

CodedUI – Gut vorbereitet ist halb getestet

Nico Orschel, MVP @ AIT, DE

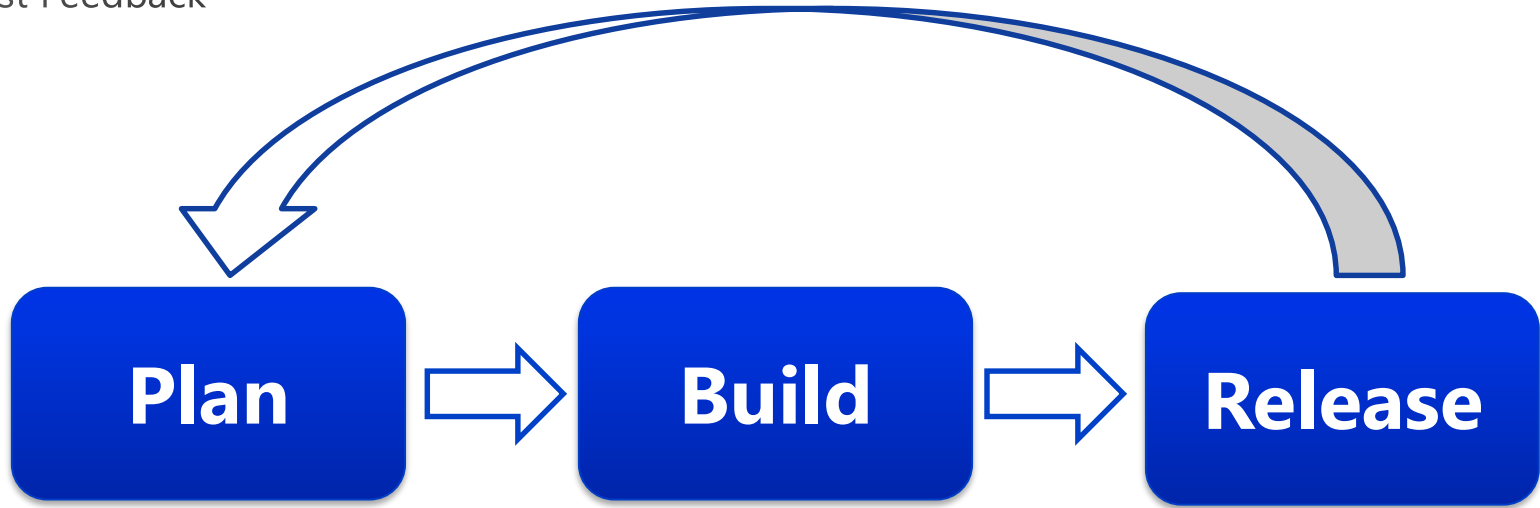
Marc Müller, MVP @ 4tecture, CH



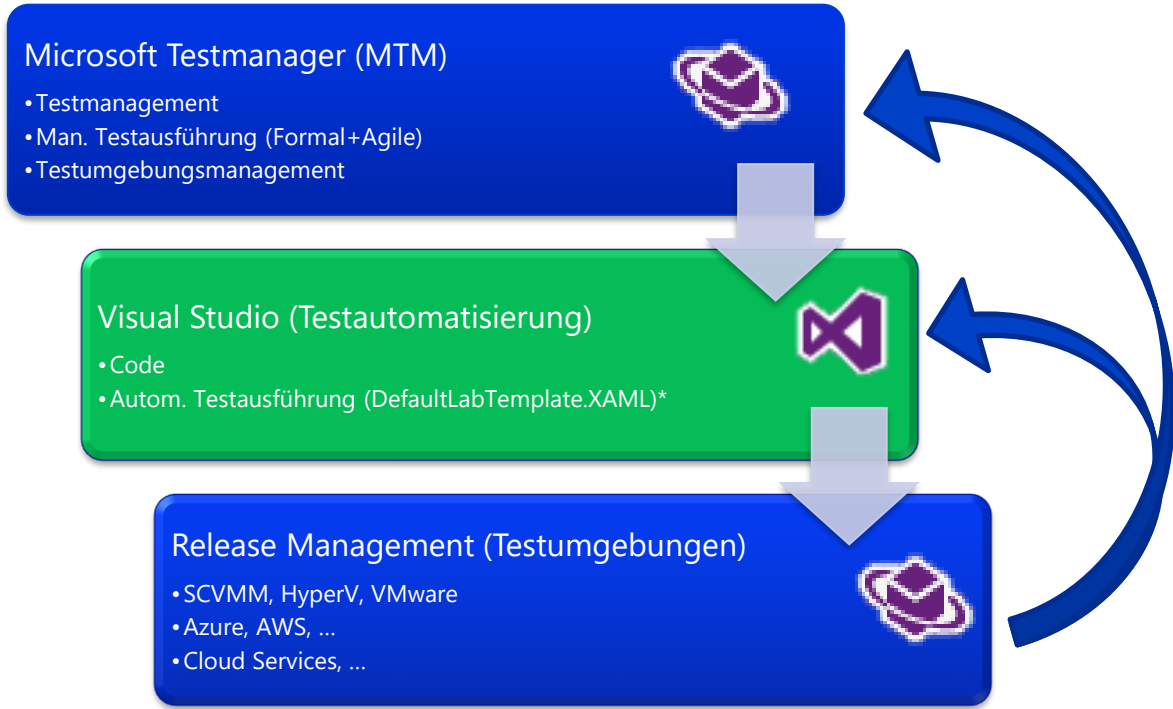
Warum Testautomatisierung

Ziele der modernen Software-Entwicklung

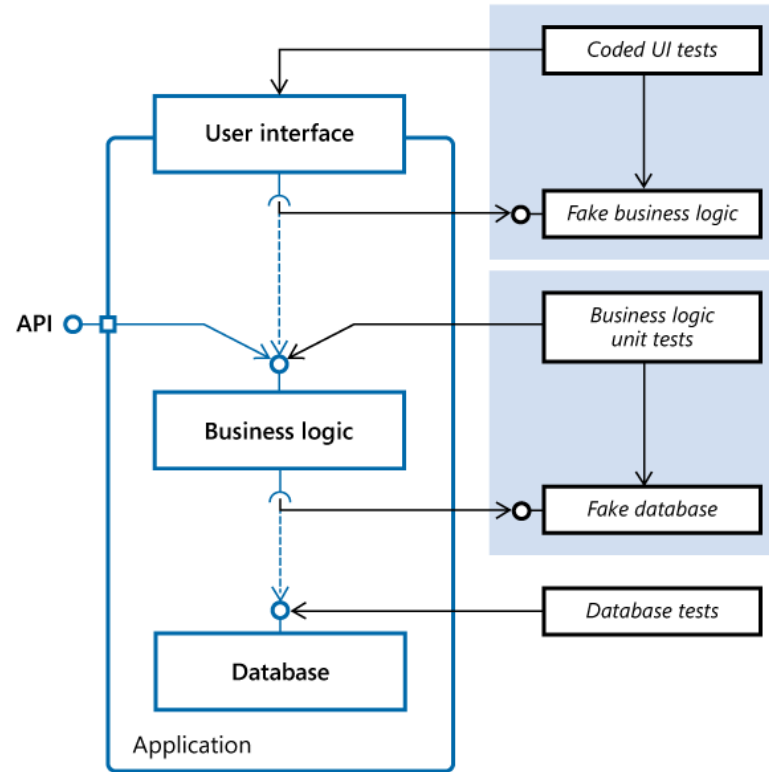
- Cycle-Times reduzieren
- Fast Feedback



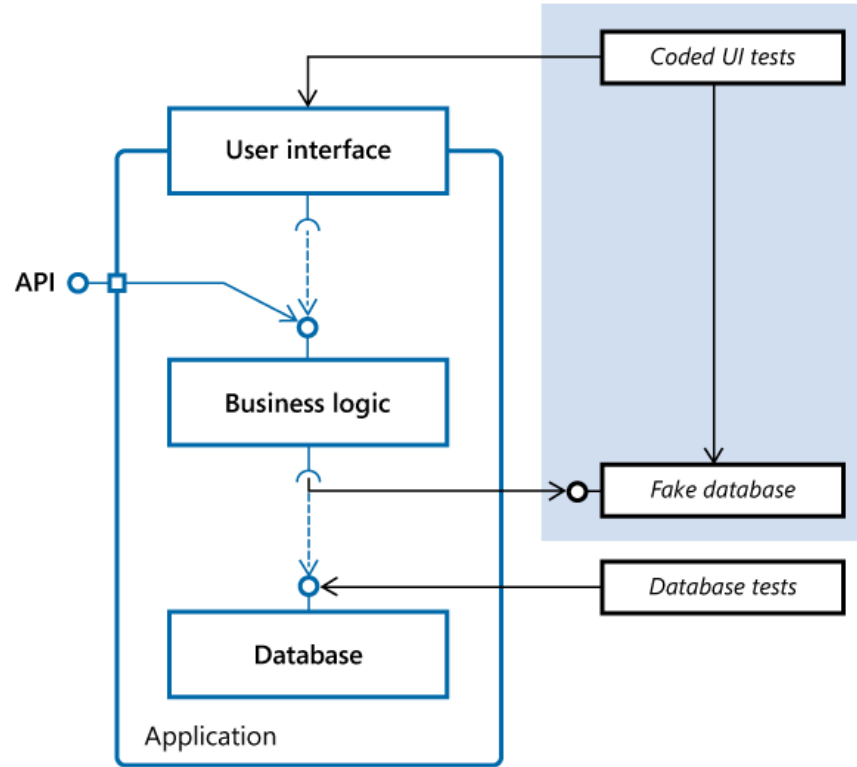
Wo sind wir in der TFS/VS Welt?



Testing auf unterschiedlichen Ebenen



CodedUI in kleinen Projekten



Warum CodedUI

CodedUI ist eine Testautomatisierungsplattform

CodedUI unterstützt die meisten Windows UI Technologien

Gleiche API für Rich Client und Web Applikationen

Running Coded UI Tests

Ausführung möglich über:

- Visual Studio Test Explorer
- Lab Management (Standard + SCVMM Env.) / Release Management
- MTM
- Build

Weitere Informationen

- <http://blog.aitgmbh.de/2010/09/13/codedui-tests-ohne-lab-management-ausfhren/>

