Andreas Henrich
Stefanie Gooren-Sieber (Hrsg.)

Web Engineering

Aspekte der systematischen Entwicklung von Web-Anwendungen

Beiträge des Seminars

im Wintersemester 2011/2012

Lehrstuhl für Medieninformatik

Otto-Friedrich-Universität Bamberg

Inhaltsverzeichnis

Modellierungsmethoden für Web-Anwendungen: Anforderungen, Vergleich und Best Practice

Tobias Kiehl

Otto-Friedrich-Universität Bamberg, Lehrstuhl für Medieninformatik
tobias.kiehl@stud.uni-bamberg.de

Zusammenfassung Für die Modellierung von Web-Anwendungen stehen eine Vielzahl von Methoden zur Verfügung, welche hinsichtlich der für die Anwender relevanten Aspekte eine sehr große Heterogenität aufweisen. So unterscheiden sich die Methoden bspw. hinsichtlich methodischer Wurzeln, Modellierungsschritten, Werkzeugverfügbarkeit, Entwicklungsreife und Verständlichkeit. Ein praxisbezogener und systematischer Methodenvergleich zur Bestimmung eines Best-Practice-Ansatzes ist in der Literatur nicht verfügbar. Daher ist es das Ziel dieser Arbeit, gängige Modellierungsmethoden anhand eines Anforderungskatalogs zu vergleichen und einen Best-Practice-Ansatz herauszuarbeiten. Ausgangspunkt dieser Arbeit stellt der Beitrag *Modellierung von Web-Anwendungen* von Schwinger und Koch [1] dar.

Schlagworte: Anforderungen an Modellierungsmethoden für Web-Anwendungen, Best-Practice-Methode, Methodenbewertung, Methodenvergleich, OOHDM, UWE, WebML

1 Verbreitung und Anwendung von Modellierungsmethoden für Web-Anwendungen

Das World Wide Web hat in den letzten Jahren eine rasante Entwicklung hinter sich und ist stetig und überall verfügbar. Ebenso entwickelten sich Web-Anwendungen von statischen Seiten zu komplexen Web-Applikationen mit dynamischen Inhalten und auf heterogene Nutzergruppen angepassten Funktionalitäten. Bei der Realisierung dieser Anwendungen kommen eine Vielzahl von Web-Technologien (vgl. [2, S. 9-52]) zum Einsatz. Bei der Konzeption von Web-Anwendungen wird jedoch, trotz der hohen Komplexität, häufig auf Modellierungsmethoden und strukturierte Vorgehensweisen verzichtet [3, S. 77].

Dies mag an der im Vergleich zum Software Engineering kurzen Entwicklungsgeschichte des Web Engineering liegen [4, S. 1, 5]. Auch wird die Erstellung von Web-Anwendungen oft nicht als vollwertige Anwendungsentwicklung betrachtet [4, S. 5]. Zudem fördert die hohe Nutzerunterstützung zahlreicher Entwicklungswerkzeuge (HTML-Editoren, Wizards, Content Management Systeme) Quick-and-Dirty-Programmierung [5, S. 243] oder auch Implement-Test-Release-Vorgehensweisen ohne methodisches Fundament [3, S. 77]. Darüber hinaus sind bestehende Modellierungsmethoden oft nicht für die Modellierung von Web-Anwendungen geeignet [1, S. 49-51].

Aus dem mangelnden Einsatz von Modellierungsmethoden resultierenden typischerweise eine ganze Reihe von Problemen (vgl. [1, S. 49][5, S. 247-248][3, S. 77]), wie unzureichende Erfüllung von Anforderungen, hoher Wartungsaufwand, niedrige Produktivität und geringe Gebrauchstauglichkeit.

In Anbetracht dieser Probleme wurde vor etwa 10 Jahren erkannt, dass neue Modellierungsmethoden benötigt werden, um effiziente Lösungen zu erhalten [6, S. 7-10][7, S. 246]. Schwinger und Koch betrachten hierzu eine ganze Reihe von Methoden mit unterschiedlichen Ursprüngen [1, S. 70]. In Bezug auf die damals aktuellen Modellierungsmethoden für Web-Anwendungen (Stand 2004) kommen sie zu dem Schluss, dass hinsichtlich der definierten Anforderungen an Modellierungsmethoden „zurzeit keine weit verbreitete Modellierungsmethode existiert“ [1, S. 72]. Eine Methodenempfehlung wird nicht gegeben.

Ziel der Arbeit ist es, die aktuell (Stand 2011) gängigen Modellierungsmethoden einer kritischen Betrachtung zu unterziehen und einen praxisbezogenen, umfassenden Vergleich zu ermöglichen. Hierzu wird ein Kriterienkatalog anhand der in der Literatur genannten und ableitbaren Anforderungen an Modellierungsmethoden erstellt. Anhand der definierten Kriterien werden Stärken und Schwächen der Methoden aufgezeigt und versucht einen Best-Practice-Ansatz bzw. die Best-Practice-Methode zu bestimmen.

2 Anforderungen an Modellierungsmethoden

Im Vergleich zu der Anzahl an Veröffentlichungen, welche sich mit der Anwendung von Modellierungsmethoden für Web-Anwendungen beschäftigen, setzen sich nur wenige mit den Anforderungen an diese Methoden auseinander. Die in der Literatur genannten Anforderungen werden daher im Folgenden durch eigene Aspekte ergänzt und in einen gemeinsamen Kriterienkatalog eingearbeitet.

Aufgrund der unterschiedlichen Ursprünge basieren die Modellierungsmethoden auf verschiedenen Grundkonzepten. Durch die Ermittlung dieser methodischen Wurzeln kann teilweise eine Aussage über das Anwendungsfeld und die Vor- und Nachteile einer Methode getroffen werden. Zudem wird untersucht, ob die Modellierungsmethode in ein Vorgehens- oder Phasenmodell eingebettet ist und dadurch einen übergeordneten systematischen Rahmen für die Entwicklung von Web-Anwendungen bietet. Weiterhin wird auf die Verfügbarkeit eines Metamodells geachtet, da dies sowohl für das Methodenverständnis, als auch für die Modellerweiterung und den Model Driven Architecture-Ansatz (MDA) von Relevanz ist. Durch MDA wird vor allem die Produktivität und Wiederverwendbarkeit von (Teil-)Modellen beeinflusst [8, S. 160-163]. Im Kriterienkatalog werden die genannten Aspekte unter „Methodenkern“ zusammengefasst.

Bei der Modellierung von Web-Anwendungen sind zwei Schritte zu unterscheiden: Im ersten Schritt werden die Anforderungen an die Web-Anwendung erfasst und durch Modelle dargestellt werden. Im zweiten Schritt erfolgt die konzeptuelle Modellierung der Web-Anwendung. Schwinger und Koch [1] beschreiben die dazu benötigten Modelle anhand der Dimensionen Ebene, Aspekt und Kontextualität. Ähnliche Anforderungsbeschreibungen finden sich bei Kunz

[9, S. 11], Castelyn [2, S. 115-119] und Koch et al. [8], wobei Kontextualität häufig auch als Adaption bezeichnet wird. In der Dimension Ebene wird zwischen Inhalts-, Navigations- und Präsentationsmodell unterschieden. Im Vergleich dazu sind bei der Modellierung von konventionellen Anwendungssystemen nur Anwendungslogik und Benutzerschnittstellen zu modellieren. Die Dimension Aspekte gliedert die Modelle der Dimension Ebene in Struktur- und Verhaltensmodelle. Diese bilden bei der vorliegenden Untersuchung kein separates Kriterium, sondern werden an geeigneten Stellen bei der Untersuchung der jeweiligen Modellebene mit betrachtet, sofern entsprechende Modelle verfügbar sind. Kontextualität wird von Schwinger und Koch [1] als neue Web-Anwendungsspezifische Dimension eingeführt, welche sich aufgrund von heterogenen Benutzergruppen, unterschiedlichen Nutzungsorten und verschiedenen Endgeräten ergibt. Kontextualität bzw. Adaption beeinflusst die Modelle in den Dimensionen Ebene und Aspekt. In Anbetracht der verschiedenen Modelle ist eine weitgehend orthogonale, überschneidungsfreie Erstellung der einzelnen Modelle wünschenswert, da sich dies positiv auf das Modellverständnis und den Modellierungsaufwand auswirkt. Im Kriterienkatalog werden diese Aspekte unter „Modelle“ zusammengefasst.

Die Werkzeugunterstützung findet bei Schwinger und Koch [1] Berücksichtigung und wird im Folgenden übernommen. Da Werkzeuge die Erstellung und Pflege von Modellen sowie die Generierung von Code und Prototyping unterstützen können, ist die Relevanz von Werkzeugen in den letzten Jahren stark angewachsen [8, S. 162]. Für Anwender spielen die Produktverfügbarkeit für eine bestimmte Plattform, die Werkzeugreife, die erwartete Weiterentwicklung sowie die Anschaffungs- oder Nutzungskosten für ein Werkzeug eine Rolle. Im Kriterienkatalog werden die genannten Aspekte unter „Werkzeuge“ aufgeführt. Die Kriterien Werkzeugreife und erwartete Weiterentwicklung sind aus Übersichtsgründen im Punkt Entwicklungsbewertung zusammengefasst.

Da die Modellierung nicht zum Selbstzweck durchgeführt wird, sondern übergeordnete Ziele verfolgt, sind die Methoden auch hinsichtlich dieser Ziele zu bewerten. Koch et al. [8] nennen in diesem Zusammenhang ein einfaches Modellverständnis [8, S. 161]. Zudem ist zu berücksichtigen für welchen Anwenderkreis eine Methode geeignet ist und welches Fachwissen bei der Verwendung einer bestimmten Modellierungsmethode benötigt wird. Ebenso sind für die Anwendung die Methodenreife und die Entwicklungsperspektive einer Modellierungsmethode relevant, da sich eine Methode an die Veränderungen im Web in gewissem Rahmen anpassen sollte. Gerade bei der Verwendung eines MDA-Ansatzes ist die mittel- bis langfristige Weiterentwicklung eines Methode von Bedeutung. Diese Aspekte werden im Kriterienkatalog unter „Anwendung“ zusammengefasst.

Die genannten Anforderungen bilden den folgenden Kriterienkatalog:

Kriterien	Betrachtung von ...
Methodenkern	
- Ursprung	Grundkonzept, ggf. abgeleitbares Anwendungsfeld
- Vorgehensmodell	Verfügbarkeit und Art eines Vorgehenskonzeptes
- Metamodell	Verfügbarkeit und Verständnis

Modelle	
- Anforderungsmodell	Verfügbarkeit, eigene Modellkonzepte, Orthogonalität
- Inhaltsmodell	Verfügbarkeit, eigene Modellkonzepte, Orthogonalität
- Navigationsmodell	Verfügbarkeit, eigene Modellkonzepte, Orthogonalität
- Präsentationsmodell	Verfügbarkeit, eigene Modellkonzepte, Orthogonalität
- Adpationsmodell	Verfügbarkeit, eigene Modellkonzepte, Orthogonalität
Werkzeuge	
- Produkte	Verfügbarkeit, Bezeichnung und Bezugsquellen
- Plattformen	Verfügbarkeit für diverse Entwicklungsumgebungen
- Entwicklungsbewertung	Werkzeugreife und erwartete Weiterentwicklung
- Kosten	Kosten der Werkzeugbeschaffung
Anwendung	
- Anwenderkreis	Anwendung in Forschung, Industrie oder durch Gruppen mit bestimmtem Fachwissen
- Methodenverständnis	Gesamteindruck hinsichtlich des Verständnis
- Methodenreife	Einschätzung des aktuellen Entwicklungsstatus
- Weiterentwicklung	erwartete Entwicklung der Methode

3 Methodenüberblick und Bestimmung gängiger Modellierungsmethoden für Web-Anwendungen

Ausgangspunkt der Betrachtung sind die von Schwinger und Koch [1] vorgestellten Modellierungsmethoden. Abbildung 1 stellt diese und weitere, neue Modellierungsmethoden sowie deren zeitliche Einordnung dar. Gerichtete Kanten zeigen konzeptionelle Verwandtschaften zwischen den Methoden auf.

Grundlegend ist erkennbar, dass die Wurzeln der einzelnen Methoden entweder auf das Entity-Relationship-Modell (ERM), die objektorientierte Object-Modeling Technique (OMT) oder auf die Unified Modeling Language (UML) zurückzuführen sind. Während sich eine Vielzahl von Methoden in der Zeit von 1993 bis 2000 entwickelten, gab es in den folgenden Jahren vergleichsweise wenige Methodenneuentwicklungen. Ein Blick in die nach 2005 veröffentliche Literatur zeigt, dass in der Forschung weitgehend auf die bereits vorhandenen Methoden zurückgegriffen wird und diese verfeinert oder erweitert werden. Das Fachwissen zu den Methoden liegt meist in Form von englischsprachigen Konferenzbeiträgen vor.

Aus Gründen der Effizienz beschränkt sich die nachfolgende Untersuchung auf Methoden mit hoher fachlicher Reife, erkennbarer Anerkennung in der Wissenschaft und methodischer Aktualität. Als Maß hierfür lassen sich die Anzahl angenommener Konferenzbeiträge, die Häufigkeit von Referenzen von anderen Autoren auf diese Konferenzbeiträge sowie die Aktualität der Veröffentlichungen zu einer Methode heranziehen. Anhand der Literatur wird deutlich das sich vor allem der Ansatz des UML-based Web Engineerings (UWE), die Web Modeling Language (WebML) und die Object-Oriented Hypermedia Design Methodology (OOHDM) sowie deren Weiterentwicklung Sematic Hypermedia Design

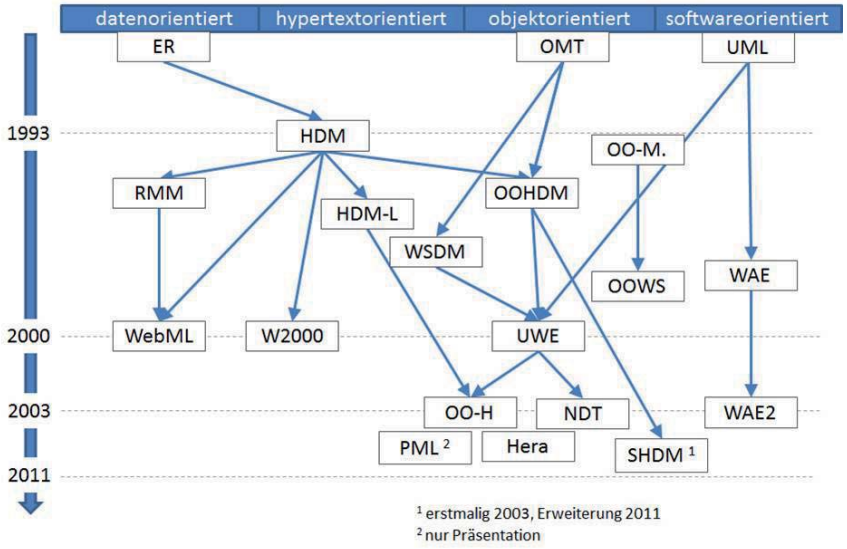


Abbildung 1. Übersicht über Modellierungsmethoden für Web-Anwendungen (eigene Darstellung nach [1, S. 70] und [10, S. 25] mit Ergänzungen)

Method (SHDM) in gewisser Weise durchgesetzt haben (vgl. [11, S. 121][12, S. 241][9, S. 14][2, S. 68-79][13, S. 29-39]). Im Folgenden werden diese Methoden hinsichtlich der in Kapitel 2 definierten Kriterien untersucht. Die Darstellung soll auch einen Ausgangspunkt zur Klärung von Detailfragen im Rahmen der Auswahl einer Modellierungsmethode für eigene Modellierungsprobleme geben.

4 Bewertung von UWE

Die offizielle Webseite von UWE findet sich unter `http://uwe.pst.ifi.lmu.de`. Erste Veröffentlichungen erschienen im Jahr 2000 von Nora Koch[1] von der Ludwig-Maximilians-Universität München.

4.1 Methodenkern

Bei UWE handelt es sich um einen objektorientierten Modellierungsansatz auf der Basis von UML [1, S. 54]. Da alle Modelle mittels UML-Profil als eine UML-Meta-Model-Erweiterung definiert sind, ist UWE UML-konform [14, S. 161]. Eine ausführliche und aktuelle Profil- und Metamodellbeschreibung ist verfügbar [15]. Ein übergeordnetes Vorgehensmodell existiert nicht, vielmehr wird das Vorgehen durch die einzelnen Modellierungsschritte definiert.

[1] `http://www.pst.ifi.lmu.de/Personen/team/koch`

4.2 Modelle

Die verfügbaren Modelle sind Anforderungs-, Inhalts-, Navigations-, Präsentations- und Adaptionsmodelle. Für jedes Modell steht mindestens eine Diagrammart zur Verfügung. Bei der Modellierung wird weitgehend auf UML-Diagramme zurückgegriffen. Es werden auf der Basis von UML und Stereotypisierung auch UWE-spezifische Modelle definiert und verwendet.

Das Modellierungsvorgehen [8, S. 163-177] lässt sich wie folgt beschreiben: Zu Beginn werden Anforderungen durch auf Web-Anwendungen angepasste Anwendungsfall- und Aktivitätsdiagramme modelliert. Zwischen prozess-, kontext- und navigationsbezogenen Anwendungsfällen wird durch verschiedene Markierungen unterschieden. Im Anschluss erfolgt die Modellierung des Inhalts mittels UML-Klassendiagrammen ohne UWE-spezifische Konzepte. Zur Berücksichtigung von Personalisierung (Adaption) kann auch ein Nutzermodell erstellt werden. Im Anschluss erfolgt die initiale Modellierung des Navigationsmodell durch Navigationsklassen und Assoziationen zwischen diesen. Die Klassen des Inhaltsmodells dienen dabei als Ausgangsbasis. Prozessbezogene Anwendungsfälle werden schrittweise in das Navigationsmodell hinein modelliert. Geschäftsprozesse werden zusätzlich durch Aktivitätsdiagramme dargestellt. Das anschließend erstellte Präsentationsmodell zeigt die Elementstruktur einer einzelnen Webseite auf. Im letzten Schritt werden für das Navigations- und das Präsentationsmodell Adaptionsmodelle modelliert, welche statische oder dynamische Änderungen in separaten Modellen darstellen. Jedes Adaptionsmodell definiert durch einen formalen Parameter einen Schnittpunkt im Navigations- oder Präsentationsmodell sowie eine alternative Aktionsfolge, durch welche die Modelladaption ermöglicht wird. Adaptionsmodelle müssen im Anschluss jedoch in die Navigations- und Präsentationsmodelle integriert bzw. hineingewebt (sog. „Weaving“) werden.

Letztlich gestaltet sich der UWE Modellierungsprozess weitgehend klar verständlich und stringent in der Umsetzung. Die klare Trennung zwischen den Modellierungsebenen trifft weitgehend zu. Zur Modellierung von kontextuellen Veränderungen kommen jedoch im Inhaltsmodell andere Ansätze zum Einsatz als im Navigations- und Präsentationsmodell. Weiterhin ist die Modellierung von Geschäftsprozessen nur indirekt durch Navigationspfade möglich.

4.3 Werkzeuge

Durch die Kompatibilität mit der Meta Object Facility (MOF) kann die Modellierung mit UWE grundlegend mit allen CASE-Werkzeugen, welche UML unterstützen, erfolgen. Die in UWE benötigten Stereotypen können in einem UML-konformen Werkzeug entweder manuell erstellt oder vielfach in Form von UML-Profilen importiert werden. Nach dem Import können UWE-Elemente in gleicher Weise wie Standard UML-Elemente verwendet werden [8, S. 162]. Es werden auf der Webseite von UWE Erweiterungen bzw. Profilimporte für die Werkzeuge ArgoUML, MagicDraw und UMLet angeboten. Weiterhin existiert das Eclipse-Plugin UWE4JSF, welches auf der Basis von UWE-Modellen Programmcode für JavaServerFaces (JSF) generiert. ArgoUML ist kostenfrei unter

`http://argouml-downloads.tigris.org` für Windows und Mac OSX verfügbar. MagicDraw ist zudem auch für Linux erhältlich, jedoch ist nur die Basisversion kostenfrei. UWLet ist nur für Windows verfügbar und kostenfrei. ArgoUWE und MagicDraw ermöglichen neben der Modellierung auch eine Konsistenzüberprüfung des Modells und unterstützen die Model Driven Architectur mit semiautomatischer Codegenerierung [8, Seit 163]. Beide Werkzeuge sind ausgereift und werden weiterentwickelt.

4.4 Anwendung

Die Modellierung kann von jeder Person durchgeführt werden, die Grundkenntnisse in UML besitzt. Im Vergleich zu den im Folgenden beschriebenen Methoden ist das Verständnis aufgrund der Verwendung von UML und der hohen Anzahl an gut sortierten und verfügbaren Publikationen auf der UWE-Webseite vereinfacht. Weiterhin wird der UWE-Ansatz von mehreren Autoren aufgegriffen und dargestellt [2][16][17][13][18].

Als neueste Entwicklungen sind Transaktionsorientierung, Personalisierung, Model Checking [8, S. 158] und die Unterstützung von Rich Internet Applications (RIA) [17][19] zu nennen. Ebenso wurden in den letzten Jahren intensive Forschungen zum Thema Model Driven Engineering (MDE) und MDA durchgeführt [20][7][21]. Das Ziel der Entwickler ist es, den gesamten Entwicklungszyklus einer Webanwendung durch UML-Diagramme zu modellieren [8, S. 161].

Zusammenfassend lässt sich festhalten, dass es sich bei UWE um eine konzeptionell und werkzeugtechnisch sehr ausgereifte Methode handelt, welche immer noch weiter entwickelt wird und eine hohe Resonanz in der Forschung aufweisen kann. Die Verbreitung in der Industrie ist anhand der Veröffentlichungen jedoch nicht bestimmbar.

5 Bewertung von WebML

Der Webauftritt von WebML ist unter `http://www.webml.org` zu finden. Die Entwicklung der Methode wird im Wesentlichen von Stefano Ceri[2] und Marco Brambilla von der Polytechnischen Universität Mailand vorangetrieben. Erste Veröffentlichungen zu WebML sind aus dem Jahr 2000 [22].

5.1 Methodenkern

WebML hat seinen Ursprung in den datenorientierten Modellierungsmethoden bzw. dem Entity-Relationship Modell [1, S. 70] und ist daher für datenintensiven Web-Anwendungen [23, S. 221] wie bspw. transaktionalorientierte Applikationen ausgelegt. Der Modellierungsablauf bei WebML lässt sich in Anlehnung an das Spiralmodell von Böhm in Phasen (Abb. 2) unterteilen, welche iterativ durchlaufen werden. Entsprechend kann die Entwicklung einer Web-Anwendung inkrementell bzw. in Teilen erfolgen. Der Entwicklungszyklus umfasst eine Analyse-,

[2] `http://dbgroup.como.polimi.it/ceri`

Modellierungs-, Implementierungs-, Evaluations- und Deploymentphase [23, S. 224]. Grafische Metamodelle zu den einzelnen WebML-spezifischen Modellen scheinen nicht verfügbar zu sein. Die Definition der WebML-spezifischen Konzepte erfolgt durch Document Type Definitions (DTD) [24, S. 5].

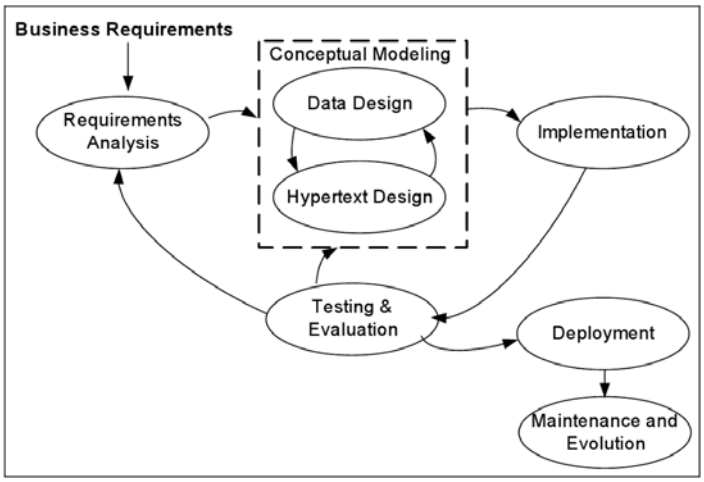


Abbildung 2. WebML-Vorgehensmodell [23, S. 224]

5.2 Modelle

Die Modellierung von Anforderungen erfolgt durch Tabellen sowie UML-Aktivitäts- und -Anwendungsfalldiagramme [23, S. 225]. Die anschließende konzeptuelle Modellierung besteht im Wesentlichen aus den zwei Subphasen (Abb. 2) Datenmodellierung und Hypertextmodellierung [23, S. 224]. Für die Datenmodellierung wird auf Entity-Relationship-Modelle oder UML-Klassendiagramme zurückgegriffen. Dabei werden keine methodenspezifischen Konzepte verwendet.

Für die Modellierung der Navigation (Hypertext) [23, S. 227-235] stehen Struktur- und Verhaltensmodelle zur Verfügung. Das Navigationsstrukturmodell, welches an ein vereinfachtes Präsentationsstrukturmodell von UWE erinnert, wird durch die WebML-spezifischen Konzepte beschrieben. Diese gliedern die Seite hierarchisch in Bereiche und Inhaltselemente auf. Bei der Modellierung des Navigationsverhaltens wird bei WebML auf maßgeschneiderte Konzepte zurückgegriffen, welche sich auch zu neuen Grundkonzepten kombinieren lassen [23, S. 221]. Diesem Ansatz steht der Nachteil gegenüber, dass man sich bei der Modellierung in WebML der semantischen Bedeutung der ca. 50 WebML-spezifischen Konzepten [24, S. 10] bewusst sein muss.

Die Modellierung von Adaption [23, S. 227-235] erfolgt ähnlich wie bei UWE: Das Datenmodell wird um kontextspezifische Klassen wie bspw. Eingabegerät, Aktivität und Ort ergänzt, welche sich visuell lediglich durch einen Rahmen oder eine Schattierung von nicht-kontextuellen Klassen im Klassendiagramm

abheben. Für kontextspezifische Darstellungen im Navigationsmodell kommen sogenannte Kontextwolken zum Einsatz, welche in ein bestehendes Navigationsmodell hinein modelliert werden und kontextabhängige, alternative Aktionsfolgen durch WebML-spezifische Konzepte aufzeigen [23, S. 249-252]. Ebenso wie bei UWE wirkt dieses Konzept nicht ausgereift. Zudem lässt sich die Adaptionsmodellierung nicht separat vom Inhalts- und Navigationsmodell durchführen.

5.3 Werkzeuge

Für die Modellierung von Inhalts- und Navigationsmodellen in WebML ist das Werkzeug WebRation[3] für die Plattformen Windows, Linux und Mac OSX verfügbar. Nur die Basisversion ist kostenfrei. Die Enterprise-Version kostet bis zu 7500 Euro. WebRatio hat einen Codegenerator und erlaubt eigene Konzepte durch XML Descriptions einzuführen [23, S. 236-238]. Für die Modellierung von Geschäftsprozessen mit der Business Process Model and Notation (BPMN) steht ein Plug-in für Eclipse zur Verfügung [23, S. 243]. WebRation stellt ein ausgereiftes Werkzeug dar, welches kommerziell weiterentwickelt wird.

5.4 Anwendung

In der Literatur werden keine bestimmten Anwendergruppen genannt. Sowohl Werkzeuge als auch Methodenkonzepte können von jedem Modellierer nach einem hohen initialen Einarbeitungsaufwand genutzt werden. Ausgehend von dem Vorgehensmodell ist die Methode vor allem für Projekte mit dem Fokus auf einer inkrementellen Entwicklung oder einer kurzen Time-to-Market (einer Basisanwendung) [23, S. 224] interessant. Dies wird durch den Model Driven Development-Ansatz unterstützt [23, S. 235f.], der auch in WebRation verankert ist.

Hinsichtlich der Entwicklungsperspektive und der Reife der Methode ist festzustellen, dass seit 2003 über 80 Publikationen zu Spezialisierungen von WebML erschienen sind. Die Erweiterungen umfassen bspw. Rich Internat Applications (RIA) [25], Semantic Web Applications [26][27] und die Einbindung von Web Services und Workflows [23, S. 236-238]. Weiterhin zeigt der Versuch WebML-Konzepte in UML darzustellen [28][29], dass die Verwendung von UML zahlreiche Vorteile bietet. Somit kann man annehmen, dass WebML-Konzepte langfristig in UML übertragen werden oder eine WebML-Weiterentwicklung ebenfalls UML nutzen wird.

Die Reife und die Anzahl der Publikationen trägt neben dem übergeordneten Vorgehensmodell erheblich zum Verständnis und bei den Anwendern bei. Die durchgeführten Industrieprojekte[4] zeigen, dass diese Methode praxistauglich ist und schnell qualitativ hochwertige Ergebnisse erzielt werden können [23, S. 252-258].

[3] `http://www.webratio.com`

[4] `http://www.webml.org/webml/page24.do`

6 Bewertung von OOHDM

Die offizielle Webseite von OOHDM ist `http://www.tecweb.inf.puc-rio.br/oohdm`. Die Methodenentwicklung geht auf Daniel Schwabe[5] und Gustavo Rossi im Jahr 1993 zurück.

6.1 Methodenkern

Anders als UWE und WebML hat OOHDM seine Wurzeln in den Hypertextorientierten Methoden [1, S. 70f.]. Ähnlich wie bei UWE bestimmt der Modellierungablauf das Vorgehen. Hierbei wird zwischen Anforderungsanalyse, konzeptueller Modellierung, Navigations- und Präsentationsmodellierung differenziert [30, S. 82]. Meta-Modelle konnten für OOHDM nicht gefunden werden.

6.2 Modelle

Im ersten Schritt erfolgt die Anforderungsmodellierung anhand von Anwendungsfällen, Szenarios und Interaktionsdiagrammen, wobei erstere und letztere nicht UML-konform sind [31, S. 307-313]. Die konzeptuelle Modellierung wird anhand der in den Anwendungsfällen identifizieren Klassen, Attribute und Beziehungen in UML-Klassendiagrammen definiert. Weitere Diagrammarten werden nicht genannt [31, S. 313-315]. Die Navigationsmodellierung wird mit OOHDM-spezifischen Konzepten durchgeführt und besteht aus einem Navigationskontext (Verhaltensmodell) und einem Navigationsschema (Strukturmodell). Das Navigationskontextmodell zeigt partielle Navigationswege zur Bearbeitung von verschiedenen Aufgaben. Die Partialmodelle müssen zu einem Gesamtmodell integriert werden. Aus den dafür benötigten Navigationselementen wird im Anschluss ein Navigationsstrukturmodell abgeleitet [31, S. 315-320].

Bei der Modellierung der Struktur der abstrakten Präsentation (Abstract data views) wird die sog. Abstract Widget Ontology verwendet, welche grundlegende Oberflächenelemente für Auswahlen, Mausklicks usw. vordefiniert und hierarchisch darstellt. Von einer modellierten Widget Ontology wird im Anschluss ein Mapping mittels Web Ontology Language (OWL) auf konkrete Präsentations- und Navigationsklassen durchgeführt. Bspw. erhält ein abstrakter Link durch das Mapping seine konkrete URL [31, S. 321-327].

Insgesamt sind die Konzepte und Bezüge der Navigations- und Präsentationsmodelle nicht immer klar nachvollziehbar. Durch die Präsentationsmodellierung kann kaum ein Eindruck der Webseitenstruktur erlangt werden, da lediglich ein Baum an Oberflächenelementen dargestellt wird. Weiterhin verlangt das Oberflächen-Mapping vom Modellierer spezifisches Wissen über die Funktionen der Web-Anwendung. Eine plattformunabhängige Modellierung ist kaum möglich. Zudem existiert kein separates Adaptionsmodell.

[5] `http://www.inf.puc-rio.br/~schwabe`

6.3 Werkzeuge

Für die Modellierung mit OOHDM steht die browserbasierte Werkzeuge HyperDE[6] zur Verfügung. HyperDE ist kostenlos. Das Werkzeug wurde jedoch seit 2009 nicht mehr weiterentwickelt, und es gibt auch keine Anzeichen dafür, dass sich dies ändern wird.

6.4 Anwendung

Wie oben bereits beschrieben, ist OOHDM teilweise nicht klar verständlich. Das Vorgehen zur Erstellung von Modellen folgt eher einem Schritt-für-Schritt-Plan statt einem übergeordneten und damit leichter verständlichem Prinzip. OOHDM wird in der Literatur meist als Entwickler-zentriert angesehen, da in der Phase der Oberflächenmodellierung Wissen über die funktionale Logik der Anwendung vorhanden sein muss [31, S. 320][13, S. 38].

