BACHELORARBEIT

im Studiengang Bachelor Informatik

Software-Test von Web-Applikationen

Ausgeführt von: Bernhard Posselt
Personenkennzeichen: 1010257029

Begutachter: MSc Benedikt Salzbrunn

Wien, 22. Mai 2013

FACHHOCHSCHULE
TECHNIKUM WIEN

Eidesstattliche Erklärung

„Ich erkläre hiermit an Eides statt, dass ich die vorliegende Arbeit selbständig angefertigt habe. Die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Die Arbeit wurde bisher weder in gleicher noch in ähnlicher Form einer anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht. Ich versichere, dass die abgegebene Version jener im Uploadtool entspricht."

Ort, Datum

Unterschrift

Kurzfassung

Die Durchführung und Erstellung von automatisierten Tests für Web-Applikationen unterscheidet sich von klassischen Applikationen. Aufgrund der komplexeren Infrastruktur und Modularisierung werden zusätzliche Testfälle und Strategien benötigt um Web-Applikationen ausreichend abzudecken und eine fortwährende Qualität zu gewährleisten. Diese Arbeit soll Möglichkeiten für den Test von Web-Applikationen anhand eines Projektes aufzeigen und Anpassungen und Anwendung der vier Testformen des V-Modells: Unit Test, Integration Test, System Test und Acceptance Test.

Schlagwörter: Webapplikationen, automatisierte Tests, Webtest

Abstract

Creation and execution of automatic web application tests is different from tests of classic applications. A more complex infrastructure and modularisation require additional testcases and strategies to guarantee a good enough test coverage which in return ensures constant quality. This thesis highlights various possibilities to test web applications based on a real world project and shows how to use and adjust the V-Model's four test methods: Unit Test, Integration Test, System Test and Acceptance Test.

Keywords: web applications, automatic tests, webtest

Inhaltsverzeichnis

1 Einführung

Durch die zunehmende Verbreitung von Smartphones und anderen mobilen Geräten erfährt der Markt für Software-Applikationen eine zunehmende Fragmentierung [1][S. 1]. Viele dieser Plattformen erfordern das Erlernen von unterschiedlichen Programmiersprachen, Frameworks und Betriebssystemen [2][3]. Will ein/eine Software-EntwicklerIn eine Applikation plattformübergreifend anbieten, erfordert dies daher einen höheren Zeit- und Kostenaufwand.

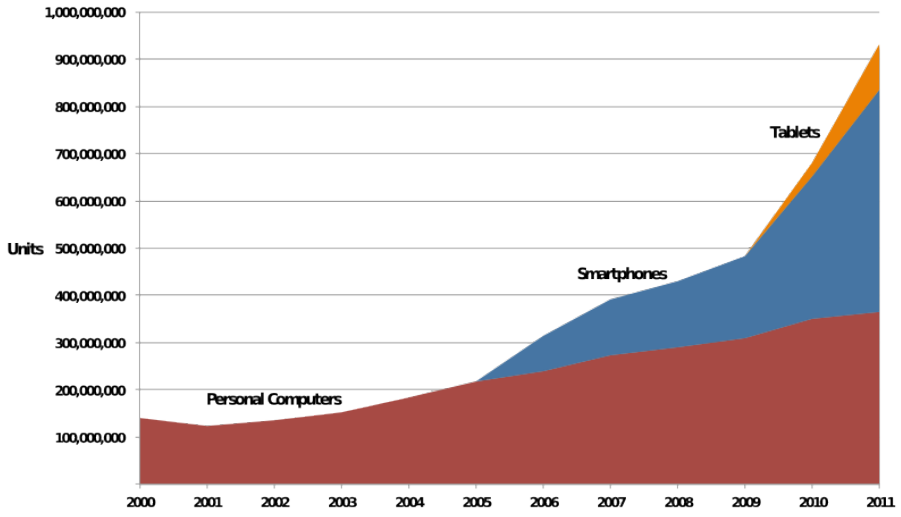

Abbildung 1.1: Entwicklung der PC und Mobilgerätverkäufe, Quelle: The Future Of Mobile, Henry Blodget. Zahlen von Gartner, IDC, Strategy Analytics, company filings, BI Intelligence estimates

