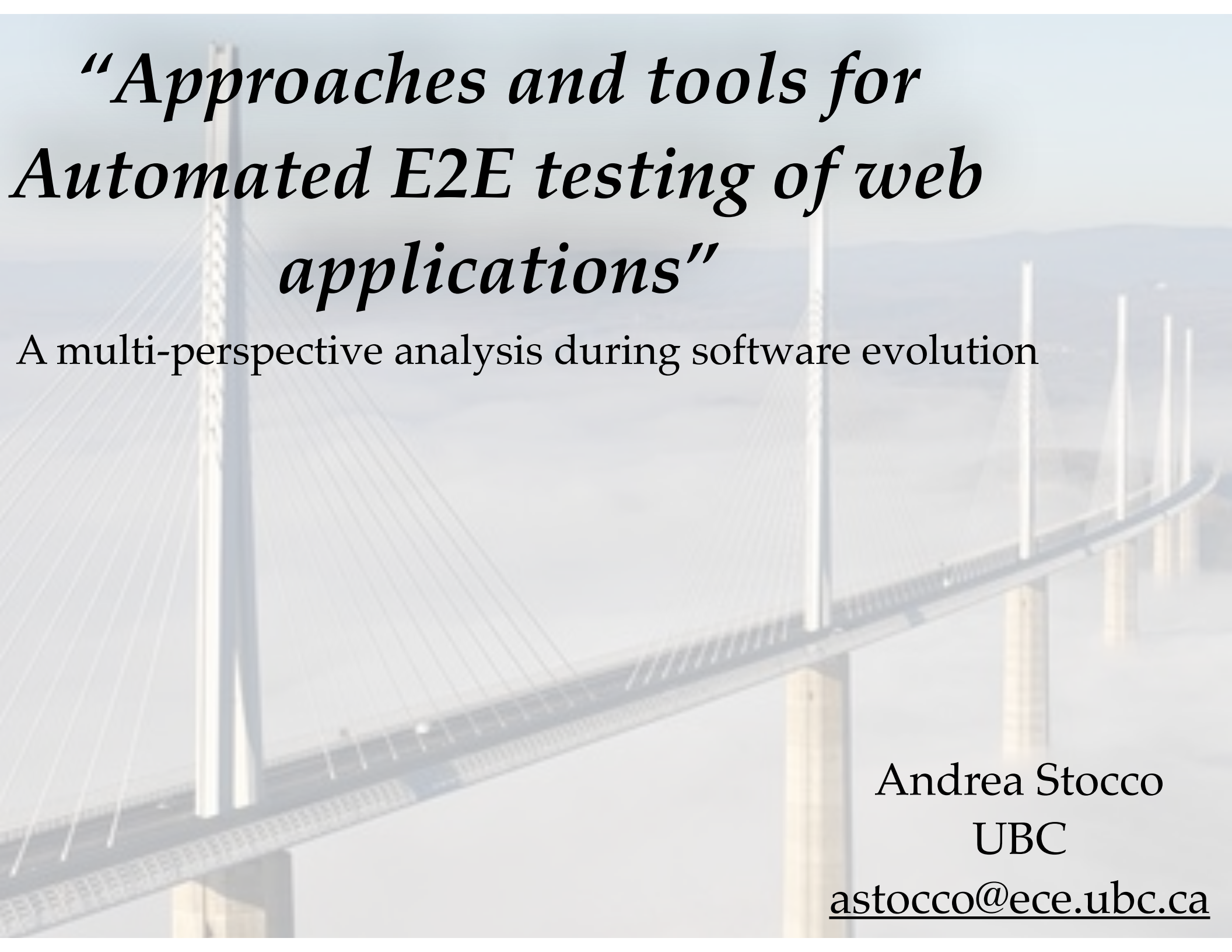


"Approaches and tools for Automated E2E testing of web applications"

A multi-perspective analysis during software evolution

Andrea Stocco
UBC
astocco@ece.ubc.ca

The World “Wild” Web

The World “Wild” Web

What does make a web application unique?

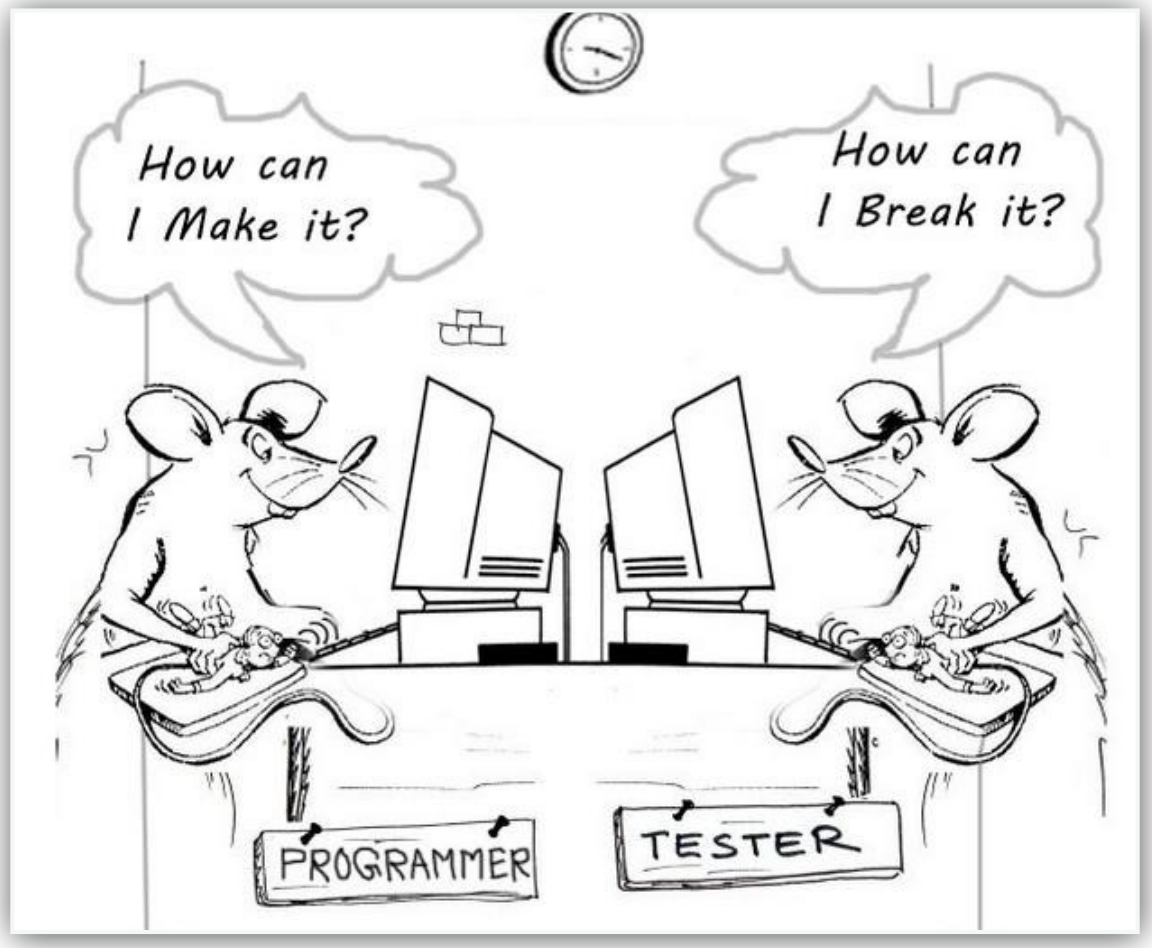
Is testing web apps different than testing a desktop app?

What makes testing web apps complex?

Modern Web Applications

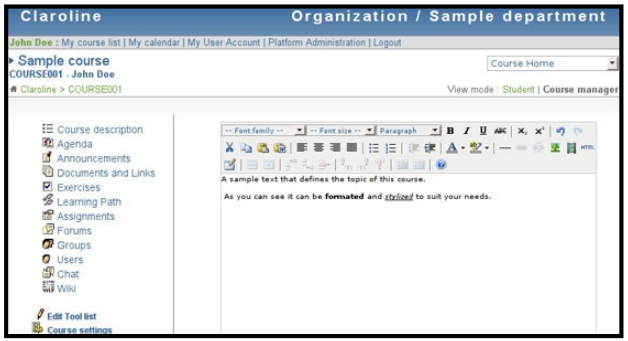
- complex **infrastructure**
- integration of **different** technologies
- **asynchronous** interactions
- **different** platforms (web, smartphones)
- used by billions of people
- **ultra-rapid** evolution

Web Development vs Testing

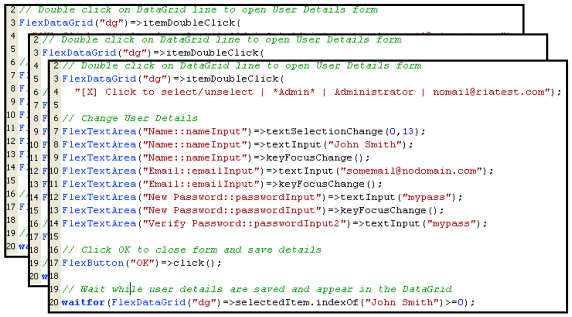


Functional Test Automation

Web App



Functional Requirements Analysis



Test Scripts Creation

Test Scripts Automation

Test Results

Bug Reports

Functional Test Automation

Web App

Functional Requirements Analysis

Test Scripts

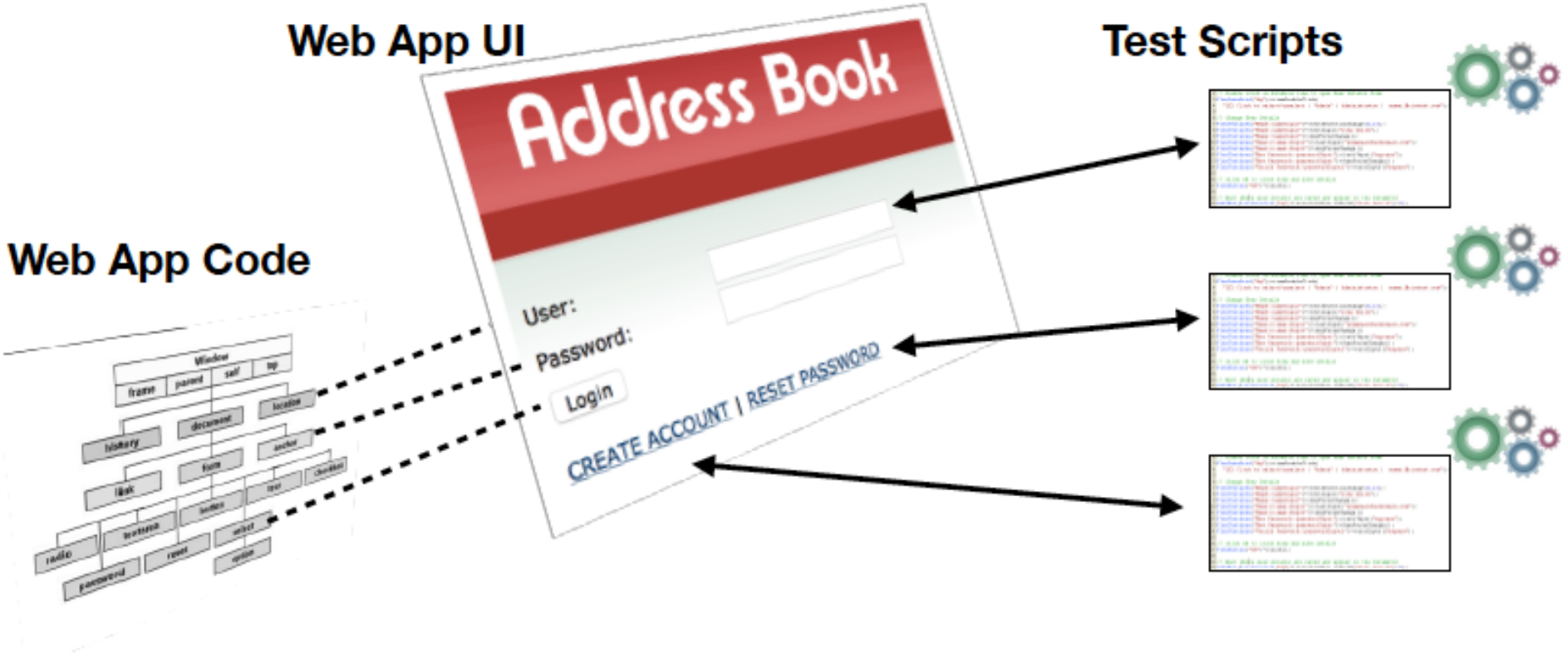
Test Scripts Automation

Why have such tools focused at the E2E (GUI) level?