Die Reife der Entwicklung und die Weiterentwicklung lässt sich nur schwer abschätzen. Im Vergleich zu WebML und UWE sind weitaus weniger Forschungsbeiträge veröffentlicht worden. Die einzig erkennbare Neuentwicklung stellt die umfassende Verbesserung der Semantic Hypermedia Design Method (SHDM) durch neue Konzepte und ein neues Werkzeug dar [32][11]. SHDM besitzt momentan jedoch unvollständige Modellierungsmöglichkeiten für Präsentations- und Adaptionsmodelle [11, S. 135] und kann daher nicht umfassend bewertet werden.

Die Verbreitung in der Praxis kann bei OOHDM nicht bestimmt werden. Als einziger Vorteil gegenüber UWE und WebML wird in der Literatur die Trennung zwischen Navigations- und Geschäftsprozessmodellen genannt [11, S. 130], welche jedoch anhand der verfügbaren Modellierungsbeispiele nicht eindeutig nachvollziehbar war. Insgesamt scheint der OOHDM-Ansatz hinsichtlich der methodischen Reife unvollständig zu sein.

7 Ergebnisüberblick

Die Ergebnisse der obigen Untersuchung sind in der folgenden Tabelle (Abkürzungen: orth. = orthogonal, teilw. = teilweise, notw. = notwendig) dargestellt. Die wesentlichen Aspekte werden im Anschluss noch einmal zusammengefasst.

Kriterien	UWE	WebML	OOHDM
Methodenkern			
- Ursprung	objektorientiert, UML	datenorieniert, ERM	hypertextorientiert
- Vorgehensmodell	entspricht Modellierungsablauf	inkrementelles Modell (nach Böhm)	entspricht Modellierungsablauf
- Metamodell	UML-Profil, grafisch, verständlich	DTD, nicht grafisch, schwer nachvollziehbar	nicht bekannt

[6] `http://www.tecweb.inf.puc-rio.br/hyperde`

Modelle			
- Anforderungs-modell	UML-Diagramme mit eigenen Konzepten, orth.	UML-Diagramme, keine eigenen Konzepte, orth.	eigene Diagramme, keine eigenen Konzepte, orth.
- Inhaltsmodell	UML-Klassen-diagramme, orth.	ERM, UML-Klassen-diagramme, orth.	UML-Klassen-diagramme, orth.
- Navigations-modell	eigenes, orth. Modell, stufen-weise Verfeinerung	eigenes, orth. Modell	eigenes, orth. Modell
- Präsentations-modell	eigenes, orth. Modell	eigenes, als Basis für Navigationsmodell	eigenes, orth. Modell
- Adpations-modell	eigenes, aspekt-orientiertes Modell, nur teilw. orth. da Integration notw.	eigenes, nicht orth., da Ergänzung von Inhaltsmodell und Navigationsmodell	keines
Werkzeuge			
- Produkte	diverse UML-Tools	nur WebRatio	nur HyperDE
- Plattform	Mac, Win., Linux	Mac, Win., Linux	browserbasiert
- Entwicklungs-bewertung	hohe Reife und Weiterentwicklung	hohe Reife und Weiterentwicklung	veraltet, keine Weiterentwicklung
- Kosten	kostenlos bis 1500 Euro (MagicDraw)	kostenlos bis 7500 Euro	kostenlos
Anwendung			
- Anwenderkreis	primär Forschung	primär Industrie	primär Forschung
- Methoden-verständnis	einfach, wie UML, jedoch stets Abstim-mung zwischen Modellen notwendig	einfaches Phasenmo-dell, nur zwei konzep-tuelle Modelle, jedoch viele eigene Konzepte	schwer, fehlendes übergeordnetes Konzept bzgl. Modellbeziehungen
- Methodenreife	sehr hoch	sehr hoch	unausgereift
- Weiter-entwicklung	ja, fachliche Aspek-te, durchgehende UML-Verwendung	ja, bzgl. neuer Web-Herausforderungen, UML-Konformität	nein, nur bei SHDM

Der Vorteil von UWE besteht in der Modellierung mit UML und dem damit einhergehenden weitreichenden Verständnis sowie einer breiten Werkzeugunterstützung. Es wird eine minimale Anzahl eigener Konzepte verwendet, welche in Veröffentlichungen auf der Website und in der Literatur gut beschrieben werden und verständlich sind. Eine weitere Stärke ist die weitgehend orthogonale Modellerstellung. Ein Vorgehensmodell, welches über die einzelnen Modellierungsschritte hinausgeht, ist dagegen nicht verfügbar. Insgesamt zeigt die Methode eine hohe Reife und wird weiterentwickelt. Die Verbreitung in der Praxis und die Produktivität sind jedoch unklar.

Dagegen liegen die Stärken von WebML in der Praxistauglichkeit, bei der prototypischen Implementierung und in dem leicht verständlichen iterativen Vorgehensmodell. Durch die WebML-spezifischen Konzepte ist der initiale Einarbei-

tungsaufwand höher als bei UWE, jedoch sind die Konzepte dafür auf Web-Anwendungen maßgeschneidert. Bei der konzeptuellen Modellierung sind im Wesentlichen nur zwei Modelle (Inhalt und Navigation) zu bewältigen. Jedoch werden Verhaltens- und Strukturmodellierung in den Modellen gemischt. Trotz dieser Schwächen, oder gerade aufgrund dieser Vereinfachung bzgl. der Modellvielfalt, scheint die Verbreitung in der Industrie [23, S. 252-258] sehr hoch zu sein, wie zahlreiche Referenzen auf der WebML- und WebRatio-Website belegen.

OOHDM zeigt in Bezug auf die beiden bisher beschriebenen Methoden kaum Vorteile. Fast durchgängig werden eigene Konzepte verwendet und nicht alle Modelle berücksichtigt. Für die Präsentationsmodellierung ist plattformspezifisches Wissen eines Entwicklers notwendig. Die Werkzeugunterstützung ist nicht mehr aktuell. Umfassende Forschungen zu RIA, Web Services oder UML-Konformität sind nicht verfügbar. Lediglich SHDM stellt eine aktuelle Weiterentwicklung dar, welche jedoch momentan noch unvollständig ist. Insgesamt erscheint OOHDM weniger ausgereift und schwerer verständlich als UWE und WebML.

8 Zusammenfassung und Ausblick

Anhand der Untersuchung konnte ermittelt werden, dass UWE, WebML und OOHDM zu den Methoden zählen, welche sich in gewissem Maße etabliert haben. Neue bahnbrechende und umfassende Methoden sind in den letzten Jahren nicht hinzugekommen. Die vorhandenen Methodenneuentwicklungen beschränkten sich meist auf ausgewählte Teilaspekte bei der Modellierung[7]. Weiterhin zeigt sich, dass die meisten in Schwinger und Koch [1] genannten Methoden heute kaum Betrachtung in Wissenschaft und Praxis finden.

Aus der Methodenbewertung geht insgesamt hervor, dass UWE ein wissenschaftlich ausgereifter Ansatz mit gut verständlichen, orthogonalen Modellen ist, wobei jedoch ein übergeordnetes Vorgehensmodell fehlt. WebML bietet dieses Vorgehensmodell bei hoher Praxistauglichkeit und ermöglicht inkrementelle Entwicklung und Prototypengenerierung. WebML stellt daher den Best-Practice-Ansatz dar. Problematisch sind dagegen die vielen eigenen Konzepte und eine Verschmelzung von Modellen (Nicht-Orthogonalität). OODHM bietet dagegen weder die Praxistauglichkeit von WebML, noch die fachliche Reife von UWE. Die Modellierung von Adaption gelingt keinem der Ansätze vollständig, wobei UWE durch die Aspektorientierung noch das beste Konzept vorweisen kann.

In Bezug auf die langfristige Entwicklung von Modellierungsmethoden für Web-Anwendungen lässt sich festhalten, dass die UML-Kompatibilität von verschiedenen Modellierungsmethoden untersucht und der Nutzen nachgewiesen wurde [29][8, S. 162]. Die von Schwinger und Koch prognostizierte Unified Web Modeling Language [1, S. 74] ist somit im Blickpunkt der Forschung, wenn gleich noch keine entsprechende Modellierungsmethode existiert. Aus den Forschungsbeiträgen lässt sich jedoch ableiten, dass die Verwendung von WebML-Konzepten keine dauerhafte Lösung darstellt. Eine Verschmelzung von UWE und WebML wäre vermutlich ein Optimum hinsichtlich Konzeption und Praxisorientierung.

[7] bspw. Presentation Modeling Language (PML) [33] für Präsentationsmodellierung

Literatur

1. Schwinger, W., Koch, N.: Modellierung von Web-Anwendungen. In Kappel, G., ed.: Web Engineering: Systematische Entwicklung von Web-Anwendungen. Dpunkt-Verl., Heidelberg (2004) 49–75
2. Casteleyn, S.: Engineering Web Applications. Springer, Dordrecht and New York (2009)
3. Chen, J.Q., Heath, R.D.: Web Application Development Methodologies. In Suh, W., ed.: Web Engineering: Principles and Techniques. Idea Group Pub., Hershey (PA) (2005) 76–97
4. Murugesan, S., Ginige, A.: Web Engineering: Introduction and Perspectives. In Brandon, D., ed.: Software Engineering for Modern Web Applications: Methodologies and Technologies. Information Science Reference, Hershey (PA) (2008) 1–24
5. Wissen, M., Ziegler, J.: Ontologiebasierte Vorgehensweise zur Modellierung komponentenbasierte Web-Anwendungen. In Szwillus, G., ed.: Interaktion in Bewegung. Teubner, Stuttgart and Leipzig and Wiesbaden (2003) 247–256
6. Wallace, D., Raggett, I., Aufgang, J.: Extreme Programming for Web Projects. Addison-Wesley, Boston and MA (2002)
7. Moreno, N., Melia, S., Koch, N., Vallecillo, A.: Addressing New Concerns in Model-Driven Web Engineering Approaches. In Hutchison, D., Naor, M., Nierstrasz, O., Bailey, J., Kanade, T., eds.: Web Information Systems Engineering - WISE 2008: 9th International Conference. Springer, Berlin and Heidelberg (2008) 426–442
8. Koch, N., Knapp, A., Zhang, G., Baumeister, H.: UML-Based Web Engineering. In Rossi, G., Olsina, L., Pastor, O., Schwabe, D., eds.: Web Engineering: Modelling and Implementing Web Applications. Springer, London (2008) 157–191
9. Kunz, A.: Web-3D-Welten systematisch erzeugen. Diplomica, Hamburg (2010)
10. Koch, N.: Methoden des Software-Engineering: Web-Software, Vorgehensmodelle und Projektmanagement. http://www.pst.ifi.lmu.de/lehre/WS0607/mse/material/25-MSEF1-WebEngineering-v2.pdf, 31.03.2012. (2007)
11. Souza Bomfim, M.H.d., Schwabe, D.: Design and Implementation of Linked Data Applications Using SHDM and Synth. In Auer, S., Díaz, O., Papadopoulos, G.A., eds.: Web Engineering: 11th International Conference, ICWE 2011. Springer, Berlin (2011) 121–136
12. Busch, M., Knapp, A., Koch, N.: Modeling Secure Navigation in Web Information Systems. In Grabis, J., Kirikova, M., eds.: Perspectives in Business Informatics Research: 10th International Conference, Bir 2011, Riga, Latvia, October 6-8, 2011, Proceedings. Springer, New York (2011) 239–253
13. Freudenstein, P.: Web Engineering for Workflow-Based Applications: Models, Systems and Methodologies. Univ.-Verl, Karlsruhe (2009)
14. Koch, N., Kraus, A.: Towards a Common Metamodel for the Development of Web Applications. In Cueva Lovelle, J.M., Del Puerto Paule Ruiz, M.d., González Rodríguez, B.M., Joyanes Aguilar, L., Labra Gayo, J.E., eds.: Web Engineering: International Conference, ICWE 2003, Oviedo, Spain, July 14-18, 2003. Proceedings. Springer, Berlin and Heidelberg (2003) 497–506
15. Kroiss, C., Koch, N., Kozuruba, S.: UWE Metamodel and Profile: User Guide and Reference. http://uwe.pst.ifi.lmu.de/download/UWE-Metamodel-Reference-v1.9.pdf, 31.03.2012. (2011)
16. Geiger, N., George, T., Hahn, M., Jubeh, R., Zündorf, A.: Using Action Charts for Reactive Web Application Modeling. In Daniel, F., Facca, F.M., eds.: Current Trends in Web Engineering: 10th International Conference on Web Engineering ICWE 2010 Workshops. Springer, Berlin and Heidelberg (2010) 49–60

17. Busch, M., Koch, N.: Rich Internet Applications: State-of-the-Art. http://uwe.pst.ifi.lmu.de/publications/maewa_rias_report.pdf, 31.03.2012. (2009)
18. Hess, T., ed.: Ubiquität, Interaktivität, Konvergenz und die Medienbranche: Ergebnisse des interdisziplinären Forschungsprojektes intermedia. Univ.-Verl. Göttingen, Göttingen (2007)
19. Scherer, S.: Entwicklung von Rich Internet Applications: Evaluierung von UWE anhand einer Fallstudie. http://uwe.pst.ifi.lmu.de/publications/SchererDA.pdf, 31.03.2012. (2010)
20. Kroiss, C., Koch, N., Knapp, A.: UWE4JSF: A Model-Driven Generation Approach for Web Applications. http://www.pst.ifi.lmu.de/~kochn/icwe2009_kroiss_uwe4jsf.pdf, 31.03.2012. (2009)
21. Kraus, A., Knapp, A., Koch, N.: Model-Driven Generation of Web Applications in UWE. http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-261/paper03.pdf, 31.03.2012. (2007)
22. Ceri, S., Fraternali, P., Bongio, A.: Web Modeling Language (WebML): A Modeling Language for Designing Web Sites. http://www9.org/w9cdrom/177/177.html, 31.03.2012. (2000)
23. Brambilla, M., Comai, S., Fraternali, P., Matera, M.: Designing Web Applications with WebML and WebRation. In Rossi, G., Olsina, L., Pastor, O., Schwabe, D., eds.: Web Engineering: Modelling and Implementing Web Applications. Springer, London (2008) 221–261
24. Moreno, N., Fraternali, P., Vallecillo, A.: WebML Modeling in UML. http://www.webml.org/webml/upload/ent5/1/IETJournal.pdf, 31.03.2012. (2007)
25. Bozzon, A., Comai, S.: Conceptual Modeling and Code Generation for Rich Internet Applications. http://www.webml.org/webml/upload/ent5/1/fp553-Bozzon.pdf, 31.03.2012. (2006)
26. Brambilla, M., Facca, F.M.: Building Semantic Web Portals with WebML. http://www.webml.org/webml/upload/ent5/1/icwe2007.pdf, 31.03.2012. (2007)
27. Brambilla, M., Ceri, S., Facca, F.M., Celino, I., Cerizza, D., Della Cefriel, E.V.: Model-Driven Design and Development of Semantic Web Service Applications. http://www.webml.org/webml/upload/ent5/1/toit_final.pdf, 31.03.2012. (2006)
28. Moreno, N., Fraternali, P., Vallecillo, A.: A UML 2.0 Profile for WebML Modeling. http://www.webml.org/webml/upload/ent5/1/vergara-mdwe2006.pdf, 31.03.2012. (2006)
29. Wimmer, M., Schauerhuber, A., Schwinger, W., Kargl, H.: On the Integration of Web Modeling Languages: Preliminary Results and Future Challenges. http://publik.tuwien.ac.at/files/pub-inf_4677.pdf, 31.03.2012. (2007)
30. Brambilla, M.: Building Semantic Web Portals with a Model-Driven Design Approach. In Cardoso, J., Lytras, M.D., eds.: Semantic Web Engineering in the Knowledge Society. Information Science Reference, Hershey (PA) (2009) 76–106
31. Rossi, G., Schwabe, D.: Model-Based Web Application Development. In Mendes, E., ed.: Web Engineering. Springer, Berlin and Heidelberg (2006) 303–333
32. Schwabe, D., Lima, F.: Modelling Applications for the Semantic Web. In Cueva Lovelle, J.M., Del Puerto Paule Ruiz, M.d., González Rodríguez, B.M., Joyanes Aguilar, L., Labra Gayo, J.E., eds.: Web Engineering: International Conference, ICWE 2003, Oviedo, Spain, July 14-18, 2003. Proceedings. Springer, Berlin and Heidelberg (2003) 417–426
33. Achilleos, A., Kapitsaki, G.M., Papadopoulos, G.A.: A Model-Driven Framework for Developing Web Service Oriented Applications. http://www.dlsi.ua.es/~santi/mdwe2011/papers/mdwe2011_submission_2.pdf, 31.03.2012 (2011)

Implementierungstechnologien für Web-Anwendungen

Manuel Leithner

Otto-Friedrich-Universität Bamberg, Lehrstuhl für Medieninformatik

Zusammenfassung. Diese Seminararbeit behandelt neue Implementierungstechnologien für Web-Anwendungen. Zunächst werden die Herausforderungen moderner Webseiten vorgestellt. Der Webstack und ein kleiner Marktüberblick der vorhandenen Technologien definieren dafür einige Grundlagen. Das darauf folgende Kapitel stellt die Tool-Suite HipHop for PHP vor, welche es ermöglicht bestehende PHP-Projekte zu beschleunigen. Der zweite Abschnitt der Arbeit führt Node.JS, eine aktuelle Neuerung am Markt für Server-basierte Anwendungssysteme ein. Ein Beispiel verdeutlicht hierbei die Funktionsweise und Einfachheit des Systems. Das Fazit grenzt beide Ansätze noch einmal voneinander ab.

Schlagworte: Herausforderungen, HipHop for PHP, Node.JS

1 Herausforderungen moderner Web-Anwendungen

Heutzutage stehen Web-Entwickler und Systemadministratoren vor immer größeren Aufgaben, um den Internetauftritt bestmöglich zu warten. Durch die rasche Entwicklung der mobilen Endgeräte hin zu leistungs- und internetfähigen Smartphones werden auch Web-Anwendungen von diesen immer stärker frequentiert. Dies führt auch zu einer Erweiterung der Frontends, welche eine Anwendung unterstützen sollte. Neben dem klassischen Desktop- (Zugriff einer nativen Anwendung über Webservices) und Web-Frontend kommt nun zusätzlich noch ein mobiles Ausgabesystem hinzu. Deshalb müssen nicht nur Änderungen am Layout - zur Herstellung der Anzeigekompatibilität der Webseite - vorgenommen, sondern auch das Gesamtsystem an die neuen Gegebenheiten angepasst werden. Hier ist vor allem dessen Leistungsfähigkeit ins Auge zu fassen. Diese ist durch folgende Formel definiert:

$$\textit{Leistungsfähigkeit des Servers} = \frac{verarbeiteteRequests}{Sekunde}$$

Je nach Kompatibilität einer Webseite für mobile Endgeräte kann sich die Nutzer- bzw. Zugriffszahl auf ein System drastisch erhöhen. Dadurch kann jenes schnell an die Grenzen der Belastbarkeit kommen. Dieser Effekt wird im Zuge von *Web 2.0* mittels der neuen einsetzbaren Technologien zusätzlich verstärkt. Auf Webseiten setzt man nun vermehrt Javascript-Frameworks wie JQuery oder die YUI-Library ein. Dadurch ist es dem User möglich ohne kompletten Reload

mit der Seite zu interagieren. Dies vermindert zwar den anfallenden Traffic, jedoch kommt es deshalb vermehrt zu speziellen Anfragen an den Webserver und somit ebenso zu mehr Requests. Dies kann bei Chatsystemen oder Livetickern beobachtet werden (z.B. Auto-Refresh alle 3-5s).

Die Skalierbarkeit eines Systems ist somit die zentrale Herausforderung moderner Web-Anwendungen. Trotz neuer Funktionen und einer Vielzahl von neuen Nutzern muss alles weiterhin stabil laufen und die Anfragen bearbeitet werden können. Hierbei kommen heutzutage entweder spezielle Performance-Optimierer (z.B. HipHop for PHP) oder neue Technologien (z.B. Node.JS) zum Einsatz, die mit neuen Lösungsansätzen versuchen, bekannte Probleme zu lösen. Beide Wege werden nun im Zuge dieser Seminararbeit erläutert und mit Beispielen veranschaulicht.

2 Grundlagen

Bevor die angesprochenen Technologien zur Sprache kommen, sollen zunächst einige Grundlagen erläutert werden. Der *Webstack* verdeutlicht den Aufbau einer Web-Anwendung anhand eines gemischten Schichtenmodells (Hardware & Software). Eine Marktanalyse wird die Verteilung einzelner Web-Technologien sowohl für Frontend-, als auch Backend-Technologien in aktuellen Webportalen darstellen.

2.1 Webstack

Wie bereits beschrieben handelt es sich beim Webstack um ein sogenanntes gemischtes Schichtenmodell. Gemischt deshalb, weil Hardware- und Software-Schichten hier vereint werden, um zum einen den komplexen Aufbau und zum anderen alle Bestandteile einer Webanwendung aufzeigen zu können.

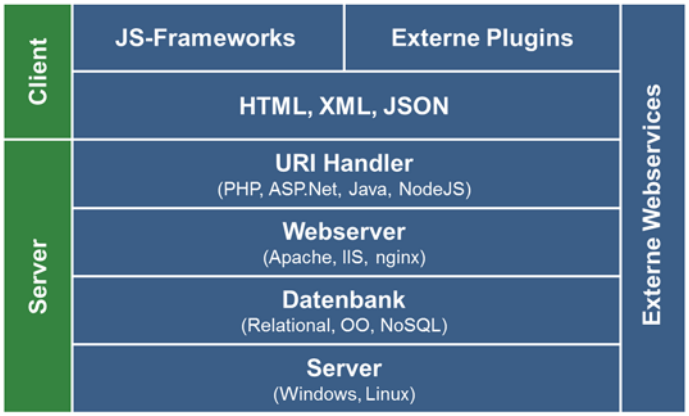


Abb. 1. Der Webstack

Abbildung 1 visualisiert jenen Webstack. Dieser ist zunächst geteilt in Client- und Server-Bestandteile. Auf Serverseite besteht die unterste Schicht aus der Server-Infrastruktur. Hierbei handelt es gewöhnlich um einfache Root-Server auf Windows- bzw. Linux-Basis, welche wiederum verteilt sind bzw. zu komplexen Rechnernetzen verbunden werden können. Aufbauend auf der Hardware-Plattform befindet sich die globale Datenbank zur Verwaltung der anfallenden Daten. Dabei kann zwischen verschiedenen Datenbankformen unterschieden werden. Relationale, Objektorientierte oder NoSQL Datenbanken gehören zu den am häufigsten genutzten. Als Webserver kommen heutzutage der frei verfügbare Apache-Server[1] oder Microsofts Pendant IIS[2] zum Einsatz. Große Portale wie GitHub, Imageshak oder SourceForge setzen zunehmend auch auf den sehr performanten *nginx*-Webserver[3]. Als oberste Schicht der Serverseite sind die verschieden *URI-Handler* zu nennen, die Anfragen vom Nutzer verarbeiten und die gewünschten Daten zur Verfügung stellen. *PHP*, *Java* und *ASP.Net* sind hier die Vorreiter. Mit dem im weiteren Verlauf der Seminararbeit vorgestellten *Node.JS* kommt jetzt ein neuer aufstrebender Typ hinzu.

Die Clientseite definiert sich zunächst über die gängigen Ausgabeformate bzw. Datenformate wie *HTML*, *XML* oder *JSON*. Darauf aufbauend kommen Javascript-Frameworks zum Einsatz, um Interaktionen auf der Webseite zu ermöglichen. Nebenbei können auch weitere *externe Plugins* verwendet werden. Dabei handelt es sich meist um Programme wie *Adobe Flash* oder *Adobe Reader*[4].

Die Schicht *Webservices* erstreckt sich über den Client- und Serverteil, damit sowohl auf Client- als auch auf Serverseite externe Webservices angesteuert und verarbeitet werden können.

2.2 Marktverteilung

Die aktuelle Marktverteilung von Backend- und Frontend-Systemen soll einen kleinen Überblick über die vorhandenen Typen geben. Als Quelle wird hier die bekannte Webseite *W3Techs* [1] herangezogen, wo monatlich neue Referenzzahlen des aktuellen Markt herausgegeben werden.

Der Frontend-Sektor wird mit 91.8% klar von *JavaScript* dominiert. Dies liegt vor allem an der Unterstützung der Scriptsprache durch die gängigen Browser. Auf den Plätzen 3-4 befinden sich verschiedene Browser-Plugins, welche von *Flash* mit 25.1% angeführt werden. Microsofts Pendant *Silverlight*[5] und Oracle's *Java*-Plugin[6] teilen sich zusammen ca. 0.5% des Marktes.

Auch zwischen verschiedenen JS-Frameworks unterscheidet *W3Techs*. Auf fast jeder zweiten Webseite kommt hier das *JQuery-Framework*[7] (45.0%) zum

[1] http://httpd.apache.org
[2] http://www.iis.net
[3] Engine-X ausgesprochen - http://www.nginx.org
[4] http://www.adobe.com
[5] http://msdn.microsoft.com/de-de/silverlight/default.aspx
[6] http://www.java.com
[7] http://www.jquery.com

Einsatz. Weitere bekannte Vertreter wie *MooTools*[8], *Prototype*[9] oder *ASP.Net Ajax*[10] haben jeweils nur einen geringen Marktanteil von ca. 3-5%.

Im Backend-Bereich ist der URI-Handler *PHP* das Zugpferd mit weiterhin steigendem Marktanteil (77.3%). Gefolgt von Microsofts *ASP.Net*, welches in den letzten Jahren einiges an Boden gutmachen konnte (21.6%). *JavaScript* als Backendsprache wird derzeit nicht aufgeführt, da der Marktanteil unter 0.1% liegt. Dies sollte sich jedoch in den nächsten Monaten ändern, da vor allem im Cloud-Computing Bereich zunehmend JavaScript als Backendsprache verwendet werden kann.

3 HipHop - Facebook's PHP Booster

HipHop for PHP ist, wie der Name bereits vermuten lässt, ein Optimierungstool speziell für die Scriptsprache PHP und die darin entwickelten Webanwendungen.

3.1 Hintergründe

Das von *Facebook* entwickelte Toolkit wurde erstmals im Februar 2010 dem Publikum vorgestellt. Die Veröffentlichung kam sehr überraschend, da es mit der *Zend-Engine*[11] bereits seit langem ein kommerzielles Optimierungs-Framework am Markt gibt, welches sehr gute Dienste leistet. Anders als die Zend-Engine war *HipHop* anfangs jedoch nur ein einfacher *Source-Code-Transformer*, welcher PHP-Code zur *Build-Time*[12] in effizienteren *C++–Code* umwandelt. Inzwischen wurde das Projekt jedoch soweit vorangetrieben, dass es im Dezember 2011 um eine *Virtuelle Maschine* - ähnlich wie bei der Zend-Engine - erweitert wurde. Wie auch die Zend-Engine ist HipHop prinzipiell für Großprojekte ausgelegt. Der hohe Anpassungsaufwand und der bei kleinen Projekten geringe Nutzen stehen hierbei in keinem Verhältnis zueinander und somit ist HipHop dafür nicht empfehlenswert.

Veröffentlicht als OpenSource-Projekt unter der *Creative-Commons-Attribution-ShareAlike-License* kann hier ein großer Vorteil gegenüber der Zend-Engine ausgemacht werden. Im Gegensatz zum kommerziellen Framework aus dem Hause Zend ist *HipHop* und desssen Source-Code auf *GitHub*[13] für jeden kostenlos zugänglich und nutzbar. Sofern sich Facebook's Opensource-Projekt in Zukunft weiter verbreiten sollte, entsteht hier eine echte Konkurrenz zur Zend-Engine, was sich sicherlich auch auf den Profit von Zend auswirken wird. Ob dies jedoch von Facebook beabsichtigt ist bzw. war, um evtl. einen weiteren Geschäftszweig zu erschließen, kann man nicht genau beziffern.

[8] http://mootools.net
[9] http://prototypejs.org
[10] http://www.asp.net/ajax
[11] http://www.zend.com
[12] Zeitpunkt, in dem der Script-Code in lauffähigen Code umgewandelt wird
[13] https://github.com/facebook/hiphop-php

3.2 Probleme und Lösungsansätze

Ein Projekt mit derzeit mehr als 650.000 Zeilen Source-Code und über 3 Jahren Entwicklungszeit ist natürlich auch mit enormen Kosten verbunden. Gründe für die Entwicklung eines solchen Vorhabens sind auftretende Probleme, die es bei einem solch riesigen sozialen Netzwerk natürlich gibt. Webseiten in der Größenordnung von Facebook haben immer mit der bereits angesprochenen Herausforderung der Skalierbarkeit zu kämpfen. Im Jahre 2009 waren auf dem Internetportal über *200 Millionen Nutzer* aktiv. Die Bearbeitung der rund *200 Milliarden Pageviews pro Monat* übernahm zum damaligen Zeitpunkt eine Rechnerinfrastruktur mit *ca. 30.000 Servern* [2]. Laut Angaben von Facebook ist die Zahl der Nutzer im Jahr 2012 auf *845 Millionen* [3] angewachsen. Vermutlich ist auch die Serverlandschaft ähnlich angewachsen. Ein weiteres Problem sind rechenintensive Module, die eine hohe CPU-Auslastung zur Folge haben. Dies sind beispielsweise Module für die Speicherung und Konvertierung von abgelegten Bildern. Die damit verbundene hohe Speicherauslastung erfordert ebenfalls eine Ausweitung des Servernetzes, was wiederum zu hohen Kosten führt. Auf Softwareseite sind auch einige Probleme zu verzeichnen. Zum einen sollte es möglich sein komplexe, in PHP entwickelte Programmlogik auch in anderen Bereichen, wie z.B. mobilen Anwendungen, nutzen zu können. Zum anderen sind Optimierungen durch Auslagerung von Anwendungslogik in PHP-Extensions meist schwer umzusetzen, da diese in C++ geschrieben werden müssen, was ein hohes Know-How voraussetzt [4].

Um diese Probleme bestmöglichst zu lösen, wurden verschiedene Ansätze in Erwägung gezogen. Das Umschreiben der Code-Basis auf eine effizientere Programmiersprache ist die naheliegenste Lösung. Dies ist jedoch aufgrund der Projektgröße nicht möglich. Zudem würde ein Wechsel auf eine andere Programmiersprache sowohl einen enormen Know-How Verlust verursachen, als auch die großen Vorteile von PHP - die schnelle Erlernbarkeit und Produktivität - aushebeln. Eine weitere Möglichkeit komplexeren Code in PHP-Extensions auszulagern ist ebenfalls nicht ausreichend. Dies würde zwar den Code beschleunigen, jedoch sind hierfür, wie bereits erwähnt, C++-Kenntnisse erforderlich, was für den einfachen PHP-Entwickler Schwierigkeiten bedeutet. Dieser Umstand zeigt, dass eine Weiterentwicklung bzw. das Umschreiben der Zend-Engine ausgeschlossen ist, da dies wiederum spezielles Wissen voraussetzt. Desweiteren wäre auch eine Wiederverwendung der in PHP entwicklten Logik in anderen Bereichen nicht möglich.

Somit spricht einiges abseits der Markstrategie von Facebook dafür, ein eigenes anpassungsfähiges Toolkit, eben jenes *HipHop*, zu entwickeln. Jenes bestand anfangs aus zwei Programmen. Dem Compiler *hphpc* und dem Interpreter *hphpi*, um die es nun geht.

3.3 Ablauf und Einschränkungen

Die Umwandlung ist in zwei Phasen unterteilt. Abbildung 2 zeigt den Ablauf der *Code-Transformations*-Phase. Nach dem Parsen des PHP-Codes wird eine

statische Analyse vorgenommen. Diese spürt hierbei verschiedene Abhängigkeiten zwischen Modulen, Funktionen oder Klassen auf. Ebenfalls werden sogenannte *"Wer deklariert/definiert was"*-Informationen gesammelt, die für eine spätere Typumwandlung benötigt werden. Die Umwandlung in einen *Abstrakten Syntax Baum (AST)*[14] schließt die statische Analyse ab.

