Beuth Hochschule für Technik Berlin

Masterarbeit

Variantenspezifische Abhängigkeitsregeln und Testfallgenerierung in TESTONA

BEUTH HOCHSCHULE
FÜR TECHNIK
BERLIN
University of Applied Sciences

Fachbereich VI - Technische Informatik - Embedded Systems

Eingereicht am: 21. Oktober 2014

Erstprüfer	:	Prof. Dr. Macos
Zweitprüfer	:	Prof. Dr. Höfig
Eingereicht von	:	Matthias Hansert
Matrikelnummer	:	s791744
Email-Adresse	:	matthansert@gmail.com

Dankessage

Inhaltsverzeichnis

Kapitel 1

Einleitung

Ziel dieser Masterarbeit ist die Erweiterung und Verbesserung des Berner & Mattner Werkzeuges TESTONA. Dieses Programm bietet Testern ein Werkzeug für eine strukturierte und systematische Ermittlung von Testszenarien und -umfänge [1]. Im Kapitel 3.1 wird weiteres zu dieses Programm und die Funktionsweise erläutert.

Die Erweiterung des Programmes besteht aus verschiedene Themen. Eins davon behandelt die Testfallgenerierung und die jeweilige Testabdeckung. Hier soll garantiert werden, dass bei einer automatischen Testfallgenerierung, eine höchstmögliche Testabdeckung erzielt wird.

Die Testfallgenerierung wird in dieser Arbeit beeinflusst, indem stärker die Produktvarianten betrachten werden. Verschiedene Varianten beinhalten verschiedene Parameter und Produktkomponenten. Die Parameterwerte definieren auch verschiedene Produktvarianten. Durch das Add-On MERAN für die Anforderungsmanagementsoftware ÏBM Rational DOORS"können Anforderungen direkt in TESTONA importiert werden. Dabei sollen automatisch die Parameterwerte zur der jeweilige Produktvariante zugeordnet werden. Aus diesem Grund kann es zu Konflikte bei der Testfallgenerierung kommen, bzw. inkohärente Testfälle.

Um solche Probleme zu vermeiden oder umgehen, gibt TESTONA den Testern die Möglichkeit Abhängigkeitsregeln anzulegen. Hier können Anfangsbedienungen sowie Sonderbedienungen definiert werden. Dabei muss wiederum geachtet werden, dass die Produktvarianten nicht verletzt werden. Weiteres zu den Themen und Begriffen wird im Kapitel 3

Im Kapitel 2 wird genauer die Aufgabe dieser Masterarbeit erläutert und in den Kapiteln 4 und 5 jeweils eine Lösung vorgeschlagen und implementiert.

Kapitel 2

Aufgabestellung

Ziel dieser Masterarbeit ist die Verbesserung der Testfallgenerierung und der Testabdeckung bei mehreren Produktvarianten, die Ersetzung von Parameter, die Prozessoptimierung sowie die Handhabung für den Benutzer in der TESTONA-Umgebung. Jedes Produkt kann unterschiedliche Produktvarianten beinhalten und jede Variante besteht aus unterschiedlichen Komponenten mit unterschiedlichen Parametern. In Abhängigkeit von der ausgewählten Variante sollen bei der Testfallgenerierung die dazugehörigen Komponenten berücksichtigt werden und die erzeugten Testfälle dargestellt werden. Besonders zu beachten sind dabei die definierten Abhängigkeitsregeln sowie die darauf bezogene Testabdeckung.

Abhängigkeitsregeln werden definiert um redundante Testfälle zu vermeiden, bzw. um Vorbedingungen für die Testfälle zu erstellen. Da Varianten verschiedene Baumelemente beinhalten, kann es dazu kommen, dass Baumelemente für Abhängigkeitsregeln nicht vorhanden sind. Dadurch könnte TESTONA bei der Testfallgenerierung die Testabdeckung verfälschen, indem die Gültigkeit eines Testfalles nicht garantiert werden kann. Um dieses Problem zu umgehen, muss bei der Erzeugung von Abhängigkeitsregeln auf mögliche Konflikte hingewiesen werden. Für den Lösungsansatz gibt es verschiedene Thesen die analysiert werden müssen, um eine optimale Prozessoptimierung zu erreichen.