Unterstützte Plattformen

Technology	Support	Comment
IE 8-11	Yes	Older IEs are not supported anymore
WinForms 2.0+, WPF	Yes	3rd party controls could be a problem
Windows Store Apps	Yes	XAML based Store Apps
Chrome, FireFox	Yes (> VS 2012 Upd. 4)	Latest version via Selenium Adapter
Silverlight	No	Unofficial support for Silverlight v4/v5 (VS 2010 / 2012 only)
Flash, Java	No	3rd Party-Vendor like Ranorex
Windows Win32 / MFC	Partially	May work with known issues
SharePoint	Yes (> VS 2012 Upd. 2)	2007+
Windows 8.1 Store Apps	Yes	> = VS 2013
Windows Phone 8.1. Apps	Yes	> = VS 2013 Update 2

UI-Ansteuerungstechnologien

Technology	UI Test Implementation Model
Windows Forms	Microsoft Active Accessibility (MSAA)
Windows Presentation Foundation	UI Automation (UIA)
Internet Explorer	MSHTML
Firefox (VS 2010)	JavaScript and Firefox DOM
Firefox / Chrome (VS 2012+)	Selenium
Silverlight	Code Injection and reflection

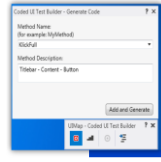
Multiple ways to create UI tests

Record and Playback



- Default Approach
- Poor Code Generation
- Poor Maintainability

Record UI Maps



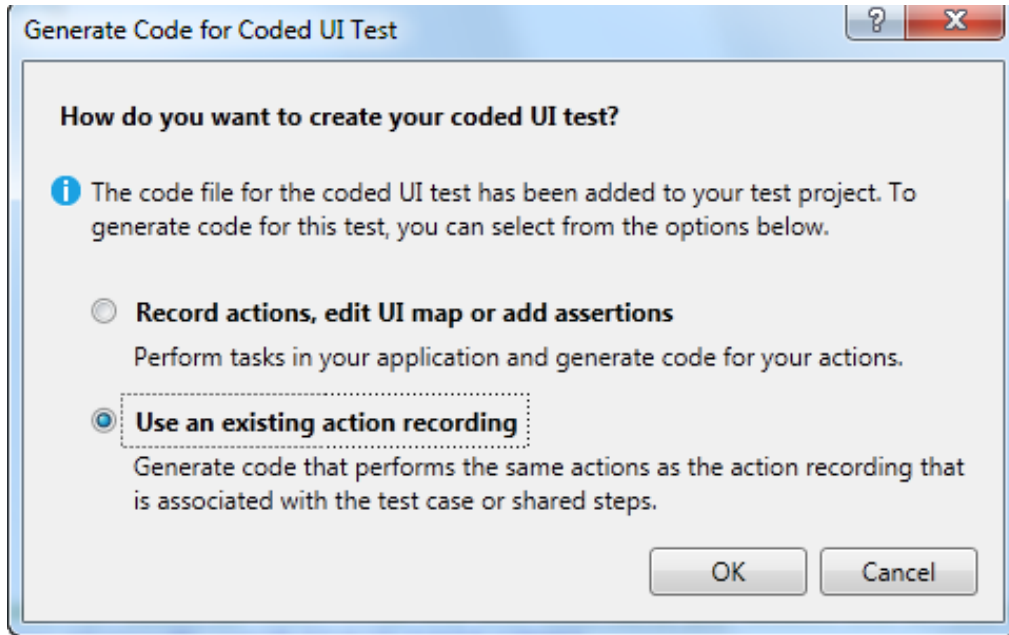
- Better Maintainability
- Supports large test repositories

Hand Code Tests

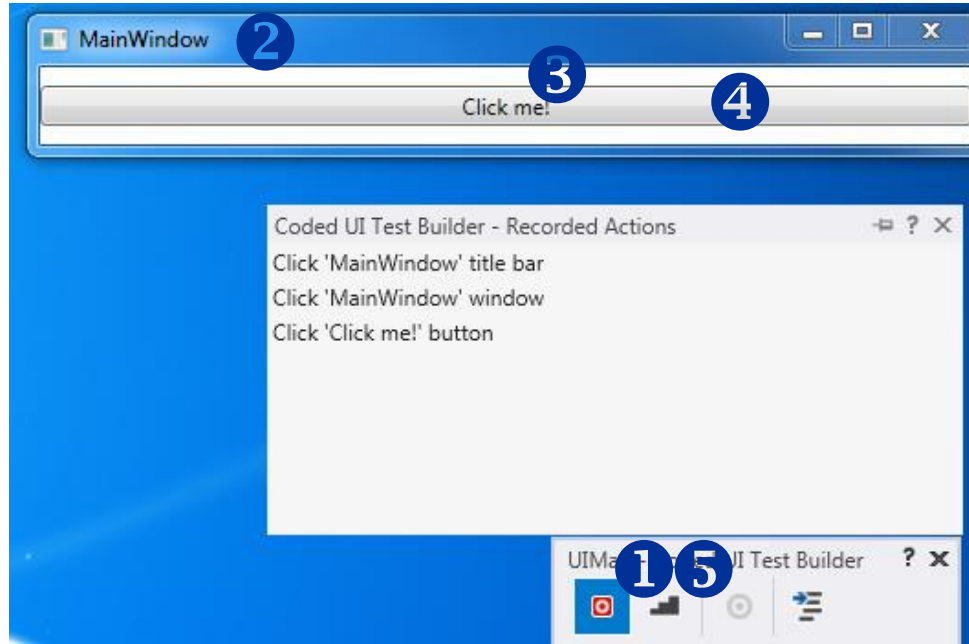


- Expert Mode
- Cleanest Automation Model

How-to generate CodedUI Tests?

