Unittest planen

Modultest oder auch Unittest gehören zu den grundlegen Teststufen. Hierbei werden einzelne Funktionen einer Software getestet. In agilen Vorgehensmodellen, wie z.B. SCRUM, wird zur Qualitätssicherung eine sehr häufige Testausführung angestrebt. Dies lässt sich nur erreichen, wenn die Tests vollständig automatisiert sind.

Manuelle Tests	**Automatisierte Tests**
• Händisches testen einer Software ohne ein Automatisierungstool. • Tester sitzt vor dem Bildschirm und führt alle geplanten Tests durch und protokolliert die Ergebnisse.	• Es wird ein Programm geschrieben, mit dessen Hilfe weitere geplante Tests automatisch und unabhängig von einem Tester ausgeführt werden. Auch die Testergebnisse werden automatisch protokolliert.

Merkmale eines guten Unittest sind: F.I.R.S.T.

FAST	Die Testausführung soll möglichst schnell sein, damit sie möglichst oft ausgeführt werden kann. Ideal eignen sich dazu vollständig automatisierte Tests.
INDEPENDENT	Unittest sollen unabhängig voneinander sein, damit sie in beliebiger Reihenfolge, parallel oder einzeln ausgeführt werden können.
REPEATABLE	Führt man einen Unittest mehrfach aus, muss er immer das gleiche Ergebnis liefern.
SELF-VALIDATING	Ein Unittest soll entweder fehlschlagen oder gut gehen. Diese Entscheidung muss der Test treffen. Es dürfen keine weiteren manuellen Prüfungen nötig sein.
TIMELY	Man soll Unittests vor der Entwicklung des Produktivcodes schreben.

Als weitere Kriterien sind:

BEGRENZTHEIT	Es wird pro Test immer nur eine Eigenschaft oder Einheit getestet, z.B. eine Methode.
VERSTÄNDLICHKEIT	Der Quelltext der Tests und dessen Inhalt sind leicht verständlich und so kurz und knapp wie möglich gehalten.
WARTBARKEIT	Der Quelltext der Tests hat auch eine hohe Codequalität, z.B. wurden die üblichen Programmierkonventionen eingehalten.
RELEVANZ	Es werden nur notwendige Programmteile getestet.

Fast alle Programmiersprachen bieten Möglichkeiten zum Schreiben automatisierter Unittests an. In der Regel ist es üblich, aber nicht zwingend notwendig, dass der Unittest in der gleichen Sprache wie das Testobjekt geschrieben wird.

Testfälle formulieren

Black-Box- und White-Box-Tests sind geeignet, um Modultests durchzuführen. In beiden Fällen stellt sich jedoch die Frage wie viele und welche Testdaten man zum Testen benötigt. Das Grundziel muss es zunächst sein, eine hohe Testabdeckung zu erreichen. Deshalb gibt es Verfahren, die bei der Auswahl der Testdaten helfen.

Der Black-Box-Test

Verfahren beim Black-Box-test sind beispielsweise die Äquivalenzklassenbildung und die Grenzwertanalyse. Grundlage für die Gewinnung entsprechender Testdaten beim Black-Box-Test sind die Anforderungsspezifikationen.

Black-Box-Test

Äquivalenzklassenbildung

Mit diesem Verfahren wird versucht, eine möglichst hohe Testabdeckung mit sehr wenigen Tests zu erreichen. Es werden gültige und ungültige Äquivalenzklassen unterschieden.

Beispiel: eine Methode soll optimale Wetter zum Joggen unter Prüfung der Temperaturen mit Rückgabe TRUE oder FALSE liefern. Temperatur über 25 Grad ist schlecht zum Joggen (FALSE). Temperaturen mit 25 und darunter sind gut zum Joggen (TRUE).

Äquivalenzklasse	1	2
Wertebereich	Temperatur <= 25	Temperatur > 25
Mögliche Testwerte	14.0, 23.0, 12.99 …	26.0, 31.7, 29.6
Erwartetes Ergebnis	TRUE	FALSE

Grenzwertanalyse

Die Grenzwertanalyse stellt eine Erweiterung der Äquivalenzklassenbildung dar. In vielen Fällen treten die Fehler in den Programmen hauptsächlich an den Grenzen der entsprechenden Äquivalenzklassen auf, z.B. durch setzen falscher Operatoren (<, <=,…). Deswegen werden bei der Grenzwertanalyse Testdaten ausgewählt, welche die Grenzen einer Äquivalenzklasse und damit den Übergang zu einer anderen Äquivalenzklasse darstellen.

Beispiel: wie bei der Äquivalenz-klassenbildung (links).

Äquivalenzklasse	1	2
Wertebereich	Temperatur <= 25	Temperatur > 25
Mögliche Testwerte	25.0	25.01
Erwartetes Ergebnis	TRUE	FALSE

Der White-Box-Test

Beim White-Box-Test verhält es sich etwas anders. Hierbei wird das Programm auf der Grundlage der Anforderungsspezifikation erstellt. Allerdings werden die Testdaten dann vor allem aus der Programmlogik abgeleitet. Zwei der Verfahren werden hier näher vorgestellt.

