Testautomatisierung mit Selenium

- Teilautomatisierte Generierung von Page Objects -

Fakultät für Informatik und Mathematik
der Hochschule München

Masterarbeit

vorgelegt von

Matthias Karl
Matrikel-Nr: 03280712

im <Datum>

Prüfer: Prof. Dr. Ullrich Hafner

Eidesstattliche Erklärung

Hiermit versichere ich, die vorliegende Studienarbeit selbstständig und nur unter Verwendung der von mir angegebenen Quellen und Hilfsmittel verfasst zu haben. Sowohl inhaltlich als auch wörtlich entnommene Inhalte wurden als solche kenntlich gemacht. Die Arbeit hat in dieser oder vergleichbarer Form noch keinem anderem Prüfungsgremium vorgelegen.

Datum: _____ Unterschrift: _____

Zusammenfassung / Abstract

Inhaltsverzeichnis

1 Einleitung

2 Grundlagen

2.1 Software-Qualität

Nahezu jeder Programmierer ist schon einmal mit dem Begriff der Software-Qualität in Berührung gekommen. Diesen Qualitätsbegriff jedoch genau zu fassen erweist sich als schwierig. Die DIN-ISO-Norm 9126 definiert Software-Qualität wie folgt:
„Software-Qualität ist die Gesamtheit der Merkmale und Merkmalswerte eines Software-Produkts, die sich auf dessen Eignung beziehen, festgelegte Erforderniss zu erfüllen."[ISO01] Aus dieser Definition wird deutlich, dass es sich bei dem Begriff der Software-Qualität eine multikausale Größe handelt. Das bedeutet, dass zur Bestimmung der Qualität einer Software nicht ein einzelnes Kriterium existiert. Vielmehr verbergen sich hinter dem Begriff eine ganze Reihe verschiedener Kriterien die je nach den gestellten Anforderungen in ihrer Relevanz variieren.[Hof13, vgl. Seite 6 ff.] Sammlungen solcher Kriterien werden in sogenannten Qualitätsmodellen zusammengefasst. Die DIN-ISO-Norm 9126 bietet selbst ein solches Qualitätsmodell und definiert damit eine Reihe von wesentlichen Merkmalen, die für die Beurteilung der Software-Qualität eine Rolle spielen. Diese Merkmale sind in der Abbildung 2.1 zusammengefasst. Eine nähere Definition der einzelnen Begriffe des Qualitätsmodells kann beispielsweise dem Buch Software-Qualität von Dirk W. Hoffmann entnommen werden. [Hof13, Seite 7 ff.] Um die Qualität einer Software zu Steigern, bietet die moderne Software-Qualitätssicherung eine Vielzahl von Methoden und Techniken. Ein Teil der Methoden geht dabei davon aus, dass ein qualitativ hochwertiger Prozess der Produkterstellung die Entstehung von qualitativ hochwertigen Produkten begünstigt. Das Augenmerk wird hierbei also auf die Prozessqualität gelegt. Diese Methoden fallen in den Bereich der Prozessqualität. Die klassischen Vorgehensmodelle der Softwarentwicklung werden z.B. hier eingeordnet. Einen weiteren Bereich bilden die Methoden die zur Verbesserung der Produktqualität dienen. Bei diesen Methoden wird das Softwareprodukt direkt bezüglich der Qualitätsmerkmale überprüft. Dieser Bereich unterteilt sich in die konstruktiven und analytischen Qualitätssicherung. Unter konstruktiver Qualitätssicherung versteht man den Einsatz von z.B. Methoden, Werkzeugen oder Standards die dafür sorgen, dass ein Produkt bestimmte Forderungen erfüllt.

