Writing Maintainable Test Automation

Marcel de Vries
Global MD & CTO Xebia|Xpirit
@marcelv | https://fluentbytes.com

Microsoft Partner
Gold DevOps
Gold Cloud Platform
Gold Data Analytics
Gold Application Integration
Gold Application Development

GitHub Verified Partner

Xebia | Xpirit

Challenges:
Automation
Maintainability
When to test what

When to test what

Why do we test software?

When to test what

WHAT do we test?

When to test what

Categorizing tests

Business facing

Supporting the Team

Critique Product

Functional tests
Examples
Story tests
Prototypes
Simulations

1

2

Exploratory testing
scenarios
Usability testing
User acceptance testing
Alpha/Beta

3

Unit tests
Component tests
System / Integration tests

4

Performance tests
Load tests
Security testing
“~ility” testing

Technology Facing

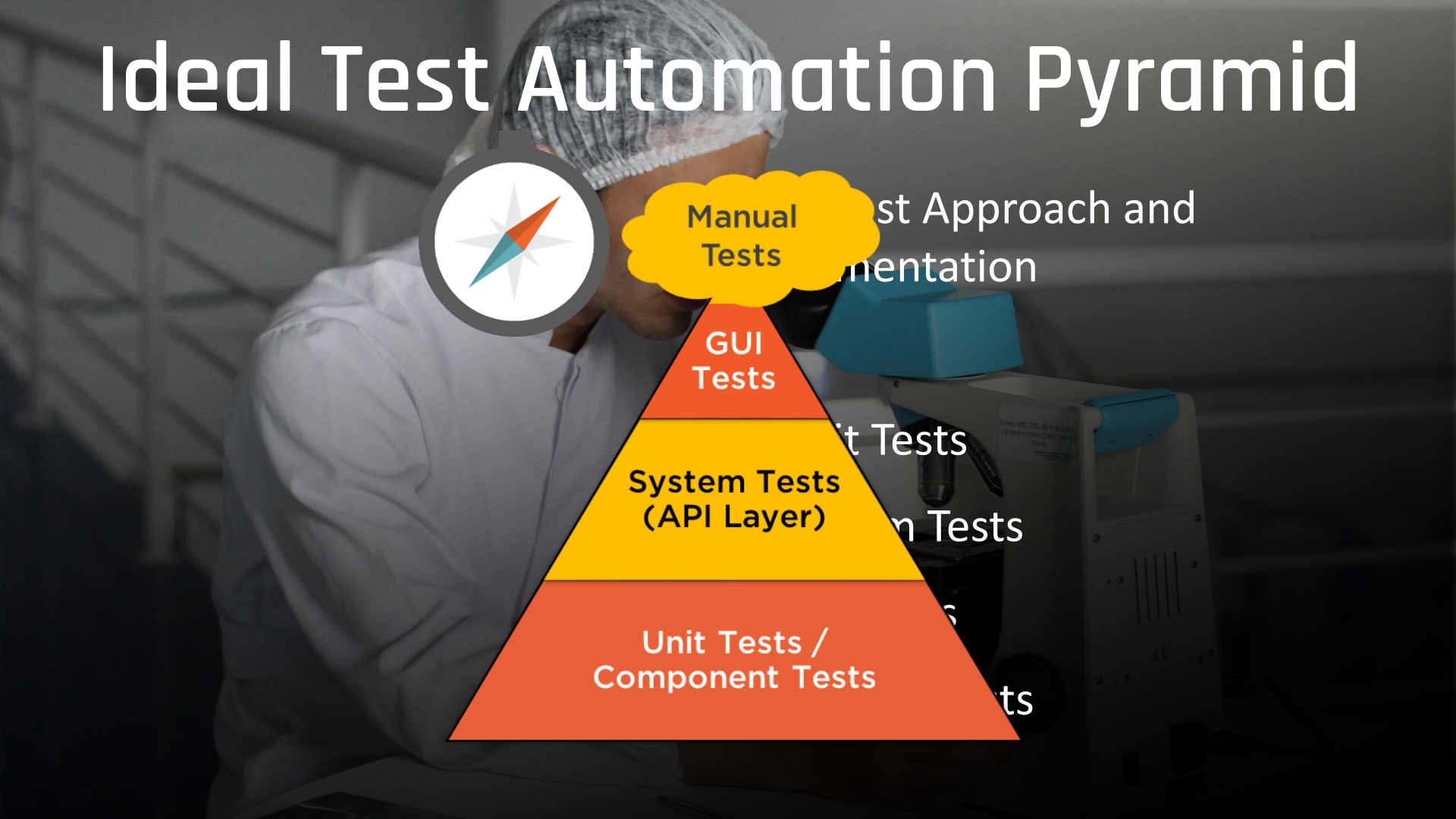
Source: Brian Marick (http://www.exampler.com/old-blog/2003/08/22/#agile-testing-project-2)

Automation

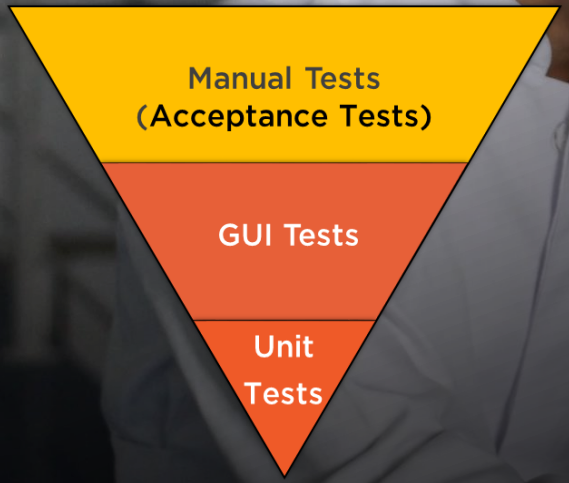
HOW do we test?

