Domain Specific Software Engineering for Web Sites

Simon Lang

15. Juni 2015
Version 1.0.0

Studiengang Informatik 5 Ba 2012
Seminar Sicherheitsanwendungen/PKI
Dozent Daniel Liebhart
Schule ZHAW - School of Engineering

Kurzfassung

Schlagwörter: Software Engineering, Domain Specific Software Engineering, Web Sites, Internet

Inhaltsverzeichnis

Abbildungsverzeichnis

1 *Responsive vs Adaptive Design for UI.* URL: http://blog.zymr.com/responsive-vs-adaptive-design-for-ui (besucht am 31. 05. 2015).

2 *The Difference Between Adaptive Design And Responsive Design | Search Engine People.* URL: http://www.searchenginepeople.com/blog/the-difference-between-adaptive-design-and-responsive-design.html (besucht am 31. 05. 2015).

3 *AjaxWorld Talk: Building Rich Internet Applications Using Microsoft Silverlight 2 - Brad Abrams - Site Home - MSDN Blogs.* URL: http://blogs.msdn.com/b/brada/archive/2008/10/20/ajaxworld-talk-building-rich-internet-applications-using-microsoft-silverlight-2.aspx (besucht am 07. 06. 2015).

4 *Waterfall model-de - Wasserfallmodell – Wikipedia.* URL: https://de.wikipedia.org/wiki/Wasserfallmodell#/media/File:Waterfall_model-de.svg (besucht am 03. 06. 2015).

5 *Scrum process-de - Scrum – Wikipedia.* URL: https://de.wikipedia.org/wiki/Scrum#/media/File:Scrum_process-de.svg (besucht am 03. 06. 2015).

Tabellenverzeichnis

Akronyme

Bezeichnung	Beschreibung
AJAX	Asynchronous JavaScript and XML
AWD	Adaptive Web Design
CA	Certification Agency
CSS	Cascading Style Sheets
EDGE	Enhanced Data Rates for GSM Evolution
GPRS	General Packet Radio Service
HSDPA	High-Speed Downlink Packet Access
HTTPS	HyperText Transfer Protocol Secure
JS	Java Script
KMU	kleine und mittlere Unternehmen
LTE	Long Term Evolution
RIA	Rich Internet application
RWD	Responsive Web Design
SEO	Search Engine Optimization
SLA	Service Layer Agreement
SPA	Single Page Applications
W3C	World Wide Web Consortium
WAP	Wireless Application Protocol
WWW	World Wide Web
ZHAW	Zürcher Hochschule für Angewandte Wissenschaften

Glossar

HTML

HTML (engl. Hypertext Markup Language) ist eine textbasierte Auszeichnungssprache zur Strukturierung digitaler Dokumente wie Texte mit Hyperlinks, Bildern und anderen Inhalten. HTML-Dokumente sind die Grundlage des World Wide Web und werden von Webbrowsern dargestellt.

KAPITEL 1

Einleitung

1.1 Ziele

Jede Software, die entwickelt wird, ist unterschiedlich und doch haben sie einiges gemeinsam. Diese Arbeit widmet sich der Softwareentwicklung für Webseiten und versucht darzustellen, was in diesem Bereich im speziellen zu beachten ist.

1.2 Begründung

Das Internet ist das weltweit grösste Netzwerk. Die Anzahl der Teilnehmer ist unmöglich zu bestimmen, da Endgeräte sich einloggen und auch wieder ausloggen. Laut IWS hatten im März 2007 etwa 16,9 Prozent der Weltbevölkerung Zugang zum Internet[1]. In der Schweiz verfügten im Jahr 2012 85 Prozent der Bevölkerung über einen privaten Internetzugang. Insgesamt nutzten im Herbst 2013 81% der Bevölkerung regelmässig (täglich oder mehrmals pro Woche) das Internet. Fast 40% der Bevölkerung nutzten 2012 einen Breitbandanschluss, mit einer Übertragungsrate von mehr als 256 Kbit/s[2].

Dadurch wird auch entsprechend viel Software für das weltweit grösste Netzwerk produziert. Darüber handelt diese Schreiben.

1 *World Internet Users Statistics and 2014 World Population Stats.* URL: http://www.internetworldstats.com/stats.htm (besucht am 30. 05. 2015).

2 *Internet – Wikipedia.* URL: https://de.wikipedia.org/wiki/Internet#Nutzerzahlen (besucht am 30. 05. 2015).

KAPITEL 2

Beschreibung der Aufgabe

2.1 Aufgabenstellung

2.1.1 Ausgangslage

Im Jahre 1969 entstand das Arpanet. Nach einem langsamen Start vergrösserte es sich rasant zu dem heute bekannten Internet. Die Tendenz ist immer noch rasant wachsend. Beschleunigt wird der Wachstum heutzutage durch die unzähligen mobilen Geräte, die Zugriff auf das Internet haben. Durch den rasanten Wechsel der damit verbunden Technologien muss sich auch die Softwareenticklung stets anpassen.

2.1.2 Ziele der Arbeit

Ziel der Arbeit ist es, die Schwierigkeiten und Probleme des Software Engineerings für die Webseitenentwicklung zu analysieren. Es sollen Konzepte gezeigt werden, wie Software für die verschiedenen Endgeräten mit ihren unzähligen Browsern entwickelt werden. Dazu wird eine grobe Übersicht über die Software für Webseiten gegeben.

2.1.3 Aufgabenstellung

Es soll ein Dokument zum Thema DSSE für die Webseitenentwicklung erstellt werden. Das Papier soll die Schwierigkeiten von Software in dieser Branche aufzeigen und einen groben Überblick über das Thema geben.

2.1.4 Erwartete Resultate

Folgende Ergebnisse werden am Schluss dieser Seminararbeit erwartet:

- Dokumentation
- Handout
- Präsentation

KAPITEL 3

Einführung

3.1 Über den Autor

Mein Name ist Simon Lang. Als gelernter Informatiker arbeite ich seit 2006 in der Webentwicklungsbranche. Seit 2012 studiere ich Informatik an der Zürcher Hochschule für Angewandte Wissenschaften (ZHAW). Durch mehrjährige Erfahrung in der Webentwicklung habe ich mich im Seminar zur Vorlesung *Software Engineering* für die Vertiefung im Web entschieden.

3.2 Aufbau

Die Arbeit befasst sich mit den wesentlichen Schritten der Softwareentwicklung, welche im SWEBOK[1] beschrieben sind.

- Requirements Engineering
- Software Design
- Software Construction
- Software Testing
- Software Maintenance / Software Configuration Management

Als Wissen wird vorausgesetzt, was die grundlegenden Arbeitsschritte in den einzelnen Kabiteln des SWEBOK sind. Die Arbeit geht nicht direkt über das SWEBOK, sondern befasst sich mit den speziellen Ausprägungen für die Softwareentwicklung im Internet.

3.3 Aufbau des Internets

Das Internet ist eine Client/Server Architektur. Die Clients stellen Anfragen an einen Server, welcher eine Antwort zurückliefert. Der Client kann dabei ein Benutzer an seinem Computer sein, oder eine Maschine. In der Reisebranche ist dies zum Beispiel der Fall, wenn der Benutzer etwas bucht. Er bestätigt seinen Einkauf und der Server bucht daraufhin

1 *index • IEEE Computer Society.* URL: http://www.computer.org/web/swebok (besucht am 30. 05. 2015).

die Reise. Der Server verbindet sich dabei auf ein drittes System, zum Beispiel jenes des Reiseveranstalters.