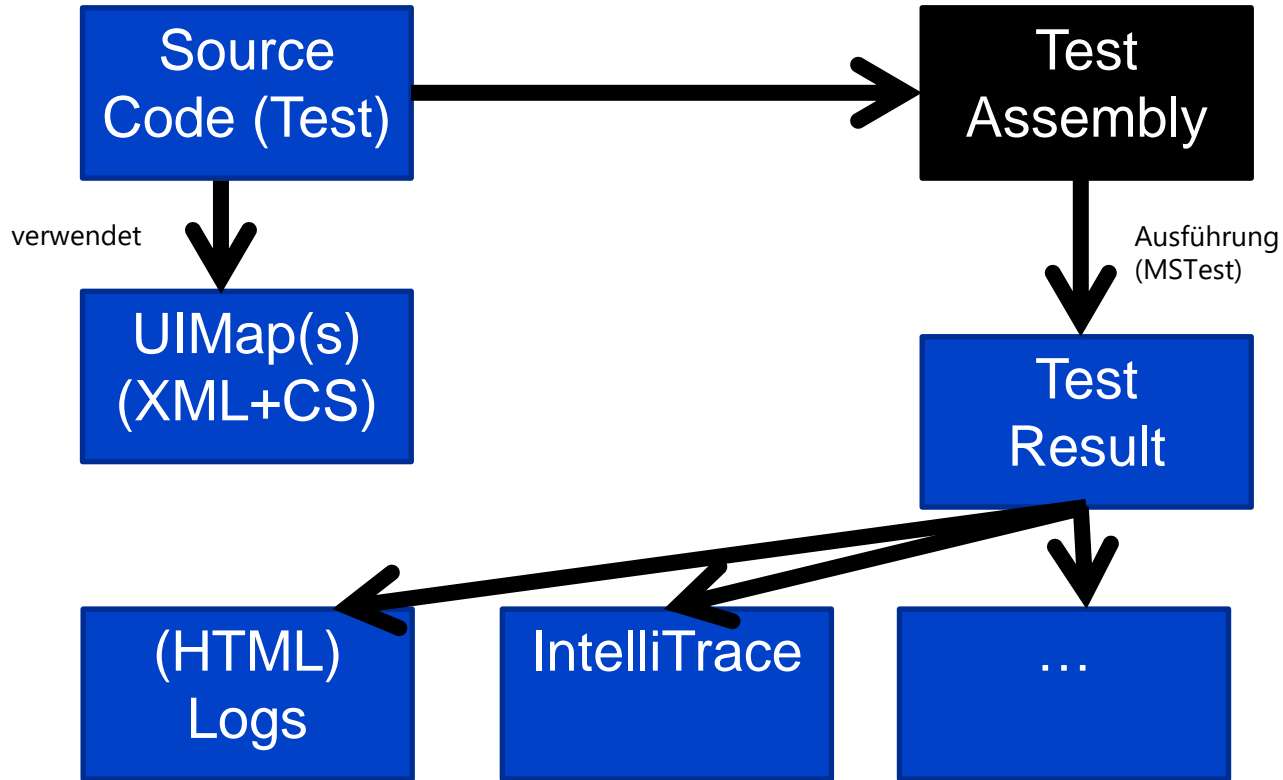


Recording Test Steps



● Klick

Understanding File Relationships



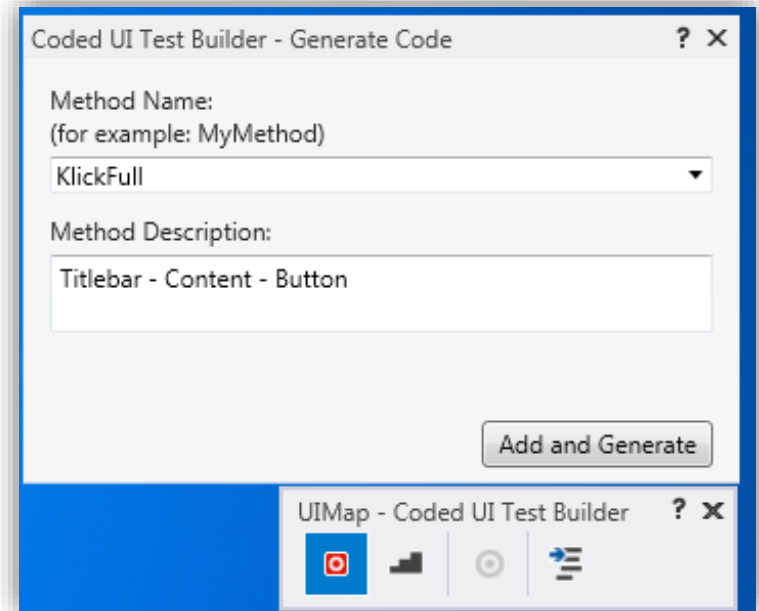
Einsatz der Recorder Controls

Aufnahme starten/ pausieren

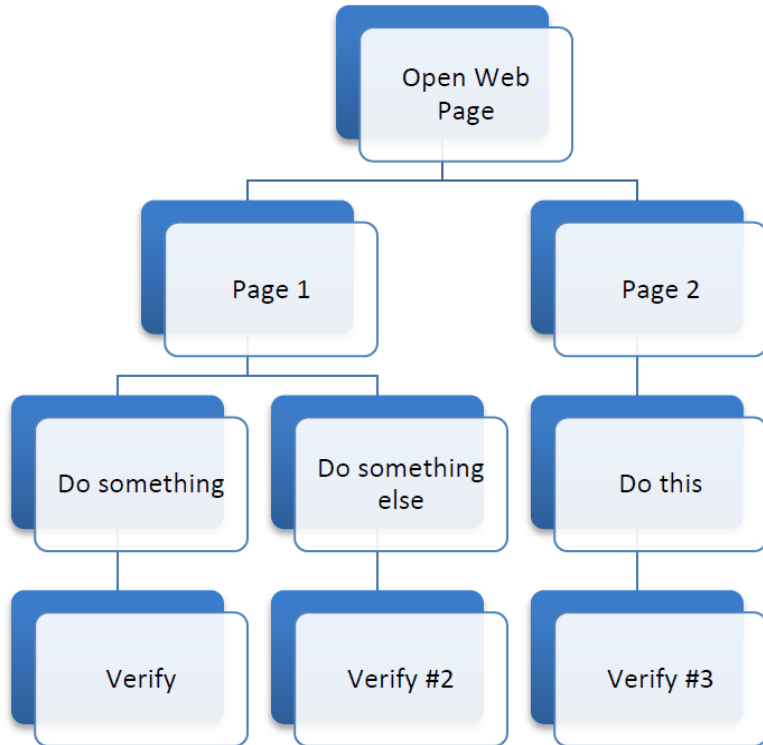
Aufgenommene Schritte
anzeigen

Aussagen hinzufügen

Code generieren



Mehrere UIMaps – Modulares Design



```
[CodedUITest]
public class MyCodedUITest {

    [TestMethod]
    public void TestCaseOne()
    {
        var util = new TestUtility();

        util.OpenWebPage();
        util.NavigateToPageOne();
        util.PageOne.DoSomething();
        util.PageOne.Verify();
    }

    [TestMethod]
    public void TestCaseTwo()
    {
        var util = new TestUtility();

        util.OpenWebPage();
        util.NavigateToPageOne();
        util.PageOne.DoSomethingElse();