Test Results

Bug Reports

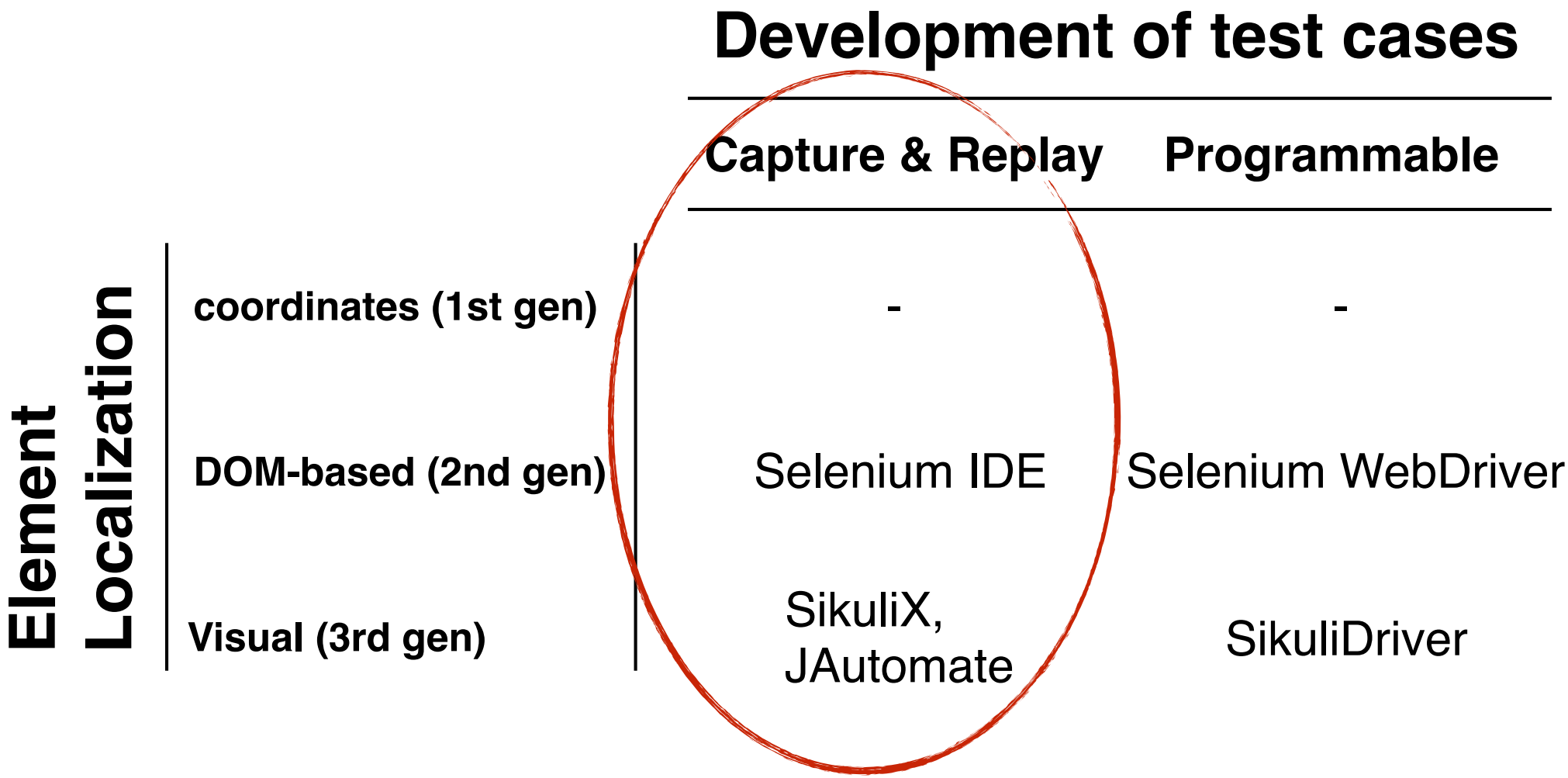
E2E Test Automation



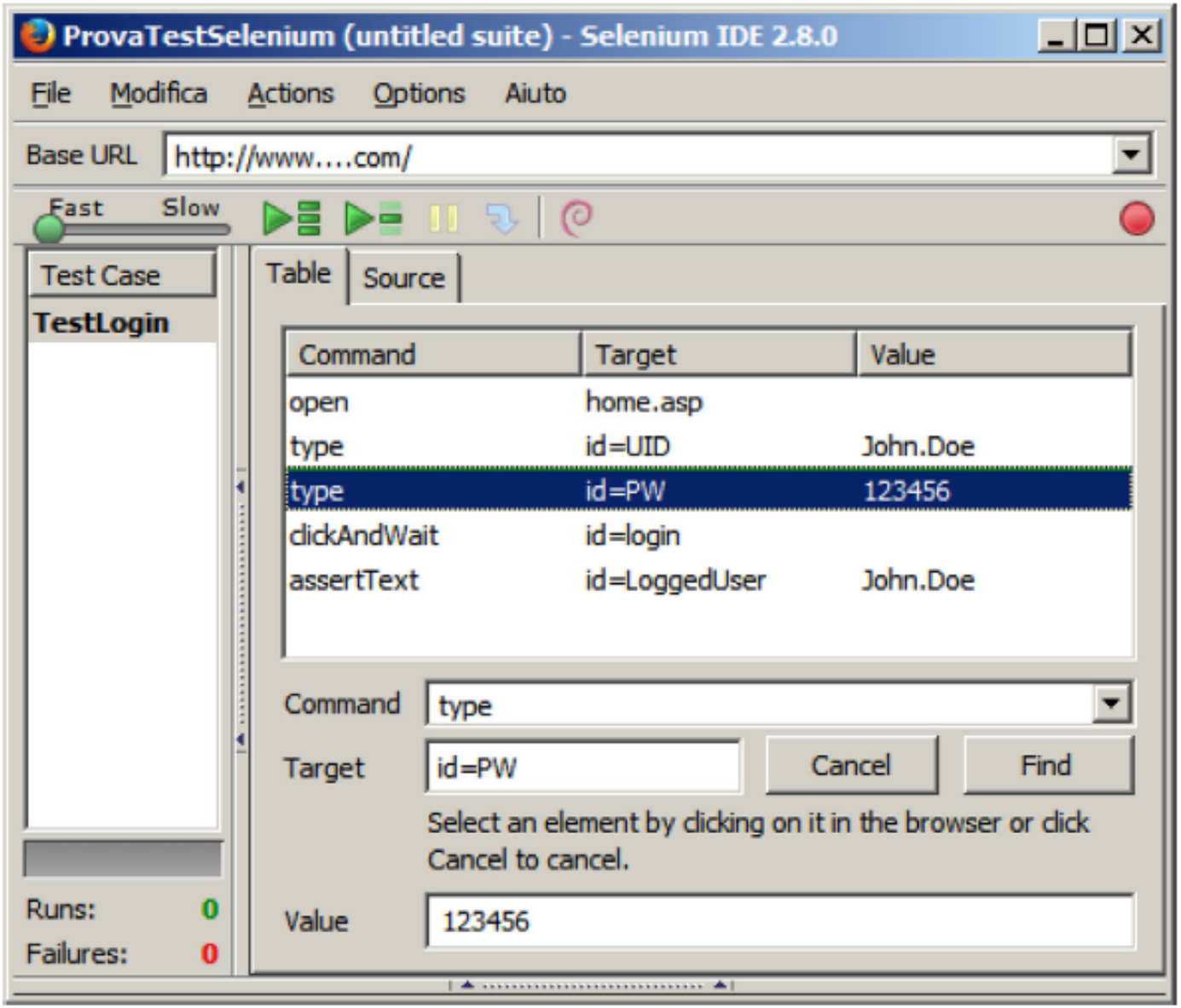
Approaches to E2E Test Automation

Element Localization	Development of test cases: Capture & Replay	Development of test cases: Programmable
coordinates (1st gen)	-	-
DOM-based (2nd gen)	Selenium IDE	Selenium WebDriver
Visual (3rd gen)	SikuliX, JAutomate	SikuliDriver

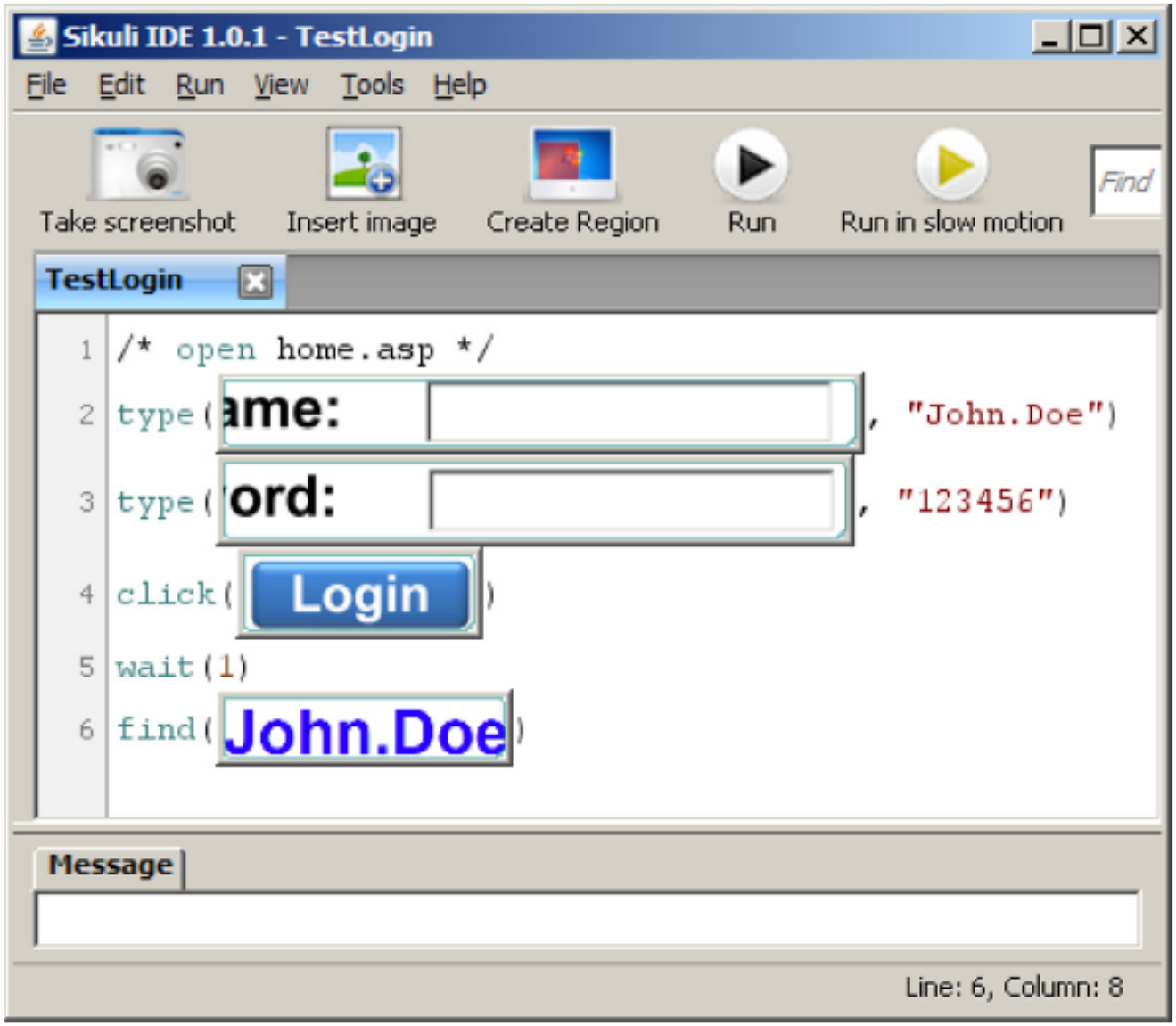
Approaches to E2E Test Automation



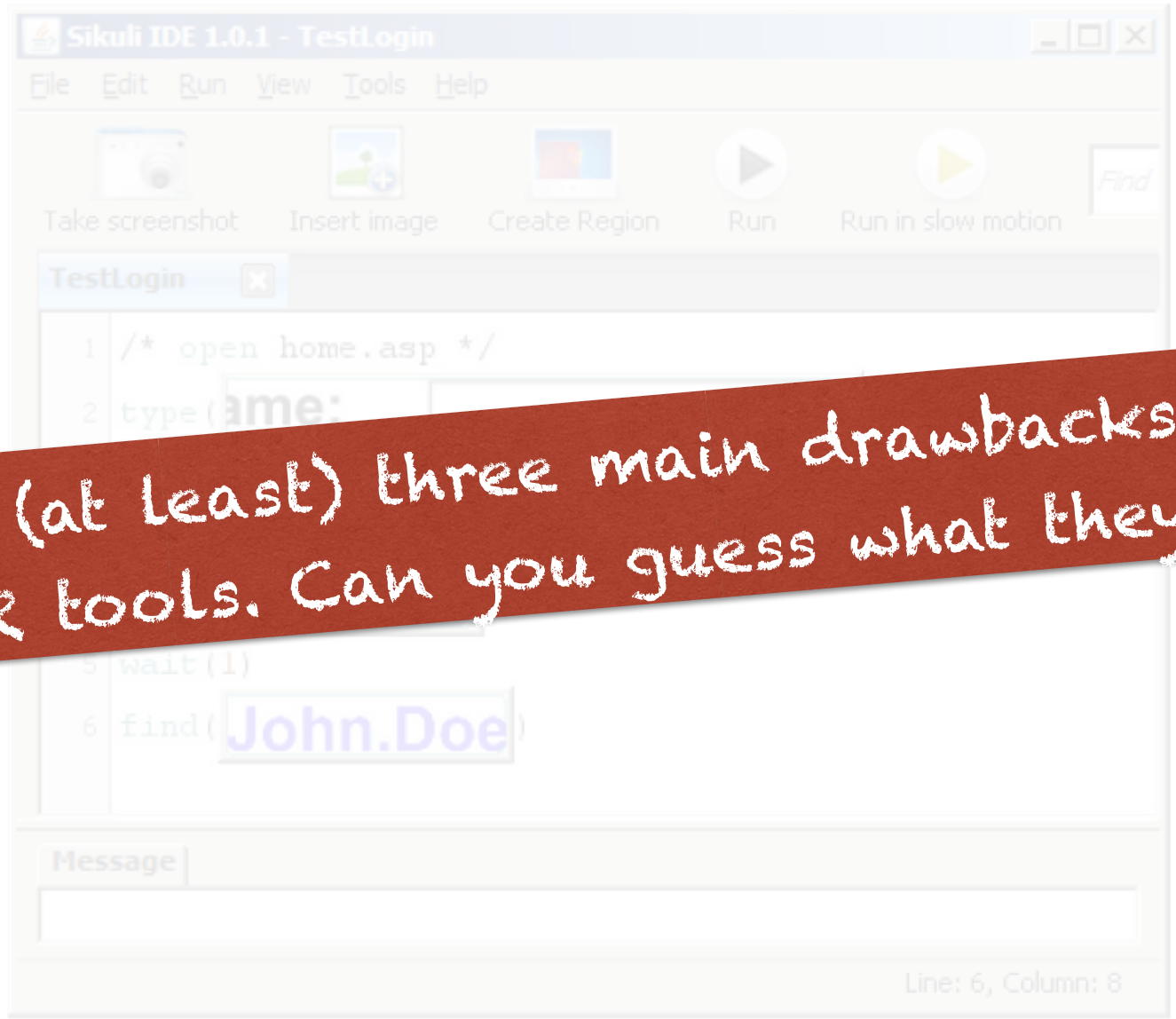
(DOM) Capture & Replay



(Visual) Capture & Replay



(Visual) Capture & Replay



There are (at least) three main drawbacks related to C&R tools. Can you guess what they are?

Capture & Replay Drawbacks

Hard-code values

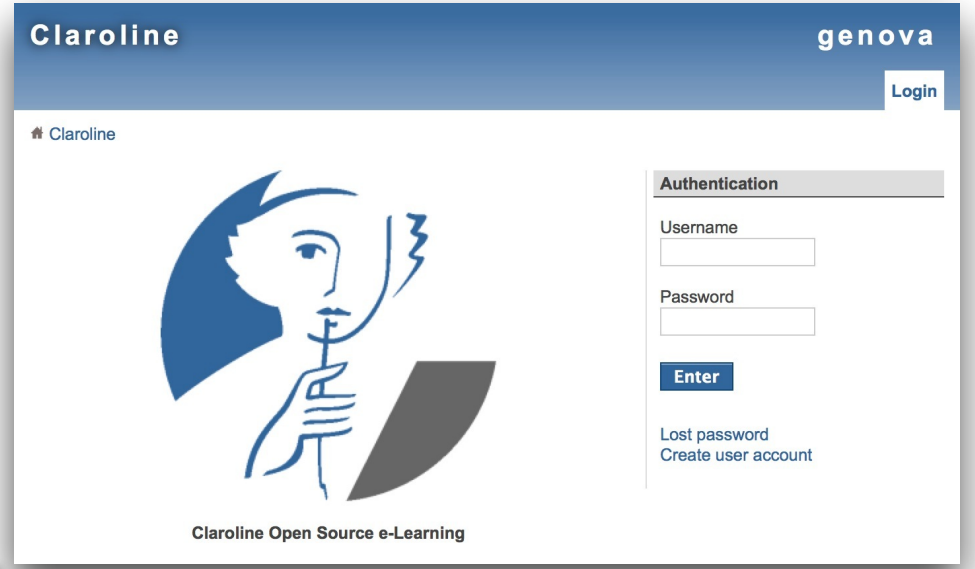
Strong coupling

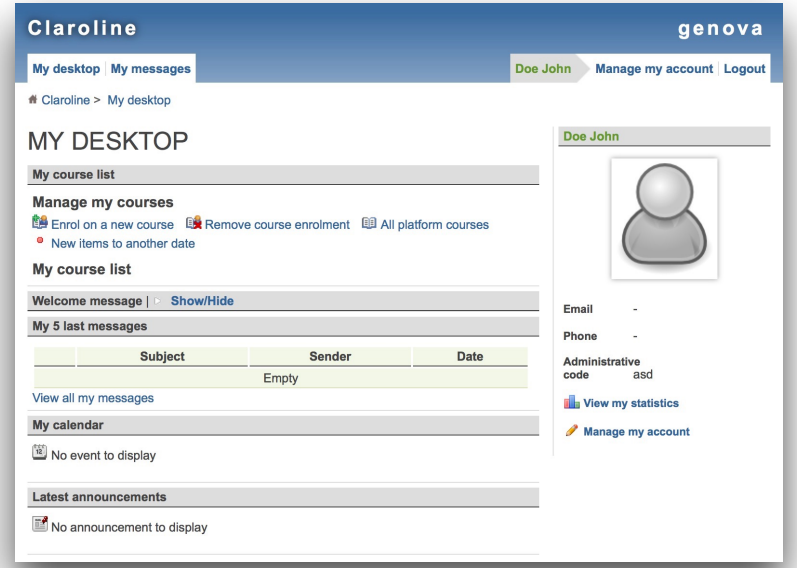
Duplicated code

Approaches to E2E Test Automation

Element Localization	Development of test cases: Capture & Replay	Development of test cases: Programmable
coordinates (1st gen)	-	-
DOM-based (2nd gen)	Selenium IDE	Selenium WebDriver
Visual (3rd gen)	SikuliX, JAutomate	SikuliDriver

(DOM) Programmable





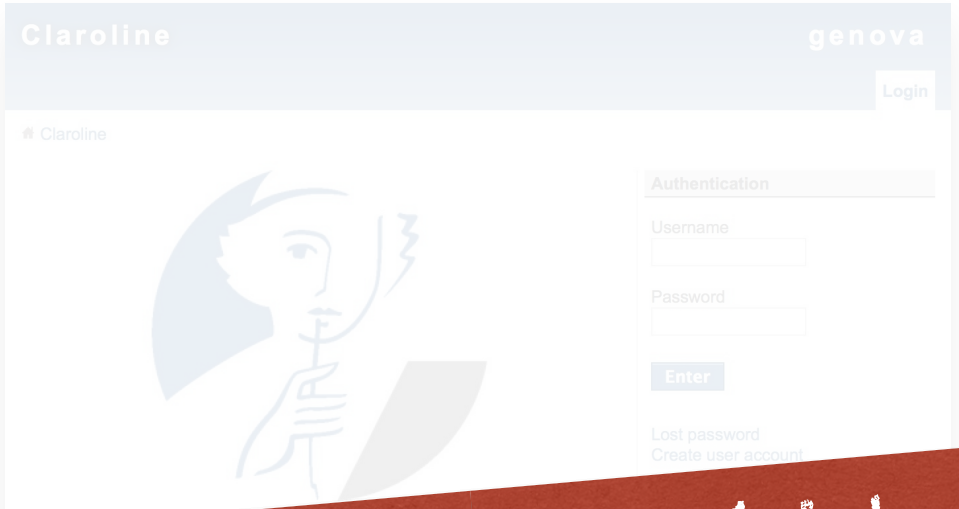

```
public class Login {
  @Test
  public void testLogin() {
    WebDriver driver = new FirefoxDriver();
    driver.get("http://localhost/claroline1115/index.php");
    driver.findElement(By.Id(id="user")).sendKeys("John Doe");
    driver.findElement(By.XPath("xpath="//form/input[2]")).sendKeys("securepassword");
    driver.findElement(By.LinkText(linkText="Enter")).click();
    Assert.assertEquals(driver.findElement(By.Id("John Doe")), "Login was successful");
  }
}
```