Um die Handhabung der Varianten bezogen auf die Testfälle und die Testgenerierung benutzerfreundlicher und effizienter zu gestalten, soll die Benutzung des Variantenmanagements durch einen Testingenieur untersucht werden. Resultierend aus den erworbenen Erkenntnissen wird das Lösungsdesign für eine Erweiterung des bestehenden Variantenmanagements in TESTONA konzipiert.

Einer der besonderen Eigenschaften von TESTONA ist die Kopplung mit Anforderungsspezifikationen die in IBM Rational DOORS definiert worden sind. Durch das DOORS Add-On MERAN können Anforderungen die in DOORS definiert sind, mit den zugehörigen Varianten verknüpft werden. Diese Varianten können in TESTONA eingebunden werden, durch eine erfolgreiche Anmeldung bei DOORS (über die TESTONA Oberfläche) und ein gezieltes Auswählen der gewünschten Varianten. Hierbei sollen die in den Anforderungen definierten Variablen (z.B. eine Geschwindigkeit oder Anzahl der Türen eines Autos) mit gespeichert werden. Im Klassifikationsbaum soll je nach ausgewählter Variante (z.B. der Name von Klassen) mit dem entsprechenden Wert ersetzt werden. Andere Lösungsmöglichkeiten werden noch untersucht.

Der derzeitige Varianten-Management-Ansatz in TESTONA ist nicht in der Lage für die Test-

fallgenerierung zwischen verschiedene Varianten zu unterscheiden. Zwar werden durch die Perspektive „Variant Management“ verschiedene Varianten unterschieden, aber die Testfälle müssen manuell mit den jeweiligen Varianten verknüpft werden. Im Falle einer automatischen Testfallgenerierung werden auch ungültige Baumelemente betrachtet (siehe Abbild 2.1 und 2.2). Um dies zu vermeiden muss der Testingenieur einzelne Generierungsregeln anlegen. Dieser Vorgang soll automatisiert und von TESTONA übernommen werden. Dabei gibt es verschiedene Betrachtungsweisen und mehrere Lösungswege. Entscheidend für die Lösung werden die erworbenen Kenntnisse über die Benutzung des Variantenmanagements durch einen Testingenieurs. Bei der Lösung ist zu beachten, dass eine komplette Testfallabdeckung garantiert werden muss.

