Modellgetriebene funktionale Systemtesterstellung mit dem MOdel Compiler for generating complete Applications (MOCCA)

Jessica Schiffmann

Zusammenfassung

Der dargestellte Ansatz - Teil einer berufsbegleitenden Promotion im kooperativen Verfahren der TU Bergakademie Freiberg und der Hochschule Kaiserslautern - beschäftigt sich mit der Möglichkeit einer leichtgewichtigen Integration der funktionalen Systemtestmodellierung in das MDA-Framework MOdel Compiler for generating complete Applications (MOCCA). Die verfolgte Zielsetzung ist hierbei eine kompakte Modellierung des prozessorientierten Systemtests, mit Zugriff auf die Benutzeroberfläche für Testausführung und Prüfung. Hierbei wird das System- und das Testverhalten in unterschiedlichen Form modelliert.

1 Einleitung

Funktionale Systemtests prüfen, „ob die Antworten des Systems korrekt sind, d.h. ob die auf dem Ausgabebildschirm erscheinenden Daten der Erwartung entsprechen.“ [1], die Modellierung dieser Testart beinhaltet somit den Prozessablauf bis zum Erreichen des Testpunktes, die Verifikation des Testpunktes und eventuelle Schritte, die die Anwendung in einen Zustand versetzen der weitere Testausführungen erlaubt. Die Modellierung erfolgt hierbei über Testfallstruktur und Testverhalten. Die Struktur dient zum Strukturieren der Testdaten und als Grundlage des Testverhaltens, da dieses die Struktur in definierter Art abarbeitet und die– in Keywords angegeben – Aktionen durchführt.

Das vorliegende Vorgehen beschreibt die Integration der funktionalen Systemtestmodellierung für GUI-basierte Systeme in das an der TU Bergakademie Freiberg entwickelte MOCCA-Framework. Beim Einsatz von MOCCA handelt sich um ein Model Driven Architecture (MDA) Vorgehen. Es dient dazu, komplette Anwendungen aus plattformunabhängigen Modellen zu generieren. MOCCA trennt hierbei das Design Model und das Target Model in einen applikationsunabhängigen und einen -abhängiges Modell. Die folgende Abbildung zeigt den Aufbau der genutzten Modelle des MOCCA-Frameworks.

Zur leichtgewichtigen Modellierung des Verhaltens von Anwendungen wurde das Framework durch die Dissertation von Herrn Liang [2] u.a. um eine Action Language eXtended Object Constraint Language (XOCL), die auf der Object Constraint Language (OCL) [3] basiert, erweitert. Diese erhöht die Möglichkeiten der Verhaltensmodellierung.

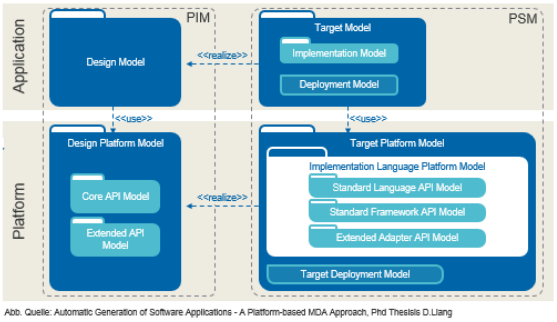

Abb. Quelle: Automatic Generation of Software Applications - A Platform-based MDA Approach, Phd Thesisis D.Liang

Abbildung 1: Aufbau MOCCA

Die Testfallstruktur wird hierbei als wiederverwendbare Struktur integriert, das Design Model enthält die Modellierung der Testdaten, sowie die Modellierung der anderen Anwendungen übertragbare Testverhaltens.

2 Testfallstruktur

Wie in der Einführung erwähnt, testen funktionale Systemtestfälle das Verhalten von Softwaresystemen anhand der Ausführung von Abläufen und Prüfung definierter Gegebenheiten des Systems im jeweiligen Zustand bzw. dem Endzustand. Dies zeigt, dass die Spezifikation eines Testfalls genau einen der möglichen Abläufe durch das zu testende System, inkl. vorgesehener Prüfungen, darstellt. Die Testfallstruktur als integraler Bestandteil eines modellgetriebenen Tests wird auch in der UML Testing Profile [5] genannt, das dargestellte Verfahren weicht von dieser Darstellung in einigen Aspekten ab.

Damit die Wiederverwendbarkeit des modellierten Verhaltens in möglichst vielen Testabläufen gewährleistet werden kann, ist die Basis der Ausführung eine möglichst kleine Komponente des Systemablaufs, hier eine GUI-Aktion. Aktionen in GUI-basierten Systemen sind Operationen auf der Oberfläche einer Anwendung. Die Auswahl in einer Combobox z.B. benötigt die Aktionsart und die Identifikation, des Weiteren noch den neuen Wert der Combobox. Damit die Möglichkeit besteht, eine Prüfung nach einer Aktion durchzuführen, muss diese Prüfung der Testaktion zugeordnet werden. Die Verifikation, anhand der Systemoberfläche, beinhaltet das Element sowie die Aktion, die zum Bestimmen des aktuellen Wertes dieses Elementes benötigt wird und den Vergleichswert der Prüfung. Den Vergleichswert von außen vorzugeben, ermöglicht eine Prüfvorgabe zu realisieren, die nicht durch das Modell selbst (also unabhängig) festgelegt werden kann. Hinzu

kommt der Verifikationstyp, bei diesem handelt es sich um die Art der booleschen Prüfung, die durchgeführt werden soll. Diese Testaktionen mit den dazugehörigen Verifikationen werden in Testfällen in einen definierten Ablauf gebracht. Hinzu kommen bei den Testfällen eine Abschlussprüfung sowie administrative Details. Folgende Abbildung zeigt die Testfallstruktur.