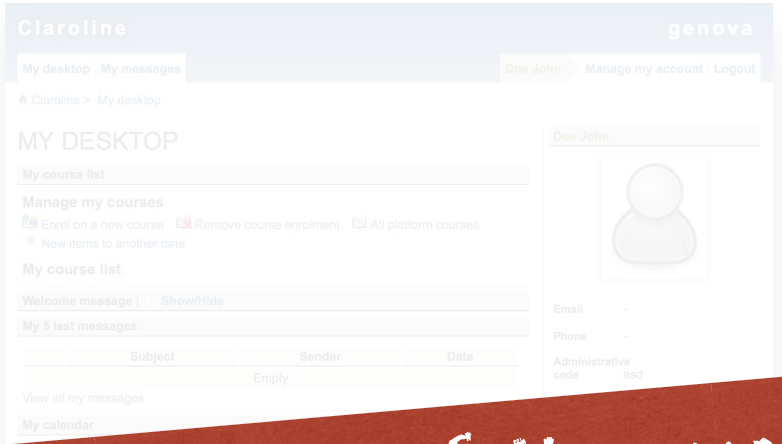
Browser specific

UI specific

Manipulation specific

(DOM) Programmable





Can you enumerate (at least) some of the main advantages that come with the use of a programmable tool w.r.t. a C&R tool?

```
FirefoxDriver();
driver.get("http://localhost/claroline1115/index.php");
driver.findElement(By.Id(id="user")).sendKeys("John Doe");
driver.findElement(By.XPath("xpath="//form/input[2]")).sendKeys("securepassword");
driver.findElement(By.LinkText(linkText="Enter")).click();
Assert.assertEquals(driver.findElement(By.Id("John Doe")), "Login was successful");
}
}
```

UI specific

Manipulation specific

Programmable tools Advantages

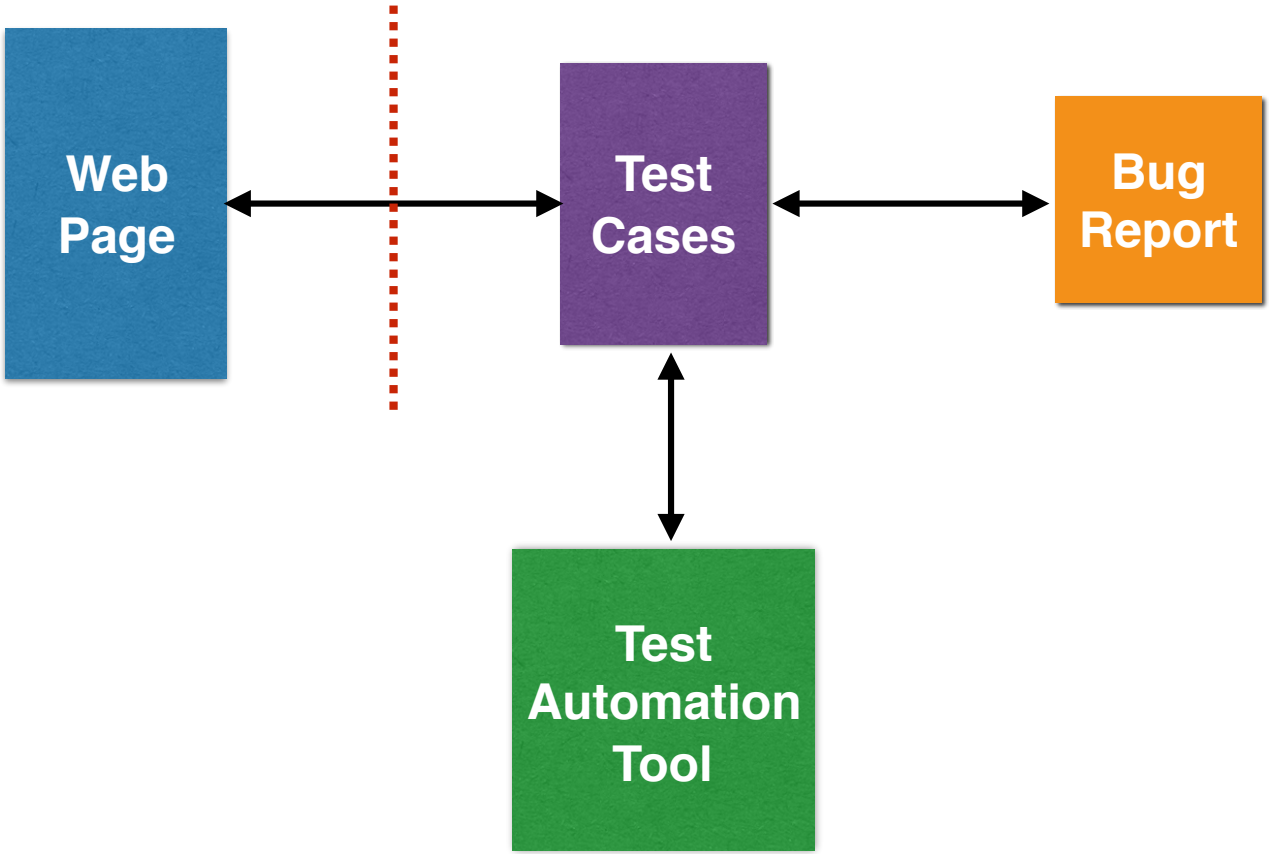
~~Hard-code values~~ **Data-driven**

~~Strong coupling~~

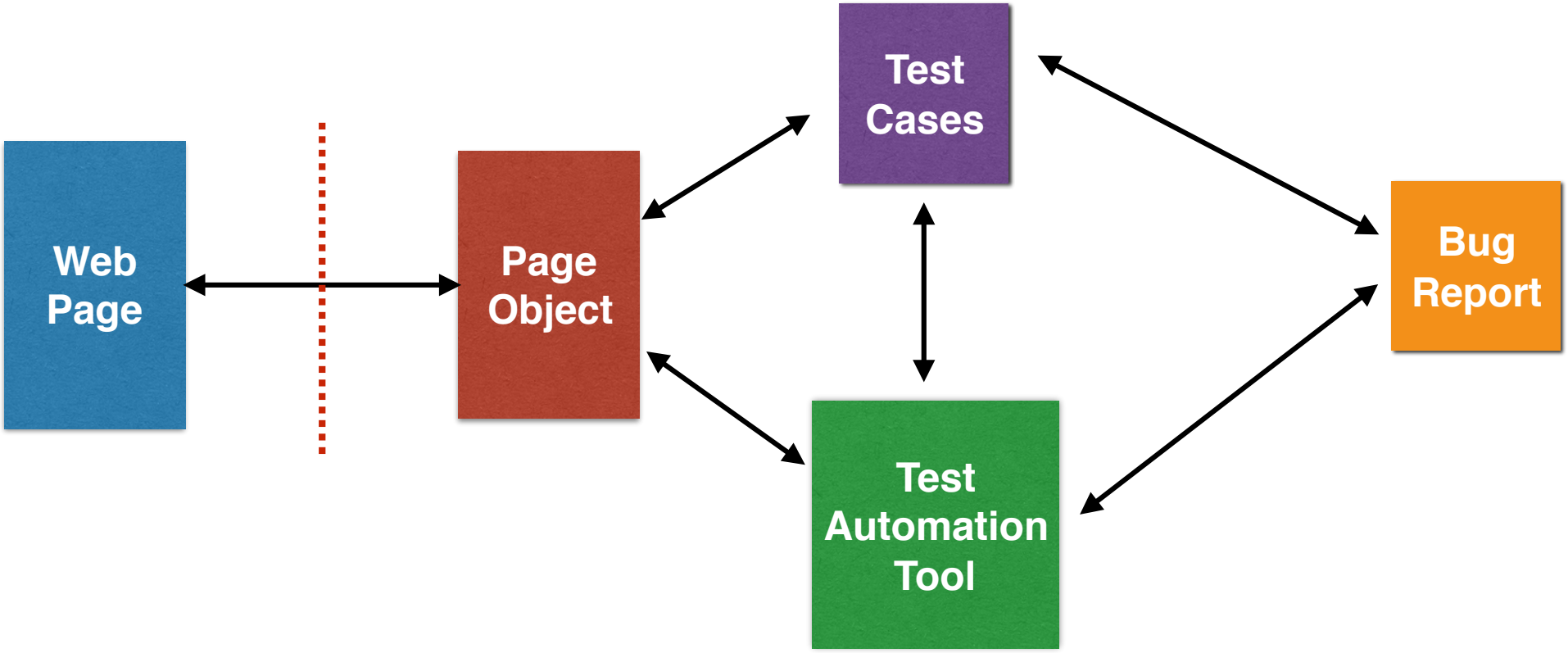
Design Patterns

~~Duplicated code~~

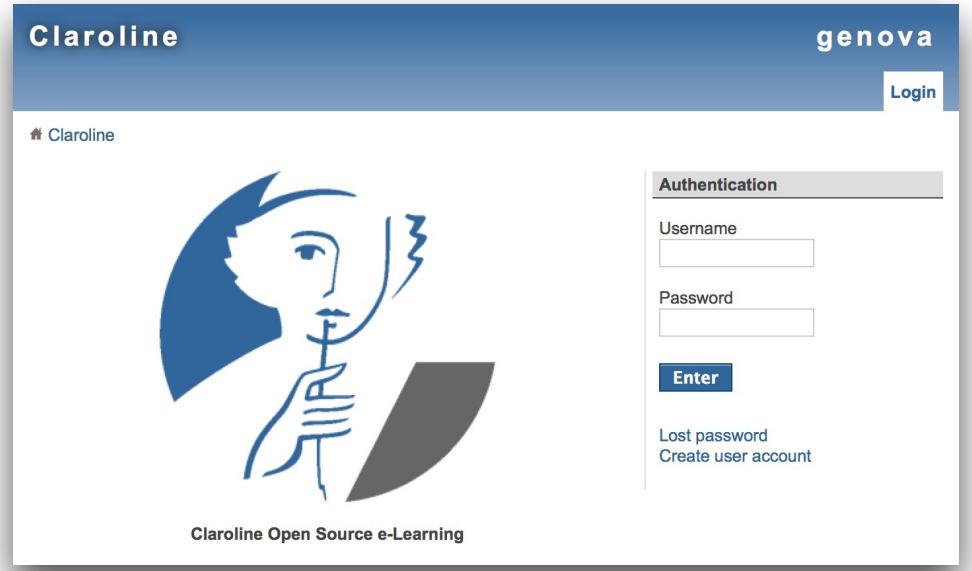
Naive web testing



Use Page Objects!

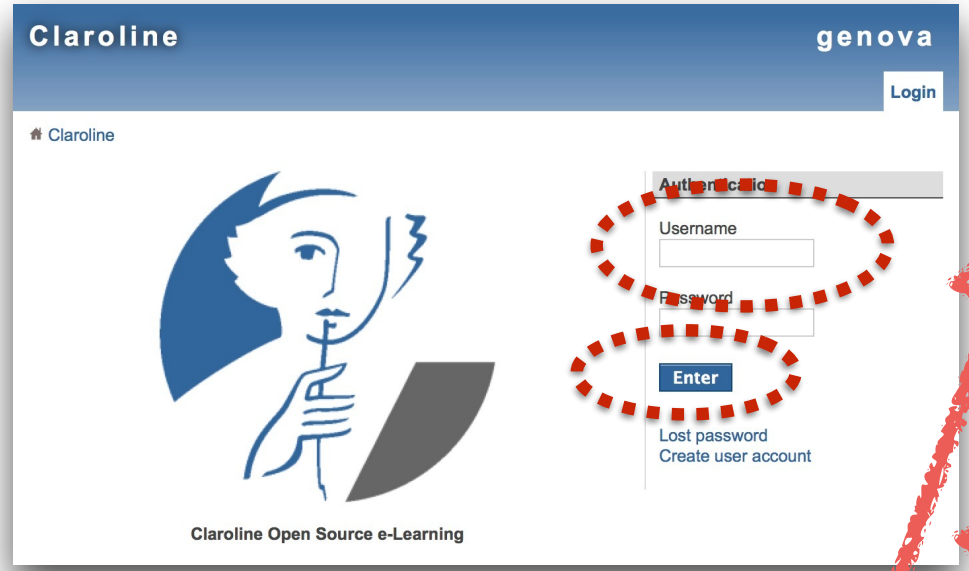


Page Object Pattern: An Example




```
public class IndexPage {
  private final WebDriver driver;

  @FindBy(id="user")
  private WebElement username;

  @FindBy(xpath="//form/input[2]")
  private WebElement password;

  @FindBy(linkText="Enter")
  private WebElement login;

  @FindBy(id="loggedUser")
  private WebElement loggedUser;

  public UserAccount goToCreateAccount(){
    createUserAccount.click();
    return new UserAccount(driver);
  }

  public HomePage login(String usr, String pwd) {
    username.sendKeys(usr);
    password.sendKeys(pwd);
    login.click();
    return new HomePage(driver);
  }
}
```

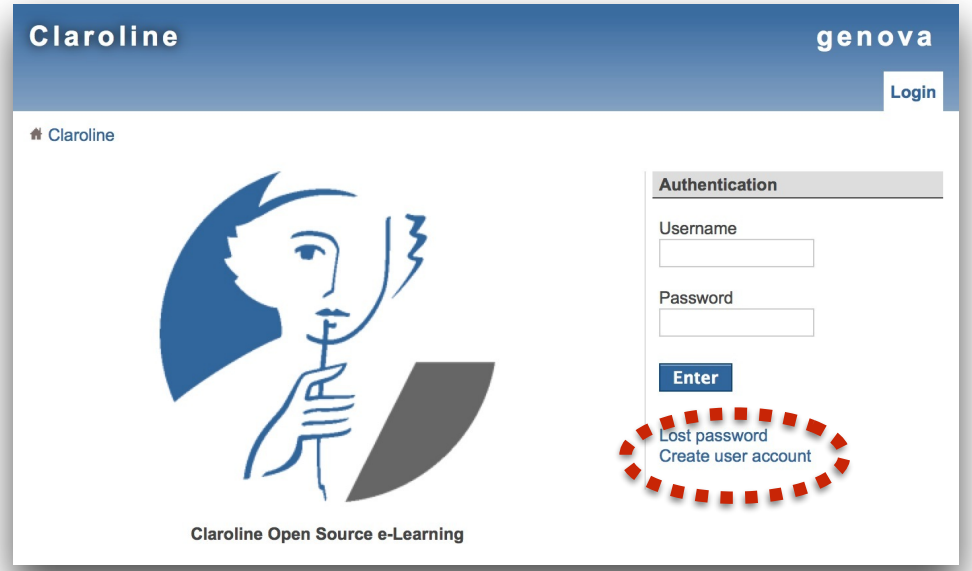
Page Object Pattern: An Example



WebElements

```
public class IndexPage {
  private final WebDriver driver;

  @FindBy(id="user")
  private WebElement username;

  @FindBy(xpath="//form/input[2]")
  private WebElement password;

  @FindBy(linkText="Enter")
  private WebElement login;

  @FindBy(id="loggedUser")
  private WebElement loggedUser;

  public UserAccount goToCreateAccount(){
    createUserAccount.click();
    return new UserAccount(driver);
  }

  public HomePage login(String usr, String pwd) {
    username.sendKeys(usr);
    password.sendKeys(pwd);
    login.click();
    return new HomePage(driver);
  }
}
```