Da viele dieser Mobilgeräte über einen Web-Browser verfügen, wird das Web als Applikations-Plattform immer attraktiver. Zudem erlauben Frameworks wie PhoneGap [4] dem/der EntwicklerIn eine plattformübergreifende Mobil-Applikation auf Basis von Web-Technologien zu erstellen, welche sich nahtlos in das System integrieren.

1.1 Problemstellung

Aufgrund der Vielfalt an unterschiedlichen Web-Browsern erhöht sich jedoch auch der Testaufwand der Applikation. Die Browser implementieren die vom W3C vorgeschlagenen Standards ab und zu anders und in unterschiedlicher Geschwindigkeit. Nicht selten kommt es vor, dass gewisse Funktionen nur in einer eingeschränkten Auswahl von Web-Browsern voll funktionsfähig sind und zusätzliche Anpassungen erfordern. [5]

Ein weiteres Problem ist das verhaltene Upgradeverhalten der NutzerInnen. Im Falle des Internet Explorers 7 dauerte es mehr als 19 Monate bis rund die Hälfte der NutzerInnen auf die neue Version umgestiegen sind. [6][S. 3]

1.2 Lösungsansatz

Um eine konstante Qualität der Web-Applikation auf allen Plattformen zu gewährleisten, muss das bisherige Testssystem angepasst werden, um der höheren Komplexität von Web-Applikationen gerecht zu werden.

Diese Bachelorarbeit soll die zusätzlichen Herausforderungen beim Testen von Web-Applikationen aufzeigen. Des Weiteren soll sie einen Überblick über mögliche Anwendungen und Anpassungen der drei häufigsten Testarten - den Unit Tests, Integration Tests, System Tests und Acceptance Tests - im Bereich der Web-Applikationen geben.

1.3 Aufbau

Kapitel 2 behandelt den Sinn und die Grenzen des Software-Tests und wie er sich in den Entwicklungsprozess einbinden lässt.

In Kapitel 3 wird auf die Unterschiede zwischen klassischen Applikationen und Web-Applikationen eingegangen. Dadurch sollen zusätzliche Probleme, die sich durch das Applikations-Modell ergeben, aufgezeigt werden.

Kapitel 4 beschreibt die Planung und den Ablauf des Testprozesses durch die Verwendung eines Testplanes.

Kapitel 5 beschäftigt sich näher mit dem praktischen Einsatz von Unit Tests, welche serverseitigen und clientseitigen Quellcode abdecken.

Kaptitel 6 nimmt sich des Themas des Integration Tests an und zeigt dessen Einsatzzweck bei Web-Applikationen.

Kaptitel 7 behandelt den Systemtest und dem Test der verschiedenen Plattformen.

Kapitel 8 beschäftigt sich mit dem Thema des Acceptance Tests und wie dieser bei Web-Applikationen umgesetzt werden kann.

2 Warum testen

Keine Applikation ist fehlerfrei[7][S. 9]. Diese Fehler führen nicht nur zu unzufriedenen Kunden, sondern auch zu hohen Kosten: „Im Jahr 2000 wurde in den USA ein Schaden durch Softwarefehler in der Auto- und Flugzeugindustrie von 1,8 Milliarden US-Dollar errechnet. Dies entspricht ca. 16 % des Softwareumsatzes.“[8][S. 15]

Die Fehler-Rate wird auf ungefähr 3 pro 1.000 Zeilen geschätzt, was bei einer aufwendigeren Applikation mit 100 Millionen Zeilen Quellcode eine durchschnittliche Anzahl von 300.000 Fehlern ergibt [9][S. 10].