<table>
<tr><th colspan="2">White-Box-Test</th></tr>
<tr><th>Anweisungsüberdeckung</th><th>Zweigüberdeckung</th></tr>
<tr><td>Bei der Anweisungsüberdeckung wird angestrebt, dass jede Anweisung im Quellcode durch das Testen mindestens einmal ausgeführt wird. Wird dieses Ziel erreicht, dann wurde eine vollständige Anweisungsüberdeckung (100%) erreicht. Somit wurde sichergestellt, dass kein Code existiert, welcher nie ausgeführt wird („toter Code“).

Beispiel Python:</td><td rowspan="2">Bei der Zweigüberdeckung (auch Entscheidungs- oder Kantenüberdeckung) wird das Verhältnis zwischen ausgeführten und möglichen Programmzweigen gemessen. Man muss jeden Zweig eines Programms mindestens einmal betreten, um eine 100%ige Zweigüberdeckung zu erreichen. Eine 100%ige Zweigüberdeckung schließt eine 100%ige Anweisungsüberdeckung mit ein. Aber anders als bei der Anweisungsüberdeckung müssen auch leere Pfade ohne Anweisung, z.B. Else-Zweige, einmal betreten werden.

Beispiel Python:</td></tr>
<tr><td><pre>Ergebnis = 0
if(a < b)
   a = -a
ergebnis = a + b
print(str(ergebnis))</pre></td></tr>
<tr><td>Bei der richtigen Auswahl an Testdaten wird in dem Beispiel nur ein Testfall benötigt, um eine vollständige Anweisungsüberdeckung zu erreichen. Dies ist z.B. der Fall, wenn der Testwert für a kleiner als für b ist (a = 10 und b =20).</td><td><pre>Ergebnis = 0
if(a < b)
   a = -a
ergebnis = a + b
print(str(ergebnis))</pre></td></tr>
<tr><td></td><td>Im Gegensatz zur Anweisungsüberdeckung ist mehr als ein Testfall nötig, um alle Zweige zu durchlaufen. Im oberen Beispiel können, wie bei der Anweisungsüberdeckung auch a = 10 und b = 20 verwendet werden. Zum anderen muss aber auch mit Daten getestet werden, bei denen a > b gilt. Z.B. a = 15 und b = 5.</td></tr>
</table>

Zudem gibt es noch die sogenannte **Pfadüberdeckung**. Hierbei muss jeder mögliche Pfad von Anfang bis Ende durchlaufen werden. Eine vollständige Pfadüberdeckung beinhaltet auch eine vollständige Zweigüberdeckung.

Ein weiterer Begriff, der zu den dynamischen Testverfahren gehört, ist der sogenannte **Regressionstest**. Hierbei handelt es sich um die Wiederholungen von Testfällen. Sollte der Code geändert worden sein (z.B. neuer Code hinzugefügt, Fehler behoben oder Durchführung von Codeoptimierung), so muss durch das nochmalige Testen anhand der bestehenden Testfälle sichergestellt werden, dass die Anwendung immer noch korrekt funktioniert. Aufgrund des Widerholungscharakters bietet sich hierbei die Automatisierung von Tests an.

Unittest durchführen

Führen Sie nun selbstständig einen Unittest anhand des folgenden Falls durch.

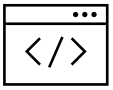
Handlungssituation

Auf der Online-Plattform des Bauamtes des Landkreises Würzburg soll potenziellen Bauwilligen gezeigt werden, ob ihr Bauvorhaben (nur Einfamilienhaus) grundsätzlich in die technische Prüfung durch das Bauamt kommt. Dazu sind 3 Eckdaten zum Bauvorhaben (Grundfläche des Bauvorhabens in m^2, Wandhöhe in m und Dachneigung in Grad) notwendig. Folgende Wertvorgaben liegen auf Basis der aktuellen Bauvorschriften des Landkreises Würzburg und des Freistaates Bayern vor:

- Die Grundfläche des Einfamilienhauses darf max. 150m^2 betragen.
- Die Wandhöhe darf 7m nicht überschreiten.
- Die Dachneigung darf max. 45° betragen.

Sofern nur eine Vorgabe nicht eingehalten ist, wird das Bauvorhaben nicht zur technischen Bauprüfung zugelassen und ist grundsätzlich abzulehnen.

Arbeitsauftrag



1. **Schreiben** Sie eine **Funktion**, die die **geforderten Anforderungen** erfüllt, in einer **Programmiersprache** Ihrer Wahl.
2. **Planen** Sie Ihre **Unittest** und formulieren Sie **passende Testfälle**. **Begründen** Sie dabei Ihr Vorgehen:
   a. Für welches **Testverfahren** haben Sie sich entschieden und **warum**? ich habe whitebox testing benutzt um alle möglichen pfade innerhalb der funktion zu checken
   b. Wie viele **Testfälle** haben Sie? 20
   c. Welche **Testabdeckung** haben Sie erreicht? 100%
3. **Protokollieren** Sie Ihre Tests gemäß der **Testprotokoll-Vorlage**.