Demo: How we test

Ideal Test Automation Pyramid



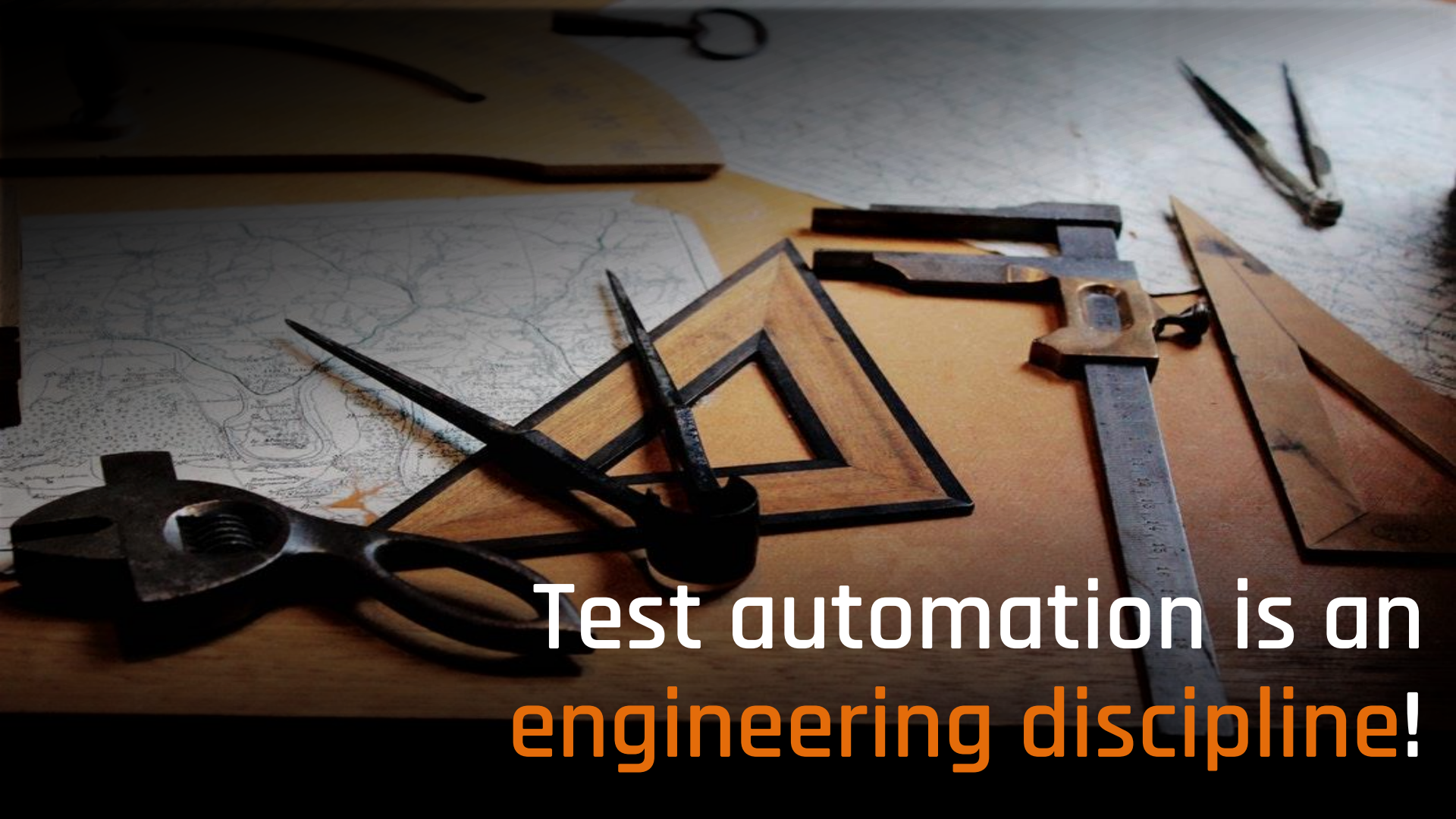
The harsh reality!



- Many tests manually executed
- Often end-to-end scenarios
- Captured as documents
- No automation or brittle automation through UI testing