Je früher diese Fehler entdeckt werden, desto kostengünstiger können diese beseitigt werden [8][S. 17]. Daher ist es wichtig, möglichst früh mit dem Testen zu beginnen und den Software-Entwicklungsprozess dementsprechend anzupassen [8][S. 16].

2.1 Manuelle Tests

2.2 Automatisierten Tests

Da die vorhandenen Tests durch die Einbindung in den Entwicklungsprozess öfters durchgeführt werden müssen, kann sich durch das manuelle Testen der immer gleichen Tests schnell eine gewisse Eintönigkeit einstellen. Das wiederum kann unter anderem dazu führen, dass Tester bestimmte Tests nicht korrekt oder ineffizient ausführen.

Dieses Problem kann durch eine Automatisierung des Testprozesses gelöst werden. Die Automatisierung erlaubt zudem bei zukünftigen Testdurchläufen eine Reduzierung auf ein Minimum des ursprünglichen Zeitaufwandes. Daher sollte versucht werden, möglichst viele dieser Tests zu automatisieren. [10][S. 22-23]

Auch können automatisierte Tests rund um die Uhr ausgeführt werden, was eine effizientere Nutzung der Zeit garantiert.

2.3 Testen im Projektzyklus

Tests können jedoch nicht nur dazu verwendet werden, um Fehler in der Applikation zu finden, sondern auch um einen Überblick über den derzeitigen Stand der Implementation zu gewinnen: Sie geben dem/der EntwicklerIn und ProjektmanagerIn ein direktes Feedback über bereits korrekt implementierte Teile der Software-Spezifikation. Auch Milestones können durch Tests definiert werden. [10][S. 2]

Zudem ist eine Abschätzung riskanter Bereiche möglich, die durch eine erhöhten Testbedarf bestimmter Bereiche offensichtlich wird. [11][S. 34]

2.4 Grenzen von Software-Tests

Das Vorhandensein von Tests kann niemals eine komplett fehlerfreie Software garantieren, nur eine Abwesenheit von bestimmten Fehlern kann garantiert werden, nämlich jenen, die explizit mit Tests abgedeckt werden [9][S. 12]. Dies resultiert unter anderem daraus, dass die Anforderungen an die Software oft nicht komplett spezifiziert oder ungenau sind und die Komplexität der Applikation im Laufe der Entwicklung stark ansteigt. Zusätzlich können die Testfälle durch die vielen möglichen Eingaben lediglich einen kleinen Bereich der Funktionsweise abdecken [12][S. 243].

Es ist jedoch nicht erstrebenswert, die ganze Applikation mit Tests abzudecken, da dies zu zu hohen Entwicklungskosten führen kann. Idealerweise werden daher nur jene kritischen Fälle mit Tests abgedeckt, deren Fehlerkosten die Kosten für die Erstellung der Tests übertreffen. [9][S. 12]

Die implementierten Testfälle müssen auch regelmäßig gewartet und aktualisiert werden, um ihre Effektivität zu gewährleisten, denn diese nimmt auf Dauer ab. Es ensteht eine sogenannte „Testresistenz“
[9][S. 12], die daraus resultiert, dass die bestehenden Tests nur bekannte Fehlerfälle abdecken und neue mögliche Fehlerfälle nicht berücksichtigen. [9][S. 12-13]

3 Testplan

Mit dem Erstellen des Projektplanes sollte gleichzeitig auch ein Testplan erstellt werden [13][S. 24]. Dieser erlaubt, den Testprozess näher zu spezifizieren und somit den erforderlichen Aufwand besser abschätzen zu können [14][S. 18] und den Testprozess eventuell an ein externes Team auszulagern. Zusätzlich kann der Plan auch dazu verwendet werden, die Abnahmekriterien exakt zu definieren, um das Projektrisiko zu verringern. [13][S. 26]

Im Testplan werden unter anderem folgende Punkte behandelt [10][S. 3]:

- Was wird getestet? Welche Bereiche besitzen eine höhere Priorität als andere?
- Wo wird getestet? Mit welchen Konfigurationen, Soft- und Hardware Plattformen respektive Versionen werden getestet?
- Wie wird getestet? mit welchen Tools wird getestet?
- Wer testet?
- Wie lange wird getestet? Bis wann muss ein Test erfolgreich absolviert werden?
- Was wird nicht getestet?