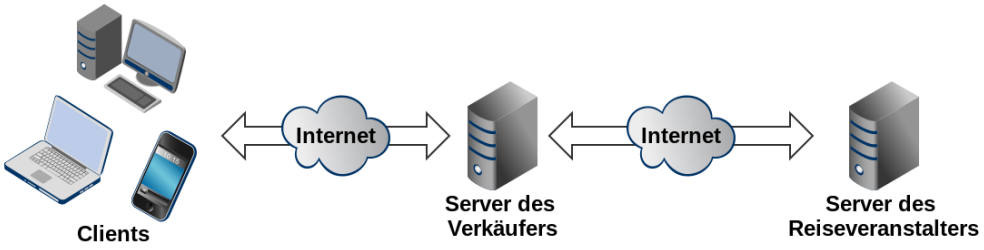

Abbildung 3.1: Aufbau des Internet

Wenn eine Webseite angezeigt werden soll, fragt der Client den Server an, und dieser liefert ein HTML Dokument aus. Dieses kann weitere Dokumente von weiteren Servern abfragen, wie zum Beispiel Bilder. Das HTML Dokument wird von einer Programmiersprache auf dem Server generiert. Das ausgelieferte Dokument kann danach von einer Client-seitigen Sprache weiter modifziert werden. Als Programmiersprache auf dem Client wird meistens Java Script (JS) eingesetzt. Die HTML-Datei sollte nur den Inhalt und die Struktur der Webseite vorgeben. Das Aussehen wird über Cascading Style Sheets (CSS) definiert.

3.4 Standards

Es gibt unzählige Standards für das World Wide Web (WWW). Die Wichtigsten werden vom World Wide Web Consortium (W3C)[1] herausgegeben. Die Standards sind folgendermassen strukturiert:

- Web Design and Applications
- Web of Devices
- Web Architecture
- Semantic Web
- XML Technology
- Web of Services
- Browsers and Authoring Tools

Vorgegeben wird die Sprache HTML und auch wie diese Dokumente aufgebaut werden sollen. Auch wie die Browser die Dateien darzustellen haben ist Teil eines Standards des W3C.

Seit einigen Jahren wurden die Standards für Mobiltelefone und Tablets erweitert. Programmierer für das Internet sollten die gängigsten Standards kennen und sich auch daran halten, so dass andere Entwickler sich einfacher in den bestehenden Code einarbeiten können.

1 *World Wide Web Consortium (W3C)*. URL: `http://www.w3.org/` (besucht am 30. 05. 2015).

3.5 Interne vs. Externe Entwicklung

Webentwickler arbeiten entweder an internen oder externen Projekten. Bei ersteren ist der Arbeitgeber der Auftraggeber, wo hingegen bei Letzteren ein externer Kunde über den Umfang der Software bestimmt. Für die Entwicklung macht dies einen grossen Unterschied, da die internen Projekte meistens über Jahre hinweg gepflegt werden. Wird die Software für ein Kunden erstellt, gibt es zuerst oft eine Projektausschreibung, wo mehrere Softwarehersteller sich bewerben können. Nach der Umsetzung und einer Garantiefrist ist der Auftrag abgeschlossen. In den Projektausschreibungen wird oft ein zu tiefer Preis angeboten, was mit einer Minderung der Qualität ausgeglichen wird.

KAPITEL 4

Requirements Engineering

Dieses Kapitel zeigt die speziellen Eigenschaften des Requirements Engineering im Bereich der Webentwicklung.

Früh in der Entwicklung einer Software muss man die Rahmenbedingungen festlegen. Zuerst wird gezeigt, für welche Geräte und Browser man sich spezialisieren und generalisieren kann.

Danach werden die wichtigsten nicht funktionalen Eigenschaften einer Webseite und zum Schluss noch weitere wichtige Aspekte des Requirement Engineerings aufgezeigt.

4.1 Endgeräte

Es gibt verschiedene Endgeräte, welche die Webseite darstellen müssen.

- PC
- Mobile
- Tablet
- E-Reader
- Spielkonsolen

Als erster Punkt des Requirements Engineering sind die Geräte aufgeführt, da davon vieles abhängt. Zum Beispiel die Grösse des Bildschirmes, die Grösse des Eingabegerätes (Maus, Finger), die Bandbreite und noch weitere Punkte.

Für die verschiedenen Bildschirmgrössen gibt es zwei verschiedene Ansätze, die im Unterkapitel 4.1.1 Responsive- vs Adaptive Web Design beschrieben werden.

Dem Problem der Bandbreite nimmt sich das Kapitel 4.1.2 Bandbreite an.

4.1.1 Responsive- vs Adaptive Web Design

[1] Beim Responsive Web Design (RWD) wird eine Seite generiert, die sich der Bildschirmgrösse anpasst, wo hingegen beim Adaptive Web Design (AWD) für verschiedene Bilschirm-

1 *Responsive vs. Adaptive Design: What's the Best Choice for Designers? - UXPin.* URL: http://blog.uxpin.com/6439/responsive-vs-adaptive-design-whats-best-choice-designers/ (besucht am 31. 05. 2015).

grössen jeweils ein anderes Layout ausgeliefert wird.

Abbildung 4.1: Responsive- vs Adaptive Web Design[a]

a *Responsive vs Adaptive Design for UI.* URL: http://blog.zymr.com/responsive-vs-adaptive-design-for-ui (besucht am 31. 05. 2015).

Beim RWD liefert der Server für jedes Endgerät das selbe HTML aus. Die Elemente ordnen sich jedoch auf den verschiedenen Bildschirmgrössen anders an. Die folgende Grafik illustriert dieses Vorgehen.

Abbildung 4.2: Responsive Web Design[a]

a *The Difference Between Adaptive Design And Responsive Design | Search Engine People.* URL: http://www.searchenginepeople.com/blog/the-difference-between-adaptive-design-and-responsive-design.html (besucht am 31. 05. 2015).

Der Vorteil an dieser Variante ist es, dass keine Überprüfung der Bildschirmgrösse nötig ist und auf dem Bildschirm immer eine angepasste Ansicht ermöglicht wird. Auch wenn man die Grösse des Browserfensters anpasst, erhält man immer eine passende Ansicht. Die Programmierung wird dadurch vereinfacht, sodass nur eine HTML Seite generiert werden muss. Jedoch wird das Layout komplizierter und es müssen mehr Daten an das Engerät ausgeliefert werden.

Beim AWD werden eigene HTML Seiten für die verschiedenen Auflösungen gepflegt. Um den selben Umfang wie das RWD zu ermöglichen, müssen die folgenden sechs Bildschirm-

breiten gepflegt werden:

- 320px
- 480px
- 760px
- 960px
- 1200px
- 1600px

Generell macht es Sinn ein RWD zu pflegen, da der Pflegeaufwand generell geringer ausfällt. Es gibt jedoch auch Gründe für ein AWD. Zum Beispiel wenn eine bestehende Seite für mobile Geräte angepasst werden muss, ist der Aufwand oft geringer, wenn man eine separate Ansicht dafür umsetzt. Die Ladezeiten für ein AWD sind meistens auch besser, was folgende Tabelle aufzeigt. Für diesen Test[1] wurden 15 Webseiten aus den *Alexa Top 100 ranking (US)* Pages ausgewählt und analysiert.

Metrik	AWD	RWD
Antwortzeit	568 ms	1'202 ms
Zeit bis das Dokument vollständig geladen ist	1'536ms	4'086 ms
Bytes Downloaded	2'474'326 kB	4'229'363 kB
Objekte Downloaded	20	61

Die AWD Webseiten sind in allen Bereichen schneller. Es kann abschliessend gesagt werden, dass AWD zu bevorzugen ist, wenn es auf die Geschwindigkeit ankommt. Dies war früher der Fall für mobile Endgeräte, fällt jedoch mit den neuen schnelleren Internetgeschwindigkeiten weniger ins Gewicht. Es sollte demnach, wenn möglich, RWD eingesetzt werden, ausser wenn die Migration einer bestehenden Webseite zu viel Aufwand bedeutet.