Demo: How we automate

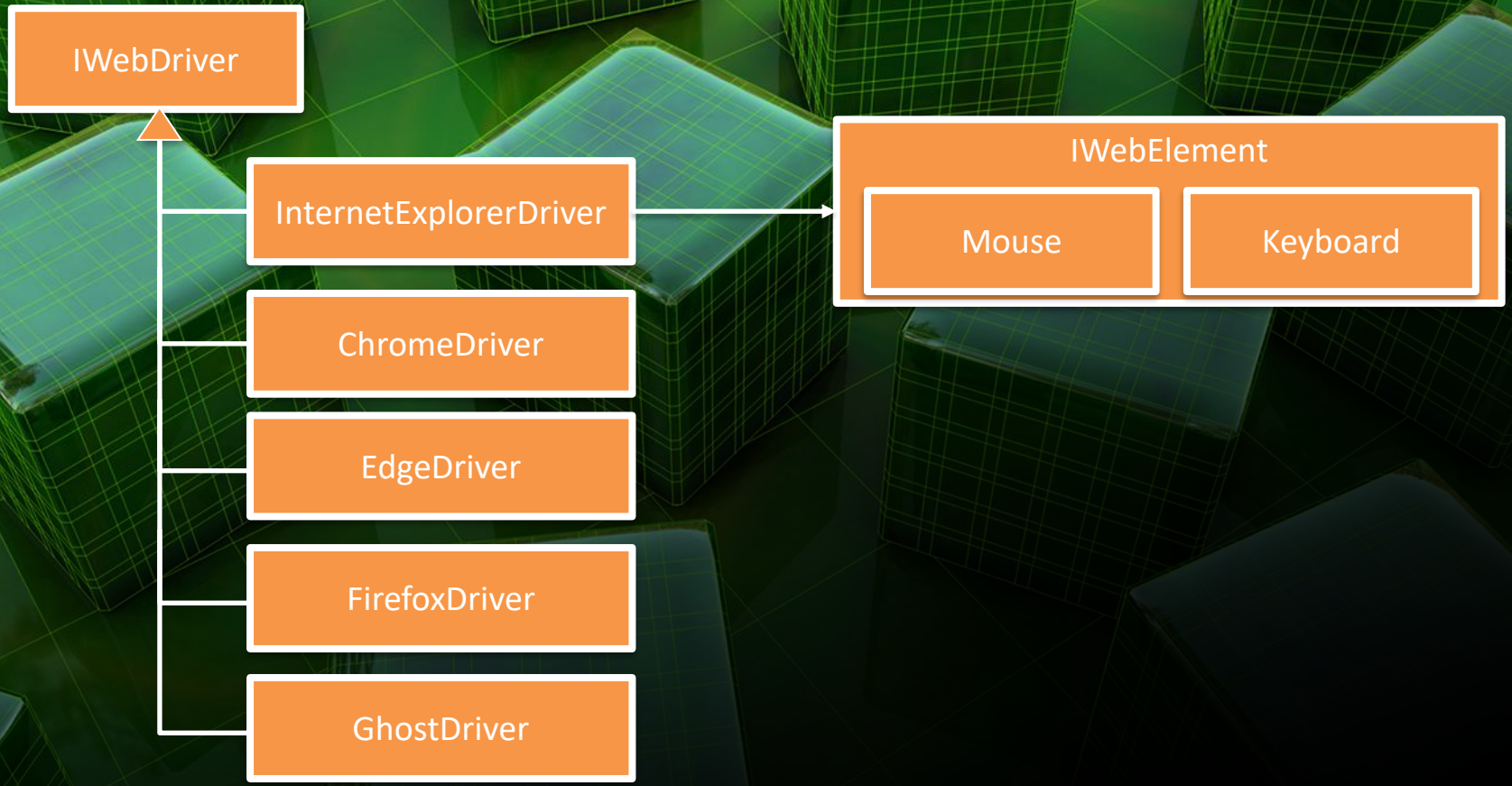

DONE!


Test automation is an
engineering discipline!


We need to apply good coding practices to Test Automation


Very short intro to the object models (Selenium)

Object Model Selenium



How Controls Are Located

```
IWebDriver driver = new EdgeDriver();
           driver.Url = "http://localhost:5266/";
           driver.Navigate();

var btn = driver.FindElement(
       By.CssSelector("tr:nth-child(2) .btn-primary"))

btn.Click();
```

Some notes on finding controls

- Finding elements by ID is most robust
- Finding on CSS or Class
  - Issue when redesign of the website
  - CssSelector is ok as long as you search for Unique stuff
- Finding on Text or Partial Text
  - Issue when different languages
- Not all drivers are the same!
  - Firefox might find by ID and chrome needs Name or vice versa…


maintainability


Maintainable tests are:
DRY
maintainability


Maintainable tests are:
SOLID
maintainability


Maintainable tests are:
DAMP
maintainability


DRY
Don't Repeat yourself!

Demo: Apply DRY


SOLID

SOLID

Single responsibility principle

There should never be more than one reason for a class to change.

SOLID

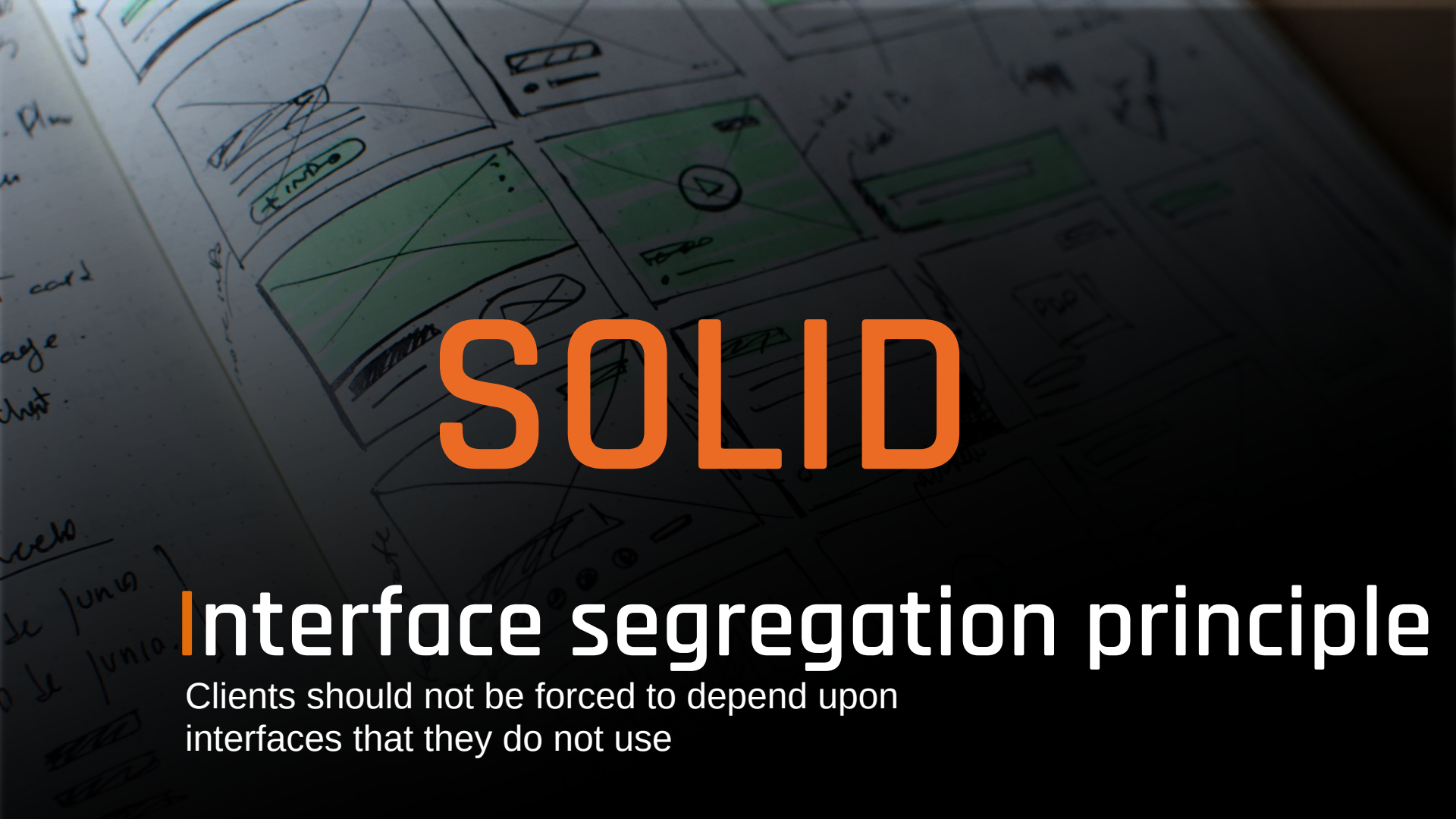
Open/closed principle

Software entities ... should be open for extension but closed for modification

SOLID

Liskov substitution principle

Functions that use pointers or references to base classes must be able to use objects of derived classes without knowing it