Der Testplan wächst mit dem Projekt und muss demensprechend regelmäßig auf Korrektheit und Angemessenheit überprüft werden. [9][S. 25]

Um den Ablauf der Tests zu beschleunigen, sollten unnötige Testdurchläufe vermieden werden. Dies wird mit der Erstellung eines Smoke-Tests erreicht der essentielle Testszenarien durchtestet. Schlagen diese fehl, sind weitere Testdurchläufe nicht notwendig und der Testdurchlauf kann abgebrochen werden. [9][S. 25-26]

3.1 Vorteile eines Testplans

Da die Entwicklung der Applikation meistens mit einem hohen Zeitdruck verbunden ist[12][S. 244], ist es wichtig, bestimmte Testfälle höher zu priorisieren als andere [11][S. 34]. Dazu werden alle essentiellen Testfälle herausgearbeitet und dokumentiert. Aufgrund der Dokumentation der Testfälle, ergeben sich zusätzlich folgende Vorteile:

- Fehlende Testfälle können schnell erkannt werden [14][S. 18]
- Redundante Testfälle werden identifziert, Testfälle können zusammegelegt werden um Zeit zu sparen [11][S. 34]
- Verschiedene Testbereiche können an mehrere Personen mit unterschiedlichen Fachkenntnissen verteilt werden um Personalengpässe zu vermeiden [14][S. 19]
- Ineffiziente Test-Tools und Test-Strategien können erkannt und verbessert werden [9][S. 25]

- Geschäftskritische Bereiche und solche mit einer erhöhten Fehleranfälligkeit werden sichtbarer [11][S. 34]. Dies ist vor allem wichtig, da Fehler ungleich verteilt sind und in bestimmten Bereichen häufiger vorkommen als in anderen [9][S. 12]
- Sinnlose Testfälle und Testbereiche können aussortiert werden
- Abhängigkeiten der Testfälle untereinander und der Implementation werden transparent und erlauben schnell auf Änderungen zu reagieren[10][S. 4]

4 Unterschiede zu klassischer Software

Im Gegensatz zu klassischen Desktop- oder Mobil-Applikationen bestehen Web-Applikationen wegen ihrer Client-Server-Architektur immer aus mehreren Modulen, die meist über ein Netzwerk miteinander verbunden sind, beispielsweise:

- Datenbank-Server
- Web-Server
- Applikations-Server
- Authentifizierungs-Server
- Web-Browser

Durch diesen modularen Aufbau ist es besonders schwer einen Fehler zu lokalisieren. Ein Fehler kann z.B. durch einen Fehler im Quellcode des Applikations-Servers oder durch ein Netzwerkproblem entstanden sein. [11][Foreword]

Außerdem gibt es eine größere Vielfalt an Plattformen, auf welchen Web-Applikationen ausgeführt wird: Auf der Serverseite sind diese Plattformen noch vom/von der BetreiberIn festlegbar, sprich welches Betriebssystem und welche Datenbank eingesetzt wird, auf der Clientseite ist dies aber schon nicht mehr möglich. Die BesucherInnen der Webseite verwenden verschiedene Web-Browser auf verschiedenen Betriebssystemen, welche beide in unterschiedlichen Versionen vorliegen können. Auch können unterschiedliche Plugins und Fonts in unterschiedlichen Versionen installiert sein. [11][Foreword]