        util.PageOne.VerifyTwo();
    }

    [TestMethod]
    public void TestCaseThree()
    {
        var util = new TestUtility();

        util.OpenWebPage();
        util.NavigateToPageTwo();
        util.PageTwo.DoThis();
        util.PageTwo.VerifyThree();
    }
}
```

Test Case 1

- OpenWebPage
- DoSomething
- Verify

Test Case 2

- OpenWebPage
- DoSomethingElse
- Verify2

Test Case 3

- OpenWebPage
- Dothis
- Verify3

Vorteile Page Object Pattern

Anwendung der SE Prinzipien

- Wartbare Tests
- SOLID, DRY, ...

Einfache Lesbarkeit der Szenarios

Test ist fokussiert auf Interaktion, kein «Plumbing»

- Separation of Concerns
- UI Interaktion ist abstrahiert
- Kann mittels UIMaps oder «Code First» implementiert werden

Code First – ohne UIMaps

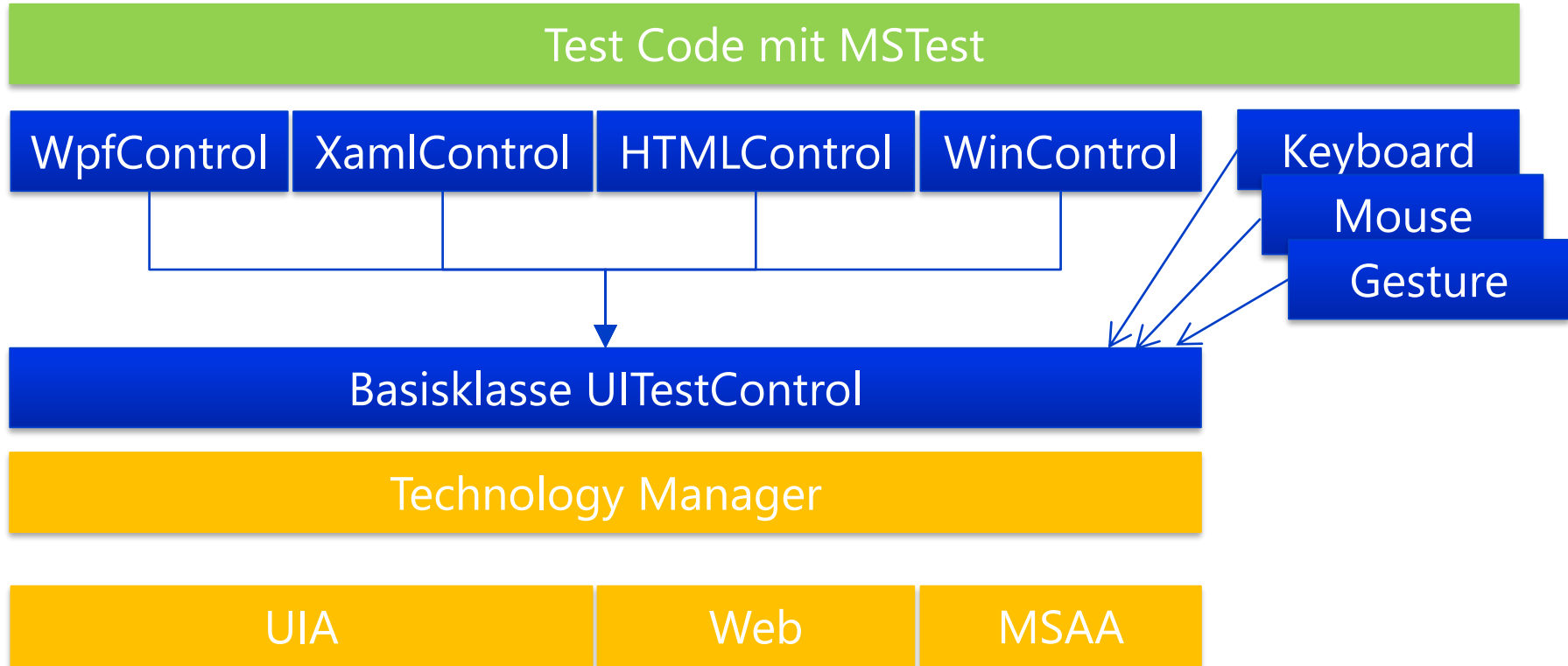
Code First Vorteile:

- Volle Kontrolle über den Code
- Einfaches Merging und Refactoring
- Bessere Team Collaboration Möglichkeiten

Code First Nachteile:

- Aufwändiger, es muss mehr Code geschrieben werden
- Detaillierte Kenntnisse über CodedUI Object Model notwendig

Technologiespezifische Umsetzung



UI Automation Controls

CodedUI abstrahiert die darunterliegende UI Technologie

- Alle Automation Controls sind von UITestControl abgeleitet.
- Ein Technology Manager lädt die richtige Implementation um mit der effektivsten Technology zu interagieren.
- Die UITestControl Klasse enthält ein TechnologyName Property

Technologieabhängige Implementierung:

- WinTable
- WpfTable
- XamlTable
- HTMLTable

Sind CLR-Objekte

Lassen sich über Properties / Methoden ändern

UI Automation Controls

```
UIMap.Designer.cs
SimpleCalculatorUI.Tests.UI.SimpleCalculatorWindow
203 [GeneratedCode("Coded UITest Builder", "12.0.21005.1")]
204 public class UISimpleCalculatorWindow : WpfWindow
205 {
206
207     1 reference
208     public UISimpleCalculatorWindow()
209     {
210         #region Search Criteria
211         this.SearchProperties[WpfWindow.PropertyNames]
212         this.SearchProperties.Add(new PropertyExpression
213         this.WindowTitles.Add("Simple Calculator");
214         #endregion
215
216     #region Properties
217     1 reference
218     public WpfEdit UIInputNumber1Edit
219     {
220     get
221     {
222         if ((this.mUIInputNumber1Edit == null))
223         {
224             this.mUIInputNumber1Edit = new WpfEdit
225             #region Search Criteria
226             this.mUIInputNumber1Edit.SearchProperties
227             this.mUIInputNumber1Edit.WindowTitles.
228             #endregion
229             return this.mUIInputNumber1Edit;
230         }
231     }
232
233     1 reference
234     public WpfEdit UIInputNumber2Edit
235     {
236     get
237     {
```

```
SimpleCalculator - WpfEdit.cs
WpfEdit.cs
1 // Type: Microsoft.VisualStudio.TestTools.UnitTesting.WpfControls.WpfEdit
2 // Assembly: Microsoft.VisualStudio.TestTools
3 // MVID: B77D54B4-4AF3-4DC7-97F2-B2F0CA9C7
4 // Assembly location: C:\Program Files (x86)\
5
6 using Microsoft.VisualStudio.TestTools.UnitTesting;
7 using System;
8 using System.Diagnostics.CodeAnalysis;
9
10 namespace Microsoft.VisualStudio.TestTools.UnitTesting
11 {
12     /// <summary>
13     /// Represents an edit control to test t
14     /// </summary>
15     [CLSCompliant(true)]
16     public class WpfEdit : WpfControl
17     {
18         /// <summary>
19         /// Initializes a new instance of the
20         /// </summary>
21         public WpfEdit();
22
23         /// <summary>
24         /// Initializes a new instance of the
25         /// <param name="parent">The <see cref="
26         public WpfEdit(UIControl parent);
27
28         /// <summary>
29         /// Gets or sets the contents of this
30         /// </summary>
31         /// <returns>
32         /// The contents of this edit control.
33         /// </returns>
34         public virtual string Text { get; set;
35         /// <summary>
```

```
SimpleCalculator - WpfControl.cs
WpfControl.cs
1 // Type: Microsoft.VisualStudio.TestTools.UnitTesting.WpfControls.WpfControl
2 // Assembly: Microsoft.VisualStudio.TestTools.UnitTesting, Version=12.0.0.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a
3 // MVID: B77D54B4-4AF3-4DC7-97F2-B2F0CA9C70B1
4 // Assembly location: C:\Program Files (x86)\
5
6 using Microsoft.VisualStudio.TestTools.UnitTesting;
7 using System;
8 using System.Collections.Generic;
9 using System.Diagnostics.CodeAnalysis;
10
11 namespace Microsoft.VisualStudio.TestTools.UnitTesting
12 {
13     /// <summary>
14     /// Represents an internal base class for te
15     /// </summary>
16     [CLSCompliant(true)]
17     public class WpfControl : UIControl
18     {
19         /// <summary>
20         /// Initializes a new instance of the <see
21         /// </summary>
22         public WpfControl();
23
24         /// <summary>
25         /// Initializes a new instance of the <see
26         /// <param name="parent">The <see cref="T:Microsoft
27         public WpfControl(UIControl parent);
28
29         /// <summary>
30         /// Gets an array of child controls in th
31         /// </summary>
32         /// <returns>
```

```
SimpleCalculator - UITestControl.cs
UITestControl.cs
10 using System.ComponentModel;
11 using System.Diagnostics.CodeAnalysis;
12 using System.Drawing;
13
14 namespace Microsoft.VisualStudio.TestTools.UnitTesting
15 {
16     /// <summary>
17     /// This class provides the ability to locate controls on a Us
18     /// </summary>
19     [CLSCompliant(true)]
20     public class UITestControl
21     {
22         /// <summary>
23         /// Initializes a new instance of the <see cref="T:Microsoft
24         /// </summary>
25         public UITestControl();
26
27         /// <summary>
28         /// Initializes a new instance of the <see cref="T:Microsoft
29         /// <param name="searchLimitContainer">The container for loc
30         public UITestControl(UIControl searchLimitContainer);
31
32         /// <summary>
33         /// Indicates whether two <see cref="T:Microsoft.VisualStudio
```