4.1.2 Bandbreite

Vor nicht allzu langer Zeit war die Bandbreite für Desktop PC's noch begrenzt und es musste sehr auf die Grösse von Bildern und Multimedia-Inhalten geachtet werden. Doch das Internet entwickelte sich rasch weiter. Heute ist es möglich mehrere Full-HD (1980x1200 Pixel) Videos gleichzeitig anzusehen. Auf mobilen Endgeräten sieht es jedoch anders aus. Das Wireless Application Protocol (WAP) wurde 1997 eingeführt. Auf grosse Grafiken musste man da noch verzichten, da das Laden zu lange dauerte. Darauf folgte General Packet Radio Service (GPRS), Enhanced Data Rates for GSM Evolution (EDGE), G3 und schlussendlich High-Speed Downlink Packet Access (HSDPA). Mit den letzten beiden sind Webseiten und Bilder kein Problem mehr, und mit HSDPA können auch Multimedia Inhalte komfortabel unterwegs angesehen werden. Die Abdeckung ist jedoch noch nicht

1 *Adaptive vs. Responsive Web Design: Which Is Right for Your Site? - Catchpoint's Blog.* URL: http://blog.catchpoint.com/2014/06/02/adaptive-vs-responsive-web-design-right-site/ (besucht am 01. 06. 2015).

flächendeckend. Vor allem in ländlichen Gebieten wird oft noch auf die älteren Verfahren zurückgegriffen. Deshalb muss für mobile Geräte die Bandbreite immer noch berücksichtigt werden.

Um die Bandbreite zu schonen, werden Caches in den Browsern eingesetzt. Grosse Bilder, die auf mehreren Seiten vorkommen, zum Beispiel Banner, werden damit nur einmal heruntergeladen und auf dem Endgerät gespeichert. Dadurch ist nur der erste Seitenaufruf langsam. Die darauf folgenden Ladezeiten lassen sich jedoch beträchtlich vermindern.

4.2 Browsers

Jeder Webentwickler kennt die Schmerzen, die die Entwicklung für verschiedene Browser mit sich bringt. Auch wenn das W3C vorgibt, wie ein HTML Dokument darzustellen ist, hat jeder Browser seine Eigenschaften auf welche Rücksicht genommen werden muss. Hier eine nicht abschliessende Liste von Browsern:

- Firefox
- Chrome
- Safari
- Internet Explorer
- Opera
- Konqueror
- Lynx

Es gibt verschiedene Versionen der Browser, welche zusätzlich berücksichtigt werden müssen und die Meisten haben noch mobile Versionen.

Es ist wichtig, dass in den frühen Phasen eines Projektes festgelegt wird, welche Browser und Versionen unterstützt werden sollen, da dies einen grossen Einfluss auf den Aufwand hat.

4.3 Nicht funktionale Eigenschaften

Nicht funktionale Eigenschaften sind solche, die nicht den Funktionsumfang der Software beschreiben. Von Kunden werden diese oft vergessen, oder zu wenig Beachtung geschenkt. Noch für den Erfolg einer Webseite sind sie fast wichtiger als die eigentliche Funktionalität. Dies wurde eindrücklich mit der Einführung des IPhones gezeigt, denn dort stand die Geschwindigkeit und die Einfachheit der Benutzung im Vordergrund. Also zwei nicht funktionale Eigenschaften.

Es gibt vier Eigenschaften für ein Projekt, die miteinander im Konflikt stehen:

- Zeit
- Kosten
- Funktionsumfang
- Nicht funktionale Eigenschaften

Man muss sich entscheiden, auf welche Punkte besonders Wert gelegt werden soll. Denn alle vier Eigenschaften können nicht erreicht werden.

In diesem Kapitel werden folgende nicht funktionale Eigenschaften beschrieben:

- Geschwindigkeit
- Sicherheit
- Bedienbarkeit

4.3.1 Geschwindigkeit

Die Geschwindigkeit ist eines der wichtigsten Aspekte einer Seite. Es gibt diverse Statistiken welche aufzeigen, dass Benutzer die Webseite verlassen, wenn man zu lange warten muss. Es gibt drei Richtwerte[1].

- 0.1 Sekunden: Benutzer haben das Gefühl die Seite reagiert augenblicklich.
- 1.0 Sekunden: Die Gedanken der Benutzer bleiben ununterbrochen.
- 10 Sekunden: So lange wartet ein Benutzer ohne einer anderen Tätigkeit nachzugehen.

Wenn auf eine Aktion länger als eine Sekunde gewartet werden muss, muss dem Benutzer angezeigt werden, dass eine Aktion im Hintergrund durchgeführt wird. Dies kann durch einen sogenannten *Spinner* visualisiert werden.

(a) Spinner 1

(b) Spinner 2

(c) Spinner 3

Abbildung 4.3: plots of....

Schnelle Webseiten führen zu weniger Frustration, tieferen Blutdruck, tieferen Flow State und hörem Umsatz[2].

Um die Geschwindigkeit zu verbessern, können Server- oder Client-seitig Anpassungen vorgenommen werden. Je komplexer die Applikation ist, desto länger sind normalerweise auch die Ladezeiten. Es wurde gezeigt, dass durch eine Verkleinerung der Anzahl Zeilen an Sourcecode tendenziell auch die Ladezeiten vermindert werden[3].

1 *Response Time Limits: Article by Jakob Nielsen.* URL: http://www.nngroup.com/articles/response-times-3-important-limits/ (besucht am 01. 06. 2015).

2 *The Psychology of Web Performance - how slow response times affect user psychology.* URL: http://www.websiteoptimization.com/speed/tweak/psychology-web-performance/ (besucht am 01. 06. 2015).

3 *Web Page Performance Thesis - web page response time measurement, modeling and monitoring.* URL: http://www.websiteoptimization.com/speed/tweak/web-page-performance-thesis/ (besucht am 01. 06. 2015).

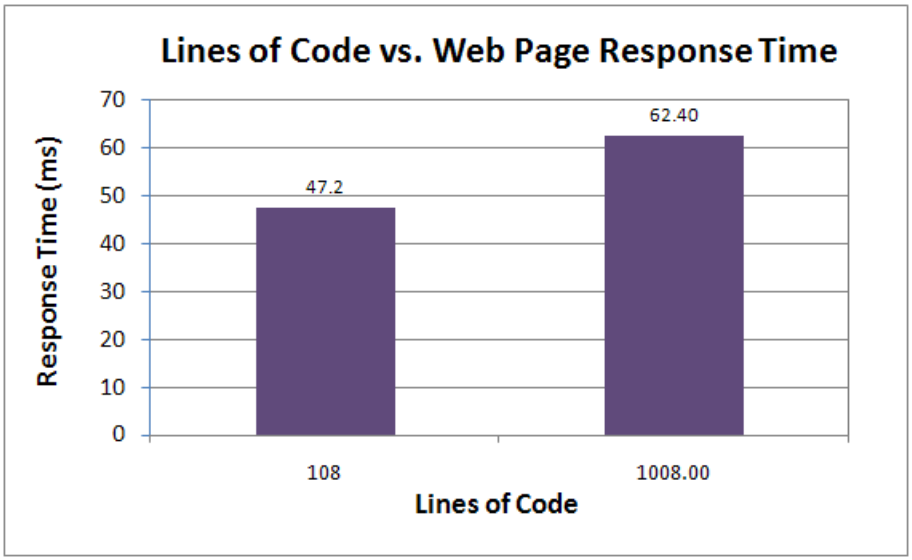

Abbildung 4.4: Einfluss der Anzahl an Codezeilen auf die Ladezeit

Einen Einfluss auf die Ladezeit haben auch die Objekte, welche von einer Seite geladen werden. Dies beinhaltet Bilder, CSS- und JS-Files. Durch so genannte *Sprite Sheets*. Das ist ein Bild, das mehrere Bilder enthält. Mittels CSS kann das passende angezeigt werden. Hier ein beispielhaftes Sprite Sheet

Abbildung 4.5: Sprite Sheet

Auf der Client-Seite gibt es die Möglichkeit Objekte nachzuladen. Dadurch wird dem Benutzer rasch eine Seite dargestellt. Eine einfache Möglichkeit ist die Umstrukturierung der HTML Datei. CSS- und JS-Dateien werden üblicherweise zu Beginn definiert. Dadurch werden sie auch als Erstes geladen, wodurch es länger dauert bis dem User eine Webseite dargestellt werden kann. Wenn man die Referenz auf diese Files ans Ende setzt, wird zuerst die Webseite geladen und dem Benutzer angezeigt. Dadurch wird die Reaktiontszeit kürzer.