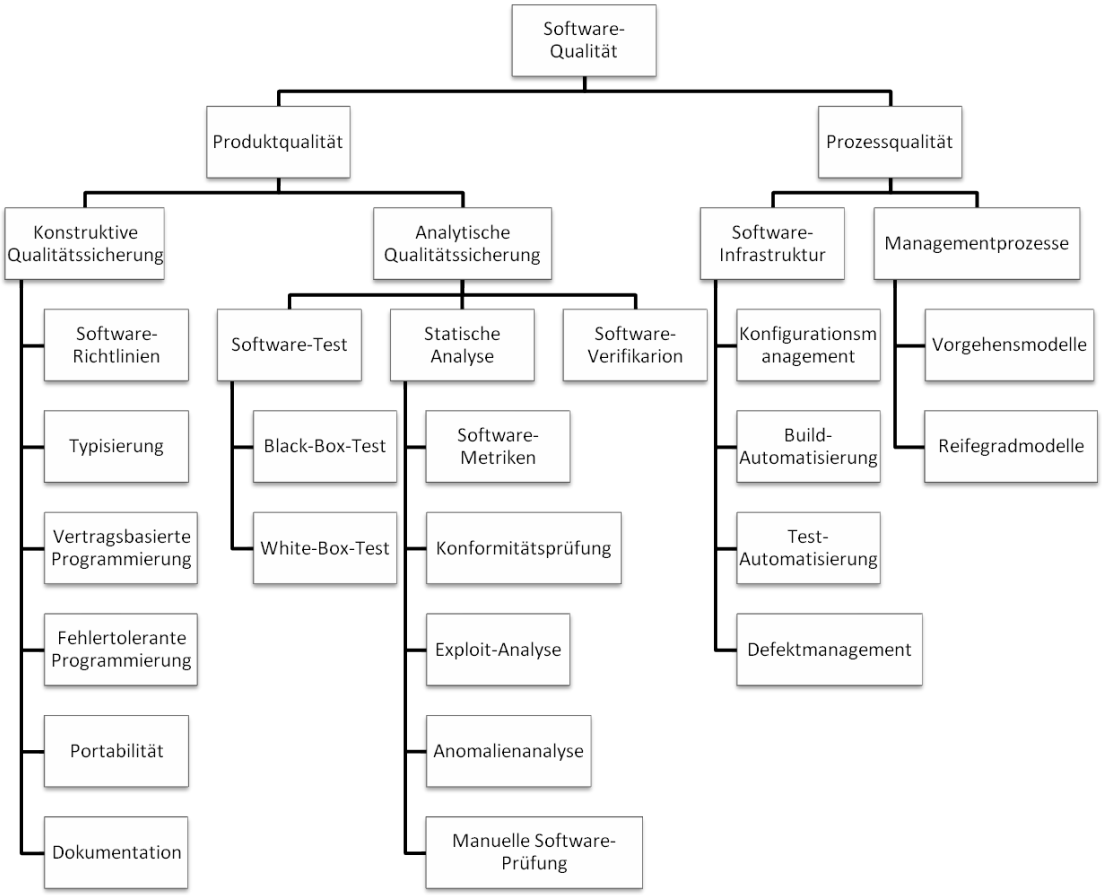
Quelle: [ISO01]

Abbildung 2.1: Qualitätsmerkmale von Softwaresystemen (ISO 9126)

Unter analytische Qualitätssicherung versteht man den Einsatz von analysierenden bzw. prüfenden Verfahren, die Aussagen über die Qualität eines Produkts machen. In diesem Bereich der Qualitätssicherung befindet sich beispielsweise der klassische Software-Test.[Hof13, vgl. Seite 19 ff.] Eine Übersicht über das gesamte Gebiet der SoftwareQualitätssicherung, wie es sich uns gegenwärtig darstellt, ist in Abbildung 2.2 dargestellt.

2.2 Softwaretest

Im laufe der Zeit wurden viele Versuche unternommen um die Qualität von Software zu steigern. Besondere Bedeutung hat dabei der Software-Test erlangt. Der IEEE Std 610.12 definiert den Begriff Test als das ausführen einer Software unter bestimmten Bedingungen mit dem Ziel, die erhaltenen Ergebnisse auszuwerten, also gegen erwartete Werte zu vergleichen. (Im Original: „An activity in which a system or component is executed under specific conditions, the results are observed or recorded, and an evaluation is made of some aspect of the system or component.“ [IEE91]) Bereits zu beginn der Softwareentwicklung hat man versucht Programme vor ihrer Auslieferung zu testen. Der dabei erzielte Erfolg entsprach nicht immer den Erwartungen. Im laufe der Jahre wurde das Testen daher auf eine immer breitere Grundlage gestellt. Es entwickelten sich Unterteilungen des Software-Tests die bis heute bestand haben. Hier wären zu nennen:



Quelle: [Hof13, vgl. Seite 20]

Abbildung 2.2: Übersicht über das Gebiet der Software-Qualitätssicherung

- White-Box-Test
- Black-Box-Test und externe Testgruppe
- Volume Test, Stress Test und Test auf Systemebene

Jeder dieser Begriffe beschreibt bestimmte Techniken, die bei konsequenter Anwendung dazu führen können Fehler in Softwareprodukten zu identifizieren. [Tha02, vgl. Seite 18]
Neben der Auswahl der richtigen Techniken für ein bestimmtes Problem spielt in der Praxis die Testkonstruktion eine zentrale Rolle. Bereits für kleine Programme ist es faktisch nicht mehr möglich das Verhalten einer Software für alle möglichen Eingaben zu überprüfen. Es muss sich daher immer auf eine vergleichsweise winzige Auswahl an Testfällen beschränkt werden. Testfälle unterscheiden sich jedoch stark in ihrer Relevanz. Die Auswahl der Testfälle hat daher einen großen Einfluss auf die Anzahl der gefundenen Fahler und damit auch auf die Qualität des Endprodukts. [Hof13, vgl. Seite 22]

Dennoch ist der Software-Test eines der am meisten verbreiteten Techniken zur Verbesserung der Softwarequalität. Diese Technik alleine reicht jedoch nicht aus um über lange Sicht gute Software zu produzieren. Ein großer Nachteil des Softwaretests ist es, dass Fehler erst in einer relativ späten Phase der Entwicklung identifiziert werden. Je später eine Fehler jedoch entdeckt wird, umso teurer wird auch seine Beseitigung. Abbildung 2.2 zeigt, dass der Software-Test nur eine von vielen Techniken des Qualitätsmanagement darstellt. Um die Probleme des Software-Tests auszugleichen, ist es daher ratsam, sich bei der Qualitätssicherung möglichst breit aufzustellen und sich nicht nur auf die analytische Qualitätssicherung in Form des Software-Tests zu verlassen. [Tha02, vgl. Seite 18]

2.3 Testautomatisierung

Das Testen von Software macht in heutigen Projekten einen großen Teil der Projektkosten aus. Studien haben gezeigt, dass das Testen für 50% und mehr der gesamten Projektkosten verantwortlich sein kann. [RW06] Mit Steigender Komplexität der Softwäre steigen diese Kosten immer weiter an. Um diese Kosten zu reduzieren haben sich im laufe der Zeit die bestehenden Testmethoden immer weiter entwickelt und auch neue Ansätze herausgebildet. Einer dieser Ansätze ist es Software-Tests möglichst automatisiert durchzuführen. Diesen Ansatz fast man mit dem Begriff Testautomatisierung zusammen. Seidl et al. definieren Testautomatisierung als „die Durchführung von ansonsten manuellen Testtätigkeiten durch Automaten.“[Sei12, Seite 7] Diese Definition zeigt, dass das Spektrum der Testautomatisierung breit gefächert ist. Testautomatisierung beschränkt sich nicht nur auf das automatisierte durchführen von Testfällen sondern erstreckt sich über alle Bereiche des Software-Test. [KAPITEL ... BEFASST SICH NÄHER MIT DEN VERSCHIEDENEN BEREICHEN UND ANSÄTZEN DER TESTAUTOMATISIERUNG]. Aus Sicht des Qualitätsmanagement ist die Testautomatisierung sowohl den Methoden zur Steigerung der Produktqualität als auch der Prozessqualität zugeordnet. Ein automatisierter Software-Test hat immer noch den selben Charakter wie ein manueller Software-Test und ist daher ein Teil der analytischen Qualitätssicherung. Allerdings erfordert Testautomatisierung auch immer infrastrukturelle Anpassungen. Automatisierte Testfälle benötigen in der Regel eine Besondere Software-Infrastruktur wie etwa ein Automatisierungsframework. Solche Maßnahmen, die den Programmentwickler aus technischer Sicht in die Lage versetzen, seiner täglichen Arbeit in geregelter und vor allem produktiver Weise nachzugehen, werden den Methoden zur Verbesserung der Prozessqualität zugeordnet. (siehe Abbildung 2.2). [Hof13, vgl. Seite 25]