SOLID

Interface segregation principle

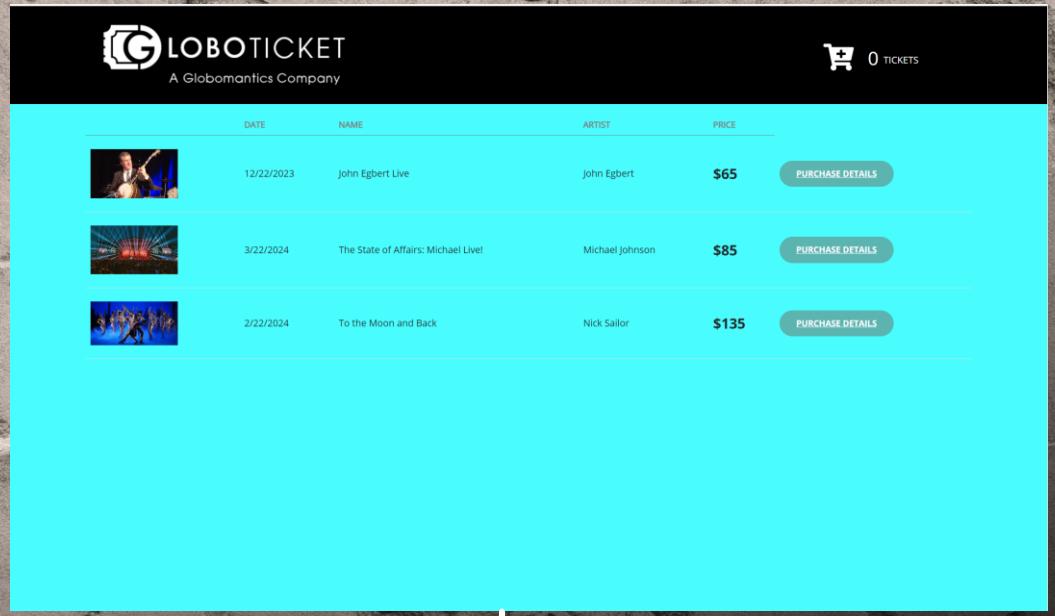
Clients should not be forced to depend upon
interfaces that they do not use

SOLID

Dependency inversion principle

Depend upon abstractions, [not] concretions.