Testautomatisierung ohne UIMaps

```
public virtual WpfControl FindControlById(WpfControl parentControl, string id)
{
    WpfControl control = new WpfControl(parentControl);
    control.SearchProperties[WpfControl.PropertyNames.AutomationId] = id;
    return control;
}
```

...

```
public void ClickManageEventsButton()
{
    WpfControl button = this.FindControlById(this.MyEventsMainWindow,
        "ManageEventsButton");
    Point clickPoint = new Point();
    button.TryGetClickablePoint(out clickPoint);
    Mouse.Click(button, clickPoint);
}
```

ACChecker

The screenshot displays the ACChecker application interface. At the top, there are three tabs: "Verifications", "Results", and "MSAA Tree". The "Results" tab is active, showing a table with the following data:

Start Time	Name	Process Name	Window Class
17:17:10	MainWindow	WpfSample.vsh...	HwndWrapper[...

Below the table, there are four summary items:

- 3 Errors
- 0 Warnings
- 0 Messages
- 0 Suppressions

The main area shows a list of errors:

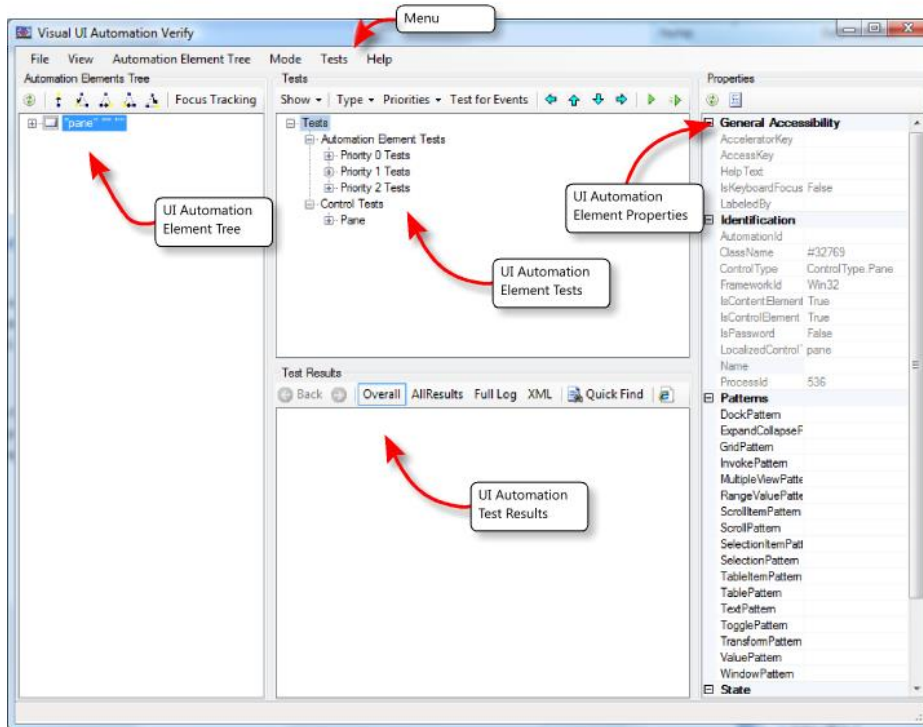
Time	ID	Text	Priority	Verification
17:17:10	ElementHasN...	Element has no name	1	VerificationR...
17:17:10	ElementHasN...	Element has no name	1	VerificationR...
17:17:10	ElementHasN...	Element has no name	1	VerificationR...

At the bottom, there is a detailed view of the selected error:

- ID: ElementHasNoName
- Verification: VerificationRoutines.CheckName
- Text: Element has no name
- Name:
- Value: 2
- Role: Text
- State: Focusable
- Rectangle: {X=158,Y=234,Width=384,Height=28}
- Window class: HwndWrapper[WpfSample.vshost.exe;;a48460]

On the right side, a preview of the application window "MainWindow" is shown. It contains a button "Click me!", a text box "Second" with the value "2" (highlighted by a red box), a "+" button, and a text box "0".