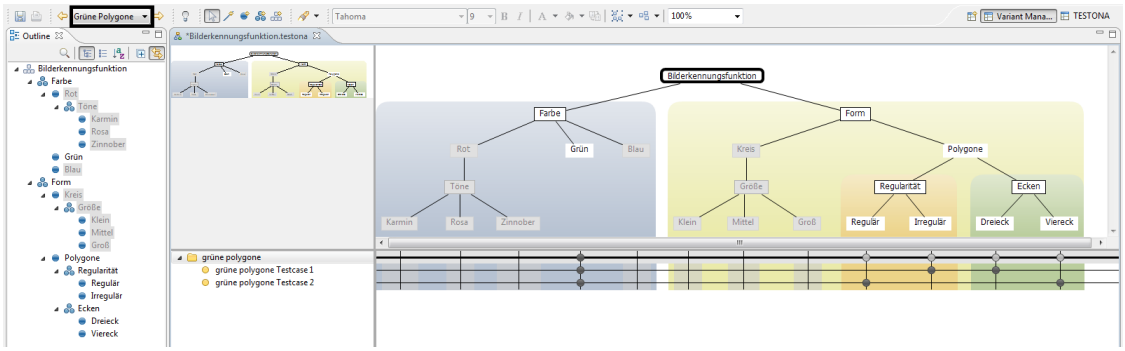

Abbildung 2.1: Design der Benutzeroberfläche

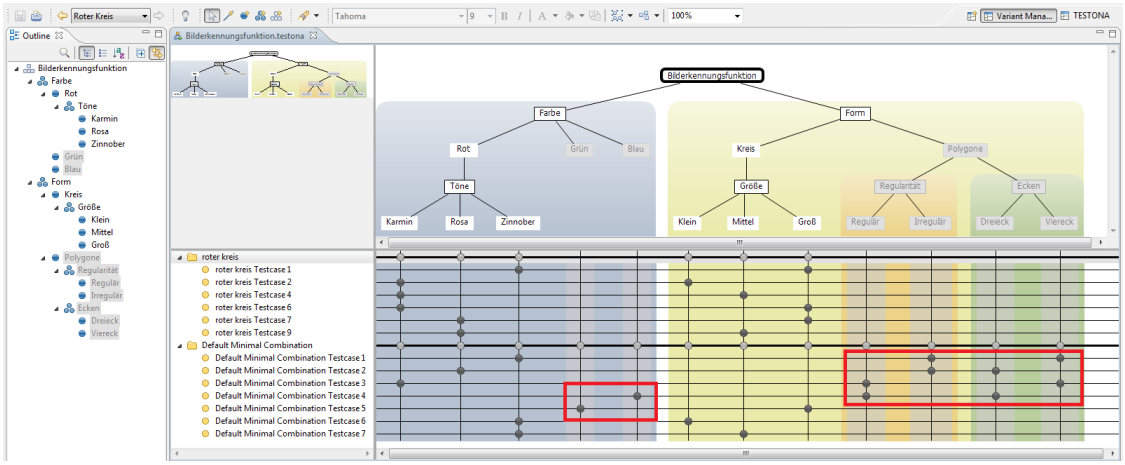

Abbildung 2.2: Design der Benutzeroberfläche

Kapitel 3

Fachliches Umfeld

Die Quellen dieses Kapitel sind aus .....

3.1 TESTONA

3.1.1 Klassifikationsbaum-Methode

In 1993 entwickelten K. Grimm und M. Grochtmann die Klassifikationsbaum-Methode zur Ermittlung funktionaler Backbox-Tests im Bereich von eingebetteter Software. Die Methode wurde im Forschungslabor von Daimler-Benz in Berlin als Weiterentwicklung der Category-Partition Method (CPM) erforscht. Gegenüber CPM hat die Klassifikationsbaum-Methode eine graphische Baum-Darstellung und hierarchische Verfeinerungen für implizite Abhängigkeiten. Als Werkzeug wurde der "Classification Tree Editor"(CTE) [1] programmiert und unterstützt Partitionierung und Testfallgenerierung. Das Werkzeug von CPM konnte nur Testfälle generieren ohne Bestimmung der Testaspekte[2].

Diese Methode besteht aus zwei wichtige Schritten:

- Bestimmung der Klassifikationen (testrelevante Aspekte) und Klassen (mögliche Ausprägungen).
- Erzeugung von Testfällen aus Kombinationen von unterschiedlichen Klassen für alle Klassifikationen

Ansatzpunkt sind die Funktionale Anforderungen (siehe 3.2) eines zu testendes Objekt. Um die Testfälle zu definieren und erzeugen, folgt die Methode das Prinzip des kombinatorischen Testentwurfs [5]. Dieses Prinzip hilft bei der Detektierung von Fehler in frühe Schritte des Testvorgangs. Nicht jeder einzelne Parameter steuert ein Fehler bei, eher werden Fehler verursacht durch die Interaktion verschiedene Parameter. Betrachten wir ein einfaches Beispiel, indem ein Programm auf Windows oder Linux laufen soll, unter Verwendung eines AMD oder Intel Prozessors und mit Unterstützung des IPv4 oder IPv6 Protokolls. Das ergibt intuitiv acht verschiedene Testfälle (2^3 Möglichkeiten). Verwenden wir dafür der kombinatorische Testentwurf "paarweise Kombination",

[1] Entwickelt von Grochtmann und Wegener[4]. Bei Berner & Mattner aus rechtlichen Gründen zu TESTONA umbenannt

No.	OS	CPU	Protokoll
1.	Windows	Intel	IPv4
2.	Windows	AMD	IPv6
3.	Linux	Intel	IPv4
4.	Linux	AMD	IPv6

Tabelle 3.1: Testfälle mittels paarweise Kombinatorik

hätten wir nur vier Testfälle (siehe Tabelle 3.1). Durch diese Methode werden alle Kombinationspaare der Parameter mindestens durch ein Testfall gedeckt[3].