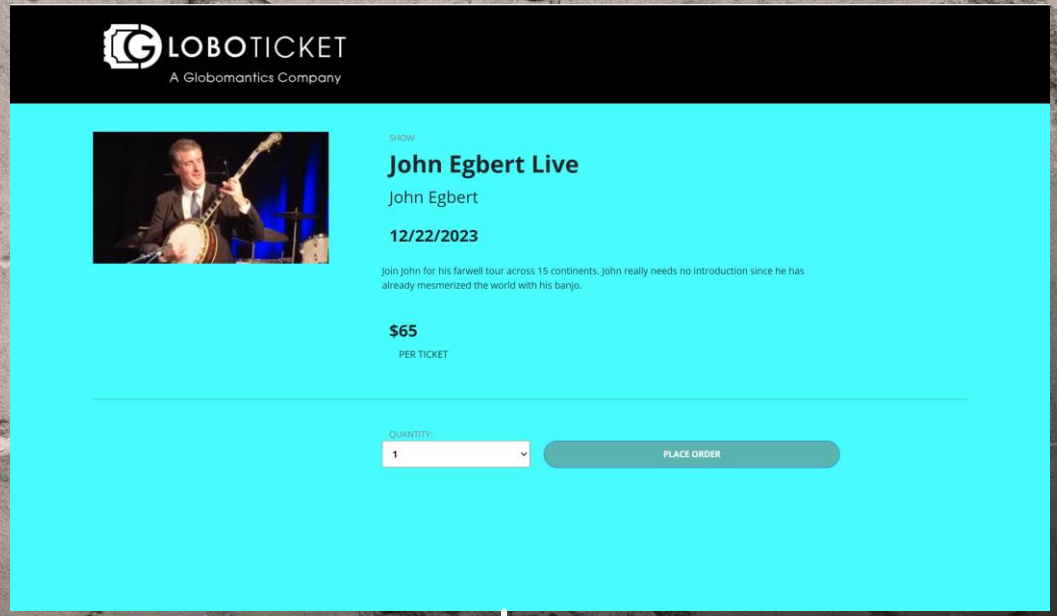

SOLID
System under test
VS
The test itself



abstraction

HomePage

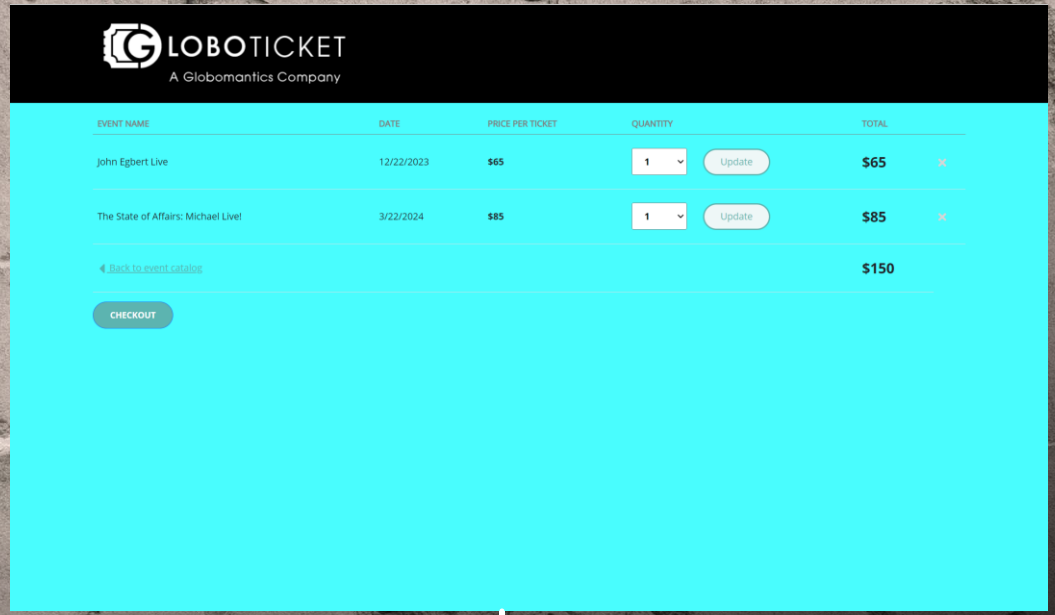
```
NavigateToHomePage()

PutProductWithNameInBasket(string name)
```



abstraction

ProductPage

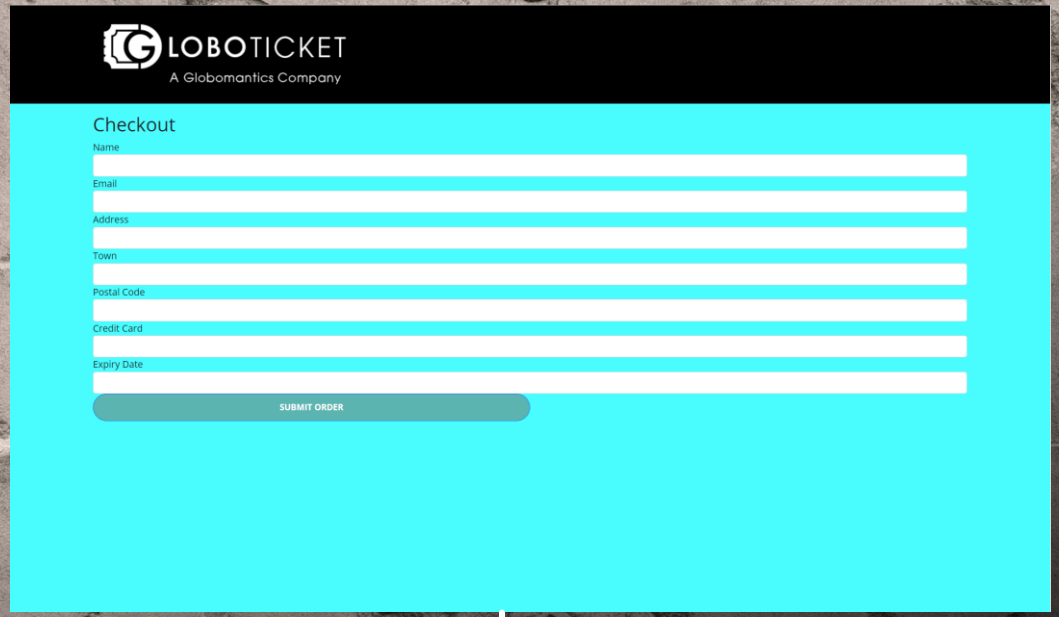
```
SetQuantity(int quantity)
PlaceOrder()
```



abstraction

ShopingBasketPage

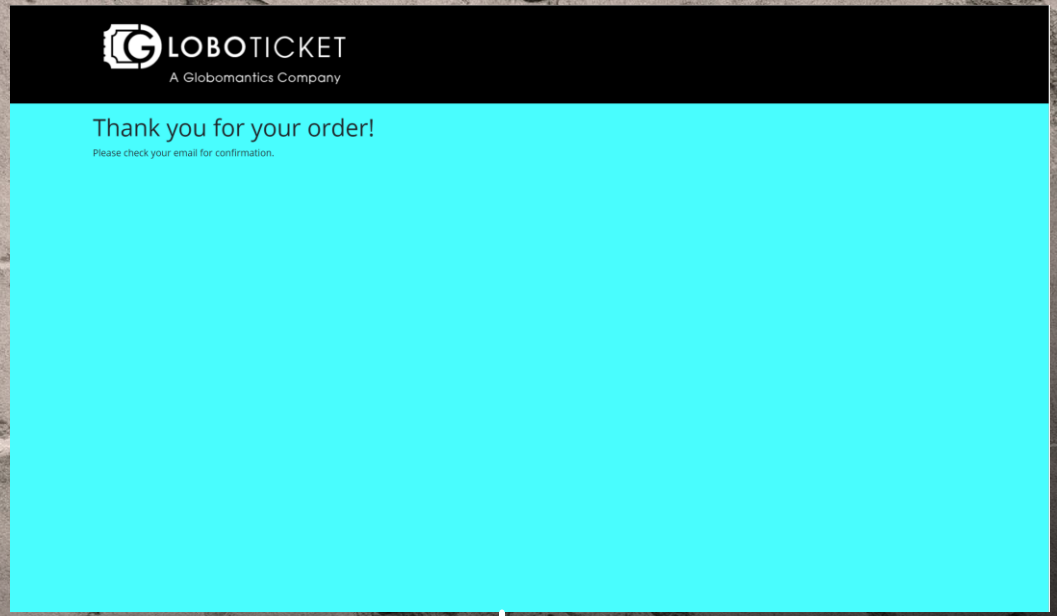
```
SetQuantity(string productName, int quantity)
RemoveItem(string productName)
BackToEventCatalog()
GotoCheckout()
```



abstraction

CheckoutPage

```
SubmitOrder(CustomerInfo customer)
```



abstraction

ThankYouPage

```
IsThankYouMessageShown()
```

Demo: Apply the S in SOLID


DAMP
Descriptive And Meaningful Phrases!