Über Asynchronous JavaScript and XML (AJAX) kann mittels JS Code auf dem Client ausgeführt werden, welcher Bilder, Dateien oder weiteren Code nachlädt. Bei dieser Variante ist jedoch zusätzliche Programmierung nötig und ist deshalb auch aufwändiger.

4.3.2 Sicherheit

Beim Requirement Engineering sollte stets ein Entwickler, oder jemand der sich mit Sicherheit auskennt, anwesend sein. Denn dieses Thema wird sonst oft vergessen.

Die meisten Sicherheitslöcher bieten Formulare. Über ein schlecht programmiertes Formular kann die gesamte Datenbank gelöscht, oder ausgelesen werden. Alternativ kann eine Schwachstelle ausgenutzt werden um Spam-Emails zu versendet oder Schadcode einzuspeisen. Es wird hier nicht auf alle Sicherheitslöcher eingegangen. Exemplarisch wird hier die Ausnutzung von einer SQL-Injection gezeigt. Die folgende harmlose Adresse sollte nur etwas auslesen:

> http://webserver/cgi-bin/find.cgi?ID=**42**

Der generierte SQL-Code sieht folgendermassen aus:

> SELECT author, subject, text FROM artikel WHERE ID=**42**;

Das Problem liegt darin, dass die 42 aus der Adresse direkt in die SQL Abfrage übernommen wird. Man könnte die Adresse nun folgendermassen verändern:

> http://webserver/cgi-bin/find.cgi?ID=**42;UPDATE+USER+SET+TYPE='admin'+WHERE+ID=23**

Die neue SQL-Abfrage sieht nun so aus:

> SELECT author, subject, text FROM artikel WHERE ID=**42;UPDATE USER SET TYPE='admin' WHERE ID=23**

Es wird immer noch der Artikel mit der ID=42 ausgelesen, gleichzeitig gibt man dem eigenen Benutzer noch Administrator-Rechte.

Solche Fehler kann eine Firma viel Geld kosten und deren Ruf vernichten, weshalb die Sicherheit einen hohen Stellenwert in jeder Software bekommen sollte.

4.3.3 Bedienbarkeit

Wenn ein Benutzer eine Webseite bedienen bzw. navigieren kann ohne zu überlegen, so hat sie eine gute Bedienbarkeit. Doch dies umzusetzen ist nicht immer einfach. Die Navigation sowie der Aufbau der Seite müssen selbsterklärend sein. Dies muss schon während des Requirements Engineering geplant werden.

Bei grösseren Webseiten wird die Bedienbarkeit mittels *Eye Tracking* überprüft. Dazu müssen Probanden ohne Vorkenntnisse vordefinierte Arbeitsschritte durchführen. Dabei wird aufgezeichnet, wo sie dabei auf der Webseite hinschauen. Dort, wo am meisten hingesehen wird, werden danach die wichtigsten Informationen platziert.

Alternativ können A/B-Tests durchgeführt werden. Möchte man zum Beispiel überprüfen, ob man mehr Produkte verkauft, wenn die Navigation am oberen Bildschirmrand positioniert ist, oder auf der linken Seite, so werden zwei Varianten programmiert. Variante A wird 50% aller Webseitenbesucher angezeigt, Variante B den anderen 50%. Dann wird gemessen, auf welcher Variante das bessere Resultat erzielt wird.

4.4 Weiteres

4.4.1 Search Engine Optimisation

Für einen erfolgreichen Webauftritt ist es essenziell, dass die Seite auf den diversen Suchmaschinen gefunden wird. Je weiter oben in der Suchresultate-Liste man erscheint desto besser. Dies ist das Ziel des Search Engine Optimization (SEO).

Suchmaschinen durchstöbern das Internet und untersuchen die Webseiten auf diverse Merkmale. Welche Merkmale ausgewertet und wie sie priorisiert werden, ist ein gut gehütetes Geheimnis.

Wichtig ist sicherlich ein semantisch korrektes HTML Markup. Denn dadurch wird der Suchmaschine mitgeteilt, was eine Überschrift ist, was die wichtigsten Schlüsselwörter sind, welche anderen Seiten mit dieser in Beziehung stehen, etc. Dadurch kann der Inhalt von der Suchmaschine richtig ausgewertet werden.

Früher konnte man die Suchmaschinen überlisten, indem man die wichtigsten Schlüsselwörter oder Texte auf mehreren Seiten präsentierte. Diese wurden als wichtig markiert. Heutige Suchmaschinen haben sich weiterentwickelt und erkennen jetzt so genannten *duplicate content*. Ist duplicate content vorhanden, wird dieser sogar als minderwertig markiert und das Suchresultat sinkt in der Suchresultat-Liste weiter herunter.

Die Suchmaschinen überprüfen auch, ob ein Text wirklich angezeigt wird. Ist ein Text nicht sichtbar für einen Benutzer, zum Beispiel durch weisse Schrift auf weissem Hintergrund, so wird der Inhalt auch nicht gewertet.

Auch die Position von Inhalten ist entscheidend. Je weiter oben etwas steht desto wichtiger ist der Inhalt und desto weiter oben ist die Seite in der Suchresultat-Liste.

Es gibt Firmen, welche sich nur auf SEO spezialisiert haben. Dies zeigt die Wichtigkeit dieser Anforderung für den Erfolg einer Webseite. Der Aufwand sollte deshalb früh alloziert werden.

4.4.2 Accessibility

Der Grossteil der Webseiten sind für „normale“ Benutzer ohne Behinderung optimiert. Auch haben viele die Wichtigkeit von SEO erkannt und ihre Webseiten dazu verbessert. Die Accessibility wird jedoch von fast keinen Webseiten umfassend implementiert. Dabei geht es darum, dass auch Benutzer mit eingeschränkten Möglichkeiten die Informationen von einer Webseite beziehen können. Damit sind hauptsächlich blinde Benutzer, also jene ohne einen Display gemeint.

Damit auch „blinde“ Benutzer die Webseite voll umfänglich bedienen können, müssen einige Punkte beachtet werden. Ein korrektes HTML Markup ist Voraussetzung. Dadurch wird dem User mitgeteilt, was ein Titel ist und was der Inhalt der Seite.

Bei Grafiken muss immer eine Bildbeschreibung angegeben werden. Dadurch weiss man, was das Bild darstellt, auch wenn man einen Browser besitzt, der keine Bilder darstellen kann.

Für ältere Benutzer, oder solche mit einer Sehschwäche, ist es hilfreich, wenn man die Schriftgrösse anpassen kann. Es muss jedoch sichergestellt werden, dass dadurch das Layout der Seite nicht zerrissen wird.

Die meisten Webseiten haben keine speziellen Vorkehrungen getroffen für Benutzer mit

einer Einschränkung. Das Einzige was meistens vorhanden ist, ist ein korrektes HTML Markup. Dieses wurde aber vielmehr für das SEO gemacht als für die Accessibility.

KAPITEL 5

Software Design

Dieses Kapitel widmet sich der Frage, was im speziellen Fall der Softwareentwicklung für Webseiten beim Design beachtet werden muss.

5.1 Single Page Applications

Die meisten Webseiten sind Multi Page Applications. Das bedeutet, dass eine neue Seite beim Server angefragt wird, wann immer er einen Link anklickt. Die Seite hat immer eine eigene Adresse über welche sie referenziert werden kann.

Bei Single Page Applications (SPA) wird initial eine Seite geladen und dann nicht mehr. Sobald ein Link angeklickt wird, geht zwar eine Anfrage an den Server, die Seite wird im Browser jedoch nicht neu geladen. Die Anfrage an den Server wird über JavaScript, im genaueren über AJAX, abgesetzt. Davon merkt der Benutzer jedoch nichts. Im Code sieht dies folgendermassen aus:

```
1 var myAjax = new Ajax.Request(
2   "datum.php",
3   { method: 'get', onComplete: zeigeDatum }
4 );
```

In diesem Beispiel wird das prototype[1] Framework verwendet, um eine Anfrage auf *www.meineseite.com/datum.php* abzusetzen. Sobald die Antwort des Servers beim Browser eintrifft, wird die Methode *zeigeDatum* aufgerufen.