UIAVerify

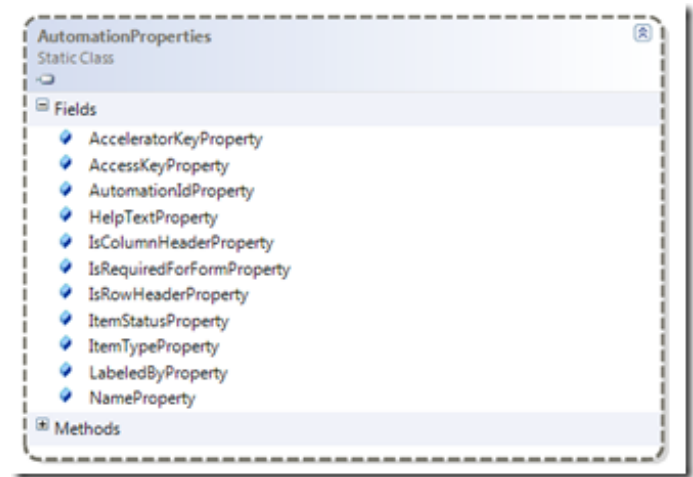


Optimierung WPF für Testautomatisierung

UIA ist fester Bestandteil von WPF

Wichtige Eigenschaften

- AutomationProperties.AutomationId
- AutomationProperties.Name



WPF Codebeispiel

<Button

AutomationProperties.AutomationId=„InsertCardButton“

AutomationProperties.Name=„InsertCardButton“

Grid.Row="1" Grid.Column="0" Grid.ColumnSpan="1"

Margin="5"

Command="{Binding Path=InsertCardCommand}">

Insert Credit Card

</Button>

UIA

User Interface Automation

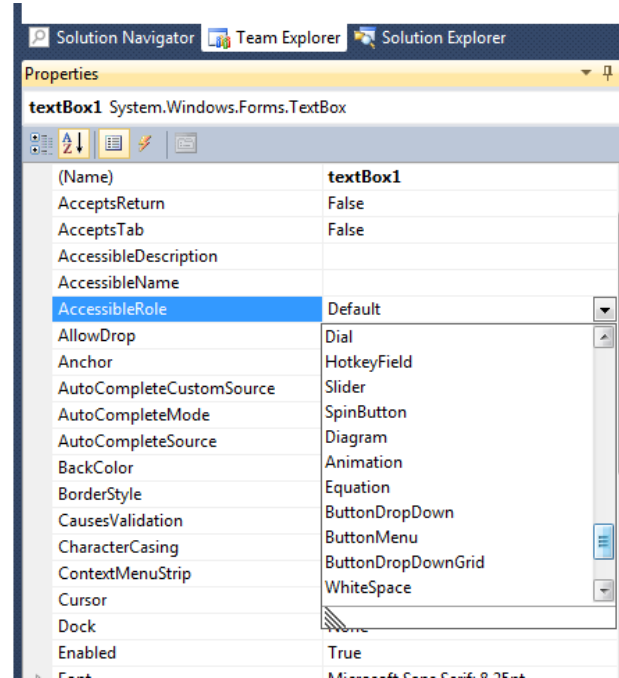
- COM-based technology
- Features COM-based and managed interfaces that were introduced with Microsoft .NET Framework
- Used for automating WPF and XAML based Applications

Optimierung WinForms für Testautomatisierung

Ansteuerung von WinForms Controls erfolgt per MSAA

Wichtige Properties

- AccessibleName
- AccessibleDescription



MSAA

Microsoft Active Accessibility

- COM-based technology
- Integrated in into Windows Operating System since Windows 98
- Used for automating Win Forms applications

Optimierung Web Applications

Ansteuerung von Webanwendungen erfolgt per IE DOM bzw. Selenium Adapter (VS 2012 Update 2)

Wichtige Eigenschaften für HTML Tags

- **class**="klassenname"
- **id**="idname"

Web

Access Document Object Model (DOM) in Browser

- Supports IE out of the box
- Cross Browser Support for Chrome and Firefox
 - by using Selenium Web Driver

Wir unterstützen Sie

KONTAKT
Nico.Orschel@aitgmbh.de
+49 151 55052624



AIT GmbH & Co. KG
Leitzstr. 45, 70469 Stuttgart
www.aitgmbh.de

BERATUNG
Agile ALM und TFS
.NET und Architektur



ENTWICKLUNG
Dienstleister für individuelle
Lösungen mit .NET und Azure



© AIT GmbH & Co. KG – Alle genannten und gezeigten Marken oder Warenzeichen sind eingetragene Marken oder eingetragene Warenzeichen ihrer jeweiligen Eigentümer und ggf. nicht gesondert gekennzeichnet. Aus dem Fehlen der Kennzeichnung kann nicht geschlossen werden, daß es sich bei einem Begriff oder einem Bild nicht um eine eingetragene Marke oder ein eingetragenes Warenzeichen handelt.

4tecture[®]

empower your software solutions

Marc Müller

Principal Consultant
für DevOps, ALM, TFS /VS, .NET



4tecture[®]

E-Mail: marc.mueller@4tecture.ch
Webseite: <http://www.4tecture.ch>
Schulungen: <http://4tecture.ch/trainings>
Blog: <http://4tecture.ch/blog>
Twitter: @muellermarc

4tecture GmbH
Aathalstrasse 84
CH-8610 Uster

+41 44 508 37 00
info@4tecture.ch





4tecture[©]
empower your software solutions

Weiterführende Informationen

- MSPress Engineering For Accessibility

http://download.microsoft.com/download/5/0/1/501FF941-E93D-423F-868B-C7BB2EC08C56/engineering_for_accessibility_eBook.pdf

- Beispiel Code

<https://github.com/marc-mueller/Demo-CodedUI>