Modul 120: Testen der Benutzerschnittstelle (GUI Testing)

- Sie kennen den Ablauf und Zweck von Tests
- Sie können die Abdeckung der Tests abschätzen
- Sie können selbständig GUI Tests planen, erstellen und durchführen
- Sie erstellen fehlerfreie GUI's

Das Thema Testing alleine könnte ein ganzes Modul füllen. Tests werden zu unterschiedlichen Zwecken und in unterschiedlichen Formen durchgeführt.

Prinzipiell ist der Erstellung von Tests die gleiche Sorgfalt wie der eigentlichen Umsetzung des Programms beizumessen. (Planung, Erstellung, Durchführung, Dokumentation,..) Eine extreme Form der Softwareentwicklung hat sich aus dem Testing heraus entwickelt, das sogenannte „Test driven development" wo zuerst die Tests und erst anschliessend die zu liefernde Software erstellt wird.

Tests können dienen zum:[1]

- Finden von Programmfehlern
- Simulieren von Mehrbenutzerumgebungen
- Beibehalten der Qualität bei Weiterentwicklungen (wenn nach 3 Jahren ein Programm angepasst wird, dient der vorhandene Test dazu die ursprüngliche Funktionsweise beizubehalten)
- Beweisen der Korrektheit eines Programms

Es gibt sehr viele Testarten, im Idealfall laufen Tests vollautomatisch ab, das spart enorm Zeit:

- Funktionstests (läuft das was laufen muss korrekt?)
- Fehlertests, Stabilitätstests (wie reagiert das Programm bei (Benutzer-)Fehlern, Fehleingaben usw.)
- Performancetests (wie schnell läuft das Programm, wie hoch ist die obere Grenze für Anzahl Benutzer usw.)

Nicht automatisierte Tests werden entweder von Entwicklern oder (besser) von einem speziellen Testteam durchgeführt:

- **Manuelle Funktions- oder Fehlertests** (möglich mit Visual Studio Test Manager)
- Code Review (4 Augen Prinzip für Sourcecode)
- Sicherheitstests
- Ergonomietests

Je nachdem welche Schicht einer Software getestet werden soll, kommen unterschiedliche Methoden zum Einsatz:

- Datenbank: Funktionstests, Konsistenztests, Performancetests,...
- Applikationsschicht: **Unittests** (möglich mit Visual Studio), Schnittstellentest, Interoperabilitätstest,...
- Benutzeroberfläche: Funktionstests, Fehlertests, **Oberflächentests** (möglich mit Visual Studio),...

Wir sehen, dass das Testing ein grosses Gebiet ist, wir werden uns auf automatische Oberflächentests (coded UI-Tests) beschränken. Keine Angst, der Begriff „coded" bedeutet nicht, dass wir viel programmieren müssen.

Freiwillige Links:

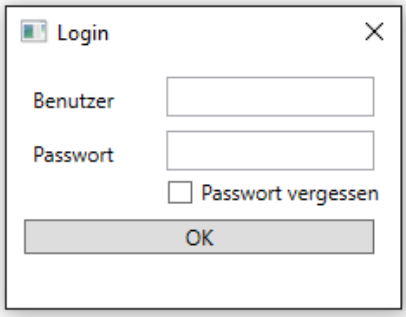
Referenz: https://msdn.microsoft.com/en-us/library/dd286726.aspx

Schnelleinstieg: https://www.youtube.com/watch?v=ONxwBDkXNj4&list=PL6tu16kXT9PrBqNFlv5sk6-63Br-Clmyd

[1] Die Aufzählungen auf dieser Seite sind nicht abschliessend, es sind die wichtigsten Elemente aus Sicht des Autors

Abdeckung

Als Abdeckungsgrad wird bezeichnet, wie viel Prozent eines Programms durch Tests erfasst werden (Test Coverage). Je höher die Abdeckung, desto grösser ist die Wahrscheinlichkeit alle Fehler zu finden. (Wie) kann alles abgedeckt werden? Wirklich alle Fälle abzudecken ist oftmals fast unmöglich. Stellen wir uns ein Loginfenster vor:



Das Fenster enthält:

Zwei Textboxes zur Eingabe

Eine Checkbox
Ein Knopf zur Bestätigung der Eingabe
Ein Label (nicht sichtbar) für die Rückmeldung

Sie sollen nun Testfälle erstellen. Wie viele Fälle sollten getestet werden?
Auf den ersten Blick sieht die Aufgabenstellung simpel aus. Es kommt darauf an, wie genau die Rückmeldung für den Benutzer sein soll. Vorschlag:

Eingabe Fall Nr	Benutzer	Psw	Vergessen	Resultat Fall Nr	Meldung
1	leer	leer	leer	1	Benutzer darf nicht leer sein
2	leer	NOK	leer	1	Benutzer darf nicht leer sein
3	leer	OK	leer	1	Benutzer darf nicht leer sein
4	NOK	leer	leer	2	Benutzer/ Psw falsch
5	NOK	NOK	leer	2	Benutzer/ Psw falsch
6	NOK	OK	leer	2	Benutzer/ Psw falsch
7	OK	leer	leer	2	Benutzer/ Psw falsch
8	OK	NOK	leer	2	Benutzer/ Psw falsch
9	OK	OK	leer	3	Login OK
10	leer	leer	checked	1	Benutzer darf nicht leer sein
11	leer	NOK	checked	1	Benutzer darf nicht leer sein
12	leer	OK	checked	1	Benutzer darf nicht leer sein
13	NOK	leer	checked	4	Benutzer ungültig (oder Fall 2)
14	NOK	NOK	checked	4	Benutzer ungültig (oder Fall 2)
15	NOK	OK	checked	4	Benutzer ungültig (oder Fall 2)
16	OK	leer	checked	5	Psw an Email gesendet
17	OK	NOK	checked	5	Psw an Email gesendet
18	OK	OK	checked	5	Psw an Email gesendet

In diesem Fall werden unterschiedliche Meldungen generiert, je nachdem ob die Textfelder leer, OK oder nicht OK (NOK) sind. Für die Checkbox gibt es zwei Zustände.

Berechnung: 3 * 3 * 2 = 18 Wobei nur 1 Resultat Fall zum erfolgreichen Login führt.

Wenn nun noch weitere Prüfungen gewünscht wären, erweitern sich die Anzahl Möglichkeiten schnell!

- Ungültige Sonderzeichen im Benutzername als Fehler ausgeben
- Mindestlänge des Passworts unterschritten
- Benutzerkonto sperren nach 3 ungültigen Versuchen

Um mit Sicherheit sagen zu können, dass ein Programm fehlerfrei funktioniert, muss für jeder der Eingabefälle ein Testfall erstellt werden, damit die Abdeckung 100% ist.

Wir sehen also, ein kleiner Formular vollständig zu testen gibt viel Arbeit, eine ganze Applikation mit hunderten von Feldern, Eingaben und verschiedenen möglichen Reihenfolgen der Eingaben erreicht schnell eine riesige Menge an Fällen.
Aus diesem Grund werden Programme oft nicht zu 100% sondern mit weniger Abdeckung getestet.
Machen Sie Berechnungen zum Abdeckungsgrad ***➔1***

AAA Muster

Damit die Tests ein wenig einheitlich aufgebaut sind und für das bessere Verständnis sowie Wartbarkeit, wird häufig das AAA Muster eingesetzt. Die drei A's stehen dabei für drei Schritte, welche während jedes Tests in der gleichen Reihenfolge durchgeführt werden:

Arrange	Vorbereiten des Tests, dies kann beinhalten: - Programm starten - Zum richtigen „Ort" im Programm springen - Daten vorbereiten - Daten „eingeben" im Programm (Tastatur, Maus)
Act	Ausführen der Funktion (oder Funktionalität) welche getestet werden soll. (Bsp: Knopf „Neu..." oder „Speichern..." drücken)
Assert	Prüfen, ob das zu erwartende Resultat eingetroffen ist. (Bsp: Eingabefelder korrekt validiert?) Im Assert Schritt wird ausgewertet, ob der Test erfolgreich ist oder nicht!