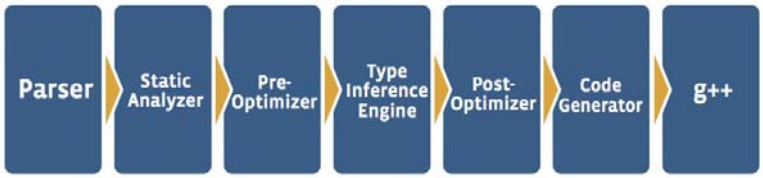


Abb. 2. Ablauf der Kompilierung

Die *Type-Inference Engine* versucht nun für die definierten Variablen möglichst naheliegende C++-Typen wie String, Integer, Object oder Array zu finden. Dies ist nötig, da PHP schwach, C++ jedoch streng typisiert ist. In Sonderfällen, falls kein Typ auffindbar ist, wird der variable Typ *Variant* verwendet. Die zwei Optimierungsschritte *Pre-Optimizer* und *Post-Optimizer* kapseln die Type-Inference und versuchen nochmals den Code bestmöglich zu optimieren. Abschließend wird der *Abstrakte Syntax Baum* in C++-Code überführt und per g++[15] in lauffähigen x64-Maschinencode kompiliert.

Die zweite Phase ist die sogenannte *Runtime*. Hierbei wird zunächst ein *Linking* durchgeführt, bei dem die verschiedenen von *HipHop* zur Verfügung gestellten Typen (z.B. String, Array, Object, usw.) mit dem erzeugten Code verbunden werden. Das *Publishing* schließt die Runtime-Phase ab. Als Ergebnis des HipHop-Prozesses entsteht eine Web-Anwendung mit integriertem Webserver, welche in einem einfachen Prozess mit mehreren Threads ausgeführt wird. Auch dies hat Vorteile gegenüber herkömmlichen Webservern. Durch den einzelnen Prozess ist es möglich Objekte zwischen den Threads zu teilen. Somit kann die Zahl der vorhandenen Datenbank-Objekte verringert werden, was ebenfalls einen Performancegewinn mit sich bringt. Das funktioniert z.B. beim Apache-Webserver in Verbindung mit dem Prefork-Module[16] nicht bzw. nur eingeschränkt.

Am Beispiel des *Variant*-Typs zeigt sich bereits, dass die Umwandlung nicht immer eins-zu-eins möglich ist. Laut Unternehmensangaben unterstützt HipHop inzwischen zwar die Funktionalitäten der PHP-Version 5.3 [4] jedoch können einige spezielle Bereiche nicht umgesetzt werden. Dies betrifft vor allem das *dynamische Coding*. Funktionen wie *Eval()*, *Create_Function()* oder *Preg_replace() in Verbinding mit /e* sind nicht verwendbar. Ebenfalls gibt es Seiteneffete bei

[14] Nicht-detailierte Abbildung des Source-Codes in eine Baumdarstellung
[15] http://gcc.gnu.org - Compiler für C++-Code
[16] http://httpd.apache.org/docs/2.0/de/mod/prefork.html - Modul zur Aufteilung des Servers auf mehrere Prozesse

Reihenfolge-abhängigen Prüfungen auf Symbole (function, class, constants). So kann es z.B. zu unregelmäßigen Abläufen kommen, wenn eine Funktion auf Existenz geprüft wird, diese aber erst zu einem späteren Zeitpunkt definiert wird.

Es ist jedoch möglich diese Probleme durch sauberes Programmieren zu umgehen, da es sich dabei eher um scriptsprachentypische Spielereien handelt.

3.4 HipHop Interpreter und Virtuelle Maschine

Wie bereits erwähnt ist neben dem *Compiler* auch ein *Interpreter (hphpi)* Bestandteil des HipHop-Projektes. Dieser wurde ebenfalls komplett neu entwickelt, da bei der Kompilierung immer das gesamte Projekt umgewandelt wird. Je nach Größe der Code-Basis beansprucht dies natürlich einige Zeit. Der typische PHP-Entwickler ist jedoch gewohnt auf Änderungen im Code direktes Feedback durch den Output zu erhalten. Hphpi ermöglicht ihm diese, aus PHP bekannten Vorteile auch in Verbindung mit HipHop. Entwickelt als Just-In-Time-Compiler agiert dieser ähnlich dem original PHP-Interpreter. Änderungen am Code werden direkt angezeigt, was ein schnelles Auffinden von Bugs sicherstellt. Durch die Just-In-Time-Kompilierung ist auch die Verwendung der *Eval()*-Funktion wieder möglich. Dies ist jedoch nur zum Debuggen nützlich, da im späteren Endprodukt der Einsatz der Funktion nicht mehr möglich ist.

Durch die Benutzung des Interpreters ist ein einmaliges Kompilieren der Gesamtanwendung nur noch bei Liveschaltung der neuen Module notwendig. Die Verwendung hat jedoch auch Nachteile. Aufgrund der Komplexität von HipHop ist der Interpreter fast zweimal so langsam wie der Standard-PHP-Interpreter. Dies kann bei häufigen kleinen Änderungen am Code schnell viel Zeit kosten. Ein weiterer Nachteil ist die Implementierung des Interpreters. Aufgrund der Ausführung als Just-In-Time-Compilers (JIT) verwendet dieser eine andere *AST*-Implementierung, was ebenfalls in Performance-Problemen endet.

Das Ersetzen des Interpreters durch die neu entwickelte *HipHop Virtuelle Maschine* im Dezember 2011 behebt diese Nachteile. Diese ist nun fester Bestandteil der HipHop-Toolsuite. Mit den Bestandteilen *Interpreter* und *Translator* ist sie ebenfalls in zwei Phasen aufgeteilt. Der Interpreter wandelt den PHP-Code in einen *Abstrakten Syntax Baum* um, anschließend in sogenannten *HipHop Byte Code* (HHBC). Dieser Code ist nun auf der Virtuellen Maschine lauffähig. Dadurch ist das Programm ca. 1.6-mal schneller als der bisherige Interpreter *hphpi*, was sich bei der Entwicklung von PHP-Modulen deutlich bemerkbar macht. Bei der Umwandlung in Byte-Code setzt der Interpreter der Virtuellen Maschine nicht auf eine JIT-Kompilierung, sondern verwendet eine *Trace-based-Translation*, welche jeden Script-Loop analysiert und daraus sogenannte *Tracelets*[17] erstellt. Derzeit steht die Entwicklung der Virtuellen Maschine bei ca. 90%, da es noch Performance-Probleme beim *Translator* gibt. Auch nutzt dieser noch kein Profiling-Feeback[18] bei den Optimierungen des Byte-Codes [5].

[17] Redefinition eines Scriptblocks. Bestandteile: TypeGuard, Body und Links zu weiteren verknüpften Tracelets

[18] Auswertung des Sourcecodes auf Schwachstellen und Optimierungsmöglichkeiten

3.5 Benchmarks

Um die Leistung eines Tools beurteilen zu können, ist es ratsam verschiedene Benchmarks zu erstellen. Laut eigenen Angaben senkte die Einführung von HipHop die CPU-Last um 50%. Da dies jedoch nicht unbedingt repräsentative Angaben sind, wurden auch von externen Portalen Benchmarks erstellt.

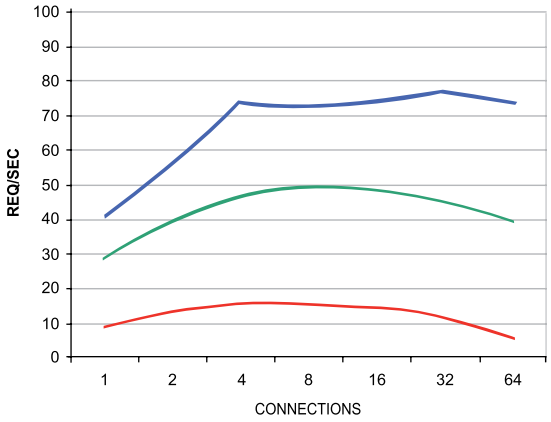


Abb. 3. Performance-Vergleich [6]

Abbildung 3 zeigt einen Vergleich der verarbeiteten Requests pro Sekunde bei installiertem Drupal[19] unter Verwendung von Standard-PHP, PHP mit APC[20] oder HipHop. Die Grafik macht deutlich, dass der Webserver mit HipHop fast fünf mal so viele Requests pro Sekunde wie die reine Standard-PHP Installation verarbeiten kann. Auch im Vergleich zu PHP mit dem APC-Cache Module ist die HipHop-Variante deutlich im Vorteil [6]. Weitere Tests zeigen, dass Wordpress mit HipHop ca. 40% schneller arbeitet als eine Vergleichskonfiguration mit dem Nginx-Webserver, einem APC-Cache und dem PHP-fpm-Modul[21] [7].

3.6 Roadmap und Fazit

Für die nahe Zukunft plant Facebook die Behebung der oben angesprochenen Probleme mit der Virtuellen Maschine. Aufgrund der guten Performance stehen ebenfalls Überlegungen aus, die komplette Code-Basis des Live-Systems auf die Virtuelle Maschine auszulagern und somit nur noch ein Tool sowohl bei der Entwicklung als auch im Betrieb zu nutzen. Aktuell wurden große Portale wie Drupal, Wikipedia oder Wordpress an die Code-Vorgaben angepasst, damit diese

[19] http://drupal.org
[20] http://pecl.php.net/package/APC
[21] http://php-fpm.org

unter HipHop lauffähig sind. Ziel ist es, zusätzliche Projekte von HipHop zu überzeugen und diese anzupassen, um einen größeren Nutzerkreis zu erhalten [5].

Die zuletzt genannten Absichten machen deutlich, dass Facebook mit dem Gedanken spielt, dieses OpenSource-Projekt weiter zu treiben, um eventuell einen weiteren Geschäftszweig im Support für HipHop zu eröffnen. Dies wird sich jedoch erst in den nächsten Jahren zeigen. Aktuell bleibt zu sagen, dass sich laut Angaben von Facebook die Entwicklung gelohnt hat und das HipHop fester Bestandteil der IT-Infrastruktur wird, da man von der Idee und der verwirklichten Leistung überzeugt ist.

4 Node.JS - Server-Side Javascript

Neben der Optimierung bestehender Technologien stellt die Implementierung eines Webprojektes in einer neuen Sprache eine weitere Möglichkeit dar, vorhandene Performanceprobleme zu beheben. Dieses Kapitel soll nun eine Einführung in das neue Node.JS Framework geben, welches aktuell viel diskutiert wird.

4.1 Einführung

Der von *Ryan Dahl* im Jahr *2009* veröffentlichte URI-Handler gilt als zukunftsweisende Technologie im Web. Die aktuelle Programmversion *0.6.13* (Stand März 2012) lässt jedoch darauf schließen, dass es sich vorerst um eine Betaversion handelt. Dennoch kommt *Node.JS* bereits auf einigen großen Portalen wie *Yahoo*[22], *Ebay*[23] oder *LinkedIn*[24] erfolgreich zum Einsatz. Auch im Bereich des Cloud-Computing können starke Zuwächse des Marktanteils verzeichnet werden. Nicht zuletzt durch das Einbinden von Node.JS in die *Microsoft-Azure-Plattform* [8].

Wie der Name bereits vermuten lässt, basieren Node.JS-Anwendungen auf Javascript. Die Idee das klassische Client-Javascript auch auf Serverseite, also außerhalb des Browsers, zu verwenden, ist jedoch nicht neu. Bereits in der Vergangenheit gab es Ansätze das zu verwirklichen. *Netscape LiveWire* [9] war der erste Webserver der Javascript auf den Server brachte. Selbst bei moderneren Webtechnologien kommt Javascript zum Einsatz. *ASP* in Verbindung mit Microsofts Javascript-Pendant *JScript*[25] macht dies ebenfalls möglich. Auch in der Java-Welt gibt es mit *RingoJS*[26] ein Javascript-Webframework.

Jedoch ist erst Node.JS derart leistungsfähig, dass sich die Vorteile der Javascript-Umgebung bemerkbar machen. Dies ist vor allem der verwendeten Plattform zu verdanken. Mit Googles V8-JavaScript-Engine[27] kommt die derzeit wohl

[22] http://www.yahoo.de
[23] http://www.ebay.de
[24] http://www.linkedin.com
[25] http://msdn.microsoft.com/de-de/library/72bd815a(v=vs.80).aspx
[26] http://www.ringojs.org
[27] http://code.google.com/p/v8

leistungsstärkste Javascript-Umgebung zum Einsatz. Durch die Just-In-Time-Kompilierung des Javascript-Codes läuft dieser wesentlich schneller als eine rein interpretierte oder in Byte-Code umgewandelte Anwendung.

Ein weiterer Vorteil von serverseitigem Javascript besteht darin, Server-Code auf Client-Seite wiederzuverwenden. So können z.B. mit ein und demselben Code Nutzereingaben in Formularfeldern sowie die danach an den Server geschickten Daten überprüft werden. Optional ist auch der Einsatz von JQuery möglich, was die Entwicklung durch weniger Code und einfacheren Zugriff auf Objekte noch einmal drastisch erleichtert.

Wie bei einigen der anderen genannten Javascript-Servertechnologien wird dies durch die Implementierung des *CommonJS-Standard*[28] ermöglicht. Dieser definiert Vorgaben, um Javascript auch außerhalb des Browsers zu verwenden. Durch verschiedene *APIs* werden Klassen und Methoden zur Verfügung gestellt, um beispielsweise einen Webserver, FTP-Server, Kommandozeilentools oder auch grafische Oberflächen zu realisieren. Node.JS unterstützt dies ebenfalls durch eine sehr kompakte und einfache API. Die wichtigsten Klassen sind hierbei *Http*, *FileSystem*, *Time* und *Sys*, mit deren Hilfe die meisten Aufgaben gemeistert werden können.

Sofern die vorhandenen Funktionen nicht ausreichen, kann die Webanwendung beliebig erweitert werden. Dies geschieht über den enthaltenen *Node Package Manager* (NPM). Abbildung 4 zeigt die Installation des *EJS-Modules*, welches ein Webserver-Framework mit sämtlichen Basisfunktionen bereitstellt. Wie

```
D:\Uni\SVN\Master 1\Seminar\Webserver>npm install ejs
npm http GET https://registry.npmjs.org/ejs
npm http 304 https://registry.npmjs.org/ejs
ejs@0.6.1 ./node_modules/ejs

D:\Uni\SVN\Master 1\Seminar\Webserver>
```

Abb. 4. Node Package Manager

auf der Grafik erkennbar ist, ist NPM dem aus Linux bekanntem Tool *APT-GET* sehr ähnlich. Mit den Befehlen *install*, *uninstall*, *search* oder *publish* lassen sich neue Module installieren, löschen, suchen oder eigene Module ins globale Repository[29] einstellen. Jenes verfügt derzeit über mehr als 6.000 Erweiterungen für so gut wie jeden Anwendungsfall. Abhängigkeiten zwischen Modulen werden von NPM automatisch bei der Installation aufgelöst und fehlende Pakete nachgeladen.

[28] http://www.commonjs.org
[29] http://www.npmjs.org

4.2 Technologie

Neben Googles V8-Engine ist auch die spezielle Architektur von Node.JS ein Grund für die gute Performance dieser Technologie. Schreib- und Lesezugriffe sind heutzutage die größten Performancekiller. Jene benötigen ca. 40-240 Millionen *CPU-Cycles*[30]. Im Gegensatz dazu dauert der Zugriff auf *RAM*- oder *Cache-Level* unter 250 Cycles [10]. Node.JS setzt deshalb auf sogenanntes *Non-Blocking IO* - also nicht blockierende Schreib- und Lesezugriffe. Dies wird durch eine eventbasierte Single-Execution-Thread Architektur ermöglicht.

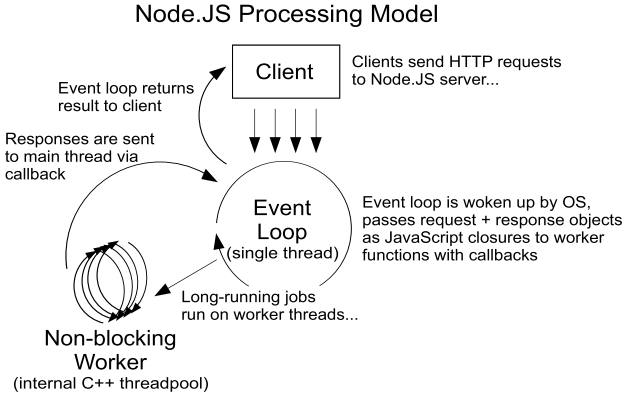


Abb. 5. Der Event-Loop[11]

Abbilding 5 zeigt die Funktionsweise eines Serverzugriffs. Zunächst senden verschiedene Clients HTTP-Requests an den Webserver. Dadurch wird der sogenannte *Event-Loop* vom Betriebssystem "aufgeweckt". Dieser einzelne Thread gibt die Aufgaben jeweils als Javascript-Closures an einen internen Threadpool weiter. Jene *Non-Blocking-Worker-Threads* bearbeiten die anfallenden Aufgaben, ohne die gesamte Anwendung zu blockieren. Nach Abschluss der Bearbeitung werfen die Javascript-Funktionen die Ergebnisse (*Responses*) per *Callback* an den Hauptthread zurück. Dieser schickt abschließend das gesamte Ergebnis an den jeweiligen Client.

Eventbasiertes Handling von Request auf Basis von Eventlistenern und Eventemittern ermöglicht nicht nur ein nicht blockierendes Anwendungssystem sondern erleichtert auch die Arbeit des Entwicklers, da dieser sich nicht mit Threads und Synchronisation auseinandersetzen muss.

[30] Taktrate einer CPU. Beispielsweise hat eine 3-MHz CPU 3 Millionen Cycles pro Sekunde

4.3 Fallstudie

Eine kurze Fallstudie soll nun die Entwicklung mit Node.JS verdeutlichen. Hierbei wird eine sogenannte *Meeting-Minutes*-Anwendung zum Dokumentieren von Besprechungen implementiert. Abbildung 6 zeigt das gewünschte Endergebnis. Auf der linken Seite können die Meetings gespeichert werden, während die rechte Seite aktuelle News per Newsfeed asynchron abruft und dem Management angezeigt.

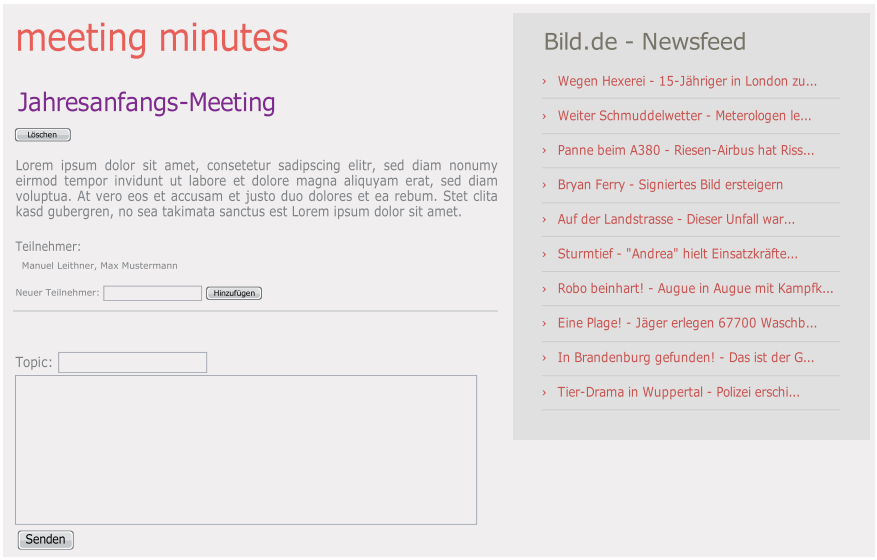


Abb. 6. Fallbeispiel

Anfangs sollen zwei verschiedene Möglichkeiten eines Webservers aufgezeigt werden, die die Unterschiede der Laufzeit verdeutlichen. Beim Aufruf des *HelloWorld.js*-Beispiels aus Listing 1.1 schreibt Node.JS "Hello World" auf die Console und beendet Programm.

```
1  // HelloWorld.js
2  console.log('Hello World\n');
3
4  // HelloWorld2.js
5  var http = require('http');
6  http.createServer(function (req, res) {
7      res.writeHead(200, {'Content-Type': 'text/plain'});
8      res.end('Hello World\n');
9  }).listen(1337, "127.0.0.1");
10 console.log('Server running at http://127.0.0.1:1337');
```

Listing 1.1. Node.JS Webserver

"Server running at http://127.0.0.1:1337" wird beim Start von *HelloWorld2.js* auf der Console und "Hello World" bei einem *HTTP-Request* ausgegeben. Dies liegt daran, dass nach der Erstellung des *http*-Objekts (Zeile 5) ein Eventlistener (Zeile 9) am Port 1337 registriert wird [12]. Die Eventarchitektur von Node.JS hindert nun die Anwendung sich nach Abarbeitung des Script zu schließen, da Node.JS stets sicherstellt, dass das Programm erst endet, wenn keine Events mehr auftreten. Da jedoch Client-Request am Port 1337 immer eintreffen können, läuft diese solange bis der Admin den Server per Hand stoppt.

Die Erstellung eines Webservers mit den gezeigten Beispielen macht jedoch nur bei sehr einfachen Programmen Sinn. Bei komplexeren Web-Anwendungen sollte man auf Module wie das bereits angesprochene *EJS* zurückgreifen, welches bereits einen kompletten Webserver zur Verfügung stellt.

```
1  var server = express.createServer();
2  server.use(express.bodyParser());
3  server.register('.html', require('ejs'));
4  server.set('views', __dirname + '/views');
5  server.set('view engine', 'ejs');
6  server.use(express.static(__dirname));
7  server.post( '/addMeeting',
8    function(request, response) {
9      database.addMeeting( request.body.topic, request.body
           .note, function(error) {
10         redirect(error, response, '/view');
11     });
12   }
13 );
```

Listing 1.2. Webserver-Setup

Listing 1.2 zeigt wie mit diesem Modul ein Webserver konfiguriert werden kann. Nach der Erstellung eines Server-Objekts wird die Render-Engine und deren Eigenschaften (Typ, Verzeichnis und Render-Engine - Zeile 2-6) definiert. Folgend können verschiedene Routings (Zeile 7-13) erstellt werden, welche jedem URL-Aufruf eine Funktion zuweisen. In diesem Fall wird beim Aufruf der Webseite in Verbindung mit "/addMeeting" ein neues Meeting eingetragen. Die Post-Methode gibt hierbei an, dass es sich um einen *POST*-Request handelt und nur ein solcher gültig ist. *GET*-Requests würden nicht verarbeitet werden.

Im Fallbeispiel ist es gewünscht beim Laden der Seite mehrere Aktionen gleichzeitig durchzuführen. Hierzu zählt zum einen die Datenbankabfrage auf Einträge bzw. das Speichern oder Löschen von Einträgen und zum anderen das Abfragen des RSS-Feeds. Für die Ausführung dieser Aufgaben benötigt man das *Async*-Modul, um asynchrone Aufrufe zu synchronisieren und am Ende der Aktion einen gemeinsamen Callback zu werfen. Listing 1.3 zeigt wie dies in der Praxis implementiert wird. Zeile 1 definiert ein neues *GET*-Routing, welches beim Aufruf der "\view"-URL aktiv wird. Die Synchronisation wird durch *asyc.series* gestartet. Alle dort definierten Funktionen werden aufgerufen

und deren geworfene Callbacks gesammelt. Der finale gemeinsame Callback (Zeile 6-10) wird am Ende der Serie verarbeitet und ebenso geworfen.

```
1  server.get( '/view', function(request, response) {
2    async.series( {
3      rss: function(callback) {...lade RSS-Feed...},
4      meetings: function(callback) {...lade Einträge...}
5    },
6    function(err, results) {
7      // Render finale Ausgabeseite
8      response.render('list.html', {title: pageTitle,
           meetings: results.meetings, rss: results.rss,
           ...});
9    });
10 });
```

Listing 1.3. Synchronisation asynchroner Aktionen

Die Einträge der Anwendungen werden in diesem Fall in einer *NoSQL*-Datenbank abgelegt. Zwar unterstützt Node.JS auch sämtliche andere Datenbank-Typen, jedoch bietet es sich bei einem solch kleinen Beispiel an, diese neue Variante zu nutzen, da der Setup-Aufwand vergleichsweise gering ausfällt. Auch die verschiedenen Operationen auf der Datenbank (Select, Insert, Delete, Update) sind vergleichsweise einfach zu implementieren. Beispielsweise genügt ein Aufruf der *save*-Methode auf einem Objekt, um dieses in der Datenbank abzulegen. Dessen Attribute werden folgend in die Datenbank geschrieben. Vorher spezifizierte Schemata sind möglich, jedoch nicht zwingend notwendig, was wiederum Zeit bei der Entwicklung spart.

Ein weiterer spannender Aspekt ist die Handhabung von Multicore-Systemen durch Node.JS. Die spezielle Architektur basiert, wie bereits erwähnt, auf einem einfachen Thread, was eine direkte Unterstützung natürlich ausschließt. Hilfe bietet hierbei das *Cluster*-Modul, welches Anfangs von der Node-Community entwickelt wurde, aber seit der Version 0.6.5 fester Bestandteil des Frameworks geworden ist. Je nach Anzahl der Prozessorkerne startet dieses Modul neue Prozesse, um so die Leistungsfähigkeit moderner Serversysteme voll ausnutzen zu können.

```
1 var cluster = require('cluster');
2 var numCPUs = require('os').cpus().length;
3 if(cluster.isMaster) {
4   for(var i=0; i<numCPUs; i++) { cluster.fork(); }
5 } else {
6   server.listen(2000);
7 }
```

Listing 1.4. Multicore-Unterstützung

Listing 1.4 zeigt die Implementierung einer Multicore-Unterstützung. Zeile 6 definiert den Hauptprozess. Sofern dieser gestartet ist, werden in Zeile 4 zusätzliche Prozesse gestartet.

5 Fazit

Steigende Nutzerzahlen und der daraus folgende höhere Traffic erfordern immer schnellere Technologien, um die Skalierbarkeit der Systeme zu gewährleisten. Die beiden vorgestellten Varianten zeigen, wie dies heutzutage in die Praxis umgesetzt werden kann.

Die Optimierung bestehender Technologien durch Toolkits wie *HipHop* oder der *Zend-Engine* für PHP bieten den Vorteil, dass es im Unternehmen zu keinem Know-How-Verlust kommt, da vorhandene Entwickler weiter mit der bereits bekannten Sprache bzw. Technologie arbeiten können. Auch kommt es zu keinem Mehraufwand ein bestehendes Projekt neu aufzusetzen oder von Grund auf neu zu implementieren. Ein Nachteil dieses Vorgehens kann jedoch sein, dass die vorhandenen Probleme nur kurzfristig aufgeschoben werden, da bei weiter steigenden Nutzerzahlen dieselben Probleme wieder auftreten werden. Die Ursache für diese Performance-Probleme liegt nämlich meist tiefer und durch oberflächliche Optimierungen werden nur die Technologien bis zur Grenze ausgenutzt. Jedoch könnte das Serversystem meistens noch viel mehr leisten.

Um auch das zu beheben, wäre der zweite vorgestellte Ansatz der richtige Weg. Durch die Reimplementierung des Projekts in einer neuen zukunftsweisenden Technologie können bereits frühzeitig aus anderen Projekten bekannte Schwachpunkte vermieden werden. Node.JS beispielsweise beseitigt durch sein *Non-Blocking-IO*-Verfahren den Missstand, die Anwendung während Schreib- oder Lesevorgängen zu blockieren, was hohe Performancegewinne ermöglicht. Das Wagnis auf eine neue Technologie zu setzen ist jedoch meist mit einem Know-How-Verlust verbunden, da diese noch sehr neu ist und viele Entwickler noch keine Erfahrungen damit gemacht haben.

Schlussendlich ist es eine sehr schwierige Entscheidung den richtigen Ansatz zu wählen. Fehlendes Know-How im Team ist oft ein Grund für das Scheitern bzw. Misslingen eines Projektes. Deshalb sollte eine vermeintlich neue bessere Technologie dennoch vermieden werden, sofern kein oder nur wenig Wissen über diese vorhanden ist.

References

1. W3Techs: extensive and reliable web technology surveys,
   unter *http://www.w3techs.com*,
   zuletzt geprüft am 23.03.2012
2. Jens Ihlenfeld: Wie Facebook die Daten von 300 Millionen Nutzern verkraftet,
   unter *http://www.golem.de/0910/70585.html*,
   zuletzt geprüft am 19.03.2012
3. Stern.de: Hintergrund: Facebook in Zahlen,
   unter *http://www.stern.de/news2/aktuell/facebook-in-zahlen-1781443.html*,
   zuletzt geprüft am 23.03.2012
4. Facebook: Facebook Technology Tasting: HipHop for PHP,
   unter *http://www.ustream.tv/recorded/4409735*,
   zuletzt geprüft am 23.03.2012
5. Facebook: The HipHop Virtual Machine,
   unter https://www.facebook.com/note.php?note_id=10150415177928920
6. Webtutor.pl: Drupal 7: HipHop for PHP vs APC - benchmark, unter
   http://php.webtutor.pl/en/2011/05/17/drupal-hiphop-for-php-vs-apc-benchmark,
   zuletzt geprüft am 23.03.2012
7. HPHP Playground: Hardened WordPress 3 Benchmark Part I,
   unter *http://huichen.org/en/2010/06/wordpress-3-benchmark*,
   zuletzt geprüft am 23.03.2012
8. Node.JS: Official Website,
   unter *http://nodejs.org*,
   zuletzt geprüft am 23.03.2012
9. The Computer Language Company Inc.: Netscape LiveWire Definition,
   unter *http://encyclopedia2.thefreedictionary.com/Netscape+LiveWire*,
   zuletzt geprüft am 23.03.2012
10. Mixu's tech blog: Understanding the node.js event loop,
    unter *http://blog.mixu.net/2011/02/01/understanding-the-node-js-event-loop*,
    zuletzt geprüft am 23.03.2012
11. Aaronontheweb: Intro to Node.JS for .NET Developers,
    unter *http://www.aaronstannard.com/post/2011/12/14/Intro-to-NodeJS-for-NET-Developers.aspx*,
    zuletzt geprüft am 23.03.2012
12. Herron, D.: Node Web development: A practical introduction to Node, the exciting new server-side JavaScript Web development stack. Packt Pub., Birmingham, U.K

Betrieb und Wartung von Web-Anwendungen

Dietlinde Flavia Lump

Otto-Friedrich-Universität Bamberg, Lehrstuhl für Medieninformatik
dietlinde-flavia.lump@stud.uni-bamberg.de

Zusammenfassung. Die Phase Betrieb und Wartung ist für viele Web-Anwendungen wichtiger, als zunächst allgemein vermutet wird. In dieser Arbeit soll auf drei Schwerpunkte eingegangen werden, die in dieser Phase essentiell sind: Zunächst erfolgt ein kurzer Einblick in die Web-Analyse mit dem Ziel den Erfolg einer Web-Anwendung zu messen, indem die Besucher und deren Verhalten analysiert werden. Als nächstes steht der Inhalt/Content der Web-Anwendung im Vordergrund und die Möglichkeit zur Unterstützung der Content-Pflege durch Web Content Management Systeme. Auf Weiterentwicklungen im Bereich der Web Content Management Systeme wird ebenfalls näher eingegangen. Abschließend wird ein Überblick über klassische Maßnahmen, wie auch neueren Trends zur Bewerbung einer Web-Anwendung gegeben.

Schlagworte: Betrieb, Wartung, Web-Analyse, Logfile-Analyse, Zählpixel-Analyse, Web Content Management System (WCMS), E-Mail-Marketing, Banner-Marketing, Affiliate-Marketing, Suchmaschinenmarketing, Social-Media-Marketing, Mobile-Marketing, Video-Marketing

1 Einleitende Gedanken und Aufbau der Arbeit

Die Phase Betrieb und Wartung von Web-Anwendungen beginnt ab dem Tag, an dem die Webseite online, der breiten Masse zur Verfügung gestellt wird. Durch die kontinuierliche/permanente Verbesserung und Weiterentwicklung der Web-Anwendung ist die Grenze zwischen den Phasen „Entwicklung“ und „Betrieb/Wartung“ nicht immer klar erkennbar. Der, vor allem bei Web-Anwendungen herrschende und wachsende Konkurrenzdruck und Time-to-Market Druck, hat zur Folge, dass oft zu schnell entwickelt wird und sorgt somit dafür, dass die Phase Betrieb und Wartung immer wichtiger wird.