2.4 Testprozess

Um Software-Tests effektiv und Strukturiert durchzuführen wird eine verfeinerter Ablaufplan für die einzelnen Testaufgaben benötigt. Diesen Ablaufplan fassen Splinner und Linz im fundamentalen Testprozess zusammen. Die einzelnen Arbeitsschritte die im Lebenszyklus eines Software-Tests anfallen werden dabei verschiedenen Phasen zugeordnet. Durch den Testprozess wird die Aufgabe des Testens so in kleinere Testaufgaben untergliedert.
Testaufgaben, die man dabei unterscheidet sind:

- Testplanung und Steuerung
- Testanalyse und Testdesign
- Testrealisierung und Testdurchführung
- Testauswertung und Bericht
- Abschluss der Testaktivitäten

Obgleich die Aufgaben in sequenzieller Reihenfolge im Testprozess angegeben sind, können sie sich überschneiden und teilweise auch gleichzeitig durchgeführt werden. Auf Grundlage des fundamentalen Testprozesses nach Splinner und Linz werden im folgenden diese Teilaufgaben näher beschrieben. [SL07, S.20ff] Für jede Teilaufgabe wird kurz darauf eingegangen worauf bei der Testautomatisierung zu achten ist. Dabei wird sich an die Erkenntnisse von Seidl et al. angelehnt. [Sei12, Seite 9 ff.]

2.4.1 Testplanung und Steuerung

Das testen eines Softwareprojektes stellt eine umfangreiche Aufgabe dar. Um diese zu bewältigen wird eine sorgfältige Planung zu beginn des Softwareprojekts benötigt. Ziel dieser Planung ist es, die Rahmenbedingungen für die Testaktivitäten festzulegen. Nachdem Aufgaben und die Zielsetzung der Tests bestimmt wurden, können die Resourcen die für die Durchführung der Aufgaben benötigt werden geplant werden. Eine Weitere Kernaufgabe der Planung ist es, eine geeignete Teststrategie zu erarbeiten. In Kapitel 2.2 wurde bereits erwähnt, dass das vollständige Testen einer Anwendung in der Regel nicht möglich ist. Die einzelnen Systemteile müssen daher nach Schwere der zu erwartenden Fehlerwirkung priorisiert werden. Je nach schwere der zu erwarteten Auswirkungen, kann dann die Intensität bestimmt werden, mit der ein einzelner Systemteil getestet werden soll. Das ziel der Teststrategie ist es also eine optimale Verteilung der Tests auf das Softwaresystem zu erreichen.

Steht das Softwareprojekt unter einem hohen Zeitdruck müssen Testfälle zusätzlich Priorisiert werden. Um zu verhindern, dass das Testen zu einem endlosen Prozess wird, müssen auch geeignete Testendekriterien festgelegt werden. Anhand dieser Kriterien kann später entschieden werden ob der Testprozess abgeschlossen werden kann. Oftmals wird im Rahmen der Tests eine besondere Werkzeugunterstützung oder Infrastruktur benötigt. Derartige Punkte müssen auch bereits in der frühen Planungsphase berücksichtigt werden. Wie Kapitel 2.2 gezeigt hat, ist das im besonderen der Fall wenn in der Planung beschlossen wird, dass die Testfälle automatisiert durchgeführt werden sollen. Die Gesammten erarbeiteten Rahmenbedingungen werden in einem Testkonzept dokumentiert. Eine mögliche Vorlage für dieses Dokument bietet die internationale Norm IEEE 829-2008 [IEE08]. Neben der frühzeitigen Planung der Tests muss während des gesamten Testprozesses eine Steuerung erfolgen. Hierfür werden die Ergebnisse und Fortschritte der Tests und des Projekts laufend erhoben, geprüft und bewertet. Werden Probleme erkannt, kann so rechtzeitig gegengesteuert werden. Bereits in dieser Frühen Phase werden auch wichtige Grundsteine für eine spätere Testautomatisierung gesetzt. Es muss entschieden werden, in welchen Teststufen und Testbereichen eine Automatisierung eingesetzt werden soll. Vor allem ist die frage zu klären ob und in welchem Ausmaß eine Automatisierung überhaupt sinnvoll ist. Entscheidet man sich für eine Testautomatisierung hat das in der Regel große Auswirkung auf die einzusetzenden Ressourcen und die Zeitliche Planung und Aufwandsschätzung. Diese Informationen fließen auch in das Testkonzept mit ein und erweitern und verändern dort die Rahmenbedingungen.