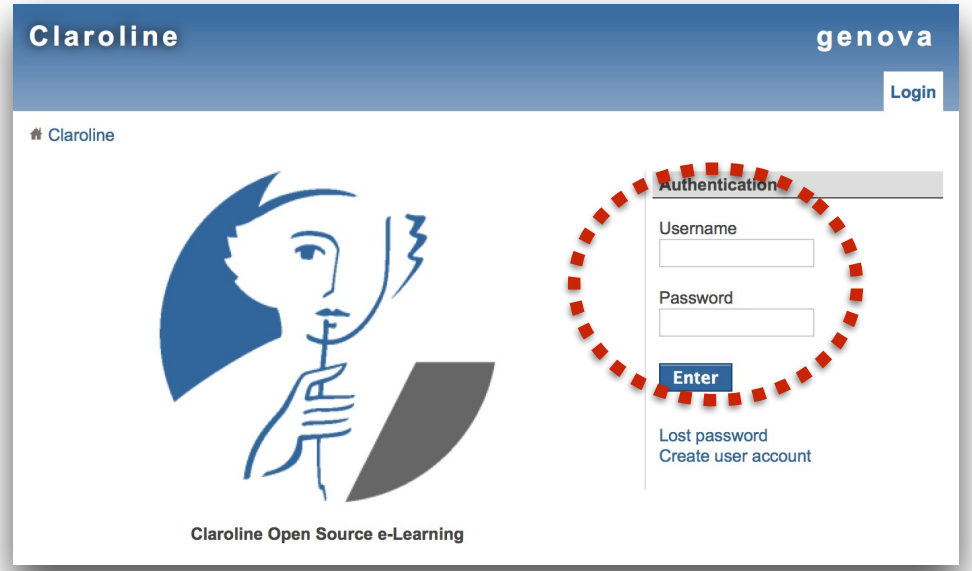
Page Object Pattern: An Example



Navigations

```
public class IndexPage {
  private final WebDriver driver;

  @FindBy(id="user")
  private WebElement username;

  @FindBy(xpath="//form/input[2]")
  private WebElement password;

  @FindBy(linkText="Enter")
  private WebElement login;

  @FindBy(id="loggedUser")
  private WebElement loggedUser;

  public UserAccount goToCreateAccount(){
    createUserAccount.click();
    return new UserAccount(driver);
  }

  public HomePage login(String usr, String pwd) {
    username.sendKeys(usr);
    password.sendKeys(pwd);
    login.click();
    return new HomePage(driver);
  }
}
```

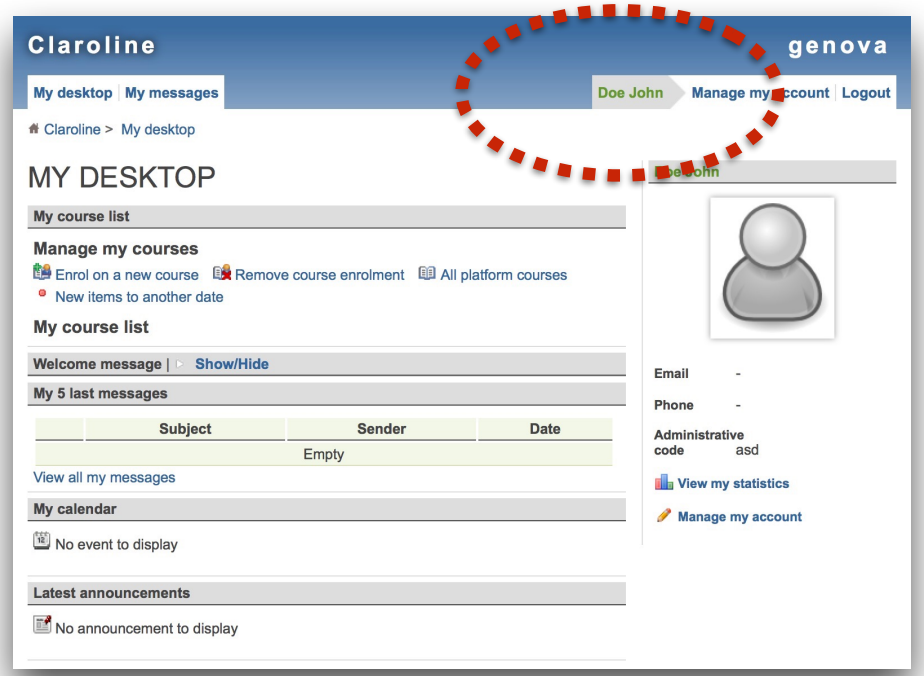
Page Object Pattern: An Example




```
public class IndexPage {
  private final WebDriver driver;

  @FindBy(id="user")
  private WebElement username;

  @FindBy(xpath="//form/input[2]")
  private WebElement password;

  @FindBy(linkText="Enter")
  private WebElement login;

  @FindBy(id="loggedUser")
  private WebElement loggedUser;

  public UserAccount goToCreateAccount(){
    createUserAccount.click();
    return new UserAccount(driver);
  }

  public HomePage login(String usr, String pwd) {
    username.sendKeys(usr);
    password.sendKeys(pwd);
    login.click();
    return new HomePage(driver);
  }
}
```

Actions

Page Object Pattern: An Example



Getters

```
public class HomePage {
  private final WebDriver driver;

  @FindBy(xpath="//li/a[1]")
  private WebElement myDesktop;

  @FindBy(xpath="//li/a[2]")
  private WebElement myMessages;

  @FindBy(linkText="Enrol")
  private WebElement enrolACourse;

  @FindBy(id="loggedUser")
  private WebElement loggedUser;

  public CoursePage goToEnrol(){
    enrolACourse.click();
    return new CoursePage(driver);
  }
...
  public String getLoggedUser() {
    return loggedUser.getText();
  }
}
```

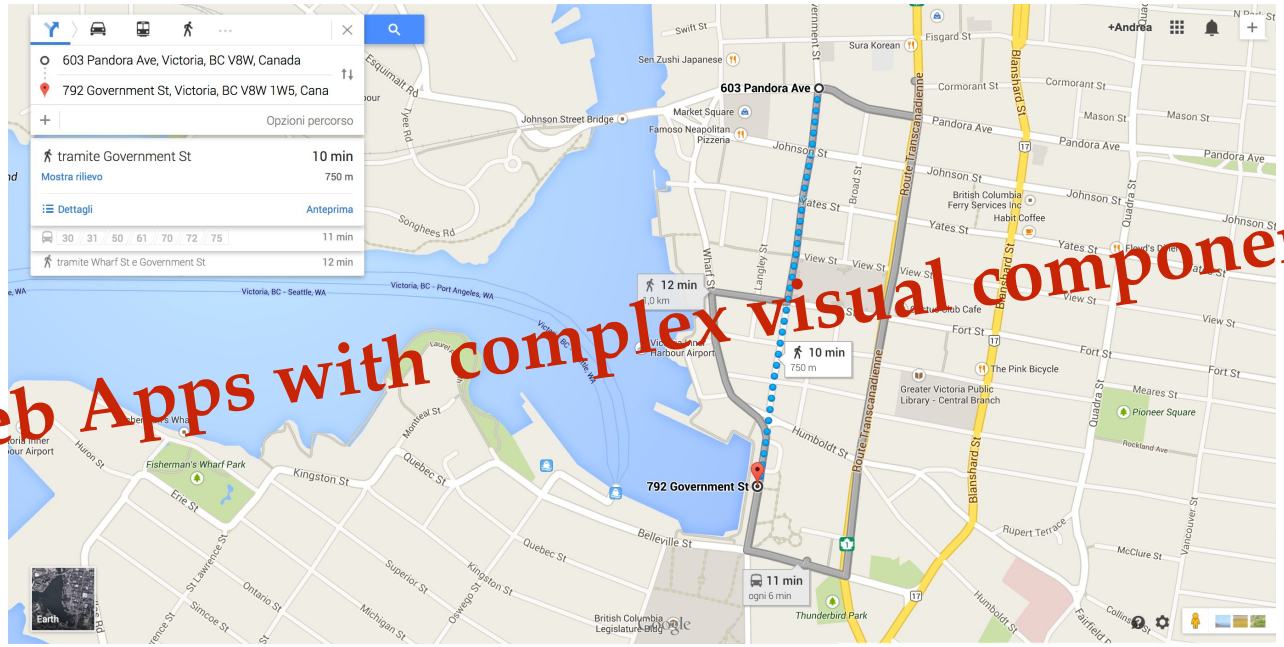
Test without abstraction

```
public class Login {
  @Test
  public void testLogin() {
    WebDriver driver = new FirefoxDriver();
    driver.get("http://localhost/claroline1115/index.php");
    driver.findElement(By.Id(id="user")).sendKeys("John Doe");
    driver.findElement(By.XPath("xpath="//form/input[2]")).sendKeys("securepassword");
    driver.findElement(By.LinkText(linkText="Enter")).click();
    Assert.assertEquals(driver.findElement(By.Id("John Doe")), "Login was successful");
   }
}
```

Test with Page Object abstraction

```
public class Login {
  @Test
  public void testLogin {
    Utils.connectToWebAppUnderTest();

    IndexPage ip = new IndexPage(driver);

    HomePage hp = ip.login("John Doe", "securepassword");

    assertEquals(hp.getLoggedUser(), "Doe John");
  }
}
```

Visual Test Automation



3rd generation: **image-recognition**

sikuli

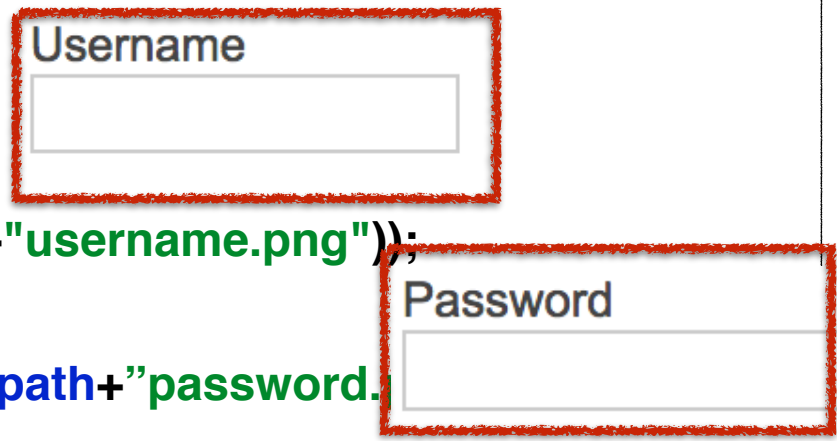
JAutomate

(Visual) Programmable

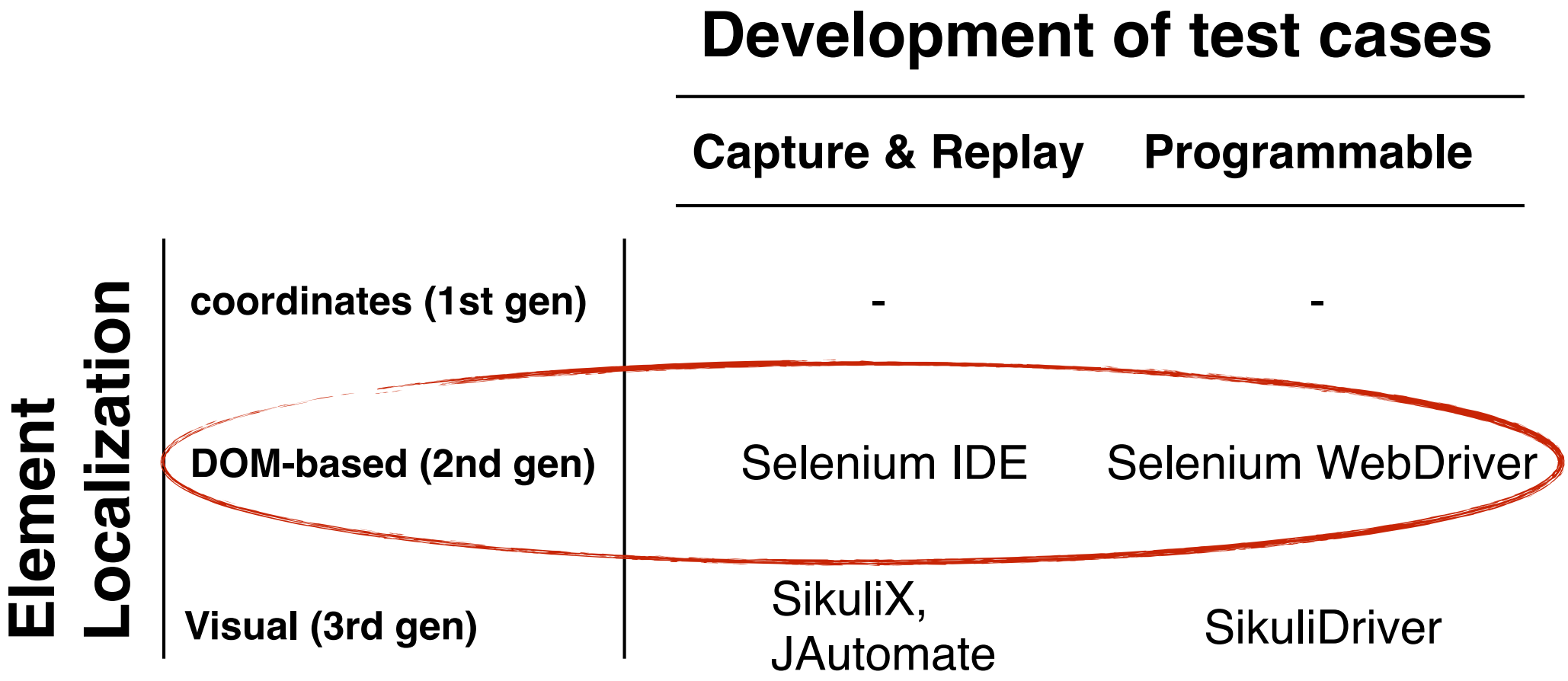
sikuli

```
public class Login {
  @Test
  public void testLogin() {
     WebDriver driver = new FirefoxDriver();
     driver.get("http://localhost/claroline1115/index.php");

     ImageTarget user = new ImageTarget(new File(path+"username.png"));
     sendKeys(user, "John Doe");

     ImageTarget password = new ImageTarget(new File(path+"password.
     sendKeys(password, "thenameofmycat");

    ImageTarget login = new ImageTarget(new File(path+"login.png"));
    click(login);

    …
  }
}
```

Visual Locators



Approaches to E2E Test Automation



DOM-based locators

- **id**
- **css**
- **name**
- **linkText**
- **XPath**

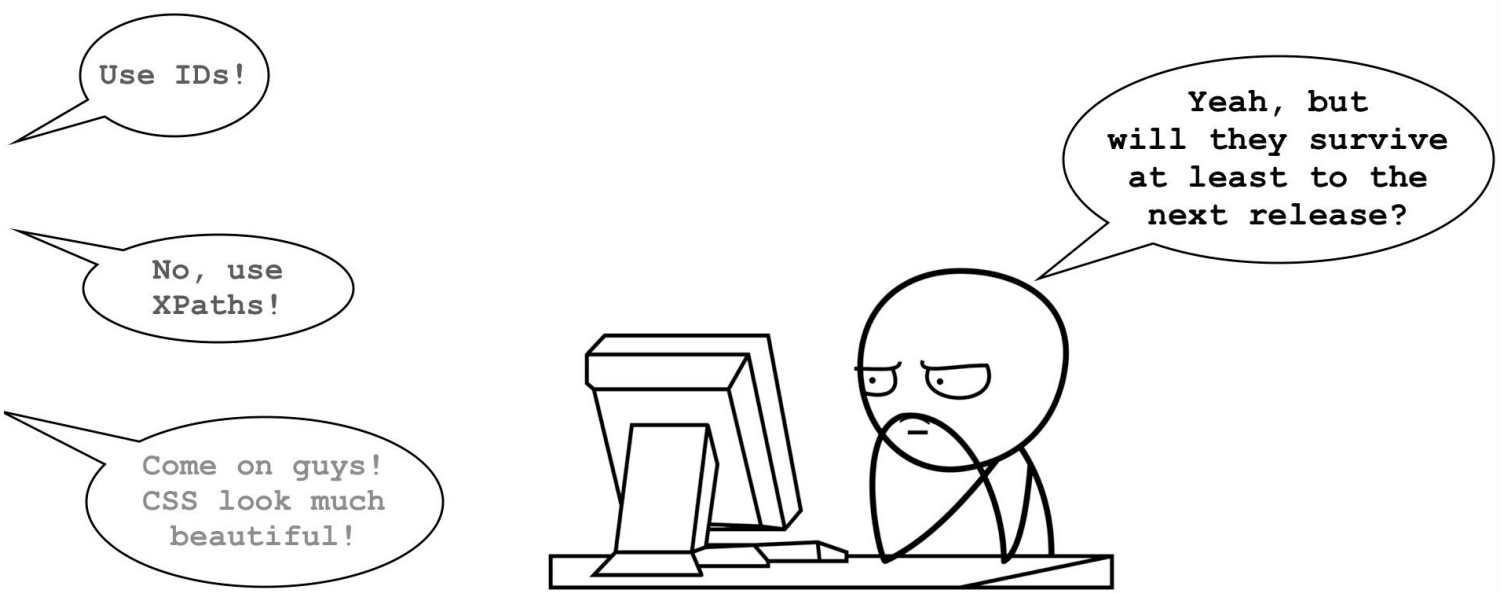
Username:
Password:
Login

```
<form name="loginform" action="homepage.asp" method="post">
 Username: <input type="text" id="UID" name="username"><br>
 Password: <input type="text" id="PW" name="password">
 <a href="javascript:loginform.submit()"
</form>
```

Which one should I use?