Zudem verlagert sich auch immer mehr Applikations-Logik von der Server- auf die Clientseite[11][S. 13]. Durch das Verwenden von *Events*, welche unter anderem durch BenutzerInnen-Eingaben ausgelöst werden können, stellt clientseitiger Code eine größere Herausforderung für den/die TesterIn dar als Serverseitiger: Events können in unterschiedlicher Reihenfolge und Kombination auftreten, manche Aktionen lösen sogar mehrere Events aus. [11][S. 18]

Eine weitere Herausforderung stellt das Session Modell von Web-Applikationen dar: viele Web-Applikationen verwenden nur eine Session pro NutzerIn, erlauben aber mehrere gleichzeitige Logins. Werden mehrere Instanzen der Applikation gestartet - z.B. loggt sich der/die NutzerIn auf dem Mobiltelefon und dem Laptop auf der Webseite ein - kann dies zu Synchronisationsproblemen zwischen den einzelnen Instanzen führen: Wird in einer Instanz ein Eintrag gelöscht, kann dieser durch eine fehlerhafte Synchronisation in einer andere Instanz immer noch existieren und in weiterer Folge zu Fehlern führen. [11][S. 20]

Diese Vielfalt an verschiedenen möglichen Konfigurationen und Herausforderungen erfordert eine neue Herangehensweise an das Thema Software-Test: Die bestehenden Techniken sind „zwar auch notwendig, aber nicht ausreichend, um die Qualität der Applikation sicherzustellen“ [13][S. 18]

5 Projektumfeld

6 Unit Test

7 Integration Test

8 System Test

9 Acceptance Test

10 Zusammenfassung

Literaturverzeichnis

[1] F. L. und Weiguo Ye, "Operating system battle in the ecosystem of smartphone industry," 2009.

[2] "Android Developer Website," Website, Google, 2013, http://developer.android.com/ [Zugang am 20.05.2013].

[3] "iOS Dev Center," Website, Apple, 2013, https://developer.apple.com/devcenter/ios/index.action [Zugang am 20.05.2013].

[4] "PhoneGap Website," Website, Adobe, 2013, http://phonegap.com/ [Zugang am 20.05.2013].

[5] "Can I use... Compatibility tables for support of HTML5, CSS3, SVG and more in desktop and mobile browsers," Website, 2013, http://caniuse.com/ [Zugang am 20.05.2013].

[6] G. O. u. M. M. Stefan Frei, Thomas Duebendorfer, "Understanding the web browser threat: Examination of vulnerable online web browser populations and the ïnsecurity iceberg‖," *ETH Zurich Tech Report Nr. 288*, 2008.

[7] M. Schmidt, "An Empirical Investigation of Human-Based and Tool-Supported Testing Strategies," Master's thesis, Technische Universität Wien, 2011.

[8] F. Richter, "Betriebliche Einführung von automatischen Softwaretests unter Berücksichtigung ökonomischer und technischer Rahmenbedingungen," Master's thesis, Technische Universität Wien, 2007.

[9] M. Oberreiter, "Evaluierung, Konzeption und Härtung eines Testprozesses für automatisierte Regressionstests in einem mittelgroßen Entwicklungsszenario," Master's thesis, Technische Universität Wien, 2011.

[10] A. Waser, "Test Automation A Case Study," Master's thesis, Technische Universität Wien, 2002.

[11] H. Q. Nguyen, *Testing Applications on the Web*. Wiley Computer Publishing, 2001.

[12] D. W. Hoffmann, *Software-Qualität*. Springer, 2008.

[13] S. Avci, "Evaluieren von automatisierten Tests bei Web-Applikationen," Master's thesis, Technische Universität Wien, 2010.

[14] C. Dobritzhofer, "Towards Test Methodologies for Developing Large Systems," Master's thesis, Technische Universität Wien, 1994.

Abbildungsverzeichnis

Abkürzungsverzeichnis

www	World Wide Web
W3C	World Wide Web Consortium