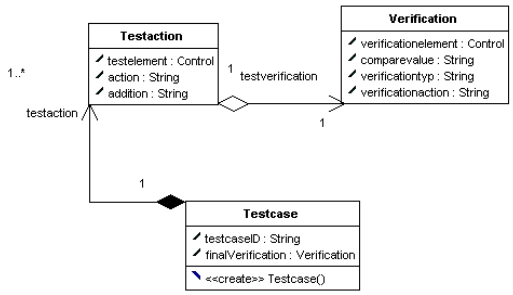

Abbildung 2: Testfallstruktur

3 Testdaten

Die Testdaten werden über die Testfallstruktur definiert und dem Testablauf zur Verfügung gestellt.

4 Testablauf

Der Testablauf ist eine Ausführung der Testfälle (Typ Testcase) auf der Oberfläche der Anwendung, aus diesem Grund werden die Abläufe zum Testfall in das Anwendungsmodell, welches selbstverständlich Aufbau- und Verhalten der Oberfläche beinhaltet, integriert. Durch diese Integration sind die Elemente der Anwendung für die Testaktionen einfach zugreifbar. Des Weiteren können dieselben Transformations-mechanismen von MOCCA für Anwendungs- und Testmodell genutzt werden. Mit dem Nutzen der Systemoberflächenelemente werden die Aktionen in derselben Weise ausgeführt, die dem Benutzer zur Verfügung stehen, die Tests werden somit mit großer Authentizität durchgeführt.

Dennoch wird das Testverhalten auf andere Weise modelliert, als das Systemverhalten, um die Test unabhängig zu machen. Beide Verhalten sind in der bereits erwähnten Sprache XOCL gemacht und können somit direkt in die zu transformierenden Modelle integriert werden. Abbildungen 3 und 4 zeigen das modellierte System-verhalten für den Klick auf den Button ‚1'. Die Einbindung des Events und das Eventverhalten werden hierfür einzeln modelliert.

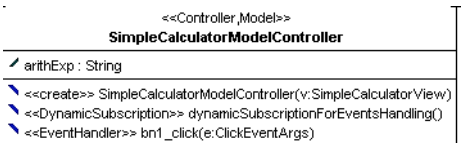

Abbildung 3: EventHandler Modellierung im Systemmodell

```
oqb bn1_click() {language OCL}

begin
  update self.arithExp = self.arithExp.concat(e.oclAsType(Button).text);
  update self.view.taResult.text = self.arithExp;
end
```

Abbildung 4: Eventverhalten nach dem Buttonevent

Die Testausführung der jeweiligen Komponente wird vom Allgemeinen zum Genauen definiert, so dass die Methoden selbst schlank gehalten werden können. Abbildung 5 zeigt das modellierte Testverhalten für den Klick auf einen Button, diese Methode wird durch die Testdaten, die in der Testfallstruktur definiert werden und im Ablauf mit XOCL angesprochen wird.

```
oqb executeButtonClick() {language OCL}

begin
 update testButton.doClick();
end
```

Abbildung 5: XOCL Testmodellierung Button betätigen

5 Zusammenfassung

Die Darstellung der Testfälle in der gegebenen Struktur, die gleichzeitig die Definition der Testdaten und den Aufbau des Testverhaltens ermöglicht schafft eine Systemtestmodellierung, die kompakt und übertragbar ist. Hierbei wurde großen Wert auf die Einnahme der Benutzersicht und die Kopplung der Identifikation der Elemente an das Anwendungsmodell und die Einnahme der Benutzersicht gelegt. Dies reduziert den Wartungsaufwand der Testfälle und erzeugt eine hohe Authentizität der Testausführung.

Referenzen

[1] Harry M. Sneed, Manfred Baumgartner, Richard Seidl, Der Systemtest, Hanser Verlag, 2. Auflage, 2009

[2] Dong Liang, Diss., Automatic Generation of Software Applications - A Platform-based MDA Approach, Freiberg, 2013

[3] OCL Spezifikation, Version 2.4, http://www.omg.org/spec/OCL/2.4/, Aufruf 08.01.2017

[4] Matthias Daigl, Rolf Glunz, ISO 29119: Die Softwaretest-Normen verstehen und anwenden, D.Punkt Verlag, 2016

[5] OMG UML Testing Profile, Version 1.2, http://www.omg.org/spec/UTP/1.2/PDF, Aufruf 30.01.2017