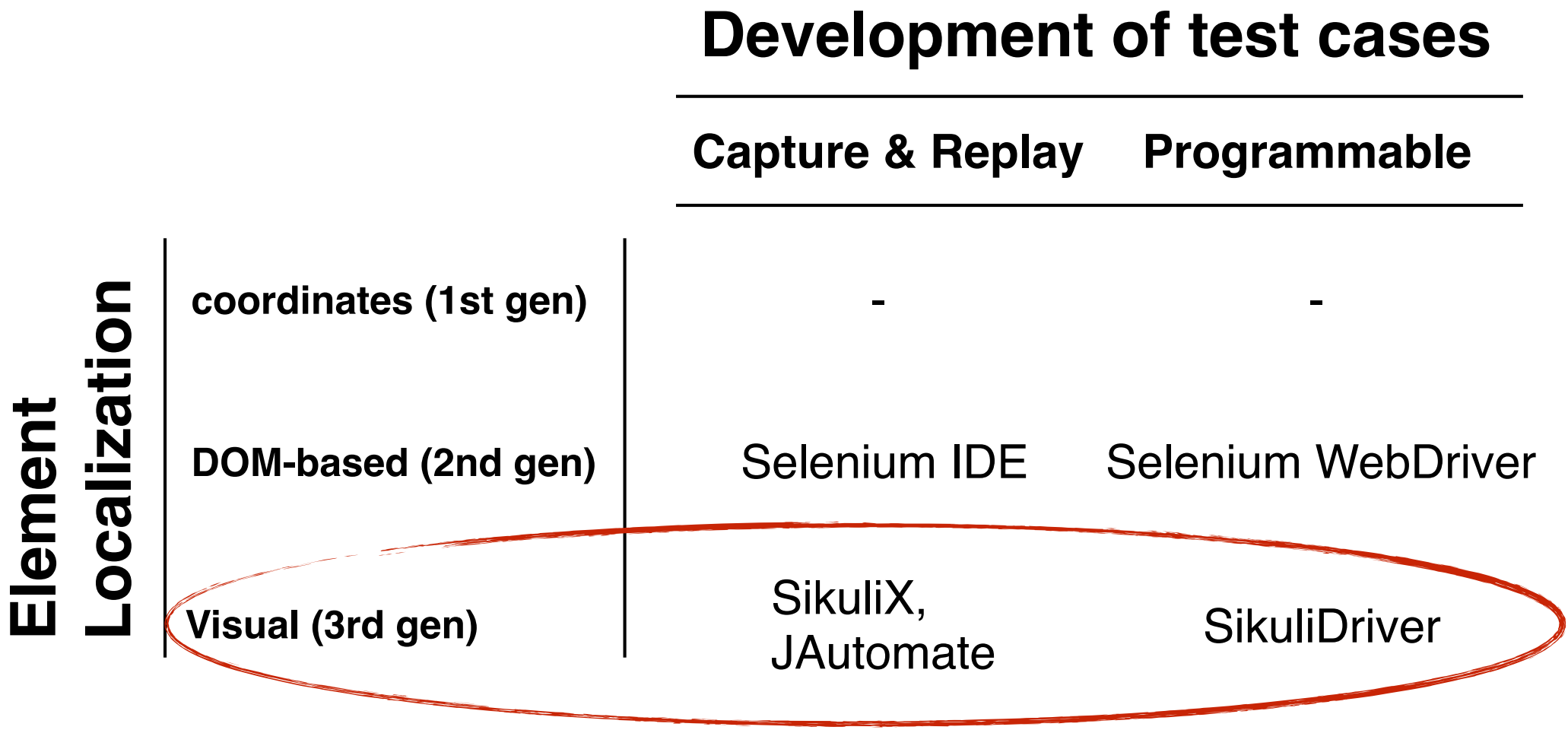
```
page
.com/login.asp");
.findElement(By.id("UID")).sendKeys("John.Doe");
driver.findElement(By.id("PW")).sendKeys(123456);
driver.findElement(By.linkText("Login")).click();
// we are in the 'HomePage.asp' page
assertEquals("John.Doe", driver.findElement(By.id("uname")).getText);
driver.close();
}
```

DOM Locators and SW Evolution



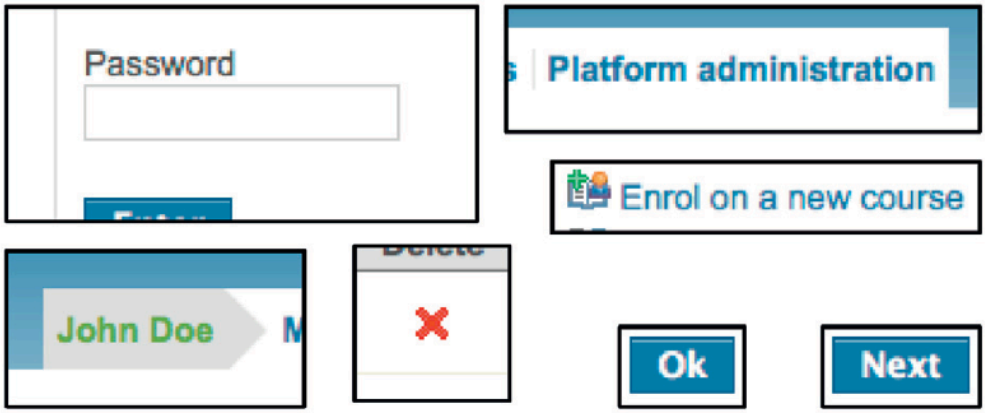
- **ids** are preferable but *(i)* not present *(ii)* auto-generated *(iii)* not unique
- other strategies are applicable **only** in **specific** contexts
- XPath **complex** and brittle but **powerful** and flexible
- a **careful** choice of locators can save you precious time (and headache)

Approaches to E2E Test Automation

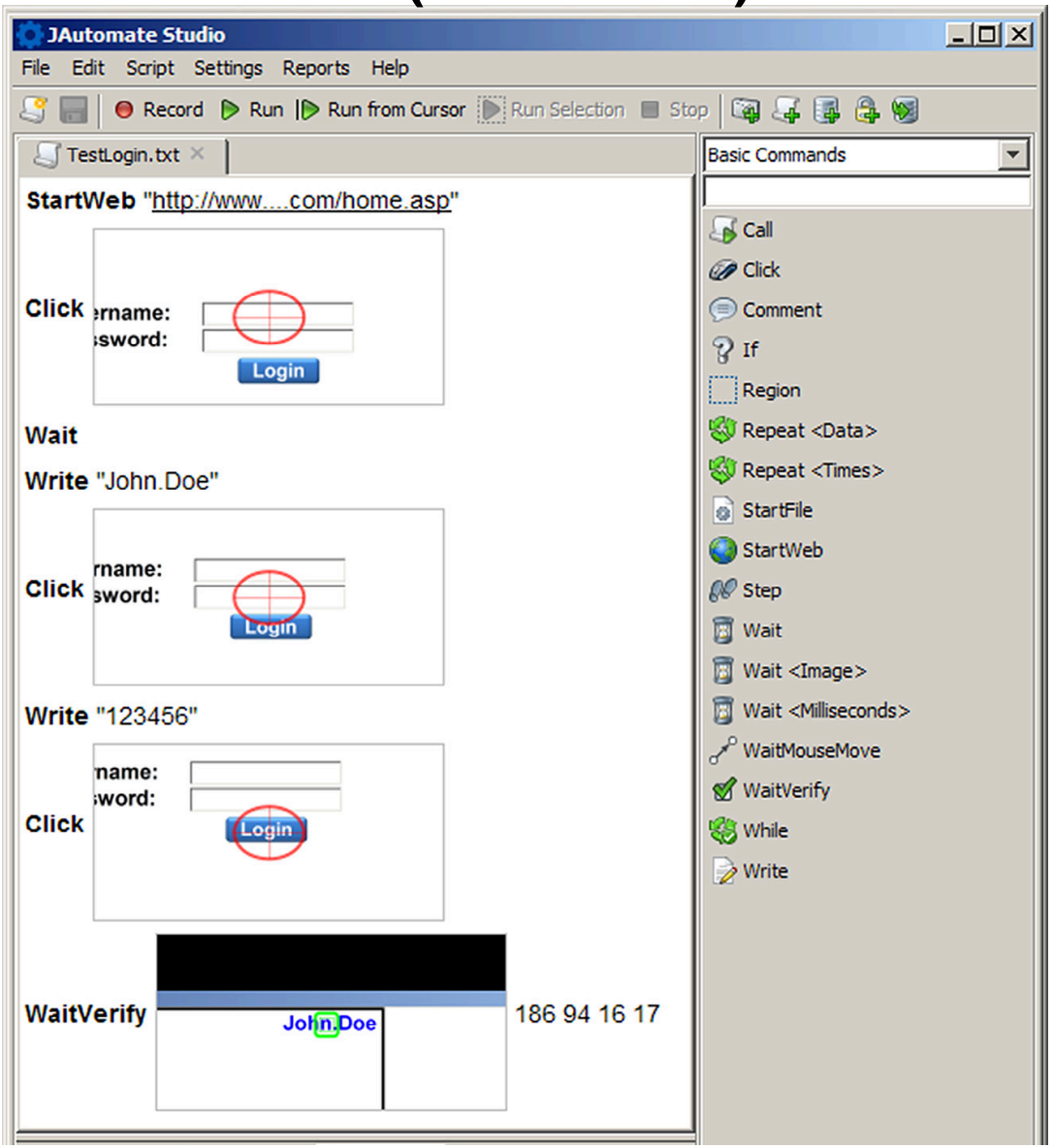


Visual locators

manually crafted (SikuliX and SikuliAPI)

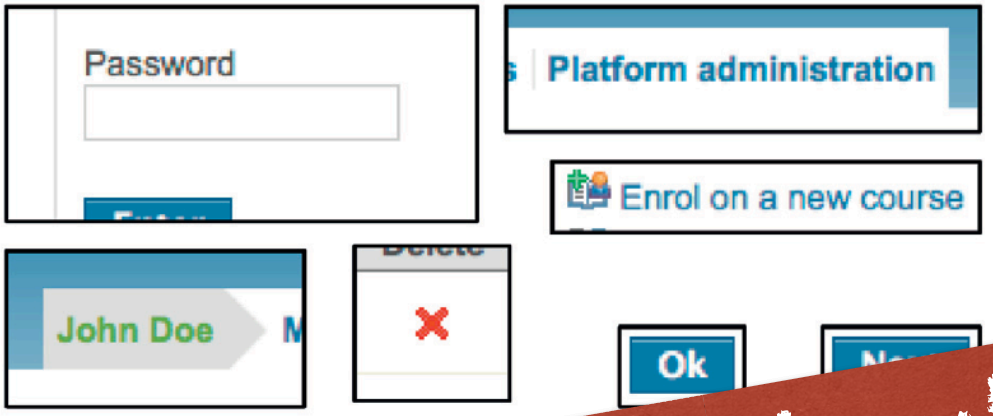


autogenerated (JAutomate)

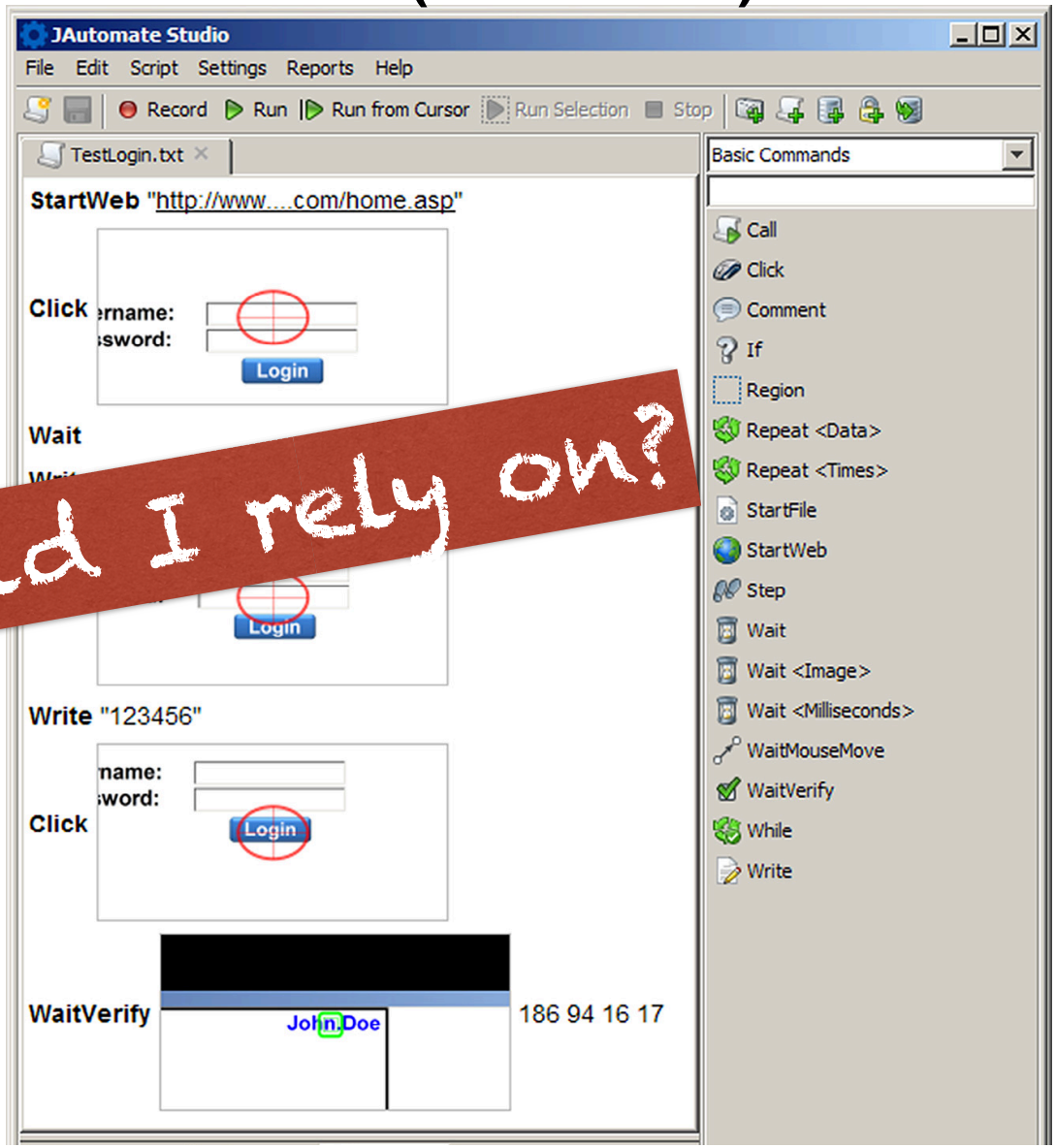


Visual locators

manually crafted (SikuliX and SikuliAPI)



autogenerated (JAutomate)

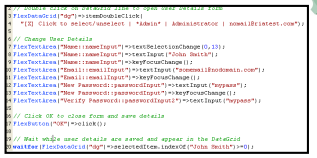


Which one should I rely on?