Die Effizienz von diesen einfachen kombinatorischen Entwurf ist bei komplexeren System zu sehen. Hat ein System 20 verschiedene Schalter und jeder Schalter 10 verschiedene Einstellungen, so gibt es 10^{20} verschiedene Kombinationen. Durch Anwendung der paarweise Kombination muss der Tester nur 180 Testfälle betrachten.

Ein Experiment hat gezeigt, dass durch die Verwendung von die paarweise Kombinatorik die gleichen oder meistens mehrere Fehler entdeckt wurden, als mit manuelle Testauswahl[2]. Paarweise Kombinatorik ist am meisten verbreitet, aber man kann durchaus auch Drei-Werde-Kombinatorik verwenden. TESTONA implementiert standardmäßig Minimalabdeckung, Paarweise-, Drei-Wege- und N-Kombinatorik (wo N die maximale Anzahl an möglichen Parameter im Klassifikationsbaum ist, auch vollständige Kombinatorik genannt)[3].

3.1.2 Testfälle und Testfallgenerierung

Unter ein Testfall ist zu verstehen, die Beschreibung eines elementaren Zustands eines Testobjekts. Hierfür werden Eingangsdaten benötigt (Parameterwerte, Vorbedingungen) und ein erwarteter Folgezustand.

3.1.3 Abhängigsleitsregeln

3.2 Variantenmanagement und IBM Rational DOORS

3.3 Entwicklungsumgebung und Programmiersprache

[2]Basierend auf funktionelle und technische Anforderungen, Use-Cases

3.3.1 Eclipse

3.3.2 Plugins

3.3.3 Java

3.3.4 Java SWT

Kapitel 4

Lösungsansätze

4.1 Oberfläche Design

4.2 Parameterspeicherung

4.3 Anhämgigskeitsregeln und Testfallgenerierung

Kapitel 5

Systementwurf

5.1 Variantenmanagement und Parameter

5.2 Testfallgenerierung

Kapitel 6

Zusammenfassung und Ausblick

Was war wirklich wichtig bei der Arbeit?
Wie sieht das Ergebnis aus?
Wie schätzen Sie das Ergebnis ein?
Gab es Randbedingungen, Ereignisse, die die Arbeit wesentlich beeinflußt haben?
Gibt es noch offene Probleme?
Wie könnten diese vermutlich gelöst werden?

Abbildungsverzeichnis

Listings

Anhang A

Anhang

A.1 CD

Inhalt:

- Quellen
- PDF-Datei dieser Arbeit

A.2 code 1

```
1hier kommt java code
```

A.3 code 2

```
1hier kommt auch java code
```

Literaturverzeichnis

[1] Berner & Mattner, http://www.testona.net. *TESTONA*, Oktober 2014.

[2] M. Grochtmann and K. Grimm. *Classification Trees For Partition testing, Software testing, Verification & Reliability, Volume 3, Number 2*. Wiley, 1993.

[3] IEEE Computer Society, https://courses.cs.ut.ee/MTAT.03.159/2013_spring/uploads/Main/SWT_comb-paper1.pdf. *Combinatorial Software Testing*, 2009.

[4] Quality Week 1995, http://www.systematic-testing.com/documents/qualityweek1995_1.pdf. *Test Case Design Using Classification Trees and the Classification-Tree Editor CTE*, 1995.

[5] Wikipedia, http://de.wikipedia.org/wiki/Klassifikationsbaum-Methode. *Klassifikationsbaum-Methode*, Oktober 2014.