Der Vorteil von SPAs liegt darin, dass nur ein Teil der Webseite neu geladen werden muss. Sprich derjenige, welcher durch die Aktion verändert werden soll. Dadurch werden die Antwortzeiten der Benutzeraktionen minimiert.

Ein weiterer Vorteil liegt darin, dass der Server sich nicht um die Darstellung der Informationen kümmern muss, da dies vom Client erledigt wird. Der Server liefert nur Informationen aus, welche Anzuzeigen sind. Der Client entscheidet dann wie diese dargestellt werden sollen.

1 *Prototype JavaScript framework: a foundation for ambitious web applications.* URL: http://prototypejs.org/ (besucht am 07. 06. 2015).

SPAs haben jedoch auch einige Herausforderungen. Dadurch, dass die gesamte Seite nur eine Adresse besitzt, wird es schwieriger für Suchen, die Seite zu analysieren. Auch der Vor- und Zurück-Knopf der Browser funktioniert nicht mit SPAs. Es gibt zwei Lösungen für das Problem:

- location.hash
- pushState

In HTML gibt es Ankerlinks. Diese werden benötigt, um auf einer Seite an eine spezifische Stelle zu springen. Der Link *www.meineseite.com* öffnet eine Seite. *www.meineseite.com#kontakt* zeigt die gleiche Seite an, springt jedoch zur Textstelle *kontakt*. Es ist ein eigener Link und auch die History Funktion (Vor- & Zurück-Button) des Browsers erkennt eine eigene Adresse. Der Browser lädt sich dabei jedoch nicht neu. Sprich diese Funktionalität kann für SPAs verwendet werden. Dazu wird zum Beispiel folgender Link generiert: *www.meineseite.com#/products/fridge*. Die eigentliche Seite ist dabei *www.meineseite.com* und es wird die Textstelle */products/fridge* angesprungen. Diese gibt es nicht auf der Seite, kann jedoch über JS nachgeladen werden.

Die neuere Möglichkeit, um SEO für SPA zu ermöglichen, ist die *pushState* Funktionalität von JS.

```
1 window.history.pushState(data, "Kühlschrank", "/products/fridge");
```

Dieser Aufruf setzt die Adresse des Browsers auf *www.meineseite.com/products/fridge*, ohne die Seite dabei neu zu laden.

5.2 Rich Internet Applications

RIAs sind Webseiten, welche einer dynamischen Desktopanwendung ähnlich sind. Dabei werden Drag-and-Drop, 3D-Effekte, Animationen und Unterstützung diverser Videoformaten ermöglicht. Zum Teil müssen RIAs nicht zwingend in einem Browser ausgeführt werden, sondern benötigen lediglich Zugang zum Internet.

Abbildung 5.1: Beispiel für eine RIA[a]

a *AjaxWorld Talk: Building Rich Internet Applications Using Microsoft Silverlight 2 - Brad Abrams - Site Home - MSDN Blogs*. URL: http://blogs.msdn.com/b/brada/archive/2008/10/20/ajaxworld-talk-building-rich-internet-applications-using-microsoft-silverlight-2.aspx (besucht am 07. 06. 2015).

Eine RIA kann traditionell über HTML und JS funktionieren, oder über Plugins von drittanbietern laufen. Zum Beispiel Flash oder Java. Neue Trends zeigen, dass RIA auf Basis von Plugins sukzessive durch HTML und JS abgelöst werden[2].

Der Vorteil besteht darin, dass das Look-and-Feel von bekannten Anwendungen aus dem Desktop Bereich übernommen wird, wodurch die Benutzerakzeptanz verbessert wird. Wenn ein Plugin verwendet wird, so werden auch die Problematiken, welche die verschiedenen Browser mit sich bringen, umgangen.

Nachteilig dabei ist, dass das Laden der ersten Seite eventuell länger Dauert, da viele Elemente dargestellt werden müssen. Auch wird mehr Rechenleistung vom Client benötigt, was im speziellen einen Einfluss auf mobile Geräte haben kann. Auch gibt es dieselbe Problematik mit dem SEO, wie sie im Abschnitt 5.1 Single Page Applications beschrieben ist.

2 *Rich Internet Application (RIA) Market Share / Global Usage*. URL: http://www.statowl.com/custom_ria_market_penetration.php (besucht am 07. 06. 2015).

KAPITEL 6

Software Construction

Bei der Entwicklung von Software gibt es diverse Prozesse. Weit verbreitet sind das Wasserfall-, Scrum- und Kanban Modell. Obwohl diese nicht spezifisch für die Domäne der Webentwicklung sind, sollen sie kurz beschrieben werden.

Danach wird vorgestellt, wie HTML Dokumente korrekt erstellt werden und zum Schluss, welche Programmiersprachen in der Webentwicklung eingesetzt werden.

6.1 Vorgehensmodelle

Für kleine und mittlere Unternehmen (KMU) Unternehmen haben sich in der Webentwicklung drei Vorhersagemodelle etabliert. Das Wasserfall, Scrum und Kanban Modell.

6.1.1 Wasserfallmodell

Ersteres ist ein strikt lineares (nicht iteratives) Model in welchem eine Phase nach der anderen abgearbeitet wird. Das Ergebnis der vorherigen Phase fliesst dabei in die Nächste mit ein. Das Vorgehen ist sehr starr, da alles, was zu Beginn definiert wurde, während des Projektes nicht mehr angepasst werden kann.

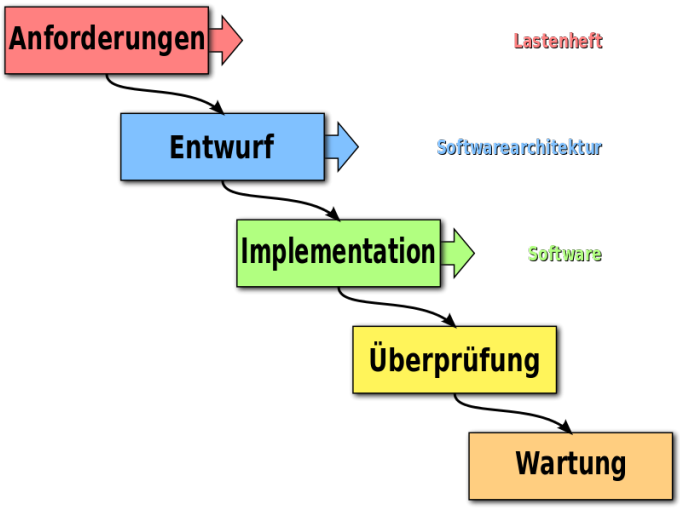

Abbildung 6.1: Wasserfallmodel[b]

b *Waterfall model-de - Wasserfallmodell – Wikipedia*. URL: https://de.wikipedia.org/wiki/Wasserfallmodell#/media/File:Waterfall_model-de.svg (besucht am 03. 06. 2015).

6.1.2 Scrum

Scrum hat sich zum Ziel gesetzt, den Kunden früher in den Entwicklungsprozess mit einzubeziehen. Durch eine iterative Vorgehensweise hat der Kunde regelmässig die Möglichkeit, Einfluss zu nehmen.

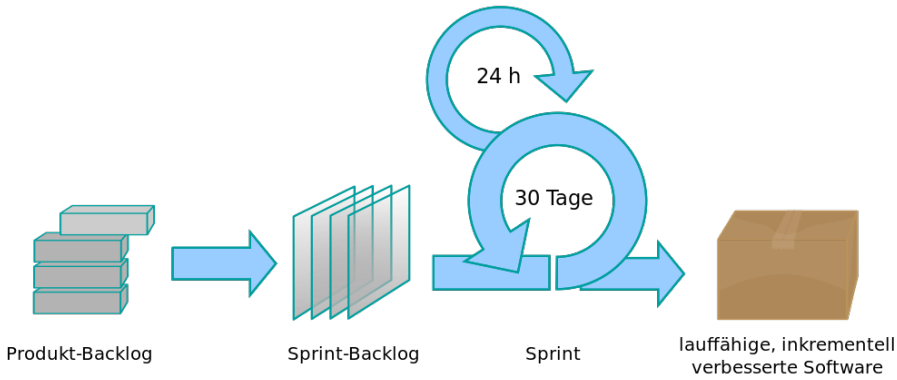

Abbildung 6.2: Scrum[a]

a *Scrum process-de - Scrum – Wikipedia*. URL: https://de.wikipedia.org/wiki/Scrum#/media/File:Scrum_process-de.svg (besucht am 03. 06. 2015).