2.4.2 Testanalyse und Testdesign

In dieser Phase wird zunächst die Qualität der Testbasis überprüft. Alle Dokumente die für die Erstellung der Testfälle benötigt werden müssen in ausreichendem Detailgrad vorhanden sind. Mit Hilfe der qualitätsgesicherten Dokumente kann die eigentliche Testfallerstellung beginnen. Anhand der Informationen aus dem Testkonzept und den Spezifikationen werden mittels strukturierter Testfallerstellungsmethoden logische Testfälle erstellt. Diese logischen Testfälle können dann in einer späteren Phase konkretisiert werden, indem ihnen z.B. tatsächliche Eingabewerte zugeordnet werden. Für jeden dieser Testfälle müssen die möglichen Rand- und Vorbedingungen so wie ein erwartetes Ergebnis bestimmt werden. Um letzteres ermitteln zu können, muss ein so genanntes Testorakel befragt werden. Hierbei handelt es sich um eine Quelle, die auf das erwartete Ergebnis schließen lässt. Als ein solches Testorakel kann beispielsweise die Spezifikation der Software dienen. In dieser Phase beginnt auch die Umsetzung der Testfälle in der Testautomatisierung. Abgestimmt auf die ausgewähl-

ten Automatisierungswerkzeuge und die zu testende Software muss die Umgebung für die Testautomatisierung vorbereitet werden. Anhand der Vorgaben des Testkonzeptes können dann jene Testfälle und Testabläufe ausgewählt werden die im Zuge der Testautomatisierung implementiert werden sollen. Hierbei wird noch einmal die technische Umsetzbarkeit der ausgewählten Testfälle geprüft. Bei der Auswahl der Testfälle sollte zu beginn eine möglichst breite Testabdeckung angestrebt werden. Problemfelder können dann später durch weitere Testfälle in in der Tiefe getestet werden.

2.4.3 Testrealisierung und Testdurchführung

In diesem Schritt des Testprozesses werden aus den Logischen Testfällen der vorangegangenen Phase konkrete Testfälle gebildet. Diese Testfälle werden anhand ihrer fachlichen und technischen Zusammengehörigkeit zu Testszenarien gruppiert und anhand der Vorgaben aus dem Testkonzept priorisiert. Sobald die zu testende Software zur Verfügung steht, kann mit der Abarbeitung der Testfälle begonnen werden. Die dabei erhaltenen Ergebnisse werden vollständig protokolliert. Auf Abweichungen von den erwarteten Ergebnissen muss entsprechend reagiert werden. Korrekturen und nachgehende Veränderungen am Testobjekt werden durch eine Wiederholung der Testläufe abgedeckt. Aus Sicht der Testautomatisierung beginnt in dieser Phase die technische Umsetzung der Testfälle. In vielen Fällen bedeutet das Programmiertätigkeit. Diese Programmiertätigkeiten sind wiederum anfällig für eigene Fehler und müssen daher in angemessener Weise qualitätsgesichert werden. Auch bei der Testautomatisierung ist eine Zusammenfassung von Testfällen sinnvoll. Auf diese weise kann man funktionalen und logischen Abhängigkeiten zwischen den Testfällen gerecht werden. Nach der Implementierung können die geplanten Testfälle durchgeführt werden. Auch bei der Automatisierung ist eine Genaue Protokollierung der Ergebnisse wichtig. Nur durch eine sehr genaue Protokollierung ist es später möglich, aufgetretene Fehler zu lokalisieren. Im Vergleich zu manuellen Tests werden automatisierte Testfälle in der Regel häufiger wiederholt. Es ist daher besonders darauf zu achten, dass erneute Testläufe nicht die Protokolle von vorangegangenen Tests überschreiben sondern ein neues Testprotokoll starten.