CONSIDER THE FOLLOWING TEST, WHAT ARE WE TESTING?

```
driver.Navigate().GoToUrl("http://localhost:5266/");
driver.Manage().Window.Maximize();

driver.Navigate().GoToUrl("http://localhost:5266/");
driver.Manage().Window.Size = new System.Drawing.Size(1616, 1034);
driver.FindElement(By.LinkText("PURCHASE DETAILS")).Click();
driver.FindElement(By.CssSelector(".btn")).Click();
driver.FindElement(By.LinkText("Back to event catalog")).Click();
driver.FindElement(By.CssSelector("tr:nth-child(2) .btn-primary")).Click();
driver.FindElement(By.Name("TicketAmount")).Click();
{
    var dropdown = driver.FindElement(By.Name("TicketAmount"));
     dropdown.FindElement(By.XPath("//option[. = '3']")).Click();
}
driver.FindElement(By.CssSelector(".btn")).Click();
driver.FindElement(By.CssSelector("tr:nth-child(2) .cancelIcon")).Click();
driver.FindElement(By.CssSelector(".navbar-brand > img")).Click();
var element =
driver.FindElement(By.XPath("/html/body/header/div/div/div[2]/div/div/div[2]/span[1]"));
Assert.IsTrue(element != null);
Assert.IsTrue(element.Text == "1");

driver.Close();
```

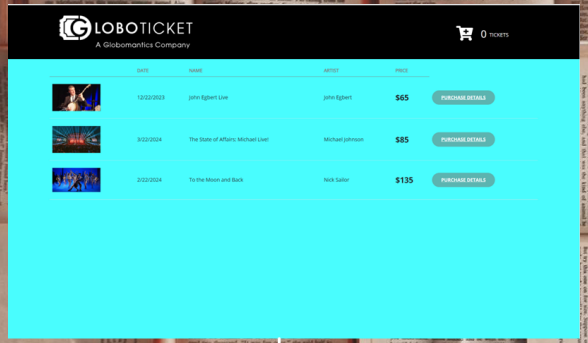
NOW, CONSIDER THE FOLLOWING TEST, WHAT ARE WE TESTING?

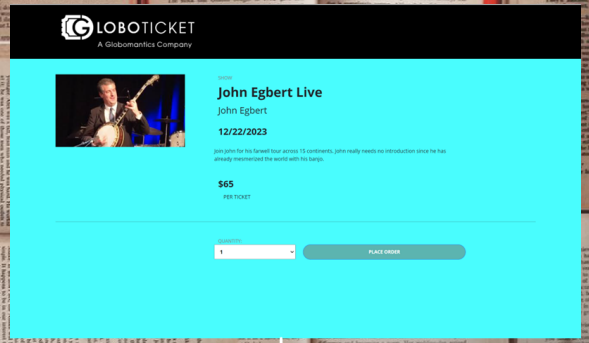
```
string productName = "John Egbert";
var homepage = new HomePage(driver);

Assert.IsTrue(
        homepage.PutProductWithNameInBasket(productName)
        .PlaceOrder()
        .SetQuantity(productName, 3)
        .GotoCheckout()
        .SubmitOrder(new CustomerMarcel())
        .IsThankYouMessageShown());
```

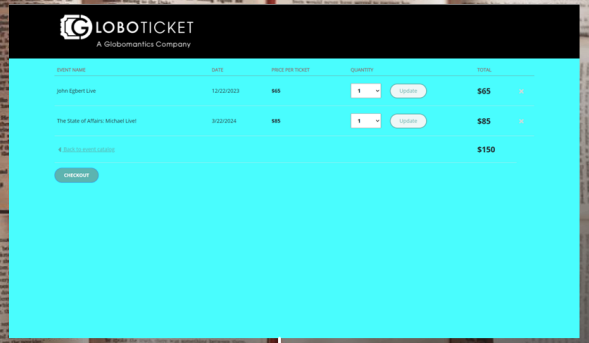
DESCRIPTIVE AND MEANINGFUL PHRASES!


But how???


Page Object Pattern







abstraction

HomePage

```
NavigateToHomePage()

PutProductWithNameInBasket(string name)
```

abstraction

ProductPage

```
SetQuantity(int quantity)
PlaceOrder()
```

abstraction

ShopingBasketPage

```
SetQuantity(string productName, int quantity)
RemoveItem(string productName)
BackToEventCatalog()
GotoCheckout()
```

Each Method Returns a Page as a result.

DAMP Tests

Page Object Pattern

- For each page create a class that abstracts the interaction with the page
- For each action create an action method that returns a page object
- For each verification create a method that executes the verification and returns true or false
- Never expose page internals, like controls, etc.!

Why Is This Good for Maintainability?

- Each page object only has knowledge of only one functional part of the system
  - S in SOLID
- Changes in the application don't affect the test scenario
  - Only the implementation behind the action and verification methods
- One change in the application will not ripple through my whole test automation framework

Applying This Principal to Your Development Team

- You can further improve maintainability of the tests
- Make the page object part of the *definition of done* of the web pages they build or modify
  - Developers are now responsible for making controls on the page better identifiable
    - Thus they will create better HTML structures & better identifiable objects
  - No test scenario will break anymore