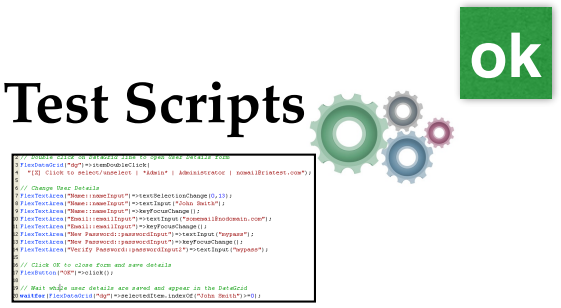
Test Case Evolution

Issues with Test Automation

Test Scripts



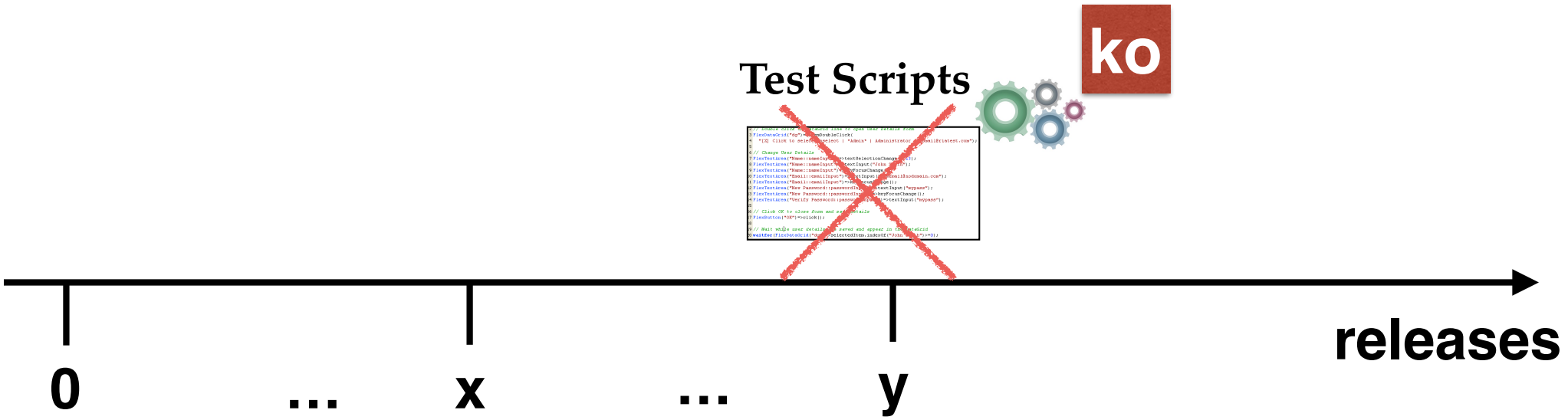
Issues with Test Automation



releases

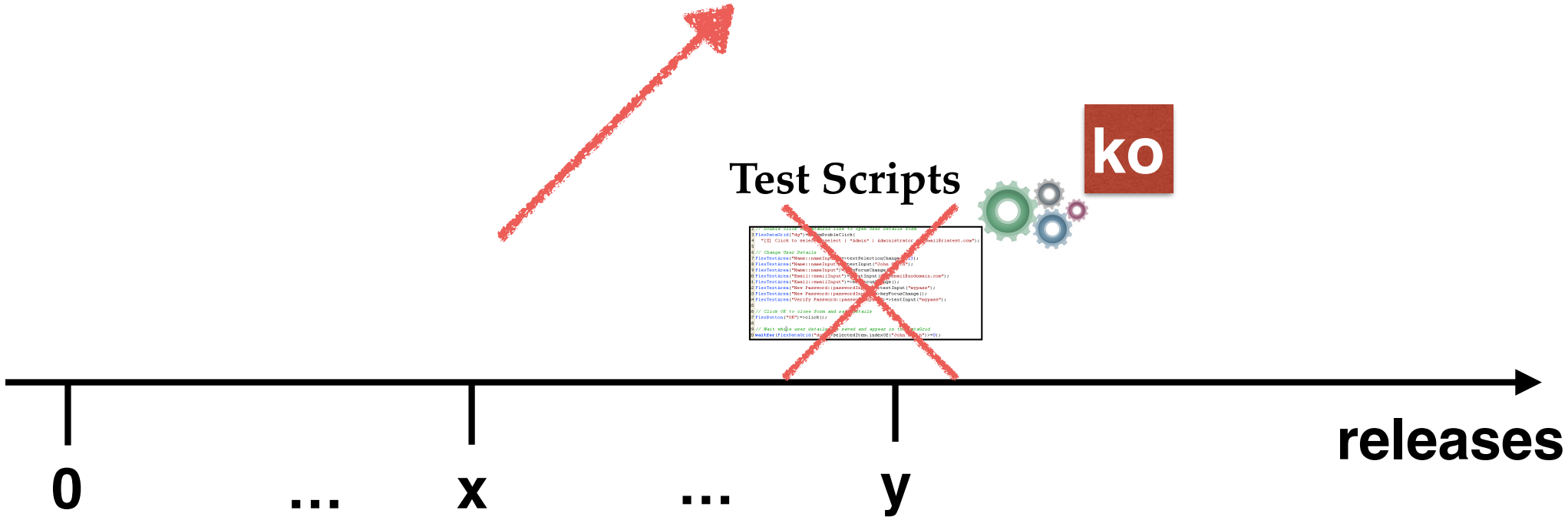
0 ... x

Issues with Test Automation



Issues with Test Automation

in web development, the releases are frequent!



Test Scripts Maintenance

- Even **minor changes of the GUI** layout can lead most of the test cases to fail. The repair activity costs e.g., the Accenture company $120 million per year

  [Grechanick et al., "*Maintaining and evolving GUI-directed test scripts,*" ICSE 2009]

- A study on Adobe's Acrobat Reader found that **74%** of its test scripts get broken between **two successive releases**

  [Memon and Soffa, "*Regression Testing of GUIs,*" SIGSOFT Software Engineering Notes 28, 2003]

- The **main source** for web test cases **fragility** are the web elements **locators**

  [Leotta et al., "*Comparing the Maintainability of Selenium WebDriver Test Suites Employing Different Locators: A Case Study.,*" JAMAICA 2013]

  [Christophe et al., "*Prevalence and Maintenance of Automated Functional Tests for Web Applications,*" ICSME 2014]

Test Evolution Patterns

Structural Change

Release N

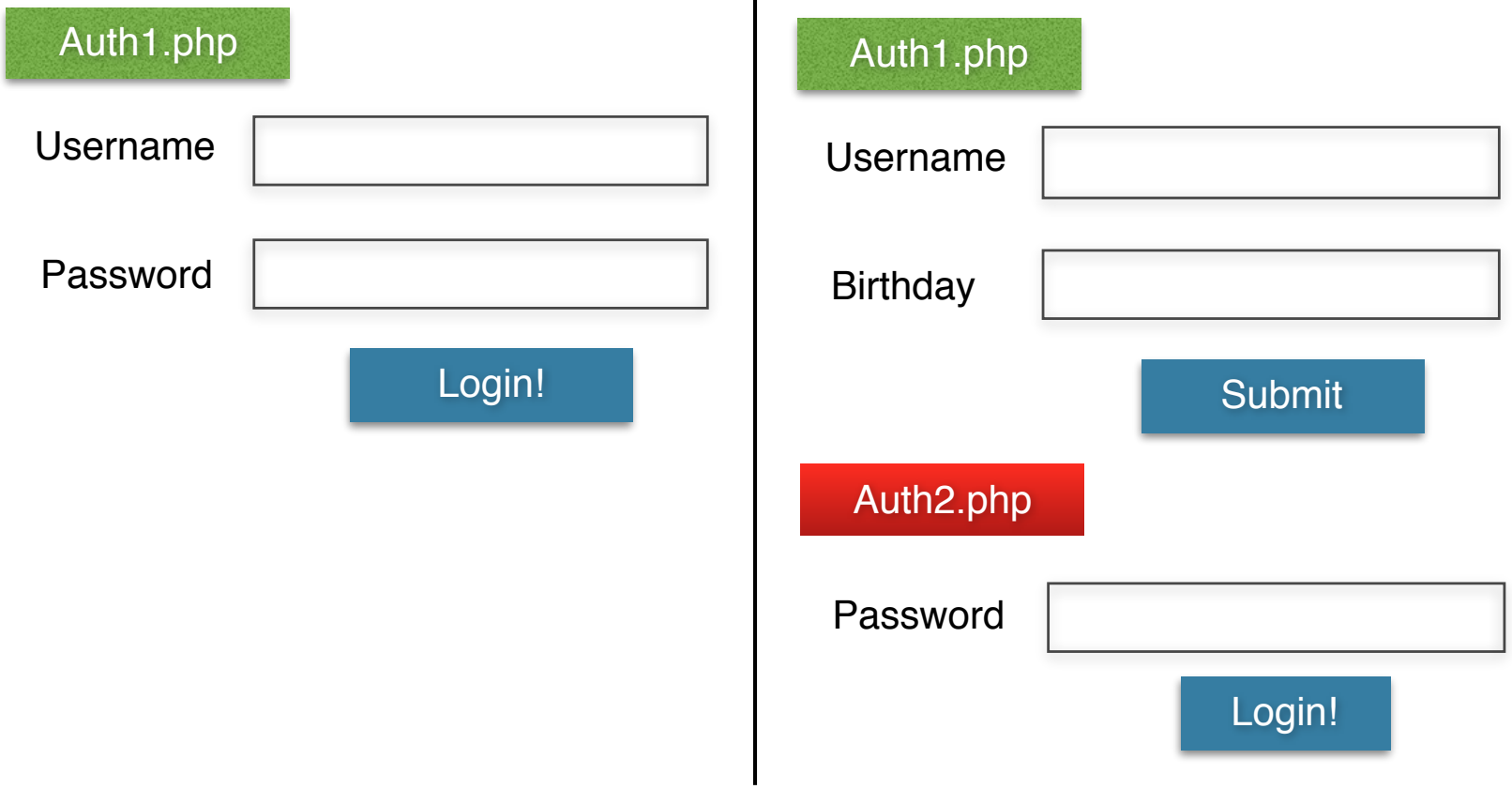

```
<form name="loginform" action="homepage.asp" method="post">
     Username : <input type="text" id="userid" name="user"><br>
     Password : <input type="text" id="pw" name="password"><br>
     <a href="javascript:loginform.submit()" id="login">Login</a>
</form>
```

Release N+1


```
<form name="loginform" action="homepage.asp" method="post">
 Username : <input type="text" id="userid" name="user"><br>
 Password : <input type="text" id="pw" name="password"><br>
 <a href="javascript:loginform.submit()" id="login">Login Now!</a>
</form>
```

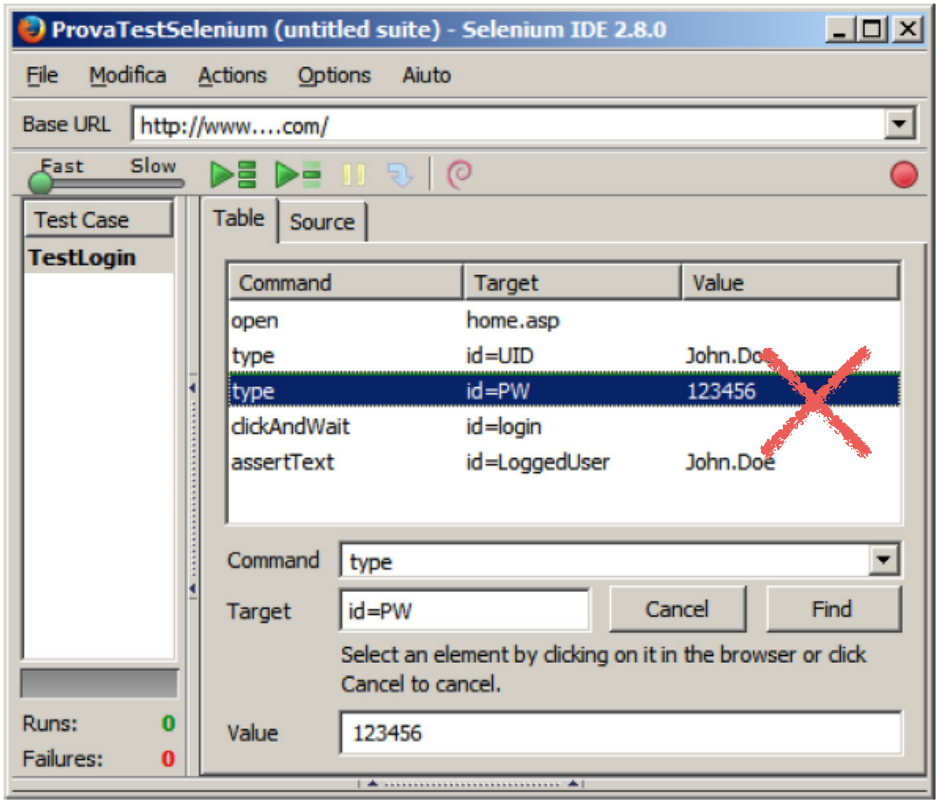
Test Evolution Patterns

Logical Change



Approaches vs Test Case Evolution

Scenario1 : C&R Approach + logical change

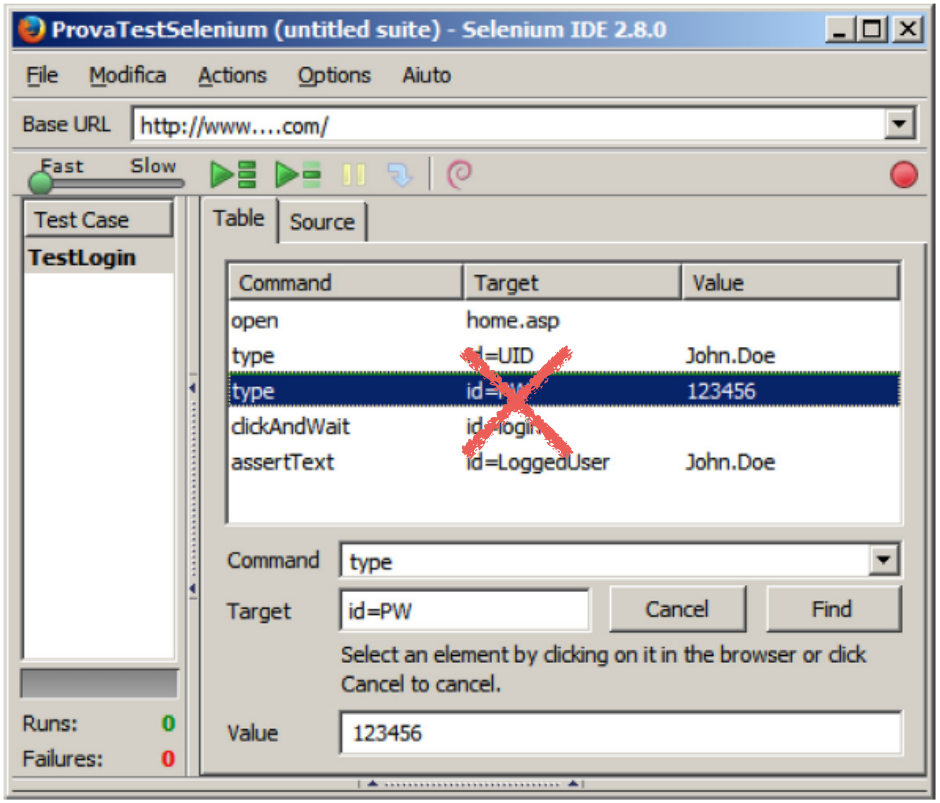


Keep the portion of the script up to the broken command.

Re-capture the execution scenario by starting from the last working command.

Approaches vs Test Case Evolution

Scenario2 : C&R Approach + structural change



Modify all the broken locators, actions, or values.