2.4.4 Testauswertung und Bericht

2.4.5 Abschluss der Testaktivitäten

2.5 Softwarelebenszyklus

2.5.1 V-Modell

2.5.2 Agil

3 Testautomatisierung

3.1 Warum Testautomatisierung

3.2 Bereiche der Testautomatisierung

Einteilung nach testprozess nicht perfekt. Dahre eine einteilung die der automatisierung besser passt: A Search-Based Approach for Cost-Effective Software Test Automation Decision Support and an Industrial Case Study

3.2.1 Testdesign

3.2.2 Testcodeerstellung

c and r

3.2.3 Testdurchführung

3.2.4 Testauswertung

3.3 Schnittstellen der Testautomatisierung zum System

Jeder testfall muss in irgendeiner weise mit dem zu testenden objekt interagieren. Analog zu manuellen tests gibt es hierfür verschiedene möglichkeiten.

3.3.1 API

3.3.1.1 JUnit

3.3.2 GUI

Web

4 Testautomatisierung mit Selenium

4.1 Selenium

4.2 Testdurchführung mit Selenium

4.3 Testcodeerstellung mit Selenium

4.3.1 Recorde-and-playback

4.3.1.1 Vorteile von Recorde-and-playback

4.3.1.2 Probleme von Recorde-and-playback

4.3.2 Manuell

4.3.3 Page Object Pattern

4.3.3.1 Vorteile des Page Object Pattern

4.3.3.2 Probleme des Page Object Pattern

5 Teilautomatisierte Generierung von Page Objects

5.1 übersicht über die Idee

5.2 einordnung des Testharness und gui in die Geamtstruktur (Deploymentdiagramm)

5.3 übersicht über Aufbau des Systems

5.3.1 pro modul ein kapitel

5.4 Vorteile und Probleme

5.5 Anwendung

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[Hof13] Hoffmann, Dirk W.: *Software-Qualität*. 2013. Berlin : Springer, 2013. – ISBN 9783540763222

[IEE91] IEEE: IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries. In: *IEEE Std 610* (1991), Januar, S. 1–217. http://dx.doi.org/10.1109/IEEESTD.1991.106963. – DOI 10.1109/IEEESTD.1991.106963

[IEE08] IEEE: *IEEE Std 829-2008 IEEE Standard for Software and System Test Documentation*. Juli 2008

[ISO01] ISO/IEC: *ISO/IEC 9126. Software engineering – Product quality*. 2001. ISO/IEC, 2001

[RW06] Ramler, Rudolf ; Wolfmaier, Klaus: Economic Perspectives in Test Automation: Balancing Automated and Manual Testing with Opportunity Cost. In: *Proceedings of the 2006 International Workshop on Automation of Software Test*. New York, NY, USA : ACM, 2006 (AST '06). – ISBN 1–59593–408–1, 85–91

[Sei12] Seidl, Richard: *Basiswissen Testautomatisierung / Richard Seidl ; Manfred Baumgartner ; Thomas Bucsics*. 1. Aufl. Heidelberg : dpunkt-Verl., 2012. – ISBN 978–3–89864–724–3

[SL07] Spillner, Andreas ; Linz, Tilo: *Basiswissen Softwaretest*. 3. Aufl. Heidelberg : dpunkt-Verl., 2007. – ISBN 3–89864–358–1

[Tha02] Thaller, Georg E.: *Software-Test*. 2., aktualisierte und erw. Aufl. Hannover : Heise, 2002. – ISBN 3–88229–198–2