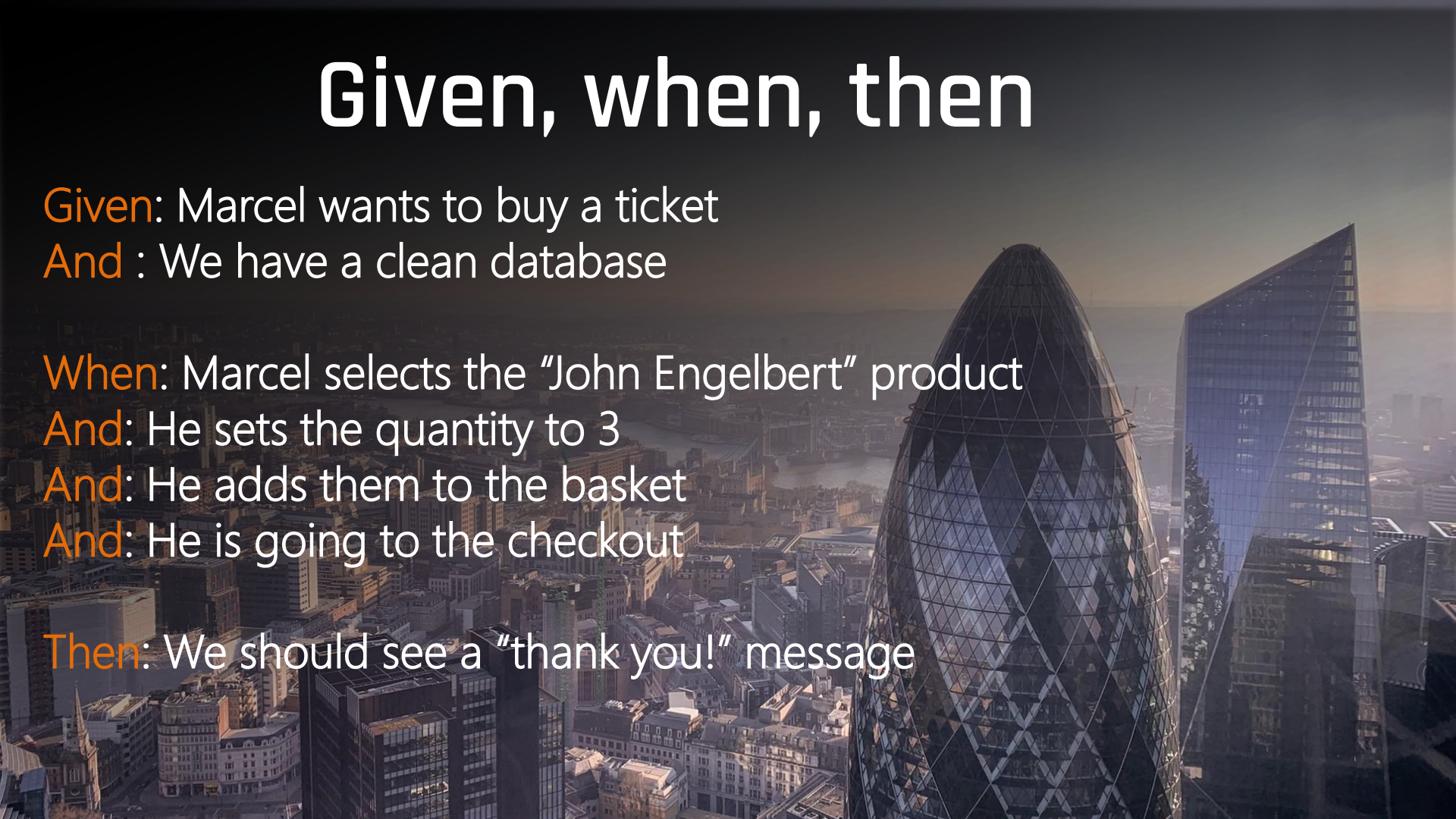
Demo: Apply DAMP

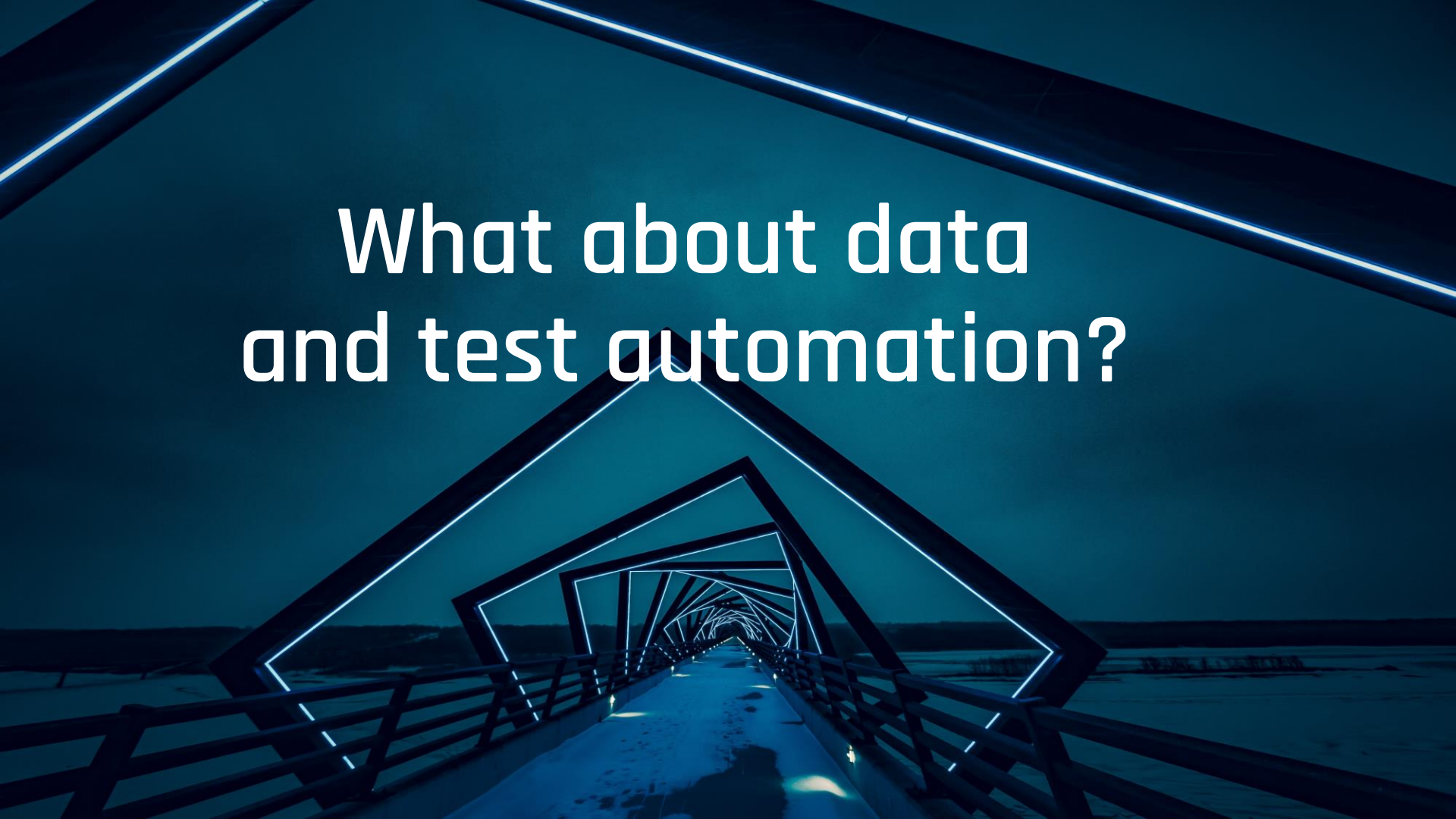
Given, when, then

Given: Marcel wants to buy a ticket
And : We have a clean database

When: Marcel selects the “John Engelbert” product
And: He sets the quantity to 3
And: He adds them to the basket
And: He is going to the checkout

Then: We should see a “thank you!” message




What about data
and test automation?

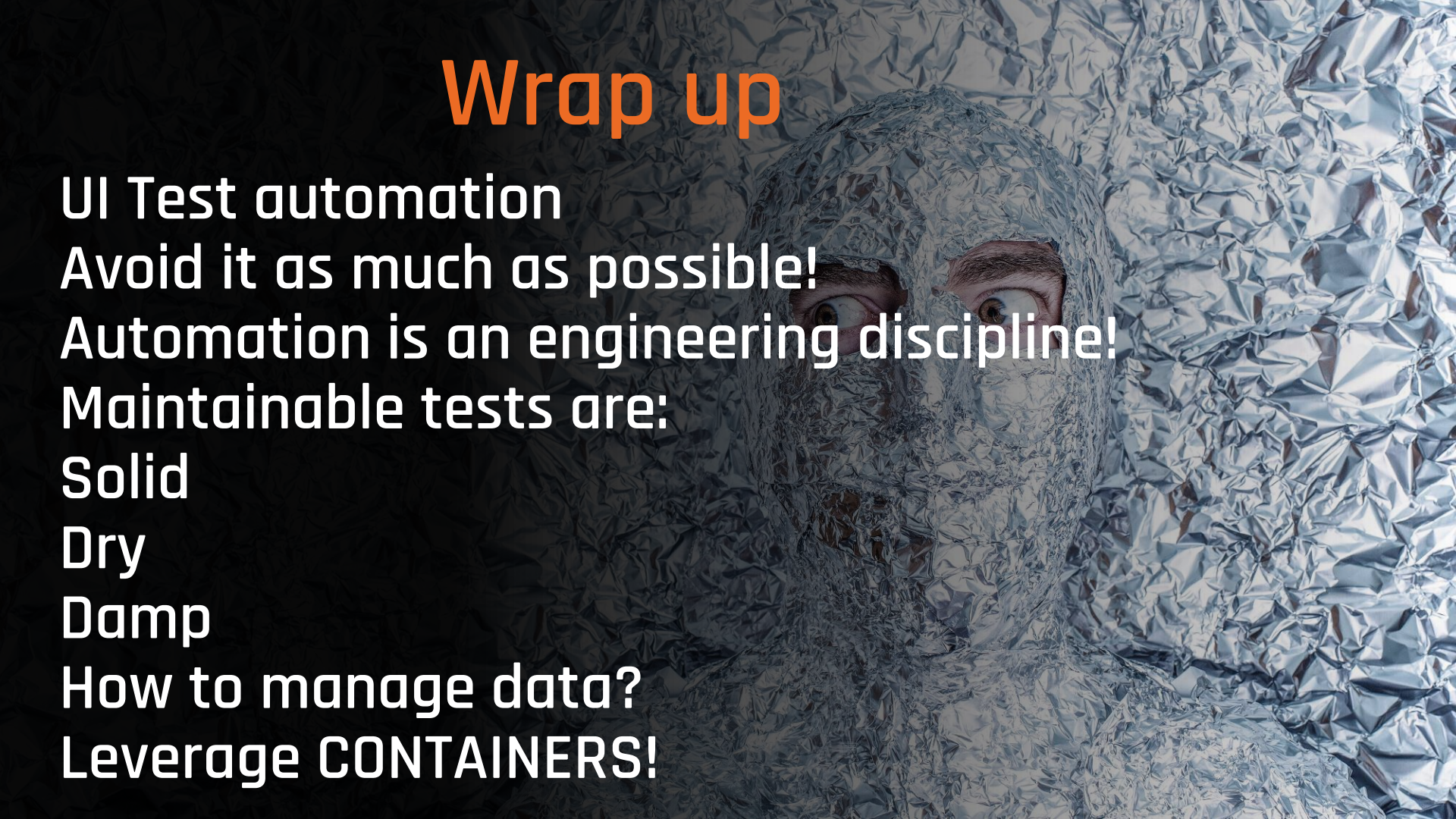
Leverage containers!

Containers provide immutable infrastructure

For production you should not store data in a container but……

For testing purposes, having the data in the container is a bliss!
Start a container for a given scenario and you can always run it over and over again
All side effects are discarded by starting a new container!

Wrap up



UI Test automation
Avoid it as much as possible!
Automation is an engineering discipline!
Maintainable tests are:
Solid
Dry
Damp
How to manage data?
Leverage CONTAINERS!

Microsoft® Most Valuable Professional

thanks!

@marcelv

Fluentbytes.com

Marcel de Vries

CTO

Xpirit Netherlands

@marcelv http://fluentbytes.com

Attributions

Unsplash.com, pictures used for backgrounds

giphy.com for animated gifs

do epic Sh!t

Xpirit