Approaches vs Test Case Evolution

Scenario3 : Programmable Approach + logical change

```
public class Login {
  @Test
  public void testLogin() {
    WebDriver driver = new FirefoxDriver();
    driver.get("http://localhost/claroline1115/index.php");
    driver.findElement(By.Id(id="user")).sendKeys("John Doe");
    driver.findElement(By.XPath("xpath="//form/
   input[2]")).sendKeys("securepassword");
    driver.findElement(By.LinkText(linkText="Enter")).click();
    Assert.assertEquals(driver.findElement(By.Id("John Doe")),
   "Login was successful");
    }
}
```

Modify the page object actions and, eventually, the tests.

```
public class IndexPage {
   private final WebDriver driver;
   @FindBy(id="user")
   private WebElement username;
   @FindBy(xpath="//form/input[2]")
   private WebElement password;
   @FindBy(linkText="Enter")
   private WebElement login;
   @FindBy(id="loggedUser")
   private WebElement loggedUser;

   public UserAccount goToCreateAccount(){
      createUserAccount.click();
      return new UserAccount(driver);
   }

   public HomePage login(String usr, String pwd) {
      username.sendKeys(usr);
      password.sendKeys(pwd);
      login.click();
      return new HomePage(driver);
   }
}
```

Approaches vs Test Case Evolution

Scenario4 : Programmable Approach + structural change

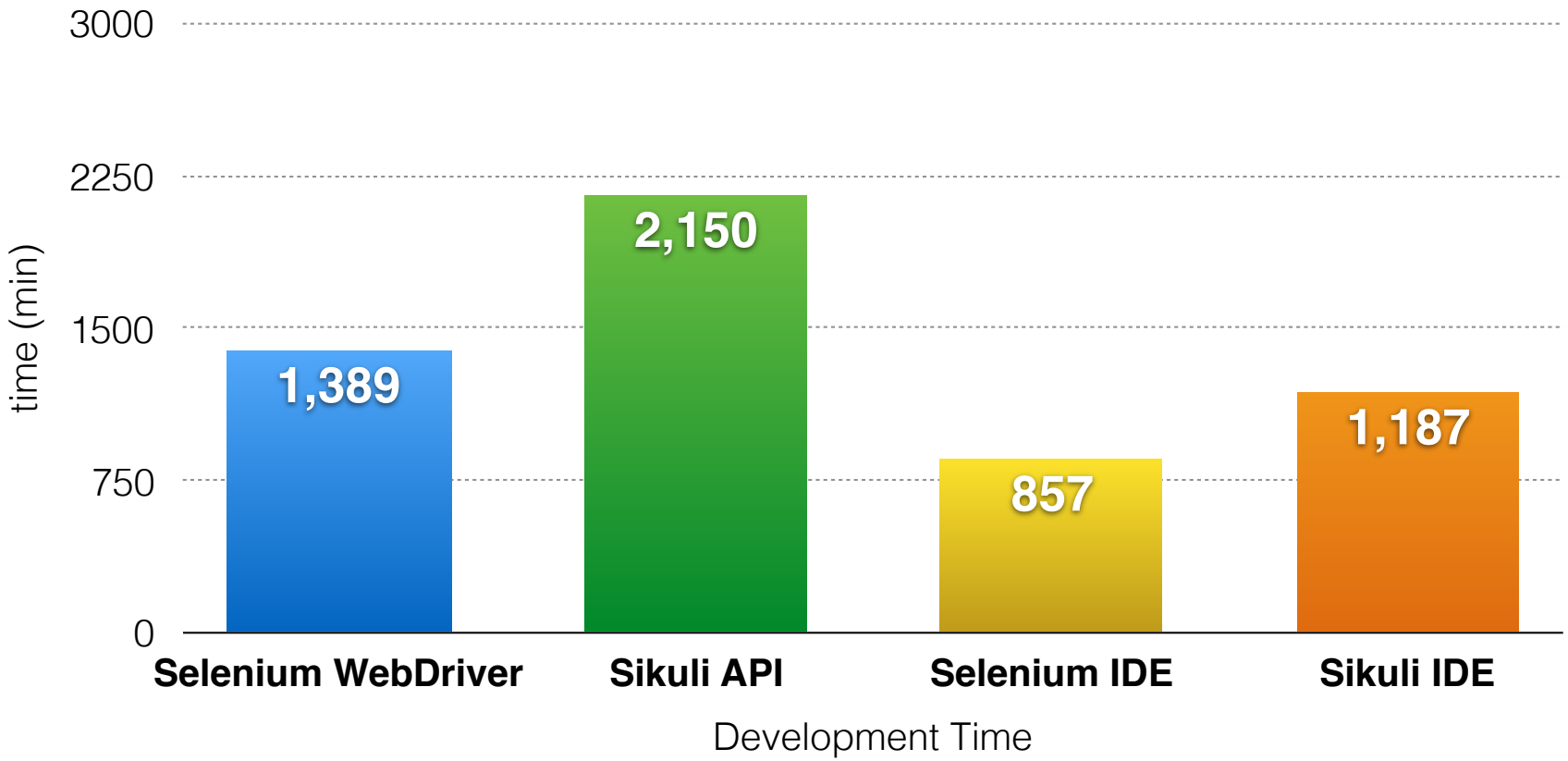
```
public class Login {
  @Test
  public void testLogin() {
    WebDriver driver = new FirefoxDriver();
    driver.get("http://localhost/claroline1115/index.php");
    driver.findElement(By.Id(id="user")).sendKeys("John Doe");
    driver.findElement(By.XPath("xpath="//form/
   input[2]")).sendKeys("securepassword");
    driver.findElement(By.LinkText(linkText="Enter")).click();
    Assert.assertEquals(driver.findElement(By.Id("John Doe")),
   "Login was successful");
    }
}
```

Modify only the page object(s).

No modifications to the tests needed!!!

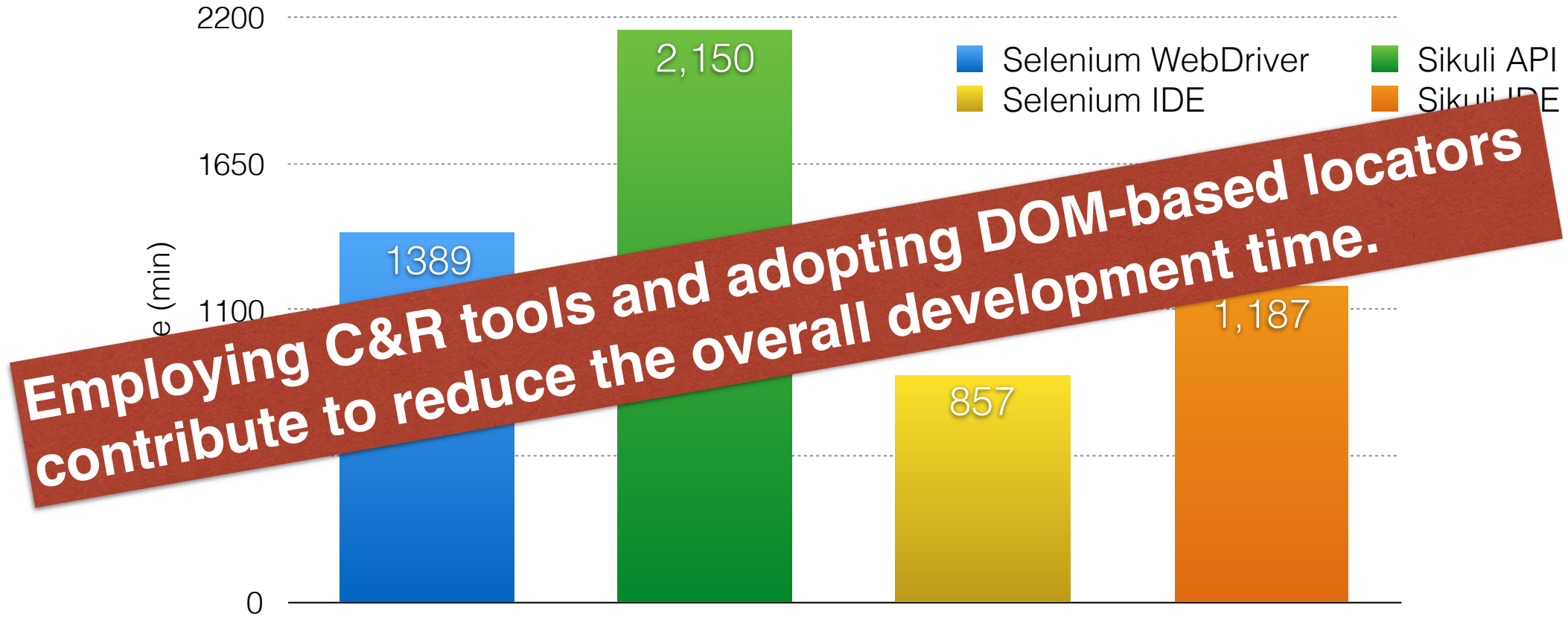
```
public class IndexPage {
   private final WebDriver driver;
   @FindBy(id="user")
   private WebElement username;
   @FindBy(xpath="//form/input[2]")
   private WebElement password;
   @FindBy(linkText="Enter")
   private WebElement login;
   @FindBy(id="loggedUser")
   private WebElement loggedUser;

   public UserAccount goToCreateAccount(){
      createUserAccount.click();
      return new UserAccount(driver);
   }

   public HomePage login(String usr, String pwd) {
      username.sendKeys(usr);
      password.sendKeys(pwd);
      login.click();
      return new HomePage(driver);
   }
}
```

Test Suite Development Cost



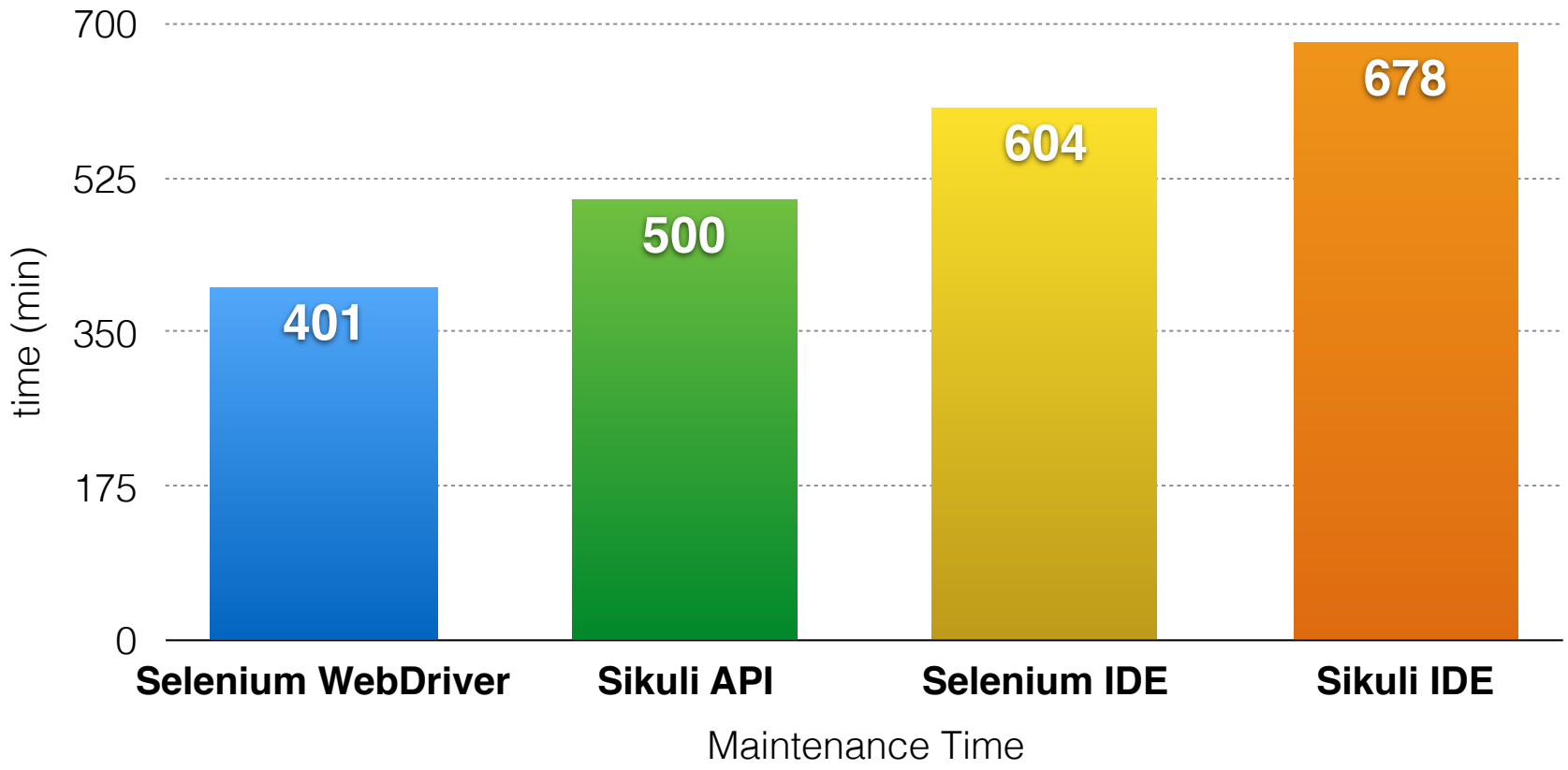
[Leotta et al., *"Approaches and Tools for Automated End-to-End Web Testing,"* Advances in Computers, Volume 101:5 (2016)]

Test Suite Development Cost



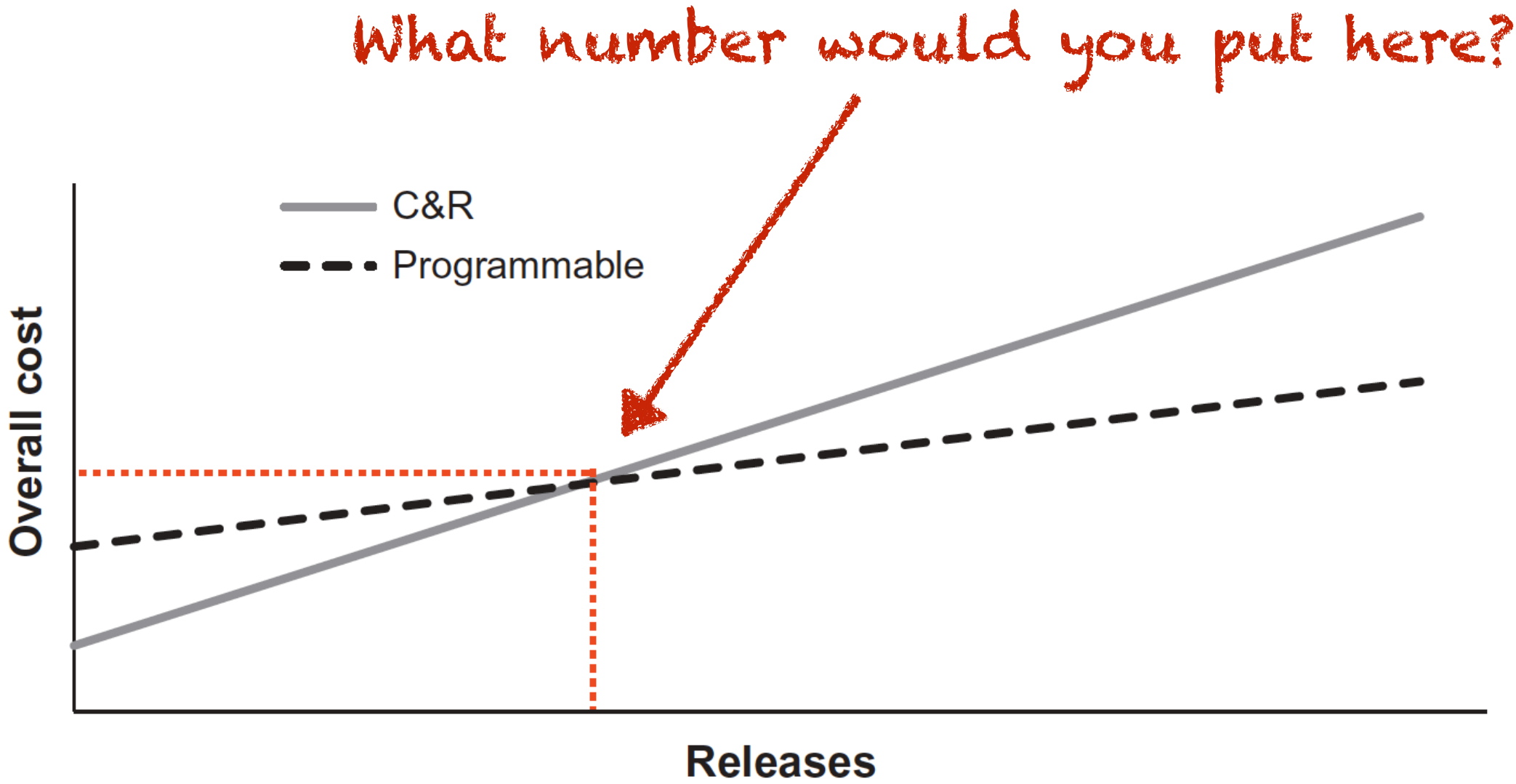
[Leotta et al., *"Approaches and Tools for Automated End-to-End Web Testing,"* Advances in Computers, Volume 101:5 (2016)]

Test Suite Maintenance Cost



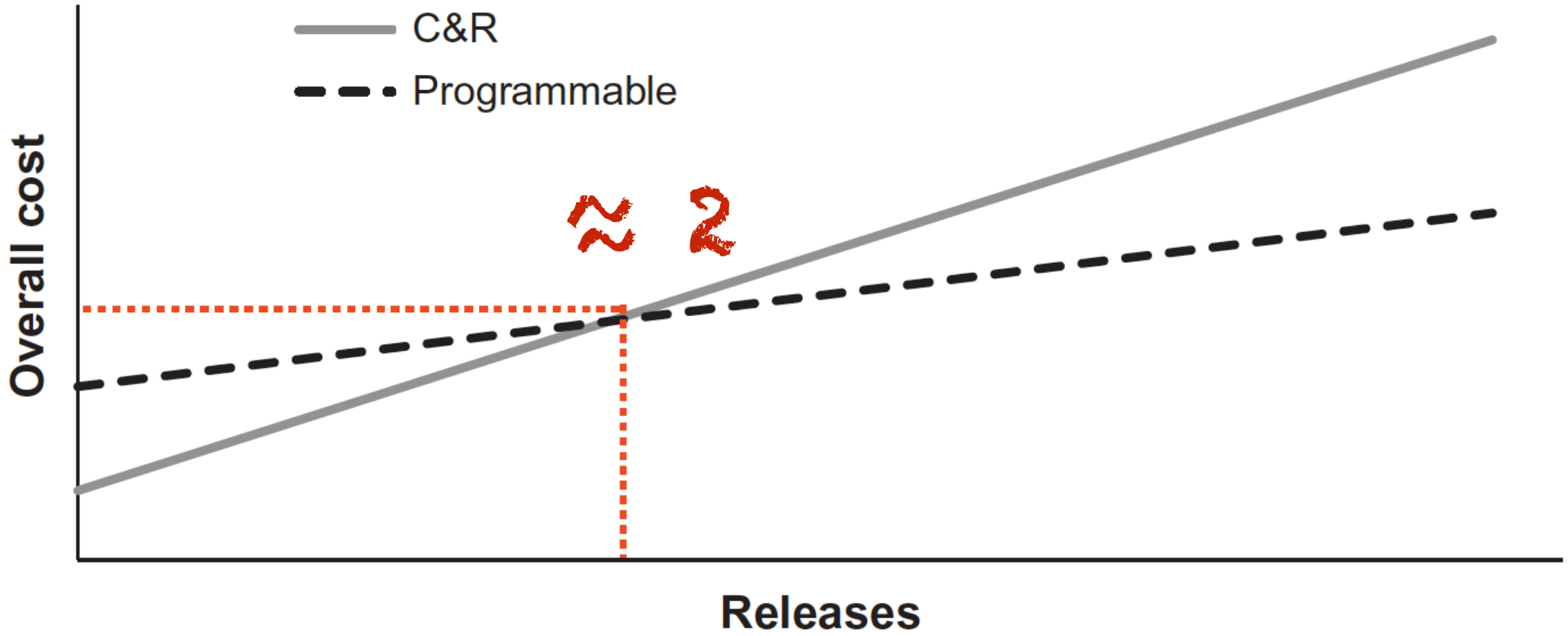
[Leotta et al., *"Approaches and Tools for Automated End-to-End Web Testing,"* Advances in Computers, Volume 101:5 (2016)]

When do they become convenient?