Betrachtet man die Tätigkeiten in der Phase Betrieb und Wartung von Web-Anwendungen, so müssen einige charakteristische Eigenschaften von Web-Anwendungen in Betracht gezogen werden:

Berücksichtigt man die nutzungsbezogenen Charakteristika, so ist bei einer Web-Anwendung die weltweite und permanente Verfügbarkeit essentiell. Die Nutzer vertrauen darauf, dass die Webseite überall und ununterbrochen erreichbar ist und hierfür muss natürlich permanent gesorgt werden. Man kann also sagen, dass bei einer Web-Anwendung der Besucher im Mittelpunkt steht und deshalb auch die Interaktion zwischen ihm und der Web-Anwendung von entscheidender Bedeutung ist.

Bei einer Web-Anwendung spielt auch der Inhalt, der sogenannte „Content“ eine große Rolle. Die Qualität und Aktualität der Inhalte einer Webseite entscheiden oftmals über den Erfolg oder Misserfolg dieser. Vor allem in Anbetracht der Schnelllebigkeit des Internets, ist eine Web-Anwendung nur dann gut, wenn sie immer aktuellen und optimierten Inhalt bietet, wofür man auch lange nach der Entwicklung einer Web-Anwendung noch sorgen sollte. Man denke immer daran, dass der Besucher nur wenige Mausklicks von den Konkurrenzseiten entfernt ist und uninteressante und langweilige Seiten sehr schnell ersetzt werden können.

Eine Webseite ist für die meisten Unternehmen wie eine Visitenkarte, und stellt das Erscheinungsbild nach außen dar. Sie hat demnach oft „Produktcharakter“ und die Erreichung eines bestimmten Bekanntheitsgrades einer solchen Internetpräsenz ist dringend erforderlich. Aufgrund dessen muss auch die Werbung einer Web-Anwendung gut durchdacht und gezielt eingesetzt werden.

Diese genannten Besonderheiten haben starken Einfluss auf die Tätigkeiten in der Phase „Betrieb und Wartung“ von Web-Anwendungen, woraus sich auch die Schwerpunkte der nachfolgenden Arbeit ergeben: Für die Messung des Erfolgs einer Webseite sollte in dieser Phase eine Web Analyse durchgeführt werden, auf die im nächsten Kapitel eingegangen wird. Die Aktualität und Qualität der Inhalte einer Web-Anwendung stehen in Kapitel drei im Vordergrund. Das vierte Kapitel widmet sich schließlich der Bewerbung einer Web-Anwendung.

Auf die Aspekte Nutzerfreundlichkeit (Usability), Performanz und Sicherheit von Web-Anwendungen, mit denen man sich in der Phase Betrieb und Wartung auch beschäftigen sollte, wird an dieser Stelle nicht näher eingegangen.

2 Web Analyse

In der Literatur findet man neben dem Begriff „Web Analyse“ häufig auch die Bezeichnungen „Web-Controlling“ oder „Web-Analytics“, wobei die Begriffe nicht immer einheitlich verwendet werden [DüRa11,S.681], [Ree08, S.16], [Etr11]. Die „Digital Analytics Association“ definiert den Begriff „Web Analytics“ wie folgt:

> *Web Analytics is the measurement, collection, analysis and reporting of Internet data for the purposes of understanding and optimizing Web usage.*

Der Begriff Web Analyse umfasst somit die Messung, die Erfassung und die Analyse von Informationen über den Besucher der Webseite und dessen Verhalten, aus denen Optimierungsempfehlungen oder Zielgruppenanalysen erarbeitet werden. Betrachtet man Web Analyse eher als Gesamtprozess, so gehören Zielfestlegung, Kennzahlenermittlung, Abweichungsanalyse und die daraus resultierende Optimierung zu den bestimmenden Elementen des Prozesses [AmBr10, S.6].
Folgende Fragen können u.a. bei einer Web Analyse geklärt werden:

Wie viele Besucher hat meine Webseite und woher kommen diese Besucher (über Suchmaschine, über Werbe-Banner, usw.)? Wie verhalten sich die Besucher auf meiner Webseite, wie ist das Navigationsverhalten? Wo sind Schwachstellen auf der Webseite, bzw. wie gut ist die Leistung der Webseite (Durchsatz, Antwortzeit)? [DüRa11,S.681] [RaSt02, S.335]

In einem ersten Schritt muss man also zunächst die Ziele der Web-Anwendung ermitteln. Diese sollten aber auch schon während der Entwicklung der Webseite festgelegt werden, sodass die Erreichung der Ziele in der Betriebsphase nur noch überwacht und gesteuert werden muss. Aus diesem Grund wird auf die Zielfestlegung an dieser Stelle nicht näher eingegangen und wir betrachten im nächsten Schritt wie die benötigten Daten erfasst werden können.

2.1 Datenerfassung/Datensammlung für die Web Analyse

In dem Gesamtprozess der Web Analyse stehen für die Datenerfassung mehrere Verfahren zur Verfügung. Grob unterscheidet man zwischen reaktiven Methoden, bei denen sich der Besucher aktiv beteiligen muss und nicht-reaktiven Methoden, bei denen der Benutzer gar nicht merkt, dass Informationen über seinen Webseitenabruf gespeichert werden. Im Folgenden wird zuerst auf die zuletzt genannten Methoden eingegangen. Die zwei wichtigsten Verfahren unter den nicht-reaktiven Methoden, sind die Logfile-Analyse und die Zählpixel-Analyse (Tracking mit JavaScript).

Die klassische Lofile-Analyse ist ein Server-basiertes Verfahren, bei der die bereits vorliegenden Protokolldateien der Webserver ausgewertet werden. Konkret werden Informationen über jede Seiten-Anfrage (Request) und Antwort (Response) vom Webserver durch sogenannte Logfiles protokolliert. Dies geschieht indem für jeden Zugriff auf einen Web-Server, für jedes abgerufene Element standardmäßig Einträge in eine Logbuch-ähnliche Datei erzeugt werden. Die Einträge erfolgen Zeilenweise und enthalten bspw. folgende Informationen: [Ree08, S.197] [StKr09, S.94ff]

Bezeichnung	Beispiel
Aufrufdatum	30/Dec/2011
Aufrufzeit	08:33:24
Name des Services	W3SVC1
Server-Name	Server1
IP-Adresse des Servers	84.172.93.59
Methode	GET
URL der aufgerufenen Datei	/ecl/E-Commerce-Leitfaden.pdf
Port des Servers	80
IP-Adresse des Besuchers	92.168.1.1
Verwendetes Protokoll	HTTP/1.0
Browser- und Betriebssystem -Version des Besuchers	Mozilla/5.0+(Windows;+U;+Windows+NT+6.0;+de;+rv:1.8.1.11)+Gecko/20071127+Firefox/2.0.0.11
Zuvor besuchte Webseite (Referrer)	www.ecommerce-leitfaden.de/download
Return-Code des Servers (hier: erfolgreich)	200
Win32-Statuscode (hier: erfolgreich)	0
Abgerufene Datenmenge (in Bytes)	9480422
Zum Server übertragene Datenmenge (in Bytes)	132
Benötigte Zeit (in Millisekunden)	3171

Tabelle 1. Beispiel eines Server-Logfiles

Je nach Server-Einstellungen können auch andere Informationen gespeichert werden. Diese Logeinträge können dann später durch Verwendung eines Tools, wie bspw. Xlogan[1], Analog[2] oder W3perl[3] ausgewertet werden. [RaSt02] beschreibt demgegenüber Möglichkeiten, wie die Logeinträge datenbankbasiert ausgewertet werden können, um eine hohe Skalierbarkeit und Flexibilität der Auswertungen zu gewährleisten.

Der Vorteil bei diesem Verfahren liegt darin, dass diese Logfiles von den meisten Servern automatisch generiert werden und daher keine Änderungen an der Webseite erforderlich sind. [StKr09, S.94ff] [Ree08, S.198ff]

Nachteilig ist, dass man durch Logfiles nur gewisse Informationen sammeln kann, nämlich nur solche, die der Browser dem Server bei einer Anfrage übermittelt. Beispielsweise wird bei einem Abruf einer Flash-Applikation zwar der Abruf an sich erfasst, die Aktionen die innerhalb der Applikation durchgeführt werden, bleiben aber verborgen. Auf der anderen Seite werden aber auch Informationen gespeichert, die für eine Auswertung überflüssig sind und den Aufwand unnötig erhöhen. Auch AJAX-Inhalte sind mittels Logfiles schwierig zu analysieren. Problematisch bei Logfiles ist auch das Caching, da bei erneutem Laden einer gecachten Seite der Server gar keine Meldung erhält und das Ergebnis dadurch verfälscht wird (angeblich wird dadurch bis zu 10% des Traffics gar nicht gespeichert [Has10]). Außerdem können Ergebnisse auch dadurch verfälscht werden, dass automatisierte Seitenaufrufe von Robots nicht von echten Besuchern unterschieden werden können. [StKr09, S.94f] [Ree08, S.198f]

Die Zählpixel-Analyse, auch Pixel-Methode genannt, registriert die Abrufe eines Besuchers nicht am Server, sondern über den Client des Besuchers und zählt damit zu den Client-basierten Verfahren. Mit jedem Aufruf einer Webseite an einen Webserver wird gleichzeitig ein kleines 1x1 Pixel-Bild, von einem getrennten Tracking- oder Analyse-Server heruntergeladen. Dieses Pixel-Bild ist in einem HTML Element der Webseite eingebettet und sorgt dafür, dass Daten über den Aufruf an den Analyse-Server übertragen werden. Dadurch wird also allein der Seitenabruf notiert. Um mehr Informationen über den Besucher zu gewinnen wird dieses Verfahren in Kombination mit JavaScript Code verwendet. Hierbei handelt es sich um das heutzutage geläufigere Tracking mit JavaScript, bei welchem der Code der Webseite um ein paar Zeilen JavaScript Code ergänzt wird. Wird eine Webseite vom Webserver an den Client geschickt, so wird mit dem Code, der die Webseite anzeigt, gleichzeitig ein weiterer Code, nämlich der, der den Abruf erfasst, im Internet-Browser ausgeführt und an einen zweiten Server, dem sogenannten Tracking- oder Analyse-Server übermittelt. Hierbei können bspw. Informationen über die Bildschirmauflösung oder den verwendeten Techniken und Plugins übermittelt werden. Der größte Unterschied zu dem Logfile Verfahren besteht darin, dass bei dieser Methode die technische Bereitstellung der Webseite (Webserver) von dem Tracking der Besucher (Tracking – oder Analyseserver) getrennt wird. [Ree08, S.197] [StKr09, S.98ff]

[1] Nähere Informationen unter `http://www.axeuk.com/xlogan/`
[2] Nähere Informationen unter `http://www.analog.cx/`
[3] Nähere Informationen unter `http://www.w3perl.com/`

Vorteilhaft bei diesem Verfahren ist, dass die Cache-Problematik vermieden wird und daher sichergestellt werden kann, dass die Seite auch tatsächlich immer vom Server abgerufen wird. Der technische Aufwand ist im Vergleich zum Logfile Verfahren kleiner, da nicht so viele unnötige Daten gespeichert und aussortiert werden müssen. [Ree08, S.197] [StKr09, S.98ff]

Problematisch ist dahingegen, dass die gewonnenen Daten in einem nicht standardisierten Format gespeichert werden, wodurch eine Analyse der Daten nur über das zuvor ausgewählte System durchgeführt werden kann. [Ree08, S.197] [StKr09, S.98ff]

Zu den reaktiven Methoden, gehören all diejenigen Methoden, die eine aktive Beteiligung des Nutzers voraussetzen und Informationen über Interaktionen zwischen dem Nutzer und der Webseite beinhalten. Dazu gehören bspw. Onlinebefragungen über Web-Formulare oder auch E-Mails der Besucher der Webseite. Das Problem bei diesen Methoden ist, dass die Realisierung oft sehr aufwendig ist, da Webseiten-Besucher nur ungern Zeit aufwenden um irgendwelche Formulare auszufüllen oder E-Mails zu schreiben. [EbWe04] Seit einigen Jahren gehört auch das „Social Engagement“ zu den reaktiven Methoden. Hierbei können Nutzer Inhalte durch Ratings, facebook Likes, Google+ oder twitter-tweets bewerten. [Has10] schlägt vor auf Basis dieser Inforationen ein „Social Egagement Score“ zu ermitteln, das ein Indiz für das Interesse der Besucher darstellt.

In einem nächsten Schritt müssen aus den erfassten und gespeicherten Daten Kennzahlen abgelesen werden, aus denen man letztendlich Rückschlüsse über die Besucher der Webseite und deren Verhalten ziehen kann.

2.2 Kennzahlen und Instrumente zur Analyse von Besuchereigenschaften und Besucherverhalten

Die aus den Daten abgelesenen Kennzahlen bauen in gewisser Weise aufeinander auf. Auf der untersten Ebene befindet sich die Kennzahl „**Hits**“, welche aus den einzelnen Zeilen einer Logdatei gewonnen und dadurch jedes einzelne Element einer Webseite gezählt wird. Inhaltlich gesehen ist die Kennzahl nicht besonders aussagekräftig. Auf der nächsten Ebene findet sich die Kennzahl „**Page Views**“, oder auch „Page Impressions“ genannt, zu Deutsch Seitenabrufe. Diese lässt sich sehr einfach aus dem Zählpixel Verfahren ablesen, da jeder Aufruf des 1x1 Pixel Bildes einem Seitenabruf entspricht. Die Kennzahl auf der nächsten Ebene ist die Kennzahl „**Visits**“ oder auf Deutsch „Besuche“. Hierbei werden zusammenhängende Seitenabrufe ermittelt und als ein „Visit“ definiert. Je nach Einstellung endet ein Visit nach bspw. 30 oder 20 minütiger Inaktivität des Clients nach einem Seitenabruf. Die Kennzahl der obersten Ebene ist die Kennzahl „**Visitors**“, zu Deutsch Besucher. Diese Kennzahl kann auf verschiedenen Wegen ermittelt werden: Zum einen kann die IP-Adresse verwendet werden, was sehr problematisch erscheint, da die IP-Adresse nicht genau einen Besucher, also einen Menschen identifiziert. Zum anderen können auch Cookies verwendet werden, die, sofern keine Deaktivierung von Cookies

vorgenommen wurde, auf dem Rechner des Besuchers gespeichert werden. [Ree08, S.197ff] [EbWe04, S.203] [Has10, S.87ff]

Im E-Commerce Umfeld werden auch komplexere Kennzahlen betrachtet, wie bspw. Konversionsraten oder Klicktiefen, für die teilweise auch unternehmensinterne Daten über Kunden einbezogen werden. Die Konversionsrate bezeichnet die Umwandlung eines Besuchers in einen Newsletter-Abonnenten oder in einen Kunden. Die Klicktiefe gibt an wie viele Seiten ein Besucher durchschnittlich während seines Besuches aufgerufen hat. Neben den genannten Kennzahlen gibt es noch viele andere, für weiterführende Informationen siehe [Ree08, S.37f] und [Has10].

Wichtige Hilfsmittel zur Analyse des Besucherverhaltens sind nicht nur Kennzahlen, sondern bspw. auch folgende Instrumente:

Instrument	Beschreibung
Seitenaufrufstatistik	Gibt Auskunft wie häufig einzelne Seiten einer Web-Anwendung pro Tag aufgerufen wurden
Klickpfadstatistik	Gibt Auskunft über die Wege der Besucher. Zu einer bestimmten Seite werden Vorgänger- und Nachfolgeseiten dargestellt.
Clickmap	Stellt, wie eine Folie über die Webseite dar, welche Links auf dieser geklickt wurden
Heatmap	Stellt ebenfalls dar, wo geklickt wurde, allerdings zeigt es auch Klicks auf nicht verlinkte Bereiche

Tabelle 2. Instrumente in der Web-Analyse

Geht aus den ermittelten Kennzahlen hervor, dass gewisse Ziele der Web-Anwendung nicht oder nur ungenügend erreicht wurden, so ergibt sich ein gewisser Optimierungsbedarf. Mithilfe der genannten Kennzahlen und Instrumente können dann Muster oder Hinweise abgelesen werden, die zur Herleitung eines Optimierungsansatzes eingesetzt werden können. Wichtig ist hierbei, dass das Besucherverhalten nicht nur beobachtet wird, sondern, dass auf erkannte Schwachstellen reagiert wird. Um eine stetige Steigerung der Effektivität der Web-Anwendung zu gewährleisten, genügt es nicht nur Daten zu erfassen und das Benutzerverhalten und Benutzereigenschaften zu analysieren, es müssen auch konkrete Optimierungsansätze erarbeitet werden. Hierzu nähere Informationen auch in [Ree08, S.137f] und in [Has10, S. 373].

2.3 Unterstützung der Web Analyse durch den Einsatz von Web-Analyse-Tools

Um den ganzen Prozess der Web-Analyse zu unterstützen, können unterschiedliche Tools und Programme eingesetzt werden. Unter den bekanntesten Tools gehören u.a. Google Analytics, etracker, Site Catalyst (Omniture), Webtrekk, usw. Je nach Autor, werden in der Literatur unterschiedliche Ansätze, zur Auswahl eines Systems zur Unterstützung der Web-Analyse, vertreten. [Ree08] nimmt bspw. bestimmte Kriterien

zur Hand, [Has10] betont, dass es vor allem wichtig ist zu wissen, was man mit der eigenen Webseite bezwecken will und [AmBr10] empfiehlt die Nutzung von „Use Cases“ um für konkrete Anwendungsfälle auch konkrete Lösungen zu finden.

In der Praxis gibt es den Trend, dass größere Webseiten sich nicht mehr ausschließlich auf das kostenlose Google Analytics System verlassen, sondern vermehrt auch zusätzlich andere Systeme einsetzen [IdOb11]. Dies könnte ein Anzeichen dafür sein, dass Top-Webseiten-Betreiber verstärkt nach professionellen Lösungen Ausschau halten und diesen Bereich für wichtiger erachten, als noch vor einigen Jahren.

Letztendlich muss jeder für sich entscheiden, wie er bei der Auswahl eines Web-Analyse-Tools vorgeht. Wichtig ist auf jeden Fall das bei der Auswahl eines Systems auf die Bedürfnisse, Anforderungen und technischen Voraussetzungen der Web-Anwendung bzw. des dahinterstehenden Unternehmens eingegangen wird.

3 Aktualisierung der Inhalte einer Web-Anwendung

Warum sollte man sich in der Phase Betrieb und Wartung mit dem Inhalt einer Web-Anwendung beschäftigen? Die Antwort auf diese Frage ist ganz klar: Für den Besucher einer Webseite, besteht diese lediglich aus Inhalt. Ist der Inhalt veraltet oder von schlechter Qualität, so wird die Webseite für die Nutzer uninteressant und die Konkurrenzseite ist nur wenige Klicks entfernt. Man kann also sagen, dass der Inhalt einer Web-Anwendung das „Aushängeschild“ nach außen darstellt und daher auch vor allem in der Phase Betrieb und Wartung der Pflege bedarf. Für den Begriff „Inhalt“ wird im Folgenden der englische Begriff „Content“ verwendet.

3.1 Content-Pflege

Die Pflege des Contents einer Web-Anwendung kann zum einen der **Kundengewinnung** dienen und dazu führen, dass aktuelle Inhalte das Interesse der Nutzer weckt und der Anreiz die Webseite aufzurufen erhöht wird. Zum anderen kann die Pflege des Contents auch zur **Kundenbindung** dienen, indem der Anreiz für den regelmäßigen Besuch der Webseite erhöht oder bspw. auch die Verweildauer der Benutzer verlängert wird. Bevor wir uns weiter mit der Content-Pflege beschäftigen, sollten wir den Begriff „Content“ näher spezifizieren. [EbWe04, S.198]

Der Content besteht aus unterschiedlichen Assets, wobei diese folgendermaßen klassifiziert werden können [ZTZ01, S.40f]:

Übliche Webinhalte	Texte, Bilder, Links, …
Multimediale Assets	Streamingformate für Audio und Video, Flash
Applikationsgebundene Assets	Dokumente die an Appklikationen (wie bspw Word) gebunden sind
Transaktionelle Assets	Führen Informationen über Transaktionen mit sich
Community Assets	Inhalte die von Nutzern selbst erstellt werden

Tabelle 2. Arten von Assets

Der Begriff Content beinhaltet also viel mehr als nur Text und ist ein Sammelbegriff für die unterschiedlichsten Assets. Schließlich fordert diese Variation an Assets nicht nur unterschiedliche Werkzeuge zur Erstellung und Bearbeitung, sondern verlangt auch nach unterschiedlichen Änderungsfrequenzen. Content kann demnach je nach Änderungsbedarf auch folgendermaßen klassifiziert werden:

Statische Informationen	Werden nach ihrer Erstellung nicht mehr verändert, wie bspw. Aufzeichnungen historischer Ereignisse
Dynamische Informationen	Werden in regelmäßigen oder auch unregelmäßigen Abständen aktualisiert, wie bspw. Verfügbarkeitsmeldungen
Semidynamische Informationen	Sind eine Mischform aus statischen und dynamischen Inhalten, wie bspw. Ansprechpartner eines Unternehmens

Tabelle 3. Klassifikation von Assets in drei Informationstypen

Je nach Klassifizierung des Contents sind unterschiedliche Update-Zyklen erforderlich [ZTZ01, S.42f].

Als nächstes stellt sich die Frage wie der Content auf die Webseite kommt, wie also der Prozess zur Veröffentlichung des Contents genau abläuft. Der Content durchläuft vor seiner eigentlichen Veröffentlichung mehrere Schritte, von der Erstellung zur Überprüfung der Inhalte, bis hin zur Freigabe und letztendlich Veröffentlichung der Inhalte. An diesem Prozess ist eine Vielzahl von Mitarbeitern mit unterschiedlichem Know-How beteiligt. Für die Texte sind bspw. Autoren zuständig, die Illustrationen werden von Grafikern aufbereitet und der Webmaster ist für die technischen Dinge, wie bspw. der Administration des Webservers und anderen Applikationen verantwortlich. Im klassischen Sinne ist der Webmaster auch dafür Zuständig den ganzen Content in die Webseite einzubauen, da allein er über genügend technischem Know-How verfügt. Das bedeutet wiederum, dass der Webmaster in diesem Prozess zum Bottleneck wird und der ganze Content nur über ihn veröffentlicht werden kann [ZTZ01, S.54ff].

Lösungsansätze für dieses Problem waren anfangs Editoren mit WYSIWYG-Funktionalität oder auch der Ansatz die Inhalte in einer Datenbank zu speichern. Für kleinere Webauftritte sind diese Lösungen sicherlich angebracht, aber mit steigender Komplexität der Web-Anwendung ist der Einsatz eines Web Content Management Systems (WCMS) unerlässlich. [ZTZ01, S.57] [KLJ10, S.628]

3.2 Einsatz eines WCMS und daraus resultierender Nutzen

Um das, im vorherigen Abschnitt erläuterte Problem zu lösen, wird ein Web Content Management System (WCMS) eingesetzt. Was ein WCMS genau sein soll, macht allein schon der Begriff deutlich: **Web** beinhaltet das Internet, das Intranet und das Extranet. Als **Content** versteht man alle veröffentlichten „Assets“, wie u.a. Texte, Bilder, Audio oder Video. Unter **Management** fallen alle Tätigkeiten im Zusammenhang mit dem Content, wie bspw. die Erstellung, Bearbeitung,

Verwaltung, Publikation und Archivierung. Das **System** ist die technische Grundlage, die das Management des Contents realisiert. [ZTZ02, S. 69f]

Letztendlich versucht also ein WCMS den Prozess zur Veröffentlichung von Content weitestgehend zu automatisieren, sodass das WCMS den Webmaster weitestgehend ersetzt und der Webmaster nur noch für die Administration der Technik zuständig ist. Die Autoren und Grafiker übergeben den Content an das System, das die Inhalte schließlich über Vorlagen in die Webseite einbettet.

Ein WCMS sorgt dafür, dass Inhalt und Programmierung getrennt werden. Dabei werden die Inhalte separat gespeichert und die Darstellung der Inhalte auf der Webseite wird über Vorlagen generiert. Ein WCMS berücksichtigt außerdem die unterschiedlichen Rollen der Mitarbeiter. Die Mitarbeiter werden also entsprechend ihrer Kompetenzen in den Prozess eingebunden, d.h. die Autoren sorgen für den Inhalt, die Grafiker sorgen für die Darstellung und die Techniker können sich ausschließlich um die Administration des Systems kümmern. Desweiteren unterstützt das WCMS den Workflow in dem Prozess der Veröffentlichung von Content. Man gibt also an, wie und in welcher Abfolge bestimmte Arbeiten durchzuführen sind. Das WCMS führt zudem zu kürzeren Time-to-Web Zeiten, dadurch das Inhalte direkt vom Schreibtisch der Redakteure/Autoren ins Web kommen. [ZTZ02, S.59ff] [StMa03, S.51ff]

3.3 Trends bei WCMS

Die Weiterentwicklungen und Trends im Bereich der WCMS sind zahlreich. Vor allem heutzutage ist es für ein WCMS wichtig genügend Flexibilität zu bieten, um möglichst schnell auf Veränderungen reagieren zu können und möglichst vielen Bedürfnissen gerecht zu werden.

In den letzten Jahren und auch in Zukunft müssen WCMS verstärkt darauf ausgerichtet sein vor allem multimediale Inhalte aufzubereiten und für jeglichen Umgang mit diesen, Unterstützung zu bieten. [Har08]

Ein weiterer Trend liegt darin mehr und mehr „User Generated Content“ in die Web-Anwendung einzubinden, wodurch den Nutzern die Möglichkeit gegeben wird bspw. eigene Kommentare oder Bilder in die Webseite einzufügen. Das wiederum bedeutet, dass der Nutzer in den redaktionellen Prozess einbezogen werden muss und im WCMS neben den bestehenden Benutzergruppen, wie Redakteuren oder Administratoren auch eine eigene Benutzergruppe, nämlich die der „aktiven User“ eingestellt wird. [Har08]

Ein weiterer Aspekt, der bei WCMS in Zukunft eine große Rolle spielen wird ist der, dass Web Content für die unterschiedlichsten Endgeräte aufbereitet und angepasst werden muss. Nicht nur die Variation an Technik, wie bspw. verschiedene Displaygrößen, sondern auch die verschiedenen Anforderungen auf den unterschiedlichsten Geräten muss dabei beachtet werden. Beispielsweise will man auf einem Handy, schon allein aufgrund der Displaygröße eher kurze Informationen erhalten, wobei man vielleicht auf einem Tablet-PC schon eher über Hintergrundwissen informiert werden will. [Hei11]

Die Problematik, mit der WCMS in Zukunft verstärkt konfrontiert werden, ist die der steigenden Informationsflut. Die immer wachsende Menge an Informationen muss verwaltet und schnell und effizient extrahierbar sein, wobei die Überschaubarkeit der Datenbestände gewährleistet werden muss. Nach [GaMi09] könnte man diesen und anderen Herausforderungen von WCMS begegnen, indem man Semantic-Web-Technologien in WCMS nutzt. Dabei wird zum einen das Management von semantischem Content in Form von Metadaten oder Auszeichnungen realisiert und zum anderen erweiterte Funktionalitäten mithilfe semantischer Konzepte zur Verfügung gestellt, wie bspw. die Suche über Wissensmodelle. [GaMi09, S.213]

In der Praxis versucht man verstärkt Semantic Web in WCMS zu integrieren. Da ein Großteil der Inhalte von Webseiten durch WCMS verwaltet wird, kann die Integration von Semantic Web Technologien unter anderem eine ganz neue Perspektive für die Nutzung des Internets bieten. Die Suche nach bestimmten Informationen wird erheblich vereinfacht und die Ergebnisse können einzig und allein auf benötigte Inhalte fokussiert werden. Desweiteren können aber auch eigene Inhalte um externe Informationsquellen, ohne die Nutzung diverser Web Services, mit Hilfe der Sprache SPARQL erweitert werden. Vor allem kann durch den Einsatz von Semantic Web Technologien auch ein Austausch von Daten zwischen den Systemen erfolgen. In der Praxis setzen Systeme wie bspw. Drupal, TYPO3, Joomla und WordPress verstärkt auf semantische Unterstützung, wobei die neuen Semantic-Web-Technologien schrittweise eingebaut werden [RaWo11] [BeAu04] [KoTa05]

4 Bewerbung einer Web-Anwendung

Der Erfolg einer guten Webseite hängt nicht nur vom Konzept der Seite allein ab – es reicht nicht einfach nur eine gute Seite zu entwickeln - sondern auch von der Vermarktung der Webseite während der Phase „Betrieb und Wartung“. Die beste Seite hilft nichts, wenn sie bspw. von Google nicht indexiert wird und deshalb von Besuchern nicht gefunden werden kann. Themen wie Suchmaschinenmarketing, sind für den Erfolg ebenso wichtig wie gekonntes Platzieren von Werbung in Form von Affiliate-, Banner-, sowie Newsletter-Marketing. Im Folgenden sollen zunächst traditionelle Maßnahmen, sowie anschließend neuere Trends zur Bewerbung von Web-Anwendungen erläutert werden.

4.1 Traditionelle Maßnahmen zur Bewerbung von Web-Anwendungen

Eine altbekannte Maßnahme zur Bewerbung von Web-Anwendungen ist die Nutzung von E-Mails und Newsletter. Da die Nutzer direkt und persönlich angesprochen werden, ist das E-Mail Marketing eine Form des Direktmarketings. Der Vorteil liegt darin, dass gezielt eine Gruppe von Nutzern angesprochen werden kann und auch eine individuelle Ansprache möglich ist.

Die bekannteste Form des E-Mail Marketings ist der Newsletter. Hierbei kann sich der Besucher mit seiner E-Mail Adresse für den Erhalt eines regelmäßigen Newsletters eintragen und bekommt dann je nachdem, täglich, wöchentlich oder

monatlich die neuesten Nachrichten oder Produktempfehlungen. Diese Maßnahme führt im Idealfall dazu, dass mehr und mehr Besucher zur Webseite wiederkehren und an das Angebot der Web-Anwendung Interesse finden. Daneben gibt es auch die Möglichkeit bereits bestehende Kunden per E-Mail zu kontaktieren und über Produktangebote regelmäßig zu informieren. Hierfür sollte aber eine zuvor eingeholte Einwilligung bestehen, da viele Kunden solche Werbe-E-Mails als unangenehm empfinden. [DüRa11, S.99]

Auch juristische Aspekte müssen bei der Verwaltung der E-Mail Adressen berücksichtigt werden. So wird bspw. durch das Double-Opt-In-Verfahren verhindert, dass Nutzer ohne ihr Wissen, von Dritten für Newsletter Angebote angemeldet werden. Dies geschieht, indem eine sogenannte Bestätigungsmail an die eingetragene E-Mail Adresse versendet wird und erst nach Bestätigung eines Links in dieser E-Mail, der Newsletter regelmäßig versendet wird. [DüRa11, S.120]

Eine weitere klassische Maßnahme zur Bewerbung einer Web-Anwendung ist das Banner-Marketing. Die Parallele zu den Werbeanzeigen in Zeitungen und Magazinen ist klar erkennbar, ein wichtiger Unterschied liegt aber darin, dass die Online-Version dieser Maßnahme um einiges günstiger und flexibler ist. Das Ziel eines Werbebanners ist u.a. potenzielle Besucher zunächst auf die Inhalte der entsprechenden Webseite aufmerksam zu machen und anschließend auf die Webseite zu leiten. Die meisten Banner verlinken daher auf die Webseite oder auf bestimmte Produktangebote des Werbenden. Ein weiteres Ziel der Bannerwerbung kann die „Brand Awareness“ sein. Hierbei wird versucht eine bestimmte Marke oder ein Image in der Öffentlichkeit bekannt zu machen. Ob der Einsatz von Bannerwerbung tatsächlich die genannten Ziele erreicht, ist umstritten. Fest steht, dass Banner dennoch sehr oft als Werbemaßnahme eingesetzt werden und sich großer Beliebtheit erfreuen [StKr09, S.5] [Sch11]. Entscheidet man sich für diese Werbeform, so kann man die unterschiedlichsten Arten von Bannern einsetzen. Es gibt u.a. die dezenten statischen Banner, die auffälligeren animierten Banner, die Rich-Media-Banner die Bild, Ton und Interaktivität miteinander kombinieren oder auch die Pop-Up-Banner, die auf einer Webseite teilweise nicht zu übersehen sind. [DüRa11, S.63]

Affiliate Marketing ist ebenfalls eine beliebte Form des Online-Marketings [StKr09, S.5]. Die Funktionsweise ist relativ einfach: Ein sogenannter „Affiliate“ veröffentlicht auf seiner Webseite die Werbung des Werbetreibenden, des sogenannten „Advertisers“. Demnach kann der Advertiser im Vorfeld thematisch passende private Webseiten auswählen und seine Dienstleistungen oder Produkte auf diesen veröffentlichen. Der Affiliate auf der anderen Seite bietet seinen Besuchern durch die thematisch passenden Anzeigen womöglich sogar einen Mehrwert und verdient nebenher auch noch Geld damit. Der Advertiser bezahlt dem Affiliate nämlich eine Provision, sobald der Nutzer auf die gewünschte Seite kommt oder, je nach Vergütungsmodell, sobald der Nutzer eine bestimmte Aktion auf der entsprechenden Seite durchführt. Zwischen den beiden Parteien, Advertiser und Affiliate wird normalerweise ein Partner-Netzwerk eingeschaltet, welches die Organisation, also die regulatorischen Angelegenheiten übernimmt. Der Vorteil einer

solchen Maßnahme liegt auf der Hand: Potenzielle Kunden können viel gezielter angesprochen werden und die Bezahlung erfolgt nur nach erfolgreicher Vermarktung. [DüRa11, S.80]

Suchmaschinenmarketing wird bei vielen als diejenige Maßnahme angesehen, bei der das Kosten-Umsatz Verhältnis das Beste ist [StKr09, S.70]. Suchmaschinenmarketing besteht zum einen aus Suchmaschinenoptimierung und zum anderen aus Suchmaschinenwerbung [DüRa11, S.294].

Bei der Suchmaschinenoptimierung versucht man eine Webseite möglichst auf die vorderen Ränge der Ergebnisliste einer Suchmaschine zu platzieren. Hierfür ist zunächst die Analyse der relevanten Suchbegriffe entscheidend und anschließend wird versucht das Website-Ranking zu verbessern. Die erste wichtige Säule bei der Suchmaschinenoptimierung ist der suchmaschinenfreundliche Aufbau der Webseite:

Am Anfang steht der Domain-Name, dieser sollte möglichst kurz, präzise und aussagekräftig sein. Als nächstes sollte man seine Webseite bei gängigen Suchmaschinen registrieren, so verlässt man sich nicht allein darauf, dass die Suchmaschinen die Webseite hoffentlich irgendwann einmal finden wird. Der nächste wichtige Punkt ist eine klare Informationsarchitektur der Web-Anwendung; Eine klare Navigationsstruktur ist nicht nur für die Besucher, sondern auch für Suchmaschinen wichtig. Diese sollte aber schon bei der Entwicklung feststehen und in der Phase Betrieb und Wartung nur noch ggf. ergänzt werden. Bei der Suchmaschinenoptimierung ist es außerdem wichtig die entsprechende Webseite den Suchmaschinen-Crawlern zugänglich zu machen, also die technischen Voraussetzungen zu schaffen. An dieser Stelle ist es auch hilfreich einen sauberen, validen und gültigen HTML-Quellcode vorliegen zu haben. Außerdem ist zu beachten, dass durch den Einsatz von Technologien, wie AJAX oder Flash verhindert wird, dass der Crawler die Inhalte der Webseite liest und auswertet. In der Praxis kann man bspw. durch die Google Webmaster-Tools herausfinden, wie gut eine Webseite gecrawlt wird. Desweitern kann man an vielen anderen Stellen eine Webseite für eine Suchmaschine entsprechend optimieren, wie bspw. die Optimierung der Meta-Angaben. Hierzu auch nähere Informationen unter [DüRa11, S.415ff].

Die zweite Säule bei der Suchmaschinenoptimierung ist die Verlinkung im Internet zu der entsprechenden Webseite. Externe Verlinkungen zu der eigenen Webseite können den Rang der betrachteten Webseite in der Suchergebnisliste beeinflussen. Hierbei ist nicht nur die Anzahl der Verlinkungen, sondern auch die Qualität/Größe derjenigen Webseite, auf der sich der Link befindet, zu beachten. An dieser Stelle werden Kennzahlen, wie bspw. die Linkpopularität, die Host-Popularität oder die Domain-Popularität analysiert. [DüRa11, S.415ff]

Bei der Suchmaschinenwerbung erscheint die Anzeige der Webseite nach Eingabe entsprechender Suchbegriffe im Anzeigebereich der Suchmaschine, oberhalb oder neben der Ergebnisliste. Der Vorteil liegt darin, dass nur, zum Suchbegriff passende Werbeanzeigen präsentiert werden. Webseiten-Betreiber können bspw. bei Google AdWords kostenpflichtige Anzeigen, sowohl auf der Seite mit den Google-Suchergebnissen, als auch auf anderen Seiten schalten. [DüRa11, S.293ff]

4.2 Neue Trends zur Bewerbung von Web-Anwendungen

Bei Social-Media-Marketing nutzt man soziale Netzwerke, um den direkten Dialog zu potenziellen Kunden zu suchen. Ein Vorteil liegt darin, dass relativ schnell ein breites Publikum erreicht werden kann. Doch auch diese Maßnahme muss gut durchdacht werden. Als erstes stellt sich die Frage, ob mit Social-Media-Marketing überhaupt die erwünschte Zielgruppe erreicht wird. Desweiteren besteht die Gefahr, dass die entsprechenden Aktivitäten bei der Zielgruppe nicht den erwünschten Effekt haben. Daher sollten nicht nur Kennzahlen festgelegt werden, die die Zielerreichung messen, sondern es sollte auch eine sorgfältige Strategie ausgearbeitet werden. Schließlich wird ersichtlich, dass Social-Media-Marketing nicht so leicht umzusetzen ist, wie vielleicht anfangs vermutet, sondern häufig mehr Zeit und dadurch auch Kosten in Anspruch nimmt. Bei erfolgreicher Umsetzung, kann dennoch große Aufmerksamkeit erlangt und ein breites Publikum erreicht werden.

Gegen Ende 2011 verzeichnete die Bundesnetzagentur erstmals mehr als 112 Millionen Mobilfunkanschlüsse in Deutschland – ein neuer Rekord [Bund11, S.50]. Der Ausbau der Bandbreitnetze in Deutschland und die Einführung von neuen Technologien wie bspw. LTE, sorgen für die technologischen Voraussetzungen für den Gebrauch dieser Mobilfunkanschlüsse [Bund11, S.3]. Heutzutage wollen Nutzer die Webseiten nicht nur im Browser betrachten sondern auch auf Smartphones, Tablet-PCs und anderen mobilen Endgeräten. Hier gilt es das Webangebot bspw. auf unterschiedliche Displaygrößen anzupassen, um den Informationsbedarf der Nutzer entsprechend zu befriedigen.

Im Mobile Marketing geht es grundsätzlich darum auf den mobilen Endgeräten präsent zu sein. Zum einen gelingt dies durch die Bereitstellung einer mobilen Version der Webseite oder durch eine mobile Anwendung, einer sogenannten App. Zum anderen kann man auch gezielt mobile Online-Werbung einsetzen. Hier kann man bspw. bei Google durch das Werbeprogramm „AdWords“ mobile Anzeigen schalten. Eine amerikanische Studie von MediaMind Research [MeMi11, S.10] über mobile Werbeanzeigen zeigte, dass Nutzer eher bereit sind Werbebanner auf dem Mobiltelefon anzuklicken als browserbasierte Standardbanner. Die Gründe hierfür können nicht eindeutig identifiziert werden, es gibt Vermutungen, dass neuere Formen der Online-Werbung anfangs besser bei den Nutzern ankommen als die älteren. Ein weiterer Punkt der im Kontext des Mobile- Marketings eingesetzt werden kann, ist die mobile und lokale Suche, bei der mittels GPS-Daten den Nutzern lokal passende Inhalte angeboten werden können. [DüRa11, S.189ff] Nähere Informationen über Mobile Marketing gibt es auf der Seite der Mobile Marketing Association[4].

Gemäß dem Social Media Marketing Report 2011 [Ste11] liegt Video-Marketing nach wie vor im Trend. Im Jahr 2011 planten 77% der Marketing-Abteilungen ihre Aktivitäten im Bereich Videomarketing zu steigern [Ste11, S.4]. Doch was genau ist Videomarketing. [DüRa11, S.167ff] versteht darunter das „Präsentieren von Botschaften per Video auf der eignen Website oder anderen Internetpräsenzen“. Der

[4] http://www.mmaglobal.com/

Vorteil von Videomarketing liegt darin, dass über Videos mehr vermittelt werden kann als allein über Text, Bilder oder Audio. Es können Emotionen und klare Markenbotschaften transportiert und neben einem bestimmten Image auch Kompetenz und Innovation vermittelt werden. All dies ist wichtig um den Besucher zu fesseln und ihn für das Angebot zu begeistern. Nach Erstellung eines Videos, kann dieses entweder auf der eigenen Webseite oder auf den zahlreichen Videoportalen, wie bspw. YouTube veröffentlicht werden. Bei Video-Ads hat man außerdem die Möglichkeit das eigene Werbe-Video in der Umgebung anderer Videos oder innerhalb von Videos zu platzieren. Hierfür kann man bspw. das YouTube-Konto mit dem AdWords-Konto verknüpfen und es fallen nur dann Gebühren an, wenn ein Nutzer auf das entsprechende Video klickt. [DüRa11, S.165ff]

5 Fazit und Ausblick

Zusammenfassend ist zu sagen, dass in allen drei Bereichen, Web-Analyse, Inhaltspflege und Bewerbung einer Web-Anwendung, Weiterentwicklungen stark voranschreiten: Im Bereich der Web-Analyse werden bspw. verstärkt Profilösungen nachgefragt. Bei der Auswahl eines Web Content Management Systems müssen mehr und mehr Anforderungen berücksichtigt werden, wie bspw. die Verbreitung multimedialer Inhalte, die Nutzung mobiler Endgeräte, sowie der Trend zu nutzergeneriertem Content. In dem Bereich der WCMS versucht man einige Herausforderungen, wie bspw. die der wachsenden Informationsflut, durch Einbezug von Semantic-Web-Technologien zu begegnen. In der Praxis haben schon einige bekannte Anbieter von WCMS semantische Unterstützung in ihre Systeme eingebaut, wodurch der gängige Fortschritt in diesem Bereich ersichtlich wird. Auch bei der Bewerbung einer Web-Anwendung sind einige Trends erkennbar, wie u.a. Social-Media-Marketing, Mobile-Marketing und Video-Marketing. Ersichtlich wird, dass das Marketing auf das Benutzerverhalten ausgerichtet wird.

References

[AmBr10] Amthor, A., Brommund, T.: Mehr Erfolg durch Web Analytics: Ein Leitfaden für Marketer und Entscheider. Hanser Verlag, München (2010)

[BeAu04] Beck, N., Auer, S.: Semantic Web Content Management. Website (2004) Online unter http://powl.sourceforge.net/swc/semweb_cms.pdf; besucht: März 2012.

[Bund11] Bundesnetzagentur: Tätigkeitsbericht 2010/2011 Telekommunikation Website. (2011) Online unter http://www.bundesnetzagentur.de/SharedDocs/Downloads/DE/BNetzA/Presse/Berichte/2011/TaetigkeitsberichtTK20102011pdf.pdf?__blob=publicationFile; besucht: März 2012.

[DüRa11] Düweke, E., Rabsch, S.: Erfolgreiche Websites. Galileo Computing, Bonn (2011)

[EbWe04] Ebner, A. Werthner, H.: Betrieb und Wartung von Web-Anwendungen In: Web Engineering, Kappel G. Pröll B. (Hrsg.). dpunkt.verlag, Heidelberg (2004)

[Etr11] etracker GmbH: Transparenz im Begriffs-Dschungel: Web-Controlling & Co. Website (16.05.2011) Online unter http://www.marketing-boerse.de/

Fachartikel/details/etracker-schafft-Transparenz-im-Begriffs-Dschungel-Web-Controlling--Co; besucht: März 2012.

[GaMi09] Gams, E., Mitterdorfer, D.: Semantische Content Management Systeme. Springer Verlag, Heidelberg (2009)

[Har08] Hartmann, S.: Zukunftsorientierte Redaktionssysteme. Website (2008) Online unter http://www.contentmanager.de/magazin/zukunftsorientierte_redaktionssysteme.html; besucht: März 2012.

[Has10] Hassler, M.: WebAnalytics. mitp, Heidelberg (2010)

[Hei11] Heise, D.: Web Content Management in 2012: Many Evolutions, but no Revolutions. Website (2011) Online unter http://www.cmswire.com/cms/customer-experience /web-content-management-in-2012-many-evolutions-but-no-revolutions-013912.php; besucht: März 2012.

[IdOb11] IdealObserver: Trends im Webanalyse-Markt: Google dominiert im Massenmarkt, Top-Portale wechseln zu Profilösungen. Website (08.06.2011) Online unter http://www.idealobserver.com/news/36-idealobserver/257-trends-im-webanalyse-markt-google-dominiert-im-massen markt-top-portale-wechseln-zu-profiloesungen; besucht: März 2012.

[KLJ10] Laudon, K.C., Laudon, J.P., Schoder, D.: Wirtschaftsinformatik – Eine Einführung. Pearson, München (2010)

[KoTa05] Koller, A., Pellegrini, T.: Content Management Systeme auf dem Weg zum Semantic Web. Website (17.05.2005) Online unter http://www.community-of-knowledge.de/fileadmin/user_upload/attachments/CMS_SemanticWeb.pdf; besucht: März 2012.

[MeMi11] MediaMind: Der umfassende Report für effektive Online-Werbebotschaften von Telekommunikationsunternehmen. Website (2011) Online unter http://www.mediamind.com/Data/Uploads/ResourceLibrary/MediaMind_Telecom_2011_DE.pdf?utm_source=PR%2BGermany&utm_medium=PR&utm_campaign=PR%2BGermany; besucht: März 2012.

[RaSt02] Erhard Rahm, Thomas Stöhr: Data Warehouse-Einsatz zur Web-Zugriffsanalyse in: Web & Datenbanken – Konzepte, Architekturen, Anwendungen, S. 335. Dpunkt Verlag, Heidelberg (2002)

[RaWo11] Rau, J., Wolf, L.:Semantic-Web-Technologien in Content Management Systemen nutzen: Systeme mit Zukunft. Website (26.09.2011) Online unter http://t3n.de/magazin/semantic-web-technologien-content-management-systemen-226333/2/; besucht: März 2012.

[Ree08] Reese, F.: Web Analytics - Damit aus Traffic Umsatz wird. BusinessVillage, Göttingen (2008)

[Sch11] Schmitt, C.: Banner-Werbung ist trotz minimaler Anzeigenklicks wirkungsvoll. Website (26.01.2011) Online unter http://www.media-treff.de/index.php/2011/01/26/banner-werbung-ist-trotz-minimaler-anzeigenklicks-wirkungsvoll/; besucht: März 2012.

[Ste11] Stelzner, M.A.: 2011 Social Media Marketing Industry Report –Website (2011) Online unter http://www.socialmediaexaminer.com/SocialMediaMarketingReport2011.pdf; besucht: März 2012.

[StKr09] Stahl, E., Krabichler, T. Breitschaft, M., Wittmann, G.: E-Commerce-Leitfaden. ibi research, Regensburg (2009)

[StMa03] Stahl, F., Maass, W.: Content Management Handbuch. NetAcademy Press, St. Gallen (2003)

[ZTZ02] Zschau, O., Traub, D., Zahradka, R.: Web Content Management. Galileo Business, Bonn (2002)

Testen von Web-Anwendungen

Michael Hamatschek

Otto-Friedrich-Universität Bamberg, Lehrstuhl für Medieninformatik
michael-robert.hamatschek@stud.uni-bamberg.de

Zusammenfassung Diese Arbeit beschäftigt sich mit dem Testen von Web-Anwendungen. Nach einer kurzen Einleitung werden in Kapitel 2 die Grundlagen des Testens behandelt, hier liegt der Blick auf der Problemstellung, den Testanforderungen und verschiedener Testarten. In Kapitel 3 werden die Unterschiede zwischen der Entwicklung von Desktop-Anwendungen und Web-Anwendungen beschrieben. Die agile Entwicklung von Anwendungen mit einem Schwerpunkt auf Test-Driven Development wird in Kapitel 4 erläutert. Kapitel 5 stellt kurz zwei Forschungsansätze vor und ein Fazit in Kapitel 5 schließt die Arbeit ab.

Schlagworte: Testen, Test-Driven Development, Agile Entwicklung, Web-Anwendungen

1 Einleitung

Aufgrund der immer noch steigenden Verbreitung von internetfähigen Endgeräten, sowie wie der verstärkte Wechsel von Desktop-Anwendungen hin zu Web-Anwendungen steigt der Bedarf an qualitativ hochwertigen Web-Anwendungen. Um die Qualität dieser Anwendungen zu gewährleisten müssen Vorkehrungen getroffen werden, diese zu überprüfen. Die Sicherstellung der gewünschten Funktionalität kann durch die Einführung von Tests in den Entwicklungsprozess umgesetzt werden. Wichtig ist es Tests so früh wie möglich durchzuführen, denn zu einem späteren Zeitpunkt in der Entwicklung kann es bereits nicht mehr effektiv sein Tests einzuführen.

"If test efforts soar, you are probably testing too late." Stober [1, 37]

Um den notwendigen Einsatz von Testverfahren in der Entwicklung zu verdeutlichen zeigt Abbildung 1 die relativen Kosten Fehler im Produkt zu korrigieren, die im Laufe der Anwendungsentwicklung extrem ansteigen.

Ziel ist es Tests bereits zu bei Entwicklungsbeginn zu integrieren und einen hohen Automatisierungsgrad zu erreichen. Wie dieses Ziel umgesetzt werden kann, soll in dieser Arbeit erläutert werden.

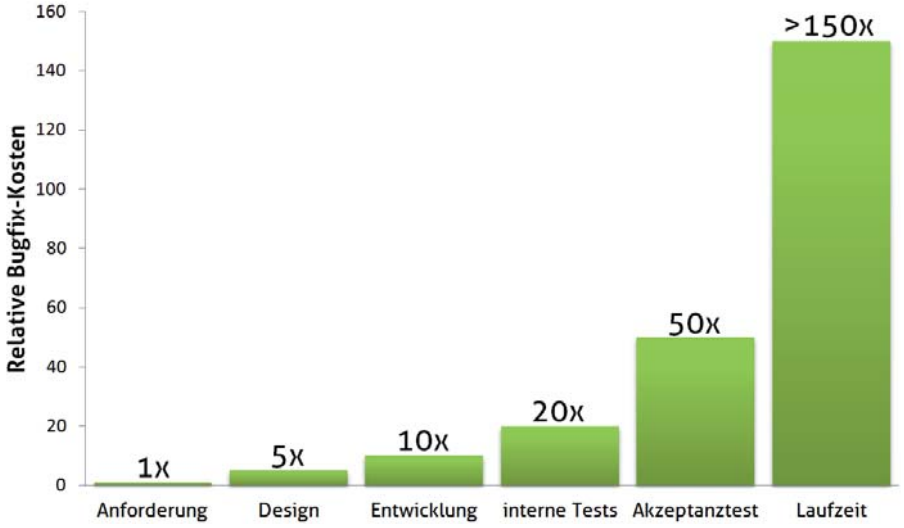


Abbildung 1. Relative Bugfixkosten nach [2]

2 Testgrundlagen

Generell stellt sich die Frage: "Warum müssen Produkte getestet werden, was muss getestet werden, wer testet und wie wird getestet?" Auf diese Fragestellung soll im weiteren Verlauf eingegangen werden.

2.1 Problemstellung

In Kappel et al. [3] wird Testen als Vorgehen zur Bestimmung der Qualität eines Produkts (in unserem Fall speziell Web-Anwendungen) betrachtet und dient in erster Linie der Qualitätssicherung. Balzert [4] unterscheidet bei der Entwicklung einer Anwendung das konstruktive und das analytische Ziel. Konstruktiv bedeutet ein fehlerfreies Programm zu entwickeln, Analytisch hingegen die Fehlerfreiheit eines Programmes nachzuweisen. Ziel eines Tests ist es fehlerhaftes Verhalten des Produkts zu entdecken und dieses im Anschluss zu beheben [3] [4]. Als Fehler wird alles bezeichnet was nicht dem erwarteten Verhalten der Anwendung entspricht oder nicht in der Anforderungsbeschreibung enthalten ist. Für die Testentwicklung und Durchführung ist es daher erforderlich, dass die Anforderungsbeschreibung vollständig und für den Entwickler verfügbar ist [3]. In der Praxis trifft das meist nicht zu, da die Anforderungen eines Produkts häufig ohne Einbindung der Entwickler geändert werden bzw. diese nicht oder zu spät über die Änderungen in Kenntnis gesetzt werden. Ein weiteres Problem ist, dass Stakeholder oft unterschiedliche Ansichten vertreten, durch die Entwicklung erschwert oder verzögert wird.

2.2 Anforderungen

Die Anforderungen an ein Produkt sind so vielfältig wie deren Nutzer. Häufig genannte Aspekte die ein Produkt erfüllen muss sind Verlässlichkeit, Kompatibilität und Verständlichkeit [3], je nach Anwendungsfall kann auch das Antwortverhalten der Anwendung von Bedeutung sein. Im Wesentlichen wird die Qualität eines (Software-)Produkts anhand der folgenden fünf Faktoren gemessen:

- Performanz
- Barrierefreiheit
- Sicherheit
- Usability
- Funktionalität

Diese Faktoren sind ausschlaggebend für den Erfolg eines Produkts, daher eignen sich diese Kategorien als Testaspekte. Im folgenden Abschnitt werden die vorhandenen Testverfahren kurz erläutert und anschließend mit den hier genannten Kategorien in Verbindung gesetzt. Das Augenmerk dieser Arbeit liegt speziell auf der Funktionalität von Web-Anwendungen, denn hier wird der größte Automatisierungsgrad im Bereich des Testens vermutet.

2.3 Testarten

Um die eben genannten Testaspekte prüfen zu können, werden verschiedene aufeinander aufbauende Testkonzepte eingesetzt.

- **Unit Tests** betrachten die kleinsten Teile (Units) einer Software. In der Regel werden hier einzelne Klassen unabhängig von anderen Klassen getestet. Diese Tests werden während der Implementierung durch den Entwickler selbst durchgeführt, da Fehler innerhalb der Tests die Lauffähigkeit und Funktionalität eines Programmes beeinträchtigen [3].
- **Integration Tests** werden von Testern oder Entwicklern (oder von beiden Parteien gemeinsam) durchgeführt und prüfen das Zusammenspiel bestimmter Einheiten (Units) [3].
- **System Tests** werden von einem extra dafür bestimmten Test-Team in einer Testumgebung durchgeführt. Durch einen System Test wird das fertige komplette System getestet [3].
- **Acceptance Tests** unterscheidet sich zum System Test, indem der Test in einem realem Umfeld mit echten Daten durchgeführt. Ziel ist es das System unter Aufsicht der Tester zu testen und die Akzeptanz des Systems durch den Nutzer zu überprüfen [3].
- **Beta Tests** sind Tests die von wohlwollenden Nutzern durchgeführt werden, um möglichst frühes Feedback zum Produkt zu erhalten. Beta Testern wird kein festgesetztes Vorgehen vorgelegt, sondern der Tester benutzt das System und gibt Rückmeldung über seine Erfahrungen [3].

Die genannten Tests können in dieser Reihenfolge stufenweise abgearbeitet werden, da die Komplexität und Bedeutung der Testverfahren aufeinander aufbaut [3]. Die Liste der genannten Testverfahren ist nicht vollständig, da diverse Aspekte nur bedingt manuell getestet werden können und somit separate Tests entwickelt werden müssen. Zusätzlich können zum Beispiel Stress-Tests eingesetzt werden, um die Performanz der Komponenten eines Systems oder das gesamte System zu testen. Die folgende Tabelle soll darstellen welche Tests für die in Abschnitt 2.2 eingeführten Kategorien verwendet werden können.

	Performanz	Barrierefreiheit	Sicherheit	Usability	Funktionalität
Unit Tests	(X)		X		X
Integration Tests	(X)		X		X
System Tests	(X)		X		X
Acceptance Tests	(X)	X	(X)	X	(X)
Beta Tests	(X)	X	(X)	X	(X)

Tabelle 1. Tests für die Kategorien
(X = kann eingesetzt werden; (X) = kann nur bedingt eingesetzt werden)

Testen ist nicht als eigenständiger Prozess zu verstehen, sondern muss in den Entwicklungsprozess eines Produktes integriert werden. Je nach Wahl des Entwicklungsprozess unterscheidet sich die Einbindung und die Relevanz der Tests. Neben den abgestuften in den Entwicklungsprozess eingebunden Konzepte unterscheidet Balzert [4] zusätzlich in dynamische und statische Testverfahren. Dynamische Tests werden stichprobenartig in realem Umfeld mit Testdaten ausgeführt (Whitebox-, Blackbox- und Greybox-Verfahren[1]), wohingegen bei statischen Testverfahren nichts ausgeführt wird, sondern hier nur der Quellcode überprüft wird (Reviews, Walkthroughs). Die bisher aufgezeigten Testverfahren lassen sich hinsichtlich Desktop-Anwendungen und Web-Anwendungen nicht differenzieren und sind auf beide Domänen anwendbar. Allerdings sind diese aufgrund ihrer Architektur teilweise unterschiedlich umzusetzen, da Desktop-Anwendungen meist ein selbstständiges Programm darstellen und Web-Anwendungen aus eine Client- und einem Server-Teil bestehen.

Bisher sind nur allgemeine Prüfkriterien betrachtet worden, im folgenden Abschnitt werden speziell die Unterschiede beim Testen von Anwendungen und Web-Anwendungen betrachtet.

3 Testen in der Anwendungsentwicklung

In der Anwendungsentwicklung gibt es verschiedene Vorgehensmodelle die bei der Entwicklung von Software eingesetzt werden. Weit verbreitete Ansätze sind

[1] Verweis auf [4] und [3]

das Wasserfallmodell, das V-Modell und das Spiralmodell. Die grundlegenden Konzepte, die in diesen Ansätzen genutzt werden können sollen in den folgenden Abschnitten betrachtet werden und auf die Besonderheiten der Anwendungen Bezug genommen werden.

3.1 Desktop-Anwendungen

In konventionellen Ansätzen wird die Entwicklung von Anwendungen in vier Hauptkomponenten untergliedert. Man spricht hier von der Spezifikation, Entwicklung, Validierung und Evolution bzw. Evaulation [4]. Im ersten Abschnitt der Entwicklung soll die benötigte Software spezifiziert werden. Hierzu wird eine Anforderungsbeschreibung entwickelt und das resultierende System modelliert. Auf Grundlage dieser Modellierung soll nun im zweiten Schritt die Software entwickelt werden. Im Anschluss an die Entwicklung muss geprüft werden ob die in der Anforderungsbeschreibung festgelegten Ziele auch erfüllt werden. Im letzten Schritt geht es darum, die entwickelte Software zu verbessern und an die Ansprüche der Nutzer anzupassen. Auch eine Rücksprungmöglichkeit in den vorherigen Prozess kann ggf. notwendig sein.

In herkömmlichen Entwicklungsansätzen wird Testen als fester Prozess integriert und muss daher streng geplant werden. Parallel zu den vier Stufen der Anwendungsentwicklung die oben genannt wurden kann auch die Entwicklung von Tests auch in vier Teilprozesse untergliedert werden [3, Seite 140]:

- **Planung** Um eine Grundlage für die auszuführenden Tests zu schaffen, müssen diese spezifiziert werden. Die einfachste Form ist die Ableitung der Tests aus der Anforderungsbeschreibung des Produkts. In der Anforderungsbeschreibung sind hauptsächlich funktionale Anforderungen formuliert, aber auch Usability-Anforderungen müssen formalisiert werden.
- **Vorbereitung** Nach der Planung der Tests müssen diese auch geschrieben (Unit Tests und System Tests), Testpersonen gesucht und Testbeschreibungen formuliert werden (Acceptance und Beta Tests).
- **Durchführung** Wenn alle Tests vorbereitet sind können diese durchgeführt werden. Wichtig für die Durchführung sind die Beachtung der zuvor spezifizierten Testbedingungen und ein zeitlich vorgegebener Rahmen.
- **Bericht** Im Anschluss an die erfolgreiche Durchführung müssen die Ergebnisse bzw. Auffälligkeiten und Fehler dokumentiert werden. Dies ist notwendig, da diese an die Entwickler weitergegeben werden müssen, um die fehlerhafte Funktionalität zu korrigieren.

Diese Teilprozesse müssen während des Entwicklunsprozesses zyklisch durchlaufen werden. Als weiterer Prozess wird in manchen Fällen auch separat auf die Optimierungsphase verwiesen [5]. Die Verbesserung der aufgetreten Fehlfunktionen ist der letzte Schritt der iterativen Vorgehensweise. Wichtig ist eine Priorisierung der aufgetretenen Fehlerfälle. Die flexible Entwicklung von Web-Anwendungen kann nicht unbedingt mit den verbreiteten Entwicklungsmodellen des Software-Development umgesetzt werden.

3.2 Web-Anwendungen

Im Vergleich zu Desktop-Anwendungen sind Web-Anwendungen schnelllebiger, da hier ein besonderes Augenmerk auf die Aktualität und Funktionalität des Dienstes gelegt wird. Wirkt eine Web-Anwendung nicht mehr aktuell oder werden Funktionen nicht unterstützt sucht der Nutzer nach einer vergleichbaren, aber besseren Alternative (z.B. StudiVZ - Facebook) und es kommt zu einer Nutzerwanderung. Nutzer werden von einer Vielzahl von Aspekten beeinflusst, z.B. eingesetzte Techniken wie JavaScript oder Flash, aber auch die Sicherheit einer Web-Anwendung spielt eine wesentliche Rolle. Ausnahmen sind Dienste die aus Gewohnheit oder äußerem Zwang genutzt werden, z.B. Facebook. Der kurze Lebenszyklus von Web-Anwendungen beeinflusst natürlich auch deren Entwicklung. Zwei Aspekte haben den großen Einfluss auf die Entwicklung von Web-Anwendungen, auf der einen Seite der hohe Verbreitungsgrad und auf der anderen Seite das Bedürfnis der Nutzer immer (und überall) aktuelle Anwendungen zu nutzen zu können, die ihre Bedürfnisse befriedigen. Folgende Abbildung 2 soll die verschiedenen Einflussfaktoren auf die Entwicklung von Web-Anwendungen visualisieren. Das Augenmerk liegt auf den Einflüssen zwischen Auftraggeber-Entwickler und Entwickler-Nutzer, da diese die Entwicklung direkt betreffen. Die Beziehung zwischen Auftraggeber und Nutzer wird nicht weiter betrachtet.

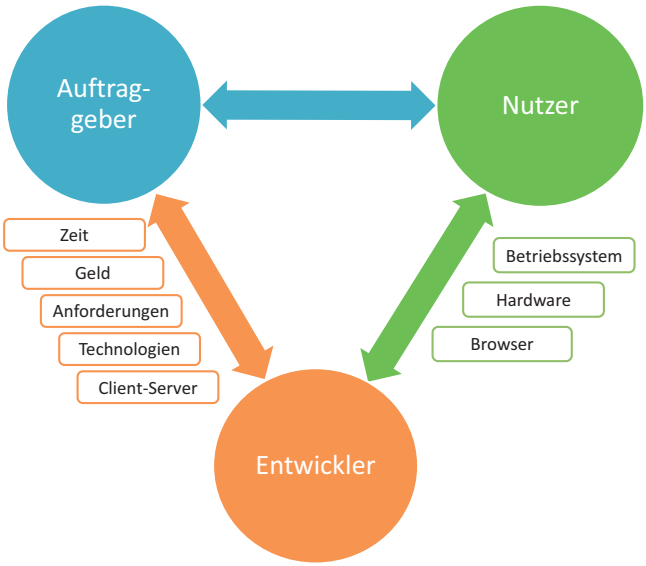


Abbildung 2. Einflussfaktoren auf Anwendungsentwicklung

Der Hauptunterschied zwischen Desktop-Anwendungen und Web-Anwendungen liegt darin, dass Desktop-Anwendungen lokal auf einem Desktop installiert werden, wohingegen Web-Anwendungen in jedem Browser aufgerufen wer-

den können, sofern alle eingesetzten Techniken unterstützt werden und eine Internetverbindung zur Verfügung steht. Neben dem Unterschied, dass Web-Anwendungen mit jedem Browser aufgerufen werden können, gibt es besondere Alleinstellungsmerkmale von Web-Anwendungen die bei deren Entwicklung beachtet und vorallem getestet werden müssen.

- Eine Schwachstelle von Web-Anwendungen sind Linkstrukturen. Zu unterscheiden sind hier Links die ins Nichts führen (broken links), da die verlinkten Seiten nicht mehr existieren oder verschoben wurden und das "back in history"-Problem. Hier ist die Rede von der im Browser implementierten Funktion auf die zuvor besuchten Seiten zurückzuspringen. Hier dürfen beispielsweise bei Einkäufen auf Online-Plattform die dem Warenkorb hinzugefügten Produkte nicht wieder entfernt werden.
- In der "always-on"-Generation müssen Web-Anwendungen rund um die Uhr für den Nutzer zur Verfügung stehen. Die 24/7 Verfügbarkeit von Web-Anwendungen [3] stellt die Systemadministratoren vor die Herausforderung Wartungsfenster so klein wie möglich zu halten und Sicherung und Updates des Systems zur Laufzeit durchzuführen. Zusätzlich müssen bei Ausfall der benötigten Infrastruktur Ausfallsysteme zur Verfügung stehen, die über einen aktuellen Abgleich der Daten verfügen.
- Ein weiteres Problem sind die Nutzer, da diese mit diversen Endgeräten auf Web-Anwendungen zugreifen. Hierdurch entstehen mehrere Probleme. Die Endgeräte können zum einen durch deren Auflösung unterschieden werden, grob kann man hier in Smartphones, Tablets und normale Desktoprechner klassifizieren[2] und auf der anderen Seite den vom Nutzer verwendeten Browser[3].

Wie bereits erwähnt ist die Auflösung der Endgeräte eine besondere Anforderung bei der Entwicklung von Web-Anwendungen. Durch den Einsatz von CSS3 bzw. Responsive Webdesign kann dieses Problem gelöst werden. Hierbei handelt es sich um die automatische Anpassung des Layouts einer Website hinsichtlich der Auflösung des Endgeräts. Ein manueller Test dieser verschiedenen Auflösungen ist kaum möglich. Für das Testen einer Web-Anwendung gibt es z.B. "Responsive Design Testing"[4] [6] von Matt Kersley, hier wird die Website automatisch in den gängigen Auflösungen mobiler Endgeräte eingebunden. In Verbindung mit den eingesetzten Browsern kommt es zur Problematik, dass diese aktuelle Web-Technologien nur nicht oder nur teilweise unterstützen. Ein Beispiel hierfür sind HTML5 und CSS3. Hier können Web-Anwendungen wie der ACID-Test[5] oder HTML5test.com[6] eingesetzt werden, um zu überprüfen ob Web-Standards unterstützt werden. Hierbei handelt es sich nur um einen kleinen

[2] Gängige Auflösungen mobiler Endgeräte: 240*320, 320*480, 480*640, 768*1024, 960*640, 1024*768, ...
[3] Firefox, Chrome, Safari, Opera und betriebssystemspezifische Browser
[4] `http://mattkersley.com/responsive/`
[5] `http://www.acidtests.org/`
[6] `http://html5test.com/`

Ausschnitt der Besonderheiten von Web-Anwendungen. In der Regel können verschiedene Anforderungen automatisiert getestet werden oder durch Monitoring-Tools wie Nagios [7] überwacht werden. Die Automatisierung von Tests ist der wichtigste Punkt in der Softwareentwicklung, da nur die Automatisierung von Prozessen die einfache Wiederholbarkeit von Tests und dadurch eine dynamische und flexible Entwicklung ermöglicht.

Als Sonderfall zwischen Desktop- und Web-Anwendungen können die sogenannten Smartphone-Apps aufgeführt werden. Diese können als native Anwendungen für ein bestimmtes Betriebssystem entwickelt werden oder als Web-Apps auf Basis von Web-Technologien. Die Präferenz sollte auf der Entwicklung von Web-Apps liegen, da hier eine Betriebssystem-Unabhängigkeit gegeben ist.

4 Agile Softwareentwicklung

Neben den herkömmlichen Testmethoden für (Web-)Anwendungen soll in diesem Abschnitt der Schwerpunkt auf die Verwendung von agilen Methoden in der Web-Entwicklung gelegt werden. Zunächst werden die Grundlagen der agilen Softwareentwicklung, ein Überblick über Scrum und im Anschluss der Einsatz des Test-Driven Developments als agile Methode beschrieben.

4.1 Grundlagen

Eine agile Herangehensweise ist ein Versuch Dinge zu vereinfachen und die Komplexität der Planung zu reduzieren, um sich somit mehr auf die Anforderungen des Kunden zu konzentrieren [1]. Ein Grundprinzip der agilen Entwicklung beschreibt, dass die Lösung durch das Entwicklungsteam gefunden wird. Innerhalb des Teams wird die Entwicklung gemeinsam voran getrieben aber nach außen unabhängig von anderen Gruppen (Partizipation und Kollaboration [1]) arbeitet. Testen ist hier eine in den Entwicklungsprozess integrierte Aktivität [3]. Zusätzlich beschreiben Beck et al. [1][7] vier wesentliche Grundsätze der agilen Softwareentwicklung als "Agile Manifesto":

- Individuen/Interaktionen sind wichtiger als Prozesse und Werkzeuge, da die Interaktion zwischen den beteiligten Entwicklern und Stakeholdern einen höheren Einfluss auf die Fertigstellung des Produkts haben, als eingesetzte Werkzeuge.
- Funktionierende Software ist wichtiger als eiine umfassende Dokumentation, denn diese wird selten von den Endnutzern genutzt. Umsetzung der Dokumentation kann direkt durch Kommentare im Code umgesetzt werden oder durch den Einsatz des Test-Driven Developments um die Funktionalität durch die angelegten Testfälle zu dokumentieren.
- Zusammenarbeit mit Stakeholder wichtiger als lang andauernde Vertragsverhandlungen.

[7] `http://www.nagios.org/`

– Reagieren auf Veränderungen ist wichtiger als striktes Folgen des Entwicklungsplans. Personelle und finanzielle Änderungen können nicht von Anfang in der Planung der Entwicklung berücksichtigt werden. Hier muss auf flexible (agile) Ansätze zurückgegriffen werden.

Desweiteren gibt es zwölf Prinzipien agiler Softwareentwicklung die unter `http://www.agilemanifesto.org/principles.html` zu finden sind. Im Zusammenhang mit agilen Entwicklungsmethoden fällt häufig der Begriff Xtreme Programming (kurz XP). Im folgenen werden zwei Praktiken aus dem Xtreme Programming betrachtet die für das Testen von Web-Anwendungen eingesetzt werden können.

Pair Programming ist wohl die bekannteste Methode aus der agilen Entwicklung. Die Idee ist, dass sich zwei Programmierer einen Computer teilen. Während einer produktiv entwickelt, überblickt der zweite Entwickler den geschriebenen Code und achtet auf (Denk-) Fehler. Nach einer gewissen Zeite werden die Rollen getauscht. Hierdurch soll die Fehleranzahl im Code reduziert werden.

Die kontinuierliche Integration des entwickelten Codes ist ein weiterer Aspekt des XP und soll die Einbindung von großen Klassen vermeiden und somit auch die Integration von Fehlern. Dieser Ansatz steht im Zusammenhang mit der testgetriebenen Entwicklung (siehe Test-Driven Development).

4.2 Scrum

Ein weiterer Ansatz ist Scrum und dient der Vereinfachung des Projektmanagements und beschreibt einen iterativen Entwicklungsprozess. Scrum baut auf drei (Nutzer-) Rollen, drei Dokumentearten und drei Meetings auf. Der Product Owner, vertritt die Stakeholder, das (Entwicklungs)-Team ist verantwortlich für die Entwicklung und Testen des Produkts und schließlich der Scrum Master der für die Qualität des Prozesses und des Produkts verantwortlich ist. Die drei Dokumente sind das Product Backlog, in dem alle Anforderungen an das Produkt festgehalten und mit Prioritäten versehen werden. Das Sprint Backlog beinhaltet alle Use-Cases die für einen Sprint festgelegt werden. Eine Iteration im Scrum-Prozess wird Sprint genannt und beinhaltet die Phasen Design, Entwicklung, Testen und Dokumentation. Ein Sprint wird innerhalb von 2 Wochen durchlaufen. Die abgeschlossenen Use-Cases werden im Sprint Result festgehalten [1, S.58f]. Zusätzlich kann der aktuelle Entwicklungsstand in sogenannten Burndown-Charts visualisiert werden [8]. Die Meetings sind unterteilt in Sprint Planning, Daily Scrum und Sprint Review. Im Sprint Planning wird der bevorstehende Sprint (2-4 Wochen je nach Projektlaufzeit und Komplexität) auf Basis des Sprint Backlogs durch den Product Owner, dem Scrum Master und dem Entwicklungsteam geplant. Das Daily Scrum ist ein tägliches Meeting das ungefähr 15 Minuten dauert und hier kurz auf die letzten 24 Stunden der Entwicklung (Erfolge und Probleme) eingegangen wird. Am Ende des Sprints findet der Sprint Review in dem Product Owner, Scrum Master und das Entwicklungsteam die erreichten Ergebnisse zu überprüfen [1, S.58f]. Dieser iterative Prozess wird bis zur Fertigstellung des Produkts durchgeführt und das Produkt an Auftraggeber übergeben. Abbildung 3 soll das Scrum-Modell veranschaulichen.

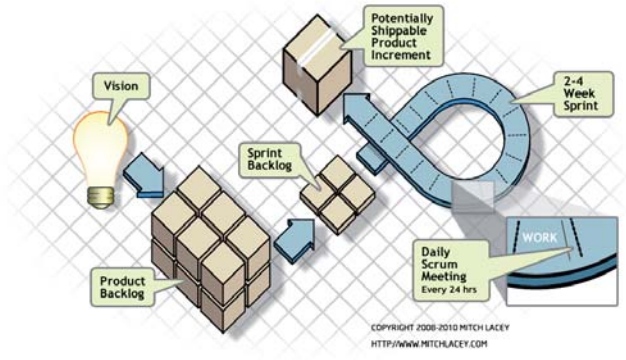


Abbildung 3. Vorgehensmodell Scrum (Quelle: `http://www.mitchlacey.com/`)

4.3 Test-Driven Development

Test-Driven Development (TDD) beschreibt eine Entwicklungsmethode in der Tests im Mittelpunkt des Entwicklungsprozesses stehen. Grundgedanke ist es jede einzelne Funktion eines Programmes mit Hilfe von Unit-Tests zu überprüfen. Aufgrund dieser Tatsache können mit TDD nur funktionale Komponenten getestet werden und nicht Aspekte wie Barrierefreiheit oder Usability. MARTIN beschreibt in [9] die Grundlagen und Anforderungen des TDD. Im Gegensatz zur üblichen Software-Entwicklung wird im TDD nicht bereits geschriebener Code getestet, sondern Tests werden geschrieben bevor der eigentliche Code geschrieben wird (Tests und Code während somit zur selben Zeit entwickelt). Im wesentlichen gibt es im TDD drei einzelne Schritten zu beachten (Abbildung 4):

- Test der fehlschlagen soll wird geschrieben, bevor eigentlicher Code geschrieben wird.
- Es wird nicht mehr getestet als notwendig.
- Genau soviel Code wird entwickelt, dass der Test nicht fehlschlägt.

Hierbei handelt es sich um einen iterativen Entwicklungsprozess, der Durchgang einer solchen Entwicklungsschleife sollte in der Regel zwischen 30 Sekunden und einer Minute liegen [10]. Wichtig ist an dieser Stelle, dass immer alle Tests durchgeführt werden, um Auswirkungen des neuentwickelten Codes auf bereits bestehende Komponenten auszuschließen oder auftretende Fehler zu korrigieren. Durch diesen Ablauf bzw. das simultane entwickeln von Test-Code und Anwendungs-Code sollten diese beiden Codestrukturen im gleichen Maße wachsen [10]. Die Verwendung von Unit-Tests macht die Entwicklung flexibel, wartbar und lesbar, da durch diese Entwicklungsstrategie zum einen immer ein funktionsfähiges Programm vorhanden ist und zum anderen man nicht von Änderungen an der Code-Basis zurückschreckt. Als zweiter wesentlicher Punkt ist die Test-Automatisierung zu nennen, da Unit-Tests auf Knopfdruck gestartet

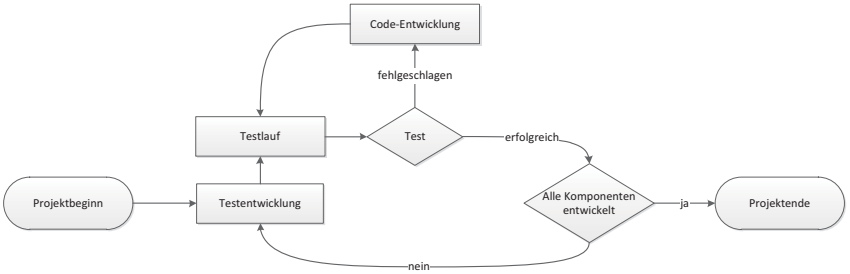


Abbildung 4. Vorgehen im Test-Driven Development

werden und ohne manuelle Eingaben während des Tests durchlaufen werden. MARTIN beschreibt in [9] die Anforderungen an gut entwickelte Tests mit dem F.I.R.S.T.-Ansatz.

1. **Fast** Test müssen schnell durchlaufen werden, da sonst aufgrund der langsamen Laufzeit die Motivation sinkt diese laufen zu lassen.
2. **Independent** Es dürfen keine Abhängigkeiten zwischen Tests vorliegen, da hier bei fehlschlagen eines Tests die hiervon Abhängigen auch nicht erfolgreich durchlaufen werden.
3. **Repeatable** Die Wiederholbarkeit von Tests bezieht sich auf die Laufzeitumgebung, das bedeutet alle Tests müssen z.B. in der Produktionsumgebung oder QA-Umgebung lauffähig sein.
4. **Self-Validating** Tests sollten zu *true* oder *false* auswerten, um ohne Überprüfung von log-Dateien feststellen zu können, ob ein Test fehlgeschlagen ist oder erfolgreich war.
5. **Timely** Zeitlich sollen Tests, nach dem Vorgehen nach TDD, vor dem Produktiv-Code geschrieben werden.

Die genannte Dauer der Iterationsschritte verlängert sich mit der Zeit, da verschiedene Tests von einer Code-Änderung betroffen sein können und diese Tests angepasst werden müssen. MARTIN schreibt in [10], dass der Durchlauf von 2200 Unit Tests[8] nicht länger als 90 Sekunden dauert. Um die Lesbarkeit von Tests und Code zu gewährleisten müssen diese unter Umständen umstruktiert werden. Hierbei ist meist die Rede von Refactoring, einer weiteren wichtigen Komponente des Extreme Programming. Unter Refactoring versteht man die Neustrukturierung der bisherigen Codebasis, um Redundanzen im Code aufzulösen, Kommentare einzufügen und die Verständlichkeit des Codes zu verbessern. Allgemein sollte zusätzlich darauf geachtet werden für Klassen, Funktionen und Variablen sprechende Namen zu verwenden. Dieser Sachverhalt zur Lesbarkeit und Verständnis der umgesetzten Funkt beitragen.

[8] Diese Tests decken ca. 90% des (Java-)Codes von FitNesse `http://fitnesse.org` ab.

Da Web-Anwendungen aus vielen verschiedenen Einzelkomponenten bestehen bietet sich dieses Konzept für deren Entwicklung an.

4.4 Einsatz von Tools

Im folgenden Abschnitt, soll mit zwei kurzen Beispielen aufgezeigt werden, wie eine Test-Automatisierung in der Entwicklung von Web-Anwendungen erreicht werden kann. Da im Test-Driven Development Unit-Tests eingesetzt werden, wird an dieser Stelle auf PHPUnit[9] verwiesen, ein Unit-Test Framework für PHP. JSUnit[10] ist ein Framework das für JavaScript Unit-Tests eingesetzt werden kann. Exemplarisch wird in Abbildung 5 ein PHPUnit-Test für die Initialisierung eines Arrays und darauf ausgeführten push- und pop-Operators dargestellt.

```
<?php
class StackTest extends PHPUnit_Framework_TestCase
{
  public function testPushAndPop()
  {
    $stack = array();
    $this->assertEquals(0, count($stack));

    array_push($stack, 'foo');
    $this->assertEquals('foo', $stack[count($stack)-1]);
    $this->assertEquals(1, count($stack));

    $this->assertEquals('foo', array_pop($stack));
    $this->assertEquals(0, count($stack));
  }
}
?>
```

Abbildung 5. PHPUnit-Test Beispiel (Quelle: `http://www.phpunit.de`)

Abbildung 6 zeigt das Ergebnis des erfolgreich durchlaufenen Unit-Tests. Mit dem zusätzlichen Parameter *--testdox* erstellt PHPUnit eine Dokumentation der durchlaufenen Testklassen und Funktionen. Für die Dokumentation wird hier der Klassenname (ohne Test) verwendet und die Funktionen anhand Groß- und Kleinschreibung getrennt.

Neben der Verwendung von Unit-Tests kann die Funktionalität einer Website auch über das Frontend bzw. Benutzereingaben getestet werden. Problematisch sind an dieser Stelle Formularfeldern oder lange Klickwege, denn diese müssen bei jedem Testdurchlauf von Hand ausgefüllt und durchgeführt werden.

[9] `http://www.phpunit.de/`
[10] `http://www.jsunit.net/`

```
> phpunit ArrayTest.php
PHPUnit 3.6.10 by Sebastian Bergmann.
.
Time: 0 seconds, Memory: 4.50Mb
OK (1 test, 5 assertions)
---
> phpunit --testdox ArrayTest.php
PHPUnit 3.6.10 by Sebastian Bergmann.

Stack
 [x] Push and pop
```

Abbildung 6. Ausgabe PHPUnit-Test Beispiel

Für diesen Anwendungsfall kann Selenium[11] eingesetzt werden. Selenium ist ein Tool zur Unterstützung der Browser-Automatisierung, auf Basis von selbst angelegten Skripten. Als Beispiel wird hier auf Selenium IDE eingegangen, ein Firefox-Plugin zur Entwicklung von Testfällen. Tests werden mit Hilfe des Plugins aufgenommen und können später zu jeder Zeit wieder abgespielt werden (Capture- & Replay-Funktionalität). In Abbildung 7 ist ein Beispiel einer Selenium Test-Suite dargestellt.

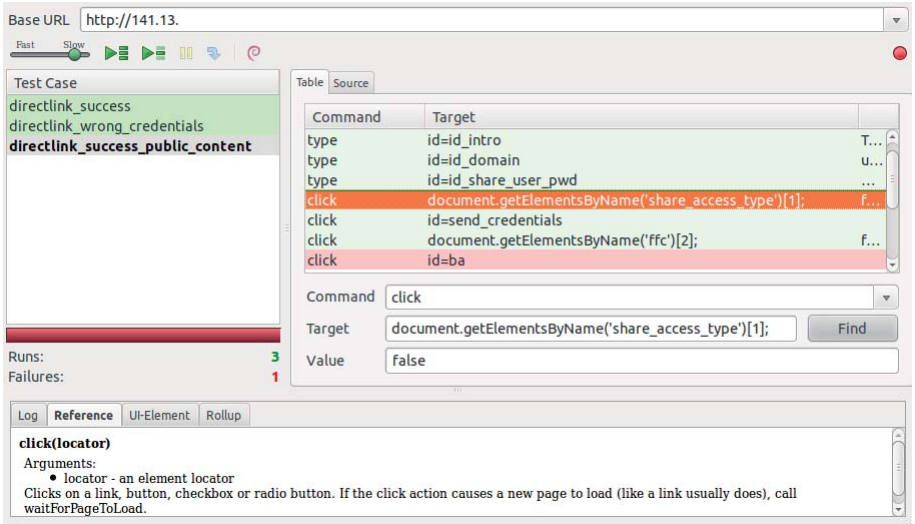


Abbildung 7. Selenium IDE Beispiel

[11] http://seleniumhq.org/

Im Vergleich zu Unit-Tests ist bei der Verwendung von Selenium zu beachten, dass die Testausführung mehr Zeit beansprucht als ein Unit-Test, da hier das Antwortverhalten des Web-Servers eine wesentliche Rolle spielt. Die beiden dargestellten Tools, sollen nur einen kurzen Einblick in die Test-Automatisierung geben und hierdurch eine schnellere Entwicklung von Web-Anwendungen unterstützen. Durch den Einsatz dieser Tools wird nicht nur die Entwicklungsgeschwindigkeit gesteigert, sondern auch die Code- bzw. Produktqualität verbessert.

5 Forschungsansätze

Neben der Verwendung der oben genannten Vorgehensmodelle, Scrum und TDD, und genannten Tools gibt es verschiedene Forschungsansätze, um eine Test-Automatisierung bei der Web-Entwicklung zu unterstützen.

5.1 WebTDD

Eine Erweiterung des Test-Driven Developments speziell für Web-Anwendungen beschreibt WebTDD. Ziel hinter WebTDD ist es Anforderungen des Produkts zu spezifizieren und im Anschluss automatisch Interaktions-Tests aus diesen Anforderungen zu generieren. Das Vorgehen des Ansatzes ist Sprint-basiert (siehe Scrum) und nutzt WebSpec als domänenspezifische Sprache. Zu Beginn werden Mockups und WebSpec Diagramme erstellt aus denen im AnschlussTests abgeleitet werden. Auf dieser Basis folgt der bekannte TDD-Ansatz, Tests werden durchgeführt und es werden nur neue Tests bzw. Funktionen erstellt, wenn die bisherigen Tests erfolgreich durchlaufen worden sind [11].

5.2 Crawljax

Bei Crawljax[12] handelt es sich um einen Ansatz bzw. Tool für das Testen von Inhalten die per AJAX in eine Website eingebunden werden. Hierbei handelt es sich um Inhalte die zur Laufzeit der Anwendung abgerufen werden und dadurch das Testen einer Web-Anwendung erschweren. In den letzten Jahren wurden herkömmliche Websites auf Basis ihrer HTML-Struktur und (ohne dynamische) Inhalte überprüft. Mit Selenium ist es bereits möglich per AJAX eingebundene Inhalte zu überprüfen, allerdings handelt es sich hierbei wie bereits erwähnt um einen Capture- & Replay-Ansatz. Das bedeutet es müssen alle erreichbaren Zustände zuerst von Hand durchgeführt werden, um diese wieder abspielen zu können. Crawljax füllt automatisch Formulare aus und löst Trigger auf einer Seite aus, um so einen State-Flow Graphen auf Basis der DOM-Tree Änderungen zu erstellen. Dieser Graph ist u.a. für das Auffinden von defekten Linkstrukturen nutzbar [12].

[12] `http://crawljax.com/`

6 Fazit

Die Probleme liegen in der Praxis, denn hier sind "Quick and Dirty"-Ansätze bei Entwicklern weit verbreitet, hierunter leidet die Codequalität des Produkts und dieses ist somit schlechter wartbar. Das Einbinden von Tests in den Entwicklungsprozess benötigt daher ein hohes Maß an Disziplin. Weiter fehlen in den meisten Projekten sowohl die finanziellen Mittel und Zeit, um Tests umsetzen zu können. Eine Umsetzung ist generell nur möglich, wenn Tests automatisiert ablaufen und keine Interaktionen, in Form von manuellen Eingaben, von Seiten der Entwickler notwendig sind. Durch den Einsatz der oben genannten Tools und Ansätze ist es möglich Tests in die Entwicklung einzubinden. Neben den genannten Ansätzen ist es sinnvoll verschiedene Frameworks einzusetzen durch die Probleme in der Entwicklung zu vermeiden. Eine Möglichkeit ist z.B. der Einsatz von JavaScript-Frameworks wie jQuery, um Browserinkompatibiltäten zu verhindern, da diese bereits für die gängigen Browser[13] getestet sind.

Literatur

1. Stober, T., Hansmann, U.: Agile Software Development: Best Practices for Large Software Development Projects. Springer Publishing Company, Incorporated (2010)
2. Priebsch, S.: Advanced OOP and Design Patterns unter http://priebsch.de/blog/advanced-oop-and-design-patterns/, zuletzt geprüft am 29.03.2012 (2009)
3. Kappel, G., Pröll, B., Reich, S., Retschitzegger, W.: Web Engineering - The Discipline of Systematic Development of Web Applications. John Wiley & Sons Ltd., England (2006)
4. Balzert, H.: Lehrbuch Grundlagen der Informatik. 2. edn. Spektrum, Heidelberg (2005)
5. Düweke, Esther und Rabsch, S.: Erfolgreiche Websites: SEO, SEM, Online-Marketing, Usability. Galileo Computing. Galileo Press GmbH (2011)
6. t3n: t3n Nr. 27 - Future Cash / Responsive Webdesign mit CSS3 unter http://t3n.de/news/sponsored-post-responsive-371942/, zuletzt geprüft am 30.03.2012. (2012)
7. Beck, K., et al.: Manifesto for agile software development unter http://www.agilemanifesto.org, zuletzt geprüft am 19.03.2012 (2001)
8. Wikipedia: Scrum unter http://de.wikipedia.org/wiki/scrum, zuletzt geprüft am 28.03.2012 (2012)
9. Martin, R.C.: Clean Code: A handbook of agile software craftsmanship. Prentice Hall, United States (2009)
10. Martin, R.C.: The Clean Coder: A Code of Conduct for Professional Programmers. Programmer's Choice. Prentice Hall (2011)
11. Robles Luna, E., Burella, J., Grigera, J., Rossi, G.: A flexible tool suite for change-aware test-driven development of web applications. (2010)
12. Mesbah, A., Van Deursen, A.: Invariant-based automatic testing of ajax user interfaces. 2009 IEEE 31st International Conference on Software Engineering (2009) 210–220

[13] Internet EXplorer 6.0+, Firefox 3.6+, Safari 5.0+, Opera, Chrome

Performanz bei Web-Anwendungen

Arndt Bergner

Otto-Friedrich-Universität Bamberg, Lehrstuhl für Medieninformatik
arndt1.bergner@stud.uni-bamberg.de

Zusammenfassung Diese Seminar-Arbeit zeigt anhand eines veränderten Wasserfallmodells einen generellen Überblick über Tätigkeiten zum Aufbau und Sicherung von Performanz bei Web-Anwendungen und verdeutlicht den Einfluss der Phasen-Entscheidungen.

Schlagworte: Performanz, Web-Anwendungen, Nicht-funktionale Anforderung, Performanz-Test, Last-Test, Stress-Test, Web-Softwareentwicklung.

1 Einleitung

Die Performanz von Web-Anwendungen ist nicht nur für die Benutzerfreundlichkeit eine wichtige Anforderung. Für deren Erfüllung beschreibt Kotsis in ihrem Aufsatz *Performanz von Web-Anwendungen* das Performanz-Testen anstatt den Gesamtkontext der Entwicklung einer auf Performanz optimierten Web-Applikation.[1, vgl. S. 298.]. Andere Phasen und Techniken werden damit ausgeklammert. Zum einen stellen Web-Seiten nur einen Teil des Spektrums dar, bei dem Performanz eine Kern-Anforderung ist. Dies kann von einer Homepage, bis zu einer Android-App reichen.[1] Zum anderen kann trotz unterschiedlicher Arten und Kontexte in einer frühen Entwicklung der Fokus auf Aufbau und Sicherung von Performanz liegen. Innerhalb der Forschung obliegt dies oft nur der Testphase, wie z.B. die Monographien zum Thema Web-Engineering von Pressman(et al.) und Casteleyn zeigen.[2,3, vgl. S. 398, vgl. S. 59] Aktuelle Forschungen gehen nur auf die Vorzüge bestimmter Technologien ein.[2]

In dieser Arbeit werden die Tätigkeiten zum Aufbau und Sicherung von Performanz einer Webanwendung mithilfe des Entwicklungsprozess dargestellt. Zunächst drei Kenngrößen definiert und anschließend anhand der Phasen eines veränderten und erweiterten Wasserfallmodells die Aktivitäten zur Performanz-Sicherung erläutert. Hervorhebungen durch die Autoren in ihren Texten wurden dabei nicht übernommen.

2 Kenngrößen

Um die Performanz einer Web-Anwendung zu Messen gibt es verschiedene Kenngrößen, die auf bestimmte Engpässe hindeuten. Viele wie z.B. die Servicezeit, die

[1] In dieser Arbeit wird der Begriff zu einer Anwendung mit einer Client-Server-Architektur generalisiert, deren Kommunikation mithilfe des Internets erfolgt.
[2] Z.B. siehe [4].

Bearbeitungsdauer einer Operation bei einer Anfrage, stammen aus der Theorie der Warteschlangennetzwerke. Im Folgenden werden nur drei Kenngrößen erläutert. Diese bilden die Quintessenz der Literatur und werden am häufigsten als Kenngrößen verwendet, wie es ebenfalls Casteleyn darstellt.[3, S. 261.]

Antwortzeiten - Response time
Antwortzeit R_i der Queue Q_i bei der Anforderung einer Ressource ist die Summe der Servicezeit S_i einer Anfrage und die durchschnittliche Wartezeit W_i. [5, vgl. S. 46f.] Bei einer Webseite wäre dies die Zeit bis zu einer bemerkbaren Reaktion durch den Web-Browser.

Durchsatz - Throughput
Der Durchsatz der Queue ist die Anzahl an „bearbeiteten Resourcenanforderungen je Zeiteinheit“[5, S. 46.]. Eine andere mögliche Definition wäre die verarbeiteten Transaktionen pro Nutzer.

Auslastung - Utilization
Die Auslastung U_i des System bei der Bearbeitung der Queue ist im Auslastungsgesetz formuliert als das Produkt aus der Servicezeit pro Anfrage S_i und dem Durchsatz x_i.[5, vgl. S. 46.] Dabei sollte die vollständige Auslastung einer Ressource niemals ein Schwellwert sein, sondern immer genug Puffer besitzen, um schwankende Benutzungen der Web-Anwendung auszugleichen.

3 Performanz innerhalb des Entwicklungsprozess'

Viele Entwicklungsprozessmodelle besitzen im Kernvergleichbare Phasen, die durch das Wasserfallmodell abgedeckt werden können. Es umfasst um das Arbeitslastmodell erweitert folgende relevante Phasen: Anforderung, Arbeitslastmodell, Entwurfsphase, Implementierung, Test sowie Weiterentwicklung. Im Folgenden werden diese Phasen genutzt, um die Tätigkeiten zur Performanz-Sicherung in Bezug auf die Entwicklung von Web-Anwendungen zu verdeutlichen.

3.1 Anforderung

Performanz wird innerhalb der Softwareentwicklung als nicht-funktionale Anforderung kategorisiert und fällt unter die Effizienz des Produkts, wie es Sommerville aufschlüsselt.[6, vgl. S. 88.] Sie kann nicht direkt z.B. mit einem Technologie-Einsatz oder der Implementierung einer Funktionalität umgesetzt werden, sondern wird global durch viele Anforderungen und Entwurfsentscheidungen beeinflusst. Meier fasst zusammen: „Performance requirements [...] are absolutely non-negotiable due to contractual obligations, service level agreements (SLAs) [= Service-Level-Agreements], or fixed business needs.“[7, S. 121.]

Für deren Erhebung eignen sich die von Sommerville beschriebenen Methoden.[6, vgl. S. 82-111] Eine Beispiel-Anforderung für einen Picasa-ähnlichen Dienst für Fotoalben könnte lauten: Der Aufbau der Einzelansicht

eines Bildes in einem eigenen Tab darf nicht länger als zwei Sekunden dauern. Ein wichtiger Anhaltspunkt für die Erhebung sind die Ziele der Unternehmung des Kunden oder können nach Meier die SLA, Industriestandards oder Nutzer-Erwartungen sein.[7, vgl. S. 48.] Zusätzlich müssen Beschränkungen und andere Anforderungen wie die Verwendung bestimmter Sicherheitsprotokolle, z.B. TLS, betrachtet werden, die sich negativ auf die Performanz auswirken können.

Um die Qualität der Performanz-Anforderungen zu steigern können diese an den SMART-Kriterien gemessen werden. „SMART ist ein Akronym und steht für die Eigenschaften spezifisch, messbar, erreichbar (attainable), realisierbar und nachvollziehbar (traceable)." [5, S. 29.]

Der erste Entwicklungsschritt unter dem Gesichtspunkt der Performanz ist eine spezialisierte Anforderungserhebung und -analyse, die sich zielgerichtet auf Performanz-Kriterien spezialisiert. Sie werden wie funktionale Anforderungen erhoben, werden jedoch durch Beschränkungen und Anforderungen beeinflusst.

3.2 Arbeitslastmodelle

Der Aufbau eines Arbeitslastmodells kann helfen, Anforderungen zu verifizieren. Dabei werden die später zum Testen verwendeten Lasten bzw. Testdaten für die Last-Tests immer genauer spezifiziert. Das Vorgehen setzt voraus, dass das Modell inkrementell mit allen neu verfügbaren Informationen der einzelnen Phasen oder eines Bestandssysteme verbessert wird. Mit der Erfassung der Anforderungen, spätestens aber mit der Entwurfsphase kann ein Model aufgebaut werden. Jedoch findet sich in der aktuellen Literatur kein konkretes, standardisiertes Vorgehen, wie es Casteleyn begründet: „Estimating the workload based on [...] requirements is not an exact discipline [...]."[3, S. 267.] Deshalb wird beispielhaft der von Meier in *Performance Testing Guidance for Web Applications* beschriebene Ablauf herangezogen, vereinfacht und erläutert.

Das Ziel eines Arbeitslastmodells fokussiert „the realism of a test, or on designing a test to address a specific requirement, goal, or performance-testing objective."[7, S. 139.]. Verschiedene Arbeitslasten können charakterisiert werden, die Kotsis weiter aufteilt: reale, synthetische und gemischte Lasten.[1, vgl. S. 304.] Synthetische sind theoretisch mögliche Lasten, die sich nicht unbedingt an die Realität anlehnen. Daher sollte der Aufbau wie Meier es eingehend beschrieben hat, realitätsnah modelliert werden.

In einem ersten Schritt sollen die Ziele bzgl. der Performanz ermittelt werden.[7, vgl. S. 139.] Meier stellt zur Formulierung zwar einige Fragen aus Entwickler-Sicht, jedoch liefern die in der Anforderungsphase erhobenen Daten ersten Aufschluss über mögliche Ziele. Auf deren Grundlage können Fragen an den Kunden gestellt werden: Für wie viele Nutzer soll das System in der Zukunft ausgelegt sein? „[How] long does the new demand need to be sustained before exhaustion of a resource compromises the service level agreements (SLAs)?"[7, S. 139f.] Mit diesen Fragen können die Schwellwerte und Verhalten des Systems spezifiziert und Ziele für verschiedener Arbeitslastmodelle deklariert werden.

In einem zweiten Schritt werden die Szenarien ermittelt, die diese Ziele abdecken können.[7, vgl. S. 140.] Quellen für diese Szenarien sind erhobene An-

wendungsfälle, die eine genaue Interaktionsfolge in ihrem Haupterfolgsszenario beschreiben.[3] Andere SWT-Aktitvitäten wie Interviews können ebenfalls helfen, Performanz-relevante Informationen zu sammeln und diesen Zielen zuzuweisen. Innerhalb des Prozess' können auch Performanz-kritische Daten ermittelt werden wie z.B. angezeigte Bildformate etc..

Als nächstes schlägt Meier vor, den möglichen Navigationspfad zu ermitteln.[7, vgl. S. 142.] Hier können detaillierte Anwendungsfälle mit einer Wahrscheinlichkeit den Schritt vereinfachen, da sie die Navigation für das Szenario beschreiben. Meier visualisiert alle Wege und zeigt, welche Pfade und Interaktionen kritisch für die Performanz sind.[7, vgl. S. 143.] Damit kann die Übersicht über Pfade behalten werden, die eine Nutzer-Session beinhaltet.

Anschließend beschreibt Meier, dass individuelle Nutzerdaten und Varianzen ermittelt werden sollen.[7, vgl. S. 143.] Dieser Punkt erscheint unmöglich für Neuentwicklungen, da keine Quellen wie Web-Server-Logs vorliegen. Kann auf Daten wie die die Dauer von Session zurückgegriffen werden, kann das Arbeitslastmodell angepasst werden.[7, vgl. S. 144.] Eine Möglichkeit der Varianz kann die Verwendung von Bedenkzeit bei virtuellen Nutzer sein.

Neben den Navigationspfaden kann ermittelt werden, wie häufig eine Aktivität vollzogen wird.[7, vgl. S. 144.] Die verschiedenen Lastmodelle beinhalten verschiedene Szenarien. Ein erster Indikator, den Meier ausspart, sind die in diesen häufig erwähnten Interaktionen. Erneut zeigt sich der Vorteil von detailliert strukturierten Anwendungsfällen, die die Szenarien abdecken. Andere Möglichkeiten, häufige Aktivitäten bei der Nutzung zu ermitteln, ist die Evaluierung eines Beta-Releases oder Interviews mit den Zuständigen des Kunden.[7, vgl. S. 145.] Wie bereits bei den wahrscheinlichen Navigationspfaden, können die Aktivitäten u.a. visuell aufbereitet werden.

Ein vorletzter Schritt ist die Ermittlung des Ziel-Last-Niveaus, das durch viele virtuelle Nutzer in ihren Sessions repräsentiert wird.[7, vgl. S. 147.] Die Threads werden durch einen Lastgenerator erstellt. Sie gehen die verschiedenen Navigationspfade der Szenarien durch. Dabei werden häufige Aktivitäten berücksichtigt und innerhalb der Session häufiger durchgeführt. Eine Session kann verschiedene Szenarien beinhalten. Meier veranschaulicht den theoretischen Ablauf des Nutzungsvolumen visuell mit folgender Grafik:

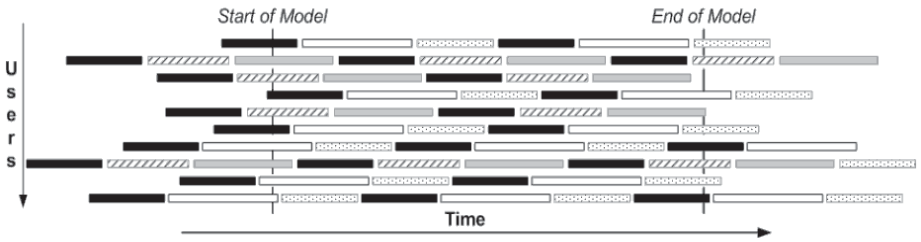


Abbildung 1. Die Server Perspektive der Nutzer-Aktivitäten.[7, S. 147.]

[3] Zur Erfassung strukturierter Anwendungsfälle, siehe [8].

Die Balken beschreiben verschiedene Aktivitäten, die eigenständige Threads (= virtuelle Nutzer) durchführen. Der schwarze Balken kann z.B. der Besuch der Startseite des erwähnten Picasa-ähnlichen Dienst sein. Danach betrachtet der Nutzer Alben, die als Daten bestimmte Bildtypen beinhalten. Er lädt ein Album herunter, kehrt zur Startseite zurück und wiederholt die Interaktion. Andere Nutzer decken gleiche und weitere Interaktionen mit dem System ab. Mit diesem Arbeitslastmodell lassen sich realitätsnahe Lasten generieren und Performanz-Anforderungen verifizieren. „[...] [The] client request stream has to be replayed through a traffic generator. The main goal of a traffic generator is to reproduce the specified traffic in the most accurate and scalable way."[9, S. 199.]

Zusammenfassend wurde ein möglicher Prozess zur Erzeugung eines Arbeitslastmodells nach Meier erläutert. Es wurden Ziele deklariert und Szenarien ermittelt, die diese erfüllen können. Danach wurden Navigationspfade ausgemacht und häufige Aktivitäten identifiziert. Die Mischung verschiedener Nutzer und deren Interaktion bildet den letzten Schritt vor der Modell-Implementierung oder Generierung durch einen Lastgenerator. Wichtig ist die anforderungs- und realitätsnahe Modellierung durch Szenarien und Anwendungsfällen. Jede Iteration präzisiert das Modell und mindert den Arbeitsaufwand für Performanz-Tests.

3.3 Entwurfsphase

Bestimmte Entscheidungen der Entwurfsphase, die noch nicht durch technische Beschränkungen oder Anforderungen geklärt sind, können Einfluss auf die Performanz, z.B. auf Antwortzeiten haben. Diese Entscheidungen geben erste Implikationen auf bestimmte Merkmale wie Antwortzeiten.

Programmiersprache
Eine erste Entscheidung ist der Einsatz der Programmiersprachen. Ein Beispiel ist die Studie von Trent(et al.), die PHP und JSP als Server-seitige Script-Sprachen nach SpecWeb2005-Benchmark-Konfiguration vergleicht.[4] „If outstanding performance and throughput is the primary goal, then the use of JSP over PHP is advisable. However, if a 5-10% difference in throughput and performance is acceptable, then the implementer [...] can achieve similar results [...]."[4, S. 181.] Wie Trent treffend erwähnt, ist eine solche Wahl auch von der Erfahrung der Entwickler, der Effizienz bei der Programmierung etc. abhängig.[4, vgl. S. 181.] Dabei sollte die ausgewählten Sprachen aufeinander abgestimmt und aktuell sein, um Performanz zu gewährleisten.

Parallelisierung
Eine weitere Möglichkeit ist die Parallelisierung der Abläufe. Die Anwendung vertikal geklont werden, so dass mehrere Anwendungsinstanzen gleichzeitig laufen und die Last verteilt wird. Ceri beschreibt den Prozess genauer: „With vertical cloning, a single server machines includes several independent processors and hosts multiple processes dedicated to the application."[11, S. 338.]

[4] Für genauere Information zu SpecWeb2005 siehe [10].

Neben dem vertikalen Klonen kann die Anwendung auch horizontal geklont werden, so dass vollständige Replikationen der Anwendung als eigenständige Systeme (Knoten) in der Front-End-Schicht existieren. Die Architektur „[...] contains multiple machines, each hosting a replica of the Web server and execution engine. In this solution, the router/firewall acts as a network dispatcher, and spreads the incoming HTTP requests to the different Web servers, to balance their workload, and augment the throughput towards the database server.“[11, S. 339] Die Replikation wird im Folgenden grafisch veranschaulicht:

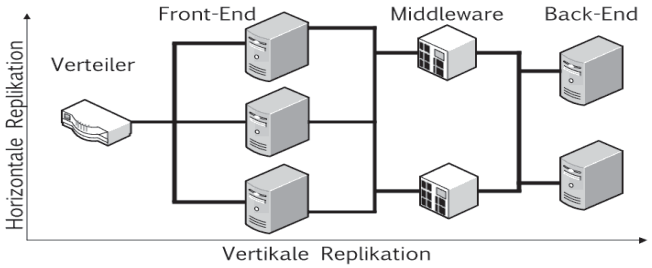


Abbildung 2. Vertikale und horizontale Replikation in Anlehnung an [9, vgl. S. 196.].

Die Anfragen an den Server laufen über einen Verteiler, hier einem „Load-Balancer“, weiter zu den horizontalen Subsystemen. Jede Schicht kann sowohl horizontal wie auch vertikal repliziert werden. Ein Beispiel für die vertikale Replikation ist, den Anwendungsserver von der Datenbank abzukoppeln.[11, vgl. S. 336.] Dadurch werden alle Datenbank-Anfrage von einem spezialisierten Server verarbeitet und nicht zusätzlich durch den Anwendungsserver.

Die Replikation kann jedoch die Komplexität des Systems erhöhen und sich wegen vermehrtem Kommunikationsaufwand negativ auf die Performanz auswirken. Z.B. könnte die Session eines Nutzers durch die Verteilung verloren gehen oder falsch zugeordnet werden. Wie Ceri beschreibt müsste der Zustand der Nutzer-Session durch den Load-Balancer geeignet verwaltet werden.[11, S. 340]

In der Entwurfsphase müssen Vor- und Nachteile von Klonen und Replikation auf die Performanz der Web-Anwendung abgewägt werden. Die Replikationen geben den einzelnen Schichten mehr Ressourcen und Möglichkeiten zur Lastverteilung. Jedoch können sie Kosten, Komplexität sowie den Kommunikationsaufwand zwischen den Schichten erhöhen.

Werkzeuge für Performanz Tests

Eine wichtige Entwurfsentscheidung ist die Wahl der einzusetzenden Messwerkzeuge, die zusätzlich auch als Lastgenerator sowie Analyse-Funktionalitäten besitzen. Das Werkzeug kann ohne viele Rechner die Anzahl an Threads, virtuelle Nutzer, erzeugen, die in einem Pool verschiedene Szenarien durchlaufen und zur Repräsentation der erzeugten Lastenmodelle werden. Automatisierte Tests und Frameworks haben, wie von Jiang prägnant zusammengefasst wird, zudem

den Vorteil, dass sie effizient, gut wiederholbar und nachvollziehbar testen sowie menschliche Fehler minimieren. [12, S. 57.] Die Verwendung von Mess-Software oder implementierter -Verfahren in der Web-Anwendung können die Ergebnisse verfälschen und sollten auf einem eigenen System ausgeführt werden.

Drei wesentliche Punkte, welche die Wahl der Mess-Software bestimmen sind zum einen die Kosten, zum anderen die Unterstützung der zu testenden Technologien sowie die gebotenen Funktionalitäten. Ein erstes Beispiel ist das Open-Source-Testwerkzeug Apache JMeter: „[It] may be used to test performance both on static and dynamic resources (files, Servlets, Perl scripts, [...]). It can be used to simulate a heavy load on a server, network or object to test its strength or to analyze overall performance [...].“[13] JMeter besitzt Einschränkungen, z.B. führt es JavaScript nicht innerhalb von HTML-Seiten aus.[13]

Nach einer Beispiel-Anfrage durch den Entwickler werden die ausgeführten Operationen aufgelistet, die einzeln selektiert und in einem Pool von mehreren Threads ausgeführt werden können. Die Ergebnisse werden durch zugeschaltene „Listener“ zusammengefasst ausgewertet.[5]

Ein zweites Beispiel ist der kostenpflichtige Loadrunner von HP in der Version 11.0. Neben den bereits genannten Funktionalität bietet er die Möglichkeit der Live-Aufnahme von Nutzerverhalten zur Abbildung des Lastmodells. Er beinhaltet auch einen Monitor, um z.B. die Auslastung des Systems während des Ablaufs zur Anwendungszeit zu verfolgen.[6] Der HP-Loadrunner ist für das Performanz-Testen verschiedener Technologien breit aufgestellt, um viele verschiedene Web-Anwendungen zu erfassen.[7]

Anhand des HP-Loadrunners kann der Einfluss der Problemstellungen z.B. beim Testen einer Web 2.0-Anwendung die Wahl eines Performanz-Test-Werkzeug erklärt werden. Zum einen können Eigenschaften des Internets wie Latenzzeiten durch einen WAN-Emulator simuliert, zum anderen verschiedene Knoten und Code-Abschnitte überwacht und analysiert werden. Mithilfe s.g. Stubs können Dienste anderer Anbieter ohne reale Nutzung verwendet werden.[8]

Diese Herausforderungen zeigen, dass ein Werkzeug je nach unterstützter Technologie, Kosten und bereitgestellten Funktionalitäten ausgewählt wird. Apaches JMeter ist eine Open-Source-Variante, während der HP-Loadrunner ein Beispiel für eine kostenpflichtige, funktionsmächtige Anwendung für das Performanz-Testen bei Web-Anwendungen darstellt.

Caching und Pooling

Caching- und Pooling-Mechanismen verbessern Antwortzeiten und Auslastung von Web-Anwendungen. Ceri(et al.) behandelt diese Konzepte als Möglichkeit zum „Tunen“ der Anwendung.[11, vgl. S. 351.] Dabei können diese Mechanismen als eine Alternative zu kostenaufwendingen Replikationen genutzt werden.

[5] Die Beschreibung wurde anhand eines JMeter-Tutoriums erstellt. Siehe [14].
[6] Die Beschreibung wurde anhand eines Loadrunner-Tutoriums erstellt. Siehe [15].
[7] Für eine Übersicht der unterstützen Web-Technologien siehe [16].
[8] Das Beispiel wurde anhand des HP-Loadrunners zusammengefasst. Siehe [17].

Pooling unterstützt dies, indem verschiedene Verbindungen aufgebaut und in s.g. Pools gebündelt werden, um bei einer Anfrage sofort genutzt zu werden. Casteleyn fügt an, dass diese Bündelung unabhängig von der Nutzung erfolgt und diese Ressourcen nicht zerstört werden, sondern weiterhin Teil des Original-Pools bleiben.[3, vgl. S. 272.] Der Aufbau und die Lösung einer Verbindung würde viel Zeit beanspruchen. Andreolini erläutert ein Beispiel: „The TCP connections [...] are typical resources handled through a pool, because they are expensive to setup and destroy.“[9, S. 204.]

Ein Beispiel eines Pooling-Mechanismus ist das „Connection Pooling“ von IBM für die DB2-Verwendung mittels JDBC und SQLJ. Der IBM Data Server Treiber beinhaltet Operationen, die auf eine Anfrage eines WebSphere Application Server hin, eine bestehende Verbindung zum Datenbestand zurückgibt.[18, vgl. S. 575.] Beim Schließen der Verbindung wird diese an den Pool zurückgesendet und steht zur Wiederverwendung bereit. Somit spart Connection Pooling Zeit beim Aufbau von Verbindungen.

Beim Caching werden häufig verwendete Daten für einen schnellen Zugriff im Zwischenspeicher gehalten und deren erneute Abfrage und damit verbundenen Operationen vermieden. „In this way, only a fraction of the incoming requests [...] is served in realtime by the application.“[11, S. 352.] Zudem sollte ermittelt werden, wo in der Architektur und wie Caching eingesetzt werden kann.

Ein Beispiel sind „Content Delivery Networks“ (CDN), genauer Limelight Deploy von Limelight Networks. Auf ein Anfrage des Clients hin und wird nach einem schnell antwortenden, geographisch nah liegenden „Edge Cache Server“ gesucht, der den gewünschten Inhalt aus dem Cache an den Client sendet. LimeLight Deploy bietet verschiedene Dienste wie „video platform, web content management [...]“[19] Vor allem statischer Inhalt sollte im Cache gehalten und Mechanismen zu Aktualisierung, z.B. durch s.g. „expiration rules“, bereitgestellt werden. CDNs verringern die Auslastung des eigenen Servers.

Pooling und Caching sind Mechanismen, die durch Entwurfsentscheidungen wie Technologie beschränkt werden, jedoch Antwortzeit und Auslastung verbessern. Dabei sollte definiert sein, was und wo im Pool oder Cache gehalten wird.

Design Pattern

Eine weiterer Faktor für die Planung oder der Software ist der Einsatz von „Design-Pattern“(DP). Ein Beispiel hierfür ist das Façade-DP. Es verkapselt Bibliotheken und Komponenten, stellt häufige Operationen per Schnittstelle bereit und minimiert Abhängigkeiten. Ein Beispiel für die Nutzung stellt die Studie von Rudzki(et al.) dar. Der Vergleich verdeutlicht das Verhalten verschiedener DP in Bezug auf die Performanz von J2EE- und .NET-Anwendungen, wenn es um rechnerferne Zugriffe auf Objekte geht.[20, S. 4.] Das Façade-DP zeigt für die J2EE-Anwendung eine gute Antwortzeit trotz steigender Anzahl an gleichzeitigen Threads. Dafür zeigen die beiden anderen Muster, das „Command“- und „Combined-Command-Pattern“ einen besseren Durchsatz.[20, vgl. Figur 4, S. 4.] Das Beispiel des Façade bei J2EE-Anwendungen zeigt, dass DP die Performanz verbessern können, jedoch bzgl. der Anwendbarkeit und „Trade-Offs“ kritisch beurteilt werden sollten.

3.4 Implementierung

Die Implementierung einer performanten Web-Anwendung ist im hohen Maß von Erfahrung und Wissen der Entwickler sowie der eingesetzten Technologien abhängig. Daher wird in diesem Abschnitt nur die Herausforderungen der Performanz-Tests an die Programmierung der Web-Anwendung erläutert, da sie zur späteren Messung der Performanz benötigt werden könnten.

Die Mess-Software sollte auf einen eigenen Server liegen, um die Ergebnisse der zu testenden Applikation nicht zu verfälschen. Ebenfalls sollten viele Zeilen Mess-Code innerhalb der Web-Anwendung vermieden werden. „Die Überfrachtung des Anwendungscodes mit Messanweisungen führt zu unerwünschten Abhängigkeiten bei der Releasebildung oder aber zu einer mangelhaften Pflege des Messcodes.“[5, S. 161.] Eine Veränderung der Implementierung macht eine Anpassung der Messpunkt notwendig. Außerdem sollte klar sein was und wo gemessen wird, um unnötige Mess-Punkte zu vermeiden.

Mess-Werkzeuge können Optionen besitzen, Messpunkte innerhalb von Komponenten zu- oder abzuschalten. Ein integriertes Werkzeug sollte ebenfalls abschaltbar sein. Schmalenbach nennt noch einen weiteren Punkt: „Wenn nach der Inbetriebnahme der Anwendung schließlich doch ausgereifte Systeme zur Performanceüberwachung eingesetzt werden, sind die fest kodierten Performancemesspunkte im Betrieb obsolet geworden.“[5, S. 162.]

3.5 Performanz-Tests

Performanz-Tests bilden die Kernaktivität des Performanz-Sicherung, um die Kenngrößen zu messen, an denen Performanz-Anforderungen verifiziert werden können. Pressman(et al.) sieht zwei Zwecksetzungen: „(1) to understand how the system responds as loading increases [...] and (2) to collect metrics that will lead to design modifications to improve performance.“[2, S. 389f.] Zunächst wird die Web-Anwendung als „black-box“ getestet und der Fokus anhand der Messdaten auf bestimmte Komponenten gesetzt, um Rückschlüsse auf deren Performanz durch „white-box“-Tests zuzulassen.

Anhand der Arbeitslastmodelle, SLAs oder der Anforderungen müssen als Vorbereitung die Testdaten definiert werden, die bestimmte Testfälle abdecken sollen. Ein Beispiel-Test-Fall könnte z.B. die Anmeldung vieler Nutzer bei einem Picasa-ähnlichen Web-Dienst zur Hauptnutzungszeit sein. Ein Arbeitslastmodell, das diese häufigen Anmeldungen mit anderen Szenarien in einem realitätsnahen „Mix“ kombiniert könnte dann die Testdaten liefern, mit denen die Lastgeneratoren konfiguriert werden. Anschließend sollte definiert werden, welche Komponenten des Systems, z.B. die CPU-Auslastung des Datenbank-Servers, betrachtet werden sollen.

Die Durchführung Tests kann anhand von Metriken festhalten werden. Meier bemerkt, dass Tests nicht nur ausgeführt, sondern per Monitor überwacht werden sollten, um unschlüssiges Systemverhalten zu entdecken.[7, vgl. S. 166.] Engpässe, s.g. „bottlenecks“, lokalisieren sich in den Komponenten und Subsysteme, die zuerst eine Sättigung oder negative Performanz aufweisen. Aufschluss

bietet die Analyse der dokumentierten Testresultate, um Optimierungen durchzuführen, genauere Tests zu planen oder den Engpass besser einzugrenzen.

Nach diesem generellen Überblick der Voraussetzungen werden Last- und Stress-Test zur Ermittlung der Performanz bei Web-Anwendungen erläutert. Anhand der Monographie *Performance Testing Guidance for Web Applications* wird ein beispielhaftes Vorgehen beschrieben. Dabei wird der Kapazitätstest nicht erläutert. Bei ihm handelt es sich um die Möglichkeit rechnerisch das Systemverhalten bei zukünftigen Lasten zu ermitteln.[9]

Last-Test

„To verify application behavior under normal and peak load conditions. [...] A load test enables you to measure response times, throughput rates, and resource-utilization levels [...]."[7, S. 29.] So können obere Lastgrenze und Engpässe ermittelt werden, Schwachstellen beseitigt und Anforderungen erfüllt werden. Die Herangehensweise für Last-Tests umfasst acht Schritte, wurde jedoch zur Hälfte unter „Anforderung" und „Arbeitslastmodelle" erläutert. Daher beginnt der beschriebene Prozess bereits mit der Erzeugung von Metriken, die bei den verschiedenen Testdurchläufen helfen, die Messdaten vergleichbar zu machen. Diese können, wie Meier vorschlägt, auch spezifisch wie z.B. Netzwerk-, Plattform- (z.B. J2EE) oder Service-Level-Metriken sein.[7, vgl. S. 210.] Eine Beispiel-Metrik für die Anmeldung an einer Web-Seite könnte folgendermaßen aussehen:

Kenngröße	**Messwert**	**Ziel**
Antwortzeit bei $Threads_{max}$	7.5 s	9.0 s
Durchsatz bei $Threads_{max}$	245	250
CPU-Auslastung des Web-Servers bei $Threads_{max}$	69%	70%

Tabelle 1. Beispiel Metrik zur Erfassung relevanter Performanz-Kenngrößen

Hier sind bereits Messwerte eingetragen, obwohl der Test noch nicht durchlaufen wurde. Die durch SLA oder Anforderungen definierten Zielwerte können auch durch die eigenen Ziele der Entwickler ersetzt werden. Es ist erkennbar, dass das System bei $Threads_{max}$ das Ziel für den Durchsatz nicht erreicht.[10]

Als sechsten Schritt werden nun spezifische Tests erzeugt, die vor allem die Zielsetzung haben, die Informationen zu erfassen, die zum Verstehen, zur Auswertung und Verbesserung der Anwendung benötigt werden.[7, vgl. S. 210.] Die Messdaten sollen zu einer höheren Genauigkeit der Untersuchungsergebnisse führen, um einen Engpass schneller ermitteln zu können. Dafür werden die Arbeitslastmodelle implementiert oder mithilfe der Lastgeneratoren abgebildet. Sollten keine Messwerkzeuge wie der Apache JMeter genutzt werden, müssen die Messpunkte mit ihren Werten geloggt und gespeichert werden.

[9] Ein Beispiel der Performanz-Voraussage siehe [21].
[10] Für weitere Beispiel von Performanz-Metriken siehe [7, S. 217f.].

Zur Durchführung des Last-Tests werden sowohl die zu testende Web-Anwendung als auch die Mess-Werkzeuge sowie Generatoren eingestellt .[7, vgl. S. 211.] Z.B. könnte ein Dienst fehlen, der dann per Stub simuliert werden müsste. Wenn beide Systeme konfiguriert sind, wird der Last-Test mit langsam steigenden Werten begonnen. Damit kann das Verhalten auch bei Zwischenwerten betrachtet werden. Meier nennt weitere Gründe zur langsamen Steigerung der Last-Ebene: „Begin load testing with a small number of users [...] and then incrementally increase the load. It is important to allow time for the system to stabilize between increases in load while evaluating the correctness of the simulation.“[7, vgl. S. 211f.] Die Last wird soweit erhöht, bis der Schwellwert ermittelt werden kann. Sollten früh Fehler auftreten, kann der Test weiterlaufen, sofern sicher ist, dass die weiteren Ergebnisse nicht verfälscht werden.

Anhand von Server-Logs oder einem Monitor-Programm können die Ergebnisse ausgelesen und durch Graphen visualisiert werden. Im Folgenden werden als Beispiel die drei Kenngrößen des Tests gezeigt. Die hier genormten Daten des Stress- und Last-Test wurden zur Vereinfachung in einen Graphen integriert. Hierbei kann eine Ziel-Linie eingezeichnet werden, um einen visuellen Vergleich zu Messkurve zu ermöglichen.

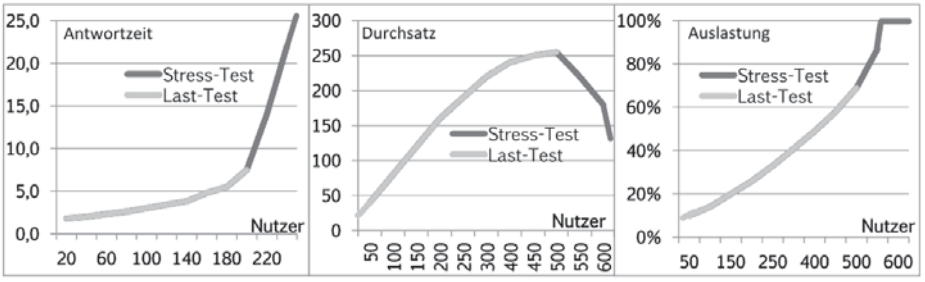


Abbildung 3. Antwortzeit (in Sek.), Durchsatz (verarb. Anfragen pro Sek.) und CPU-Auslastung eines Web-Servers in Prozent.

Der Schwellwert zur Einhaltung der Performanz-Ziele könnte z.B. bei 190 Besuchern liegen und wäre damit erfüllt. Jedoch könnte bei einem Durchsatz von 250 Anfragen pro Sekunde und insgesamt 500 gleichzeitigen Nutzern eine Nichteinhaltung der SLA bedeuten und müsste genauer untersucht werden. Der Graph der Auslastung zeigt, dass der CPU bei 500 Nutzern zu ca. 70% ausgelastet ist. Obwohl damit eine Anforderung verifiziert werden könnte, kann es sein, dass zu wenig Puffer für Interaktionen weiterer Nutzer besteht. Dies könnte dann durch einen Stress-Test genauer betrachtet werden.

Der letzte Schritt vor der erneuten Ausführung weiterer oder genaueren Definition von Tests besteht aus der Auswertung der Daten und dem Festhalten der Ergebnisse in Berichten. Die Analyse-Metriken mit den Messwerten sollte helfen, spezifischere Tests zu durchlaufen, sollte ein Engpass absehbar sein. Meier

verdeutlicht dies an einem weiteren Beispiel: „[During] the first iteration of load tests, the process shows a marked increase in memory consumption, indicating a possible memory leak. In the subsequent iterations, additional memory counters related to generations can be captured to study the memory allocation pattern [...].“[7, S.212] Die Erkenntnisse können bei einer Code-Review eingebracht werden oder zur Umgestaltung des System-Entwurfs führen. Erste Konzepte wurden unter „Entwurfsphase“ genannt und weitere werden später erläutert.

Zusammenfassend werden beim Last-Test die bereits erwähnte Erhebung der Anforderung und die Arbeitslastmodelle verwendet, um das System zu testen. Diese werden als Input genutzt, um die Metriken aufzubauen, die die Kenngrößen beinhalten. Der aufgebaute Testfall wird durchlaufen, analysiert und sollte als Output die Verbesserung der Testpläne ermöglichen und Engpässe aufzeigen.

Stress-Test

Der Stress-Test ist die Weiterführung der Last-Tests über Schwellwerte hinaus. Dabei wird beobachtet, wie sich das System bei einer Überlast verhält, z.B. ob es komplett ausfällt oder trotz völliger Auslastung weiterarbeitet. Meier definiert sie wie folgt: „[...] determining or validating performance characteristics of the system or application under test when subjected to conditions beyond those anticipated during production operations. [...] These tests are designed to determine under what conditions an application will fail, how it will fail, and what indicators can be monitored to warn of an impending failure.“[7, S. 21.] Ein Stress-Test zeigt, ob zu Engpässen auch Bugs hinzukommen, die z.B. auf Synchronisationsprobleme hindeuten und die Performanz verschlechtern.

Der Prozess zur Nutzung von Stress-Tests ähnelt dem nach Meier beschriebenen für Last-Tests. Wie zuvor werden zuerst die Ziele anhand der Anforderungen, SLAs etc. definiert, dann die häufigen und Daten-intensiven Szenarien ausgewählt. Im Aufbau der Arbeitslast kann vom normalen Herangehen abgewichen werden, indem auch Anti-Profil-Modelle erzeugt werden. Hierbei werden nach Meier die Anteile der Szenarien am Modell invertiert.[7, vgl. S. 217.] Z.B. würde das Herunterladen eines Web-Albums bei einem Piacasa-ähnlichen Web-Dienst im normalen Arbeitlastmodell ca. 24% ausmachen. In einem Anti-Profil würde dieses Szenario nun 76% betragen. Zudem kommt zu diesem Anti-Profil eine Mischung anderer Szenarien dazu. „Using an anti-profile can serve as a valuable starting point for your stress tests because it ensures that the critical scenarios are subjected to loads beyond the normal load conditions.“[S. 217.][7]

Wie bei den Last-Tests müssen in einem vierten Schritt Metriken aufgebaut werden, die die Messdaten im Verhältnis zu der zu testenden Komponente stellen und Kenngrößen zuweisen. Hierbei könnten die ermittelten Schwellwerte des Last-Tests mit aufgenommen werden, um sie als Fixpunkt zu nutzen.

In einem fünften Schritt werden die Testfälle ähnlich der Last-Tests erzeugt. Jedoch kann der Test-Zweck fehlen, der vorher durch SLAs, Anforderungen und weitere Zielsetzungen definiert war. Meier erklärt, dass der Test-Entwurf dokumentieren sollte, was zu erwartende Resultate und/oder Schlüsseldaten sind, so dass der Test als „bestanden“„fehlgeschlagen“ oder „ergebnislos“ deklariert werden kann.[7, vgl. S. 218.] Bevor die Web-Anwendung einem Stress-Test un-

terzogen wird, muss neben der bereits erwähnten Konfiguration und Einspeisung der Testdaten sichergestellt werden, dass das System mit den Lastgeneratoren die Lasten erzeugen können oder nicht selbst überlasten.[7, vgl. S. 219.]

Die Resultate werden in einem siebten Schritt ebenfalls analysiert und dokumentiert. Die Graphen im Abschnitt der Last-Tests beinhalten Beispiel-Ergebnisse eines Stress-Tests. Hier zeigt sich bei der Anmeldung an einer Webseite, dass die Kurve nach dem Schwellwert des Last-Test fast exponentiell wächst. Mehr relevante Messbereiche könnten z.B. die ersten Werte nach dem Schwellwert von 190 Threads sein, um den Puffer bis zu inakzeptablen Werten zu ermitteln.

Ebenso könnte dies beim Durchsatz geschehen, um zu ermitteln, wann die Web-Anwendung gegen SLAs verstößt. Dieser Punkt könnte bei 500 Nutzern erreicht sein, da die Web-Applikation nur noch ca. 230 Anfragen pro Sekunde beantworten kann. Die Auslastung des Web-Server-CPUs deutet daraufhin, dass das System bei 550 sich anmeldenden Nutzern zu ca. 75% ausgelastet ist, was laut einer definierten Anforderung noch innerhalb eines akzeptablen Bereichs liegt. Der Graph zeigt deutlich, dass bei einer steigenden Anzahl über 550 Nutzern die Auslastung schnell 100% bemisst und das System zusammenbrechen könnte.

Es muss abgewogen werden, bestimmten Ursachen eines solchen Zusammenbruchs nachzugehen, da eine Überlast auch sehr seltene Fälle abdeckt. Meier warnt vor solchen Problemen: „In situations where performance issues are observed, but only under conditions that are deemed to be unlikely enough to warrant tuning at the current time, you may want to consider conducting additional tests to identify an early indicator for the issue in order to avoid unwanted surprises.“[S. 219.][7] Der Entwickler sollte seine Einschätzungen und Entscheidungen dokumentieren. Wenn das System weiter gestresst wird, können in weiteren Iterationen die Metriken identifiziert, die Testfälle erzeugt, die Last simuliert und die Resultate analysiert werden.[7, vgl. Graphik „Stress Testing Steps“ S. 215.]

Der Stress-Test ist eine Weiterführung der Last-Tests über die Schwellwerte hinaus. Als Input dienen die Erkenntnisse der Last- und vorangegangene Stress-Tests sowie Szenarien und Arbeitslastmodelle.[7, vgl. S. 214.] Die Arbeitslastmodelle können z.B. durch Anti-Profile angepasst werden. Als Output werden weitere Engpässe und Bugs sichtbar und die Stress-Testabläufe verbessert.

3.6 Wartung und Weiterentwicklung - Tuning

Die ausgelieferte Web-Anwendung kann z.B. durch stark erhöhte Nutzung an ihre Grenzen stoßen. Neuentwicklungen oder zu analysierendes Bestandssystem können gleichermaßen optimiert werden. Beim s.g. „System Scale-up“ wird, wie es Andreolini(et al.) darstellt, ein ausgelasteter Knoten innerhalb der Architektur anhand von white-box-Tests ermittelt und die Hardware aufgerüstet.[9, vgl. S. 204] Die Replikation kann auch mit einem s.g. „Scale-out“ erfolgen, wobei einer Schicht weitere Knoten hinzugefügt werden.[9, vgl. S. 204f.] Solche Aufrüstungen können ohne Analysen jedoch unnötige Kosten erzeugen, daher sollten die Optimierung bereits bestehender Komponenten Vorrang haben. Erste Verbesserungen sind die im Abschnitt „Entwurfsphase“ vorgestellten Konzep-

te, deren Wirkung weitere Performanz-Tests zeigen. Im Folgenden wird je ein generelleres Beispiel für Front- und Back-End einer Web-Anwendung dargestellt.

Frontend - HTTP
Transaktionen per Hypertext-Übertragungsprotokoll (HTTP) benötigen aufgrund synchroner Bearbeitung viel Zeit. Je höher die Anzahl an z.B. GET- und POST-Aufrufen ist, desto mehr Kommunikationsaufwand besteht und umso länger dauert der Aufbau z.B. einer Web-Seite. Ein Beispiel der Verwendung von HTTP-Operationen bei Webseiten zu vermeiden findet sich in Souders Monographie zur Web-Seiten-Optimierung. Er schlägt vor, die Anzahl der zu ladenen Dateien zu minimieren, um weitere Anfragen an den Server zu vermeiden.[22, vgl. S. 15.] Die Skripte einer Webseite müssten nicht anhand hoher logischer Kohäsion modularisiert werden, sondern sich an den zu ladenden Programmteilen der Web-Anwendung orientieren. Souders sieht eine mögliche Lösung darin, „to follow the model of compiled languages and keep JavaScript modular while putting in place a build process for generating a target file from a set of specified modules.“[22, S. 16.] Dies bedeutet, Kombinationen aus zu ladenen Dateien zusammenzufassen, die als eine Datei geladen werden. Dies kann zu einer Beschleunigung der Ladezeit um bis zu 38% führen und stellt nur einen Teil von Souders vorgeschlagenen Verbesserungen dar.

Backenend - Datenbank-Optimierung
Datenbanken sind Performanz-kritische Elemente von größeren Web-Anwendungen. Je mehr Anfragen an die Datenbank gestellt werden, um so kritischer können sich ineffiziente Zugriffe oder Konfigurationen auf die Performanz auswirken. Thomas Kurdaß stellt in seiner Monographie *Taschenbuch Datenbanken* verschiedene Phasen zur Optimierung von Datenbanken vor. [23, vgl. S. 304f.] Die zweite, dritte und vierte Phase befasst sich mit Optimierungen, die nur die Datenbank betreffen. Deren Beitrag zur Performanz kann enorm sein, jedoch wird nur die anwendungsorientierte Optimierung dargestellt.

Eine Möglichkeit stellt die Reduzierung von Sekundärspeicherzugriffen dar.[23, vgl. S. 301 u. 314] Es sollten nur Daten in den Hauptspeicher geladen werden, die auch Verwendung finden. Wie Kudraß erklärt, besitzen Datenbanken einen Optimierer, der Anfragen verbessert, jedoch sollte sicher sein, dass die Anfrage korrekt formuliert ist.[23, vgl. S. 317] Anfragen mit Sortierungen, sollten vermieden werden, da sie je nach Größe der Tabellen Zeit kosten. Kudraß erläutert dies anhand von Anfragen durch das DMBS. [23, vgl. S. 319f] Die Kommunikation mit der Datenbank und die Verarbeitung der Anfragen können kostenintensive Operationen sein. Es ist daher wichtig, den Einfluss der Web-Anwendung auf seine Datenbank zu beobachten.

4 Ausblick

Beim Überblick über Tätigkeiten zum Aufbau und Sicherung der Performanz bei Web-Anwendungen, wurden Bereiche wie z.B. die Berechnung des Laufzeitverhaltens der Warteschlangen, Module oder der Gesamtanwendung z.B. durch

Automaten oder weiterer Formeln aufgrund des Platzes ausgeklammert. Ein weiteres Thema könnte die Evaluation bestehender Performanz-Werkzeuge sein.

Literatur

1. Kotsis, G.: Performanz von Web-Anwendungen. In Kappel, G., ed.: Web engineering. Dpunkt, Heidelberg (2004) 297–318
2. Pressman, R.S., [et al.]: Web engineering: A practitioner's approach. McGraw-Hill Higher Education, Boston (2009)
3. Casteleyn, S.: Engineering Web applications. Springer, New York (2009)
4. Trent, S., [et al.]: Performance comparison of PHP and JSP as server-side scripting languages. In Issarny, V., [et al.], eds.: Middleware. Springer, Berlin (2008) 164–182
5. Schmalenbach, C.: Performancemanagement für serviceorientierte Java-Anwendungen. Springer, Berlin (2007)
6. Sommerville, I.: Software engineering. 9 edn. Pearson, Boston (2011)
7. Meier, J., [et al.]: Performance Testing Guidance for Web Applications: patterns & practices. Microsoft Corporation (2007)
8. Cockburn, A.: Writing effective use cases. Addison-Wesley, Boston (2001)
9. Andreolini, M., [et al.]: Web System Reliability and Performance. In Mendes, M.E.X., [et al.], eds.: Web engineering. Springer, Berlin (2006) 181–217
10. Standard Performance Evaluation Corporation: SPECweb2005 Benchmark. Unter www.spec.org/web2005, zuletzt geprüft am 13.01.2012
11. Ceri, S., [et al.]: Designing Data-Intensive Web Applications. Elsevier Science & Technology Books, San Diego (Dec. 2002)
12. Jiang, G., [et al.]: A Quick Testing Model of Web Performance Based on Testing Flow and its Application. In: Sixth WISA. IEEE, Los Alamitos (2009) 57–61
13. The Apache Software Foundation: JMeter. Unter http://jmeter.apache.org, zuletzt geprüft am 25.02.2012
14. Youtube: jmeter getting started. Unter http://www.youtube.com/watch?v=KI6u5pclYIw zuletzt geprüft am 21.02.2012
15. HP: Performance center and Loadrunner 11.0 Protocol Bundles. Unter http://h20195.www2.hp.com/V2/GetPDF.aspx/4AA1-3191ENW.pdf, zuletzt geprüft am 21.02.2012
16. HP: LoadRunner Product Walkthrough. Unter http://h20621.www2.hp.com/video-gallery/us/en/a559e7eb609fd4b9d530795d989af630d8d203e5/r/video, zuletzt geprüft am 25.02.2012
17. HP: Load Testing Web 2.0. Part 1-4. Unter www8.hp.com/us/en/software/software-product.html?compURI=tcm:245-935779#tab=3, zuletzt geprüft am 21.02.2012
18. IBM: JDBC and SQLJ connection pooling support. Unter http://publib.boulder.ibm.com/epubs/pdf/dsnjvm04.pdf, zuletzt geprüft am 03.03.2012
19. Limelight Networks: Limelight Deploy. Unter www.limelight.com/managed-content-delivery, zuletzt geprüft am 4.3.2012
20. Hierons, R., [et al.]: Performance implications of design pattern usage in distributed applications. In Hieron, R., [et al.], eds.: ISSTA. ACM Press, New York (2006) 1–11
21. Cheung, L., [et al.]: A Study of Web Services Performance Prediction: A Client's Perspective. In: MASCOTS. IEEE, Los Angeles (2011) 75–84
22. Souders, S.: High performance web sites: Essential knowledge for frontend engineers. O'Reilly, Beijing (2007)
23. Kudrass, T.: Taschenbuch Datenbanken. Carl-Hanser, München (2007)

Sicherheit von Webanwendungen

Michael Großmann

Otto-Friedrich-Universität Bamberg, Lehrstuhl für Medieninformatik
michael-alfons.grossmann@stud.uni-bamberg.de

Zusammenfassung. Ziel der vorliegenden Arbeit ist es neben Klärung grundlegender Begriffe zur IT-Sicherheit, wie Authentizität, Vertraulichkeit, Integrität, Verbindlichkeit und Verfügbarkeit, ein Ebenenmodell des Bundesamtes für Sicherheit in der Informationstechnik(BSI) zur ganzheitlichen Betrachtung der Sicherheit von Webanwendungen vorzustellen. Anschließend wird ein Überblick über aktuelle Sicherheitsrisiken in Webanwendungen gegeben, wobei hier der Fokus auf SQL-Injection und Cross-Site-Scripting liegen soll, da diese die zwei häufigsten Sicherheitsrisiken für Webseiten darstellen, wie sie im *OWASP Top 10 Report* des Open Web Application Security Project (OWASP) veröffentlicht wurden. Der Leser soll ein Gespür für die vorgestellten Sicherheitsrisiken, sowie mit Hilfe des Ebenenmodells ein Denkmodell an die Hand bekommen, um systematisch über die Sicherheit in eigenen Webanwendung nachdenken zu können.

Schlagworte: IT-Sicherheit, Webapplikationssicherheit, IT-Security, SQL-Injection (SQLi), Cross-Site-Scripting (XSS), BSI-Ebenenmodell, Schutzziele, Authentizität, Vertraulichkeit, Integrität, Verbindlichkeit, Verfügbarkeit

1 Einleitung und Zielsetzung

"2011 wird zum Jahr der Hackerangriffe. Ob Pentagon oder Zoll, Sony oder Rewe- vor den Angriffen aus dem digitalen Untergrund scheint niemand sicher. Die Hacker attackierten Firmen wie Staaten. 2011 gab es so viele Fälle wie noch nie." [1]. Dies sind die Anfangszeilen einer der vielen Schlagzeilen der letzten Wochen. Es vergeht kaum ein Tag, an welchem keine neuen Hiobsbotschaften über geknackte Kundenkonten, gehackte Websites oder großangelegte Hackerangriffe in den Fokus der Öffentlichkeit und Politik geraten. Mitverantwortlich an diesem Chaos sind mitunter sogenannte Hacktivistengruppen wie Lulzsec oder Anonymous, die immer wieder mit scheinbarer Leichtigkeit die Sicherheitsvorkehrungen von Webseiten, sogar mancher namhafter IT-Sicherheitsfirmen, umgehen und so für oben genannte Schlagzeilen sorgen konnten. Wie kann es solchen Gruppen gelingen über die Webseiten der Unternehmen an sensible Informationen, wie beispielsweise Kreditkartendaten, zu gelangen und noch wichtiger, gibt es Möglichkeiten, wie man sich davor schützen kann?
Im Folgenden soll versucht werden einen Überblick über die Sicherheit von Webanwendungen zu geben. Neben der Klärung grundlegender Begriffe über die

Schutzziele von Webseiten, soll ein Ebenenmodell des Bundesamtes für Sicherheit in der Informationstechnik (BSI) vorgestellt werden, welches als "Denkmodell" dabei helfen soll, auf verschiedenen Ebenen ganzheitlich über die Sicherheit einer Webseite nachzudenken. Schließlich soll der Schwerpunkt der Arbeit darauf liegen zwei weitverbreitete Angriffsmethoden auf Webseiten - SQL Injection und Cross-Site-Scripting - zu erklären und erste Möglichkeiten derer Vermeidung aufzuzeigen. Als Orientierung soll hierbei der *OWASP Top 10 Report, 2010* [2] dienen, welcher im dreijährigen Rhythmus von dem Open Web Application Security Project (OWASP) veröffentlicht wird und die häufigsten Sicherheitsrisiken für Webanwendungen aufzählt. Die Arbeit erhebt dabei keinen Anspruch auf Vollständigkeit, sondern versteht sich vielmehr als erster Überblick über die Sicherheit von Webanwendungen und hat das erklärte Ziel eine erste Intuition für diese zu vermitteln.

2 Erläuterung wichtiger Grundbegriffe der IT-Sicherheit - Schutzziele

In unserer heutigen Informationsgesellschaft stellen Informationen und Daten ein wichtiges und schützenswertes Gut dar. So soll es beispielsweise bei einem Webauftritt nicht jedem gestattet sein, gewisse Informationen zu lesen oder gar verändern zu können. Informationssysteme müssen folglich Mechanismen bereitstellen den Zugriff auf diese Daten und Informationen zu beschränken bzw. zu kontrollieren. In diesem Zusammenhang wird häufig auch von Schutzzielen gesprochen. Bei Schutzzielen handelt es sich generell um sicherheitsrelevante Grundsätze zum Schutz von Daten und Informationen. Sie stützen sich auf die fünf Dimensionen Vertraulichkeit, Integrität, Authentizität, Verbindlichkeit und Verfügbarkeit und sollen im Folgenden näher erläutert werden. Eine Synthese dieser fünf Dimensionen kann kurz in der Frage "Wer darf was" zusammengefasst werden [3, S. 321]. Eine mögliche Antwort auf diese Frage zu geben, ist Hauptaufgabe der IT Sicherheit.

Authentizität (Authenticity) Unter der Authentizität eines Objektes bzw. Subjekts versteht man dessen Echtheit und Glaubwürdigkeit, welche anhand einer eindeutigen Identität und charakteristischen Eigenschaften überprüfbar ist [4, S. 6f.]. Daten sind demnach authentisch, wenn gewährleistet werden kann, dass sie auch wirklich vom genannten Absender stammen [3, S. 320]. Die Authentizität eines Nutzers wird in herkömmlichen Systemen meist, bei Anmeldung am System, mit einer Kombination aus Nutzerkennung, als eindeutiger Identität und Passwort überprüft. Im besten Falle sollte es einem Angreifer dadurch unmöglich gemacht werden Daten oder Informationen unter falschem Namen einzustellen bzw. zu verändern.

Vertraulichkeit (Confidentiality) Der Schutz von Daten gegenüber unautorisierter Informationsgewinnung bzw. Kenntnisnahme seitens einer dritten Par-

tei, wird als Vertraulichkeit bezeichnet [4, S. 8f.]. Demnach soll zum Beispiel ein als "vertraulich" gekennzeichnetes Dokument nicht von unbefugten Personen eingesehen werden können.

Integrität (Integrity) Die Unversehrtheit, Unverfälschtheit und Korrektheit von Daten wird als (Daten-) Integrität bezeichnet. Daten gelten als integer, wenn sie durch unautorisierte Dritte nicht unbemerkt manipuliert werden konnten [3, S. 320]. Datenintegrität zu gewährleisten erfordert ein Rechtekonzept zur Nutzung der Daten auf dem Informationssystem beispielsweise in Form von Schreib- oder Leserechten [4, S. 8f.]. Einem Angreifer soll es somit nicht ermöglicht werden, Daten unautorisiert zu manipulieren.

Verbindlichkeit (Non-Repudiation) Verbindlichkeit ist dann gegeben, wenn Dokumente (Daten und Informationen) an den ursprünglichen Aussteller gebunden sind, sowie die Herkunft oder der Erhalt eines Dokuments im Nachhinein zweifelsfrei nachweisbar ist und nicht von einer der Parteien abgestritten werden kann [3, S. 320]. Dies ist insbesondere im Bereich des E-Commerce von Bedeutung, um die Rechtsverbindlichkeit geschäftlich durchgeführter Transaktionen eindeutig nachweisen zu können. Während Verbindlichkeit eine gewisse Ähnlichkeit zur Authentizität aufweist liegt der Fokus hierbei mehr auf der nachträglichen Beweisbarkeit gegenüber Dritten [4, S. 11f].

Verfügbarkeit (Availability) Unter der Verfügbarkeit von Informationssystemen versteht man allgemein den Schutz vor Angriffen, welche die Ressourcen des Informationssystems derart beeinträchtigen können, dass sie für berechtigte Nutzer nicht mehr benutzbar sind, sowie von Maßnahmen, die den Zugriff gewährleisten sollen [5, S. 15]. Ein Beispiele der Umsetzung solcher Maßnahmen könnte in der Konfiguration von Quotas bestehen, welche den Zugriff auf CPU-Zeit oder Speicher reglementieren und somit die Verfügbarkeit gewährleisten sollen.

3 Das Ebenenmodell des Bundesamtes für Sicherheit in der Informationstechnik (BSI) zur Sicherheitskonzeption von Webanwendungen

Während Last- und Funktionstests bei der Erstellung von E-Business Webanwendungen heutzutage als qualitätssichernde Maßnahmen häufig gut organisiert und fest verankert sind, verhält sich die Situation meist vollkommen anders, wenn es um Sicherheitsaspekte geht. Hier fehlt es oft an einer letzten Kontrollinstanz, die beispielsweise im Rahmen eines Penetrationstests sicherstellt, dass die betrachtete Webseite frei von typischen Sicherheitslücken ist.[6, S. 33] Andererseits wird Websicherheit häufig zu einseitig betrachtet und nur unter dem Gesichtspunkt fehlerhafter Programmierung und Konfiguration gesehen. Diese Implementierungsebene ist sicherlich wichtig, jedoch kann eine Webanwendung

erst hinreichend als sicher eingestuft werden, wenn noch weitere Ebenen untersucht worden sind [7, S. 1f.]. Das im Folgenden vorzustellende Ebenenmodell des BSI greift diesen Aspekt auf, indem es versucht ein generisches Modell vorzugeben, um über die Sicherheit einer Webanwendung ganzheitlich nachzudenken. Zu betonen gilt an dieser Stelle auch, wie es Bruce Schneier, ein anerkannter Sicherheitsguru, in seinem Buch *Secret and Lies* einmal treffend bemerkte: *"IT Sicherheit ist ein Prozess, kein Produkt"* [8, Vorwort S. x]. So, kann Sicherheit nicht einfach allein durch Zukauf eines einzelnen Produktes wie bspw. einer Firewall erreicht werden, sondern muss in die Prozesse und Workflows des Unternehmens integriert werden. Das Ebenenmodell soll hierbei helfen, das Problem Websicherheit auf verschiedenen Ebenen bzw. von verschiedenen Perspektiven aus zu betrachten. Es ist generisch, da es von Einzelheiten wie verwendete Technologien und Programmiersprachen sowie projektspezifische Erfordernissen abstrahiert. Diese müssen im einzelnen Projekt jeweils ergänzt und detailliert werden. Das Ebenenmodell gliedert Websicherheit in sechs Bereiche (vgl. Abbildung 1), welche nachfolgend im einzelnen kurz erläutert werden sollen [6] [7].

	Ebene	Inhalt (Beispiele)
6	Vorschriften und Bestimmungen	**Einhaltung gesetzlicher Regelungen und unternehmensspezifischer Vorgaben** - Fehlende Angaben zum Datenschutz - Nichteinhalten von Bestimmungen, z.B. des KontraG - Preisgabe vertraulicher Informationen
5	Semantik	**Schutz vor Täuschung und Betrug** - Informationen ermöglichen Social Engineering-Angriffe - Gebrauch von Popups u.ä. erleichtern Phishing-Angriffe - Keine Absicherung für den Fall der Fälschung der Website
4	Logik	**Absicherung von Prozessen und Workflows als Ganzes** - Verwendung unsicherer Email in einem ansonsten gesicherten Workflow - Angreifbarkeit des Passworts durch nachlässig gestaltete „Passwort vergessen"-Funktion - Die Verwendung sicherer Passworte wird nicht erzwungen
3	Implementierung	**Vermeiden von Programmierfehlern, die zu Schwachstellen führen** - Cross-Site Scripting - SQL-Injection - Session Riding - Information Disclosure
2	Technologie	**Richtige Wahl und sicherer Einsatz technischer Verfahren** - unverschlüsselte Übertragung sensitiver Daten - Authentisierungsverfahren, die nicht dem Schutzbedarf angemessen sind - Ungenügende Randomness von Token
1	System	**Absicherung der auf der Systemplattform eingesetzten Software** - Fehler in der Konfiguration des Webservers - „Known Vulnerabilities" in den eingesetzten Softwareprodukten - Mangelnder Zugriffsschutz in der Datenbank
0	Netzwerk & Host	Absicherung von Host und Netzwerk

Abb. 1. Ebenenmodell des BSI

Ebene 0: Netzwerk und Host Auf dieser Ebene geht es um die Absicherung von Host und Netzwerk, was meist in den Aufgabenbereich der Netzwerkadministration fällt. Diese Ebene ist eigentlich nicht direkt der Sicherheit der Webanwendung zugeordnet, wenngleich ein sicherer Host und Netzwerk unabdingbare Voraussetzungen dafür darstellen.

Ebene 1: System Auf Ebene eins geht es um die Absicherung der auf der Systemplattform eingesetzten Software, d.h. es beinhaltet all jene Software, die eine Webanwendung benötigt, um überhaupt ablaufen zu können. Hierzu gehören Web- und Applikationsserver aber auch Datenbanken, sowie beteiligte Backendsysteme. All diese Systeme müssen bei der Betrachtung der Sicherheit einer Webseite miteinbezogen werden. So kann es passieren, dass eine Webseite vielleicht frei von Sicherheitslücken ist, die Kommunikation zur Datenbank jedoch von Dritten einsehbar und manipulierbar ist und somit der Prozess als Ganzes als unsicher einzustufen ist.

Ebene 2: Technologie Auf Ebene zwei geht es um die richtige Wahl und den sicheren Einsatz technischer Verfahren, welche mit dem jeweiligen Zweck und Schutzbedarf der Webseite abzustimmen sind. So lassen zum Beispiel unverschlüsselte Übertragung sensitiver Daten oder Authentifizierungsverfahren, die nicht dem Schutzbedarf angemessen sind, auf einen falschen Einsatz von Technologien schließen.

Ebene 3: Implementierung Die Implementierungsebene ist die offensichtlichste Ebene und beinhaltet das Vermeiden von unbeabsichtigten Programmierfehlern, meist im Zusammenhang mit mangelhafter Validierung von Nutzereingaben, die zu Schwachstellen führen können, welche von Hackern ausgenutzt werden können. Im weiteren Verlauf der Arbeit soll der Fokus auf diese Ebene gerichtet werden, indem zwei typische Sicherheitsrisiken näher untersucht werden sollen.

Ebene 4: Logik Die Ebene der Logik beinhaltet Fragen rund um die Absicherung von Prozessen und Workflows als Ganzes. Sie wird in der heutigen Unternehmenspraxis leider häufig noch unzureichend wahrgenommen. So kann die Verwendung von Email als unsicherem Kommunikationsmittel in einem ansonsten gesicherten Workflow die Sicherheit der Webseite unter Umständen komplett aushebeln. Auch die Verhinderung der Verwendung unsicherer bzw. leicht zu erratender Passwörter, Rotationszyklen von Passwörtern, sowie Fragen, ab dem wievielten Versuch ein Nutzer nach mehrmaliger Falscheingabe seiner Nutzerdaten ausgesperrt werden soll, gehören in diesen Bereich.

Ebene 5: Semantik Auf der Ebene der Semantik geht es um den Schutz vor Täuschung und Betrug. Dabei geht es hauptsächlich um den Vertrauenskontext und die Interaktion mit dem Benutzer. Sorgloser Umgang innerhalb dieser Ebene könnte zur Folge haben, dass Dritte die Webanwendung missbrauchen können, um andere Nutzer zu täuschen oder zu betrügen. Angriffe die hier zu nennen sind, sind beispielsweise Social Engineering, Phising oder Identitätsdiebstahl. Meist werden solche Angriffe erst dadurch ermöglicht, dass nicht ausreichend darüber nachgedacht wird, wie und welche Inhalte/Informationen dem Benutzer auf der Webseite zugänglich gemacht werden und wie diese Inhalte unter Umständen dazu benutzt werden können, um Andere zu täuschen bzw. zu betrügen.

Ebene 6: Vorschriften und Bestimmungen Die letzte Ebene umfasst Fragen rund um die Einhaltung gesetzlicher Regelungen und Vorschriften. Hierunter fallen neben staatlichen Reglementierungen, beispielsweise in Form von Datenschutzbestimmungen, auch Regelungen aus dem eigenen Hause sowie von Seiten Dritter. Auch wenn ein neues Feature in der Webanwendung die Sicherheit erhöht, könnte es unter Umständen datenschutzrechtlich verboten sein. Diese Umstände gilt es auf dieser Ebene zu berücksichtigen.

4 Die häufigsten Sicherheitsrisiken von Webanwendungen und Strategien zu deren Vermeidung

4.1 Übersicht - OWASP Top 10 Report 2010

Die Sicherheit von Webanwendungen geht keineswegs nur die Betreiber von großen Online-Shops oder Banking-Portalen an. Auch kleinere oder privat betriebene Websites können Ziel böswilliger Angreifer werden, wobei hier meistens nicht monetäre Aspekte ausschlaggebend sind, sondern vielmehr den gekaperten Server für eigene Zwecke missbrauchen zu können, sei es als illegaler Fileserver, als Ausgangspunkt für Denial-of-Service Attacken oder zur Verbreitung von Spam-Mails [9]. Die Sicherheitslücken können dabei vielfältig aussehen. Aus diesem Grund veröffentlicht das Open Web Application Security Project (OWASP) bereits seit mehreren Jahren einen Report, in welchem sie die 10 kritischsten Sicherheitsrisiken für Webanwendungen auflistet. Bei der OWASP handelt es sich um eine offene Community, welche sich zum Ziel gesetzt hat Entwickler, Designer, Architekten und Führungskräfte über die Risiken der wichtigsten Schwachstellen von Webanwendungen aufzuklären[1]. Abbildung 2 zeigt eine Auflistung dieser Schwachstellen aus dem OWASP Top 10 Report 2010 [2, S. 4].

OWASP Top 10 – 2007 (Previous)	OWASP Top 10 – 2010 (New)
A2 – Injection Flaws	A1 – Injection
A1 – Cross Site Scripting (XSS)	A2 – Cross-Site Scripting (XSS)
A7 – Broken Authentication and Session Management	A3 – Broken Authentication and Session Management
A4 – Insecure Direct Object Reference	A4 – Insecure Direct Object References
A5 – Cross Site Request Forgery (CSRF)	A5 – Cross-Site Request Forgery (CSRF)
<was T10 2004 A10 – Insecure Configuration Management>	A6 – Security Misconfiguration (NEW)
A8 – Insecure Cryptographic Storage	A7 – Insecure Cryptographic Storage
A10 – Failure to Restrict URL Access	A8 – Failure to Restrict URL Access
A9 – Insecure Communications	A9 – Insufficient Transport Layer Protection
<not in T10 2007>	A10 – Unvalidated Redirects and Forwards (NEW)
A3 – Malicious File Execution	<dropped from T10 2010>
A6 – Information Leakage and Improper Error Handling	<dropped from T10 2010>

Abb. 2. Übersicht der Sicherheitsrisiken aus dem OWASP Top 10 Report, 2010

[1] www.owasp.org

Die Liste erhebt dabei keinen Anspruch auf Vollständigkeit und bezieht sich hauptsächlich auf die oben beschriebene Implementierungsebene. Denkbar wären jedoch auch Angriffe auf die Systemebene in Form von Buffer-Overflow Exploits oder Denial-of-Service-Attacken. Während bei Ersterem bspw. versucht wird einen Programmierfehler im Serverbetriebssystem auszunutzen um eigenen Code zur Ausführung zu bringen, wird bei Denial-of-Service-Attacken ein Server durch unzählige Anfragen so sehr bombardiert, dass er keine weiteren Anfragen mehr bearbeiten kann. Diese und weitere mögliche Angriffe sollen hier jedoch nicht weiter thematisiert werden. Der Fokus soll im Folgenden lediglich auf den zwei häufigsten Sicherheitsrisiken, SQL-Injection und Cross-Site-Scripting, liegen. Für ein tieferes Verständnis der anderen Risiken, sei auf den Report oder weitere Literatur verwiesen.

4.2 SQL Injection (SQLi)

Definition Unter SQL-Injection (SQLi) versteht man eine Angriffsmethode, bei welcher ein Angreifer versucht Eingabedaten durch Einschleusen zusätzlichen SQL-Codes derart zu manipulieren, dass er Zugriff auf nicht autorisierte Daten einer dahinter liegenden Datenbank bekommt. Die bekannteste Form der SQL-Injektion besteht darin den zusätzlichen SQL-Code bspw. beim Ausfüllen eines Webformulars direkt in die zu übergebenden Parameter einzuschleusen, welche anschließend vom Server, im schlimmsten Falle ohne weitere Überprüfung, in einer dynamische SQL-Anweisung zusammengefügt und direkt ausgeführt wird. [10, S. 7]

SQL-Injection kann folglich immer dann auftreten, wenn übergebene Parameter bei Generierung einer dynamischen SQL Anweisung nicht richtig gefiltert bzw. überprüft werden. Eine Überprüfung sollte dabei immer serverseitig und nicht clientseitig stattfinden, da letztere von einem Angreifer einfach überwunden werden kann.

Die Gefahren von SQLi können von Datenverlust oder -verfälschung bis hin zur völligen Übernahme der Daten in der Datenbank reichen und werden von Führungskräften leider häufig immer noch unterschätzt.

Beispiel Im Folgenden soll SQLi anhand eines Beispiels erläutert werden. Ausgangspunkt ist ein gängiges Loginformular, welches zur Eingabe von Benutzername und Passwort auffordert. Abbildung 3 zeigt grafisch zunächst den normalen Ablauf bei Absendung dieses Formulars. Im Anschluss wird gezeigt, wie das Formular mittels SQL-Injection ausgenutzt werden kann.

Füllt der Benutzer die Felder aus - in diesem Fall Nutzername: admin und Passwort: admin - und sendet das Formular ab, wird es an das entsprechende Skript auf dem Webserver gesendet. Das Skript liest die übergebenen Parameter aus, erstellt dynamisch eine SQL Anweisung und führt diese gegen eine Backend-Datenbank aus. Stimmen Nutzername und Passwort mit den Daten in der Datenbank überein, quittiert das Skript die Anfrage, indem es den Nutzer erfolgreich authentifiziert. Abbildung 3 zeigt außerdem beispielhaft ein verkürztes