Beispiel Positivtest: (Funktionstest)

Arrange	Programm wird gestartet Gültige Logindaten werden eingegeben (Benutzername, Passwort)
Act	Login Knopf wird gedrückt
Assert	Wird das Login durchgeführt, das Programm geöffnet und der Hauptbildschirm angezeigt?

Beispiel Negativtest: (Fehlertest)

Arrange	Programm wird gestartet Ungültige Logindaten werden eingegeben
Act	Login Knopf wird gedrückt
Assert	Wird eine (oder die korrekte) Fehlermeldung angezeigt?

Umsetzung von „coded UI Tests" in Visual Studio

Wir werden nun 2 Testfälle in Visual Studio erstellen. Diese Funktion ist nur in den Top-Versionen von VS enthalten (VS 2013 Ultimate, VS 2015 Enterprise). Obwohl die Funktionalität „coded" heisst, müssen wir nicht wirklich programmieren. Zur schnelleren Test – Erstellung, können die 3 Schritte (AAA) aufgezeichnet werden!

Im vorbereiteten Projekt „AB120_07_Beispiele.zip" ist das obige Loginfenster korrekt umgesetzt. Anstelle von langen Erklärungen existiert ein Screencast, welcher den Testfall 9 (Login korrekt, Positivtest, Funktionstest) erstellt.

Screencast: ab120-07 Test Fall9.mp4
Erstellen eines neuen Testprojekts
Erstellen eines neuen coded UI Test
Aufzeichnung eines Testfalls
Durchführung des Tests

Bemerkungen:

- Die Aufzeichnung kann auch direkt nach der Erstellung des neuen Projekts gestartet werden. Dann ist allerdings der Name der Klasse nicht auswählbar.
- Es können selbstverständlich mehrere Fälle in einer Aufzeichnung untergerbacht werden.
- Zu den Schritten Arrange, Act und Assert ist noch ein Schritt Cleanup erstellt werden, welcher das zu testende Programm wieder schliesst.
- Die generierte Datei kann, bei Bedarf, nachträglich manuell angepasst werden.

Die Assert Methode ist im Code gut sichtbar:

```
// Sicherstellen, dass die Eigenschaft 'Name' von "Login korrekt" Bezeichnung ist gleich 'Login korrekt'
Assert.AreEqual(this.Login_AssertExpectedValues.UILoginkorrektText1Name, uILoginkorrektText1.Name, "Login Fall 9 Fehler");
```

Diese wir auch bei den anderen Testarten so verwendet.

Nun ist es an der Zeit für die letzten Übungen in diesem Modul ➔***2,3***

Aufgabenteil

1) Abdeckungsgrad

Sie müssen Testfälle für ein Programm schreiben, dieses enthält:

- 5 Checkboxes
- 3 Radiobuttons
- 2 Eingabefelder, welche nicht leer sein dürfen
- 1 Eingabefeld, welches eine Emailadresse beinhalten muss

Wie viele Fälle ergeben sich daraus?

2) Negativtest erstellen von Login

Erstellen Sie nach dem gleichen Muster einen Negativtest (Fehlertest) zum Beispiel Fall 13.
Führen Sie beide Tests aus und kontrollieren Sie das Resultat.

Nun prüfen Sie, ob die Tests allfällige Fehler im Programm erkennen. Dazu passen Sie Zeile 45 in der Klasse MainWindow wie folgt an:

```
if (passwortVergessen.IsChecked == true)
```

Welche der beiden Tests gibt jetzt einen Fehler aus?

3) Fallbeispiele: Testfälle für Abläufe

Wenden Sie das Gelernte im Fallbeispiel „GUITravel“ an!

- Aufnahme einer Modifikation eines Datensatzes, erneutes öffnen und Kontrolle ob Daten korrekt
- Aufnahme einer Erfassung eines neuen Datensatzes