[Leotta et al., "*Approaches and Tools for Automated End-to-End Web Testing*," Advances in Computers, Volume 101:5 (2016)]

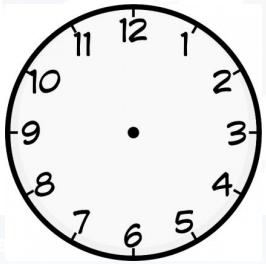
When do they become convenient?



[Leotta et al., *"Approaches and Tools for Automated End-to-End Web Testing,"* Advances in Computers, Volume 101:5 (2016)]

Why?

Augmented Maintainability



3x

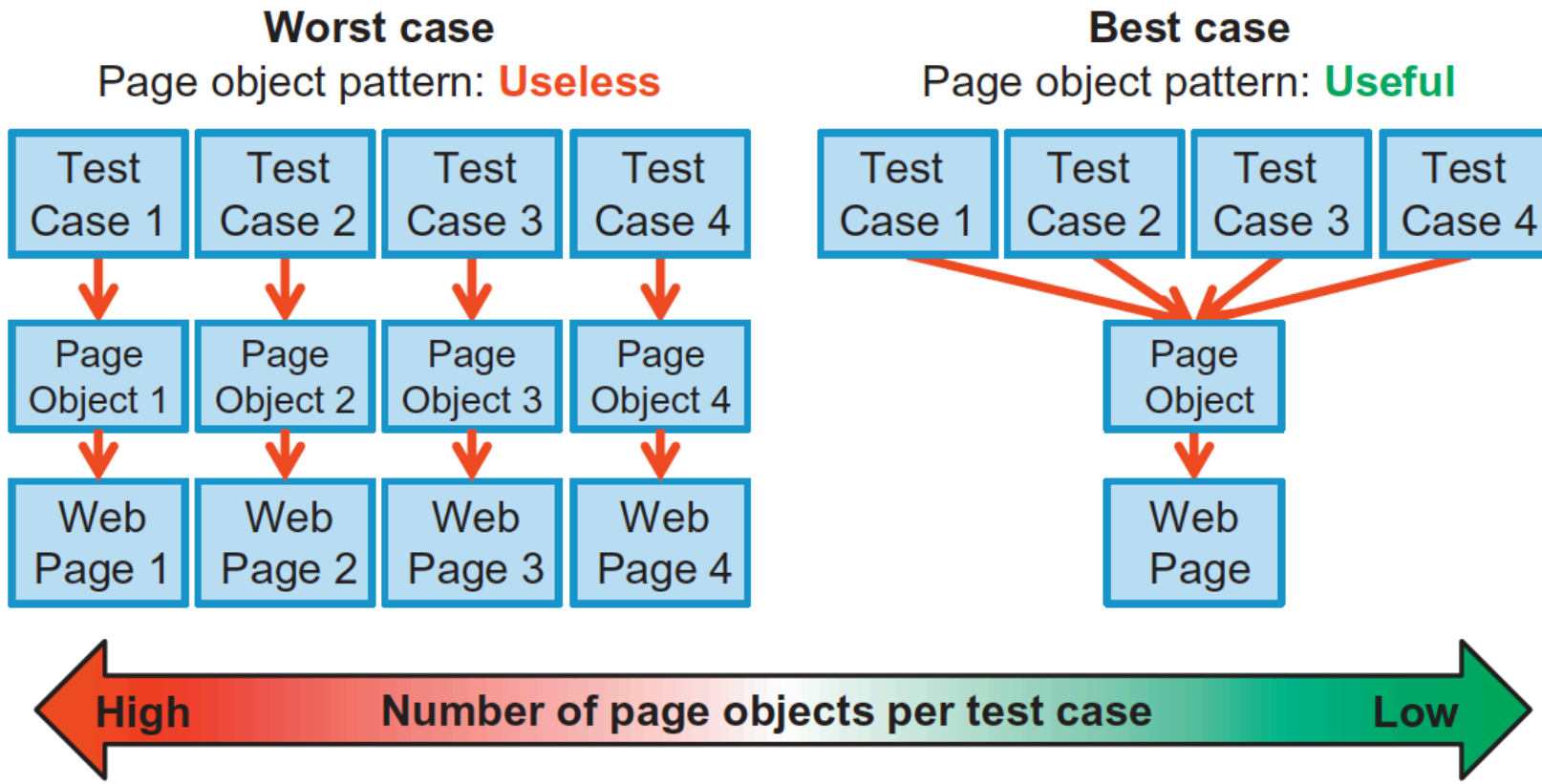
8x

[Leotta et al., *"Improving Test Suites Maintainability with the Page Object Pattern: An Industrial Case Study,"* ICSTW 2013]

```
// A simple Page Object class with Selenium WebDriver
public class IndexPage {
        private final WebDriver driver;

        @FindBy(id="user")
        private WebElement username;

        @FindBy(xpath="//form/input[2]")
        private WebElement password;

        @FindBy(linkText="Enter")
        private WebElement login;

        @FindBy(id="loggedUser")
        private WebElement loggedUser;

        public HomePage login(String usr, String pwd) {
                username.sendKeys(usr);
                password.sendKeys(pwd);
                login.click();
                return new HomePage(driver);
        }
}
```

```
@Test
public void testLogin {
        Utils.connectToWebAppUnderTest();
        IndexPage IP = new IndexPage(driver);
        HomePage hp = IP.login("John Doe", "securepassword");
        assertTrue(hp.checkLoggedUser("Doe John"));
}
```

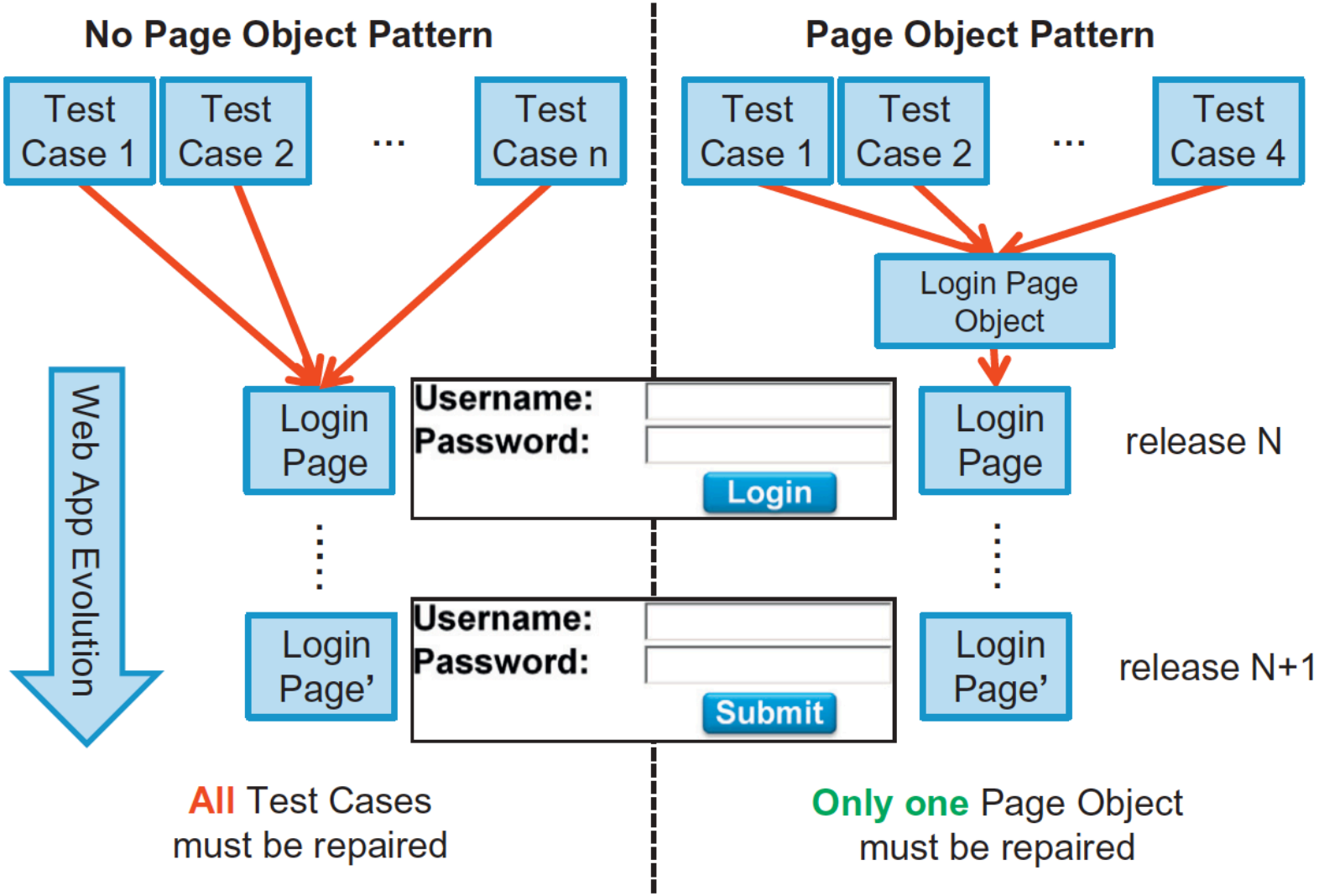
Factor 1 : Test cases vs Page objects



The web page modularity of the web application under test affects the benefits of programmable test cases.

Reusable pages objects are maintained less during software evolution.

Factor 2 : Page objects and repair effort



eg, in Selenium WebDriver the repair is:

`@FindBy(linkText="Login")` becomes `@FindBy(linkText="Submit")`

Factor 3 : Number of Locators

Elements with complex visual interaction

Add Car Test Case: Select the car manufacturer

DOM-based approach
one locator needed

```
(driver.findElement(By.id("carList")))
    .selectByVisibleText("Saab");
```

Visual approach
two locators needed

Locate the Select Web Element + Click to Open

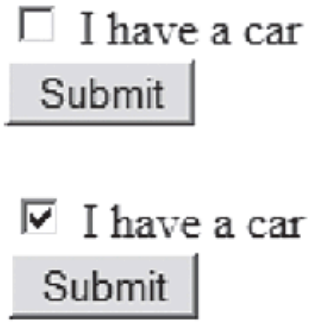
Locate the Option Value

Factor 3 : Number of Locators

Multistate Elements

Add Car Test Case
Check the Box and Submit

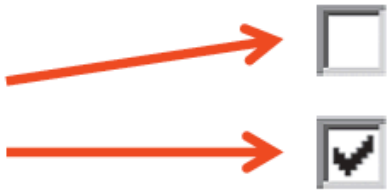
Remove Car Test Case
Uncheck the Box and Submit



The Target Web Element is always the same

(ie, the Box)

Visual approach
two locators needed

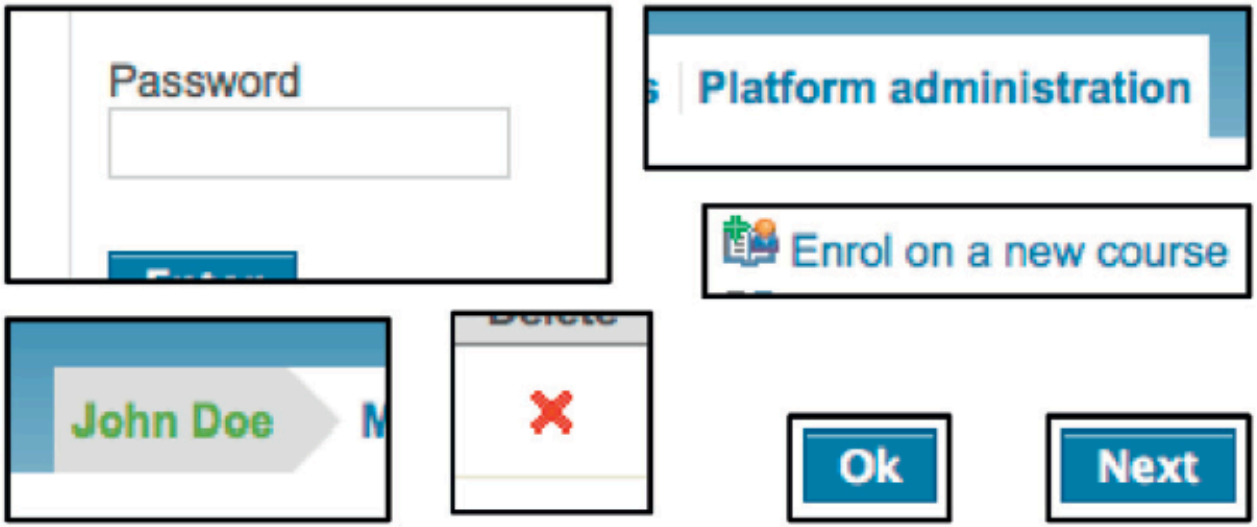


DOM-based approach
one locator needed

//input[@name="vehicle"]

Factor 3 : Number of Locators

Data Driven Test Cases



Examples of DOM-based Locators

```
//input[2]
linkText = Platform administration
//td[contains(text(),'John Doe')]
//img[@src="/web/img/edit.png?1232379976"]
//*[@id='body']/table[2]/tbody/tr/td[4]/a/img[4]
```

Other Factors

Robustness favours the DOM based

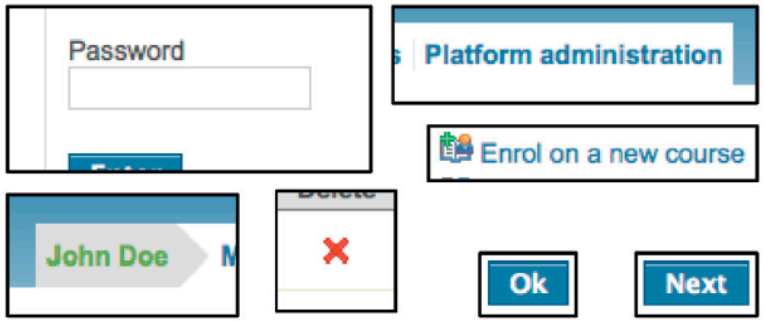
Execution Time: almost the same

Multiple Instances of the same web element

Invisible/Hidden elements

Web page loading/**synchronization**

Comprehensibility



Examples of DOM-based Locators

```
//input[2]
linkText = Platform administration
//td[contains(text(),'John Doe')]
//img[@src="/web/img/edit.png?1232379976"]
//*[@id='body']/table[2]/tbody/tr/td[4]/a/img[4]
```

Can we do better?

Hybrid Testing
join the advantages of visual and DOM-based tools
in a single testing framework

Contextual Clues
write the test script in natural language

Much still needs to be done!!!