Eine Iteration wird als Sprint bezeichnet. Er sollte nicht länger als 30 Tage dauern und an dessen Ende sollte stets eine lauffähige Version der Webseite vorgestellt werden können.

Vor jedem Sprint hat der Kunde die Möglichkeit den Sprint-Backlog zu füllen. Dort sind die Arbeitspakete definiert, die in der nächsten Iteration umgesetzt werden sollen. Dadurch sieht der Kunde wie die Software sich entwickelt und kann entsprechend Einfluss nehmen.

6.1.3 Kanban

Kanban hat das primäre Ziel, ein Arbeitspaket so rasch fertigzustellen wie möglich. Dazu werden alle Arbeiten auf eine Karte geschrieben und an eine Tafel gehängt.

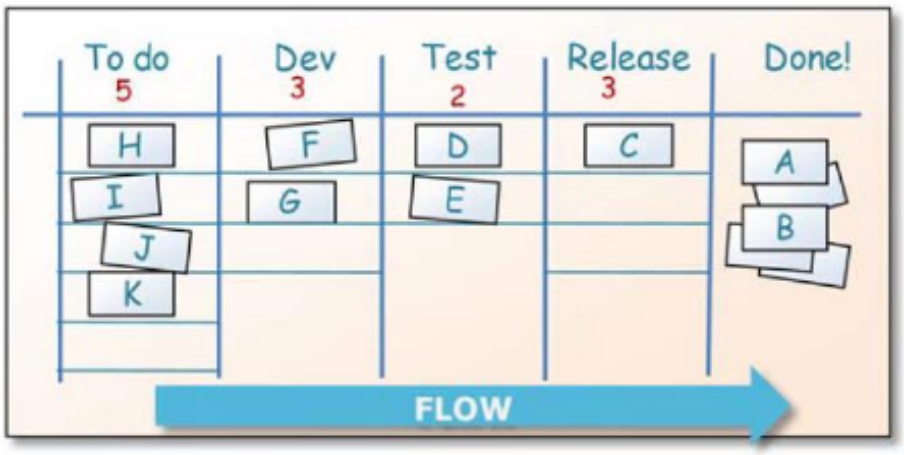

Abbildung 6.3: Kanban

Jede Spalte hat eine Zahl, welche definiert, wie viele Arbeitspakete auf einmal in einer Spalte liegen dürfen. Es wird gemessen, wie lange die Durchlaufzeit pro Paket ist und dann wird versucht, diese Zeit durch Prozessverbesserungen zu minimieren.

6.2 HTML

Jede Webseite läuft mit HTML. Dabei handelt es sich um eine Auszeichnungssprache für digitale Dokumente. Sie definiert die Struktur und den Aufbau einer Webseite.

Der Aufbau wird von dem W3C definiert:

```
1  <!DOCTYPE html>
2  <html>
3
4  <head>
5  <meta charset="UTF-8">
6  <title>Title of the document</title>
7  </head>
8
9  <body>
10 Content of the document......
11 </body>
12
13 </html>
```

Aufgebaut ist das Dokument über Tags, welche zwischen den < und > Zeichen definiert sein. Der *DOCTYPE* gibt die HTML Version an. Im *head* werden Kopfdaten definiert.

Zum Beispiel der Titel der Seite oder die Schlüsselwörter, welche von den Suchmaschinen gefunden werden sollen. Und Schlussendlich wird im *body* der Inhalt der Seite angegeben.

Im *body* können weitere Tags definiert werden. Zum Beispiel können Tabellen erzeugt werden. Der folgende Code

Zelle 1	Zelle 2	Zelle 3
Zelle 4	Zelle 5	Zelle 6
Zelle 7	Zelle 8	Zelle 9

wird folgendermassen definiert:

```
<table>
  <tr>
    <td>Zelle 1</td>
    <td>Zelle 2</td>
    <td>Zelle 3</td>
  </tr>
  <tr>
    <td>Zelle 4</td>
    <td>Zelle 5</td>
    <td>Zelle 6</td>
  </tr>
  <tr>
    <td>Zelle 7</td>
    <td>Zelle 8</td>
    <td>Zelle 9</td>
  </tr>
</table>
```

Eingeleitet wird die Tabelle mittels *<table>* begonnen und mit *</table>* wieder beendet. Eine Reihe wird mit *<tr>* definiert und eine Zelle mit *<td>*

Über die Tabellenstruktur kann der gesamte Aufbau einer Seite gesteuert werden. Für eine professionelle Entwicklung und eine gute Suchmaschinen-Unterstützung ist dies jedoch nicht empfehlenswert, denn nicht jede Seite ist automatisch eine Tabelle. Möchte man einen Textabsatz definieren, sollte man zum Beispiel den <p> Tag (paragraph) verwenden. Dabei spricht man von semantischen HTML. Suchmaschinen wissen dann, dass es sich dabei um einen Textabsatz handelt und nicht um eine Tabelle.

Der Schwierigkeitsgrad von HTML ist sehr niedrig. Jedoch ist die Umsetzung einer semantisch korrekten Seite um Faktoren schwerer zu bewerkstelligen.

6.3 Programmiersprachen

Da das Web auf einer Client-/Server-Architektur aufbaut, gibt es Programmiersprachen für beide Seiten. Eine Webseite besteht Grundsätzlich aus HTML Code, welche von weiteren Client-seitigen Programmiersprachen manipuliert werden kann. Die Server-seitigen Sprachen haben hingegen das primäre Ziel, HTML oder Client-Seitigen Code zu generieren.

Es gibt unzählige Programmiersprachen. Folgend eine Liste der populärsten Server-Programmiersprachen gemäss dem TIOBE Index[1].

- Java
- C
- C++
- C#
- Phyton

Die einzige verbreitete Programmiersprache für die Benutzung im Browser is JavaScript. Es gibt Alternativen wie ActionScript, mit welchem Flash-Dateien umgesetzt werden, oder Java um Java Applets zu programmieren. Die beiden Varianten verlieren jedoch immer mehr an Bedeutung, wie dem TIOBE Index entnommen werden kann.

Es gibt Sprachen, welche in JavaScript transcodiert werden. Zum Beispiel *Coffee Script* oder *Dart*. Beim Transcodieren wandelt der Compiler die Sprache dann in valides JavaScript um. Die Frage, welche Programmiersprache man einsetzen soll, ist sehr schwierig zu beantworten, da die Auswahl sehr gross ist. Folgende Punkte sollten dabei beachtet werden:

- Grösse und Komplexität der Software
- Problemdomäne
- Knowhow

Je nach Grösse und Komplexität spielt auch die Geschwindigkeit eine Rolle. Es gibt Sprachen, die in diesem Bereich Vorteile besitzen, wie C++, die sehr schnell sind.

Es ist möglich, alles in jeder Sprache umzusetzen. Nur eignen sich spezifische Sprachen für gewisse Arbeiten besser als andere. Zum Beispiel können mathematische Probleme mit funktionale Programmiersprachen wie Haskell oder F# einfacher und verständlicher gelöst werden. Deshalb sollte stets die passende Sprache ausgewählt werden.

Auch wenn die Problemdomäne wichtig ist, so sollte stets auch das Wissen der Programmierer berücksichtigt werden. Denn wer sich in einer Programmiersprache zuhause fühlt, der leistet auch bessere Arbeit.

1 *TIOBE Software: Tiobe Index*. URL: http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html (besucht am 04. 06. 2015).

KAPITEL 7

Software Testing

Das Testing bei der Webentwicklung ist wie üblich aufgeteilt in die Teilbereiche Unit-, Integration- und Systemtest. Dieses Kapitel nimmt sich den drei Bereichen an und stellt Werkzeuge für das Web vor.

7.1 Unit Tests

Das Unit Testing ist die unterste Ebene beim Software Testing. Deshalb werden die kleinsten Einheiten, die Methoden, überprüft. Dabei sollte sicher gestellt sein, dass genau nur diese Grösse getestet wird und keine Abhängigkeiten.

Frameworks für das Unit Testing gibt es wie Sand am Meer. Bei Java gibt es zum Beispiel JUnit, JUnitEE, TestNG, etc. Für C# gibt es NUnit, NUnitAsp, MSTest, etc. Für PHP gibt es PHPUnit, lime, SnapTest, etc. Nicht nur die Namen sind ähnlich, sondern auch deren Aufbau. Alle Funktionieren nach dem *Arrange-Act-Assert* Prinzip. Ein Beipsiel aus JUnit:

```
1  @Test
2  public void testIfFoodIsWarm() {
3    // Arrange
4    Kitchen kitchen = new Kitchen();
5    Kitchen.PrepareFood();
6
7    // Act
8    Kitchen.CookFood();
9    Food food = Kitchen.getFood();
10
11   // Assert
12   assertThat(food.Temperature, is(80));
13 }
```

Im *Arrange* wird alles vorbereitet. Beim *Act* wird die Aktion durchgeführt und schlussendlich beim *Assert* überprüft, ob alles richtig funktioniert hat.

Die Schwierigkeit besteht darin, keine Abhängigkeiten mit zu testen. Im obigen Beispiel wird eine Küche erstellt. Sie hat eine Mikrowelle in welcher das Essen aufgewärmt wird. Die Mikrowelle wird jedoch auch getestet.

Bei der Problematik kommen Mock-Frameworks zur Verwendung. Sie kapseln die Funktionalität von Abhängigkeiten weg. Auch hier gibt es für jede Sprache viele Frameworks, welche in frage kommen. Folgend ein Beispiel mit Mockito, einem Mock Framework für Java:

```
1  @Test
2  public void testIfFoodIsWarm() {
3    // Arrange
4    Microwave microwave = mock(Microwave.class);
5    when(microwave.cookFood()).thenReturn(new Food(80));
6
7    Kitchen kitchen = new Kitchen(microwave);
8    Kitchen.PrepareFood();
9
10   // Act
11   Kitchen.CookFood();
12   Food food = Kitchen.getFood();
13
14   // Assert
15   assertThat(food.Temperature, is(80));
16   verify(microwave).cookFood();
17 }
```

Zu Beginn wird ein Mock von der Mikrowelle erstellt und definiert, dass wenn die *cookFood()* Methode aufgerufen wird ein warmes Essen zurückkommt. Am Schluss wird überprüft, ob die *cookFood()* Funktion auch wirklich aufgerufen wurde.

Somit wird Funktionalität der Mikrowelle nicht geprüft, sondern nur, ob sie verwendet wurde. Für die Mikrowelle sollte es dann eigene Unit Tests geben.

7.2 Integrationstests

Eine Software besteht aus kleinen Einheiten, welche zusammengesetzt eine Komponente ergeben. Diese Komponenten werden wiederum zu grösseren Komponenten zusammengepuzzelt. Dies geht so weiter bis eine fertige Software entsteht. Um zu überprüfen, ob die Komponenten korrekt funktionieren, gibt es Integrationstests.

Als Beispiel wird ein Kühlschrank angenommen, welcher die innere Temperatur misst. Er hat eine Schnittstelle mit der über das Internet der Kältezustand abgefragt werden kann. Ein Test könnte folgendermassen aussehen:

```
1  @Test
2  public void testIfTemperatureCanBeQueried {
3    // Arrange
4    HttpClient client = new DefaultHttpClient();
5      HttpGet request = new HttpGet("https://api.home-sweet-home.com/fridge/
getTemperature");
6
7      // Act
8      HttpResponse response = client.execute(request);
9
10     // Assert
```

```
11      Fridge fridge = HttpResponse.Get(response, Fridge.class);
12      assertThat( fridge.Temperature, lowerThan(5));
13  }
```

Hier wird die Schnittstelle des Kühlschrankes überprüft, welcher jedoch aus einer Aggregation von kleineren Bausteinen besteht. Zum Beispiel dem Temperaturmesser, der Stromzufuhr, etc. All diese Einzelteile müssen funktionieren damit der Test erfolgreich ist.

Damit die Tests so früh als möglich programmiert werden können, müssen auch hier Mocks verwendet werden. Denn vielleicht funktioniert der Baustein für die Stromzufuhr noch nicht. Der Temperaturmesser ist jedoch bereits implementiert. Also muss zuerst noch der Mock für alle unfertigen Komponenten erstellt werden.

Für das Mocking bei den Integrationstests gibt es mehrere Vorgehensweisen:

- Top-Down
- Bottom-Up
- Sandwich Testing
- Big Bang

Bei allen Methoden wird zuerst der Integrationstest implementiert. Beim *Top-Down* kommt als nächstes die grösste Komponente, also der Kühlschrank. Alle anderen Teile werden als Mocks bereitgestellt. Die *Bottm-Up* Vorgehensweise beginnt mit dem kleinsten Element. Zum Beispiel dem Temperaturmesser. Dabei muss ein Mock für den Kühlschrank erstellt werden. *Sandwich Testing* ist ein Mix zwischen den ersten beiden. Es wird mit einer beliebigen Komponenten begonnen und alle anderen, also die darüber sowie die darunter müssen als Mock bereitgestellt werden. Und schlussendlich gibt es noch die *Big Bang* Vorgehensweise. Dabei wird zum Schluss die gesamte Komponente getestet und es sind keine Mocks nötig.

Für Neuentwicklungen eigenen sich Top-Down, Bottom-Up und das Sandwich Testing. Big Bang sollte nur eingesetzt werden, um bestehende Software zu testen. Denn es empfiehlt sich die Tests zu Beginn zu schreiben. Denn Tests nachträglich zu entwerfen ist stets schwieriger als wenn man es vor der Entwicklung tut[1].

7.3 Systemtests

Der letzte Schritt beim Testen ist der Systemtest. Bei den vorherigen Tests stand das technische im Zentrum. Der Systemtest sollte stets aus der Sicht des Benutzers entworfen werden.

In der Webentwicklung werden Systemtests über Hilfsprogramme realisiert, welche einen Browser starten und wie ein Benutzer über die Seite navigieren. Die Korrektheit wird wiederum über Asserts validiert.

Das bekannteste Programm ist Selenium[2, 3]. Als Beispiel wird eine Suche getestet:

1 *Madeyski10c.pdf*. URL: http://madeyski.e-informatyka.pl/download/Madeyski10c.pdf (besucht am 07. 06. 2015).
2 *Selenium - Web Browser Automation*. URL: http://docs.seleniumhq.org/ (besucht am 07. 06. 2015).
3 Roy Osherove, Hrsg. *The Art of Unit Testing*. mitp-Verlag, 2010.

```
@Test
public class GoogleSearch {
    static WebDriver driver;
    static Wait<WebDriver> wait;

    public static void main(String[] args) {
    }

    private static boolean testIfMoreThan3ElemntsFoundWithFirefoxBrowser() {
        // Arrange
          WebDriver driver = new FirefoxDriver();

        // Act
          // Navigate to site.
          driver.get("http://www.kitchen.com/");

          // Type search query.
          driver.findElement(By.name("searchword")).sendKeys("fridge");

          // Click search.
          driver.findElement(By.name("submit")).click();

          // Wait for search to complete.
          Wait.for(seconds(5));

        // Assert
          assertThat( driver.findElements(By.tagName("list-item")), greaterThan
(3));
    }
}
```

Als Erstes wird ein Firefox Browser geöffnet. Danach die Adresse *www.kitchen.com* eingegeben, das Suchwort *fridge* eingetippt und auf den Absendebutton gedrückt. Nachdem fünf Sekunden gewartet wurde, wird überprüft, ob mehr als drei Suchresultate gefunden wurden.

Dieser Test überprüft die Software, sowie ob die Interaktion mit dem Browser korrekt funktioniert. Er ist aus der Sicht des Benutzers geschrieben, denn genau so würde auch er sich bei der Suche verhalten.

KAPITEL 8

Software Maintenance

Eine Webseite kann von einer simplen HTML Datei bis zu einem komplexen Konstrukt mit mehreren Webservern und Datenbanken reichen. Deshalb ist die Pflege sehr abhängig von der Grösse der Seite.

8.1 Service Layer Agreements

Zum Beispiel bei Internetshops, Reiseportalen, etc. ist es von existenzieller Bedeutung für die Firma, dass die Seite läuft. Durch einen Ausfall können schnell Millionen von Franken Umsatz verloren gehen. Bei diesen Fällen werden Service Layer Agreements (SLAs) eingesetzt. Diese Regeln, wie lange die Seite im Jahr online bleiben muss. Der Wert liegt oft bei 99.99% oder noch höher. Es werden die Anzahl an 9'en nach dem Komma gezählt. Wenn man in einem SLA von vier 9'en spricht, muss die Seite demnach 99.9999% online sein. Dies entspricht 31 Sekunden im Jahr.

Über SLAs werden zusätzlich die Reaktionszeiten, Fehlerbehebungen oder Weiterentwicklungen definiert. Es kann abgemacht werden, dass die Seite innerhalb von zwei Sekunden eine Antwort liefert, dass Fehler binnen zwei Tagen behoben, oder eine Weiterentwicklung maximal eine Woche dauern darf.

8.2 Service Desk

Zur Maintenance gehört auch der Support für die Benutzer. Bei Fragen muss es eine Anlaufstelle geben an welche sich die Betroffenen melden können.

Es gibt drei Arten von Service Desks:

- Interne
- Externe
- Virtuelle

Das interne Service Desk wird von der Firma selber betrieben. Für das Externe wird eine eigene Firma mit dem Auftrag engagiert. Für das virtuelle Service Desk gibt es auch den Ausdruck *Follow The Sun*. Dazu gibt es eine Anlaufstelle für alle Fragen. Je nach Tageszeit erhält jedoch immer ein anderes Service Desk die Anfrage. Je nachdem, welches gerade aktiv ist. Weltweit gibt es somit drei bis vier Instanzen. Diese Möglichkeit eignet

sich für internationale Firmen und spart Kosten, da keine Nachtzuschläge auf die Löhne entrichtet werden müssen.

KAPITEL 9

Schlusswort

Das Entwickeln einer Webseite ist grundlegend sehr einfach. Das Einzige, was man können muss, sind die Grundlagen von HTML. Alles andere ist freiwillig und muss nicht beachtet werden.

Möchte man jedoch professionelle Webentwicklung tätigen, so muss einiges beachtet werden. Dies beinhaltet semantisch korrektes HTML, ein gutes SEO Konzept, gute Geschwindigkeit, Sicherheit und Bedienbarkeit, und vieles mehr.

Diese Arbeit ist eine Ansammlung von Werten und Techniketn, welche bei der Entwicklung in der Domäne des Internets beachtet werden muss. Für sich selber sieht alles sehr einfach aus. Doch die Schwierigkeit liegt darin, alles zu beachten und zu beherrschen sowie in einem akzeptablen Kostenrahmen zu bleiben.

9.1 Reflexion

Durch meine Berufserfahrung in der Webentwicklung hatte ich ein grosses Interesse an dieser Arbeit. Das Schreiben ging mir sehr leicht von der Hand. Ich kannte zwar schon alle vorgestellten Bereiche, doch bei der Recherche konnte ich immer noch neue Aspekte kennenlernen, welche ich hoffentlich schon bald in meinem Arbeitsleben recyclen kann.

Quellenverzeichnis

[1] *Adaptive vs. Responsive Web Design: Which Is Right for Your Site? - Catchpoint's Blog*. URL: http://blog.catchpoint.com/2014/06/02/adaptive-vs-responsive-web-design-right-site/ (besucht am 01. 06. 2015) (siehe S. 8).

[2] *AjaxWorld Talk: Building Rich Internet Applications Using Microsoft Silverlight 2 - Brad Abrams - Site Home - MSDN Blogs*. URL: http://blogs.msdn.com/b/brada/archive/2008/10/20/ajaxworld-talk-building-rich-internet-applications-using-microsoft-silverlight-2.aspx (besucht am 07. 06. 2015) (siehe S. 17).

[3] *index • IEEE Computer Society*. URL: http://www.computer.org/web/swebok (besucht am 30. 05. 2015) (siehe S. 3).

[4] *Internet – Wikipedia*. URL: https://de.wikipedia.org/wiki/Internet#Nutzerzahlen (besucht am 30. 05. 2015) (siehe S. 1).

[5] *Madeyski10c.pdf*. URL: http://madeyski.e-informatyka.pl/download/Madeyski10c.pdf (besucht am 07. 06. 2015) (siehe S. 25).

[6] Roy Osherove, Hrsg. *The Art of Unit Testing*. mitp-Verlag, 2010 (siehe S. 25).

[7] *Prototype JavaScript framework: a foundation for ambitious web applications*. URL: http://prototypejs.org/ (besucht am 07. 06. 2015) (siehe S. 15).

[8] *Response Time Limits: Article by Jakob Nielsen*. URL: http://www.nngroup.com/articles/response-times-3-important-limits/ (besucht am 01. 06. 2015) (siehe S. 10).

[9] *Responsive vs Adaptive Design for UI*. URL: http://blog.zymr.com/responsive-vs-adaptive-design-for-ui (besucht am 31. 05. 2015) (siehe S. 7).

[10] *Responsive vs. Adaptive Design: What's the Best Choice for Designers? - UXPin*. URL: http://blog.uxpin.com/6439/responsive-vs-adaptive-design-whats-best-choice-designers/ (besucht am 31. 05. 2015) (siehe S. 6).

[11] *Rich Internet Application (RIA) Market Share / Global Usage*. URL: http://www.statowl.com/custom_ria_market_penetration.php (besucht am 07. 06. 2015) (siehe S. 17).

[12] *Scrum process-de - Scrum – Wikipedia*. URL: https://de.wikipedia.org/wiki/Scrum#/media/File:Scrum_process-de.svg (besucht am 03. 06. 2015) (siehe S. 19).

[13] *Selenium - Web Browser Automation*. URL: http://docs.seleniumhq.org/ (besucht am 07. 06. 2015) (siehe S. 25).

[14] *The Difference Between Adaptive Design And Responsive Design | Search Engine People*. URL: http://www.searchenginepeople.com/blog/the-difference-between-adaptive-design-and-responsive-design.html (besucht am 31. 05. 2015) (siehe S. 7).

[15] *The Psychology of Web Performance - how slow response times affect user psychology*. URL: http://www.websiteoptimization.com/speed/tweak/psychology-web-performance/ (besucht am 01. 06. 2015) (siehe S. 10).

[16] *TIOBE Software: Tiobe Index*. URL: http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html (besucht am 04. 06. 2015) (siehe S. 22).

[17] *Waterfall model-de - Wasserfallmodell – Wikipedia*. URL: https://de.wikipedia.org/wiki/Wasserfallmodell#/media/File:Waterfall_model-de.svg (besucht am 03. 06. 2015) (siehe S. 19).

[18] *Web Page Performance Thesis - web page response time measurement, modeling and monitoring*. URL: http://www.websiteoptimization.com/speed/tweak/web-page-performance-thesis/ (besucht am 01. 06. 2015) (siehe S. 10).

[19] *World Internet Users Statistics and 2014 World Population Stats*. URL: http://www.internetworldstats.com/stats.htm (besucht am 30. 05. 2015) (siehe S. 1).

[20] *World Wide Web Consortium (W3C)*. URL: http://www.w3.org/ (besucht am 30. 05. 2015) (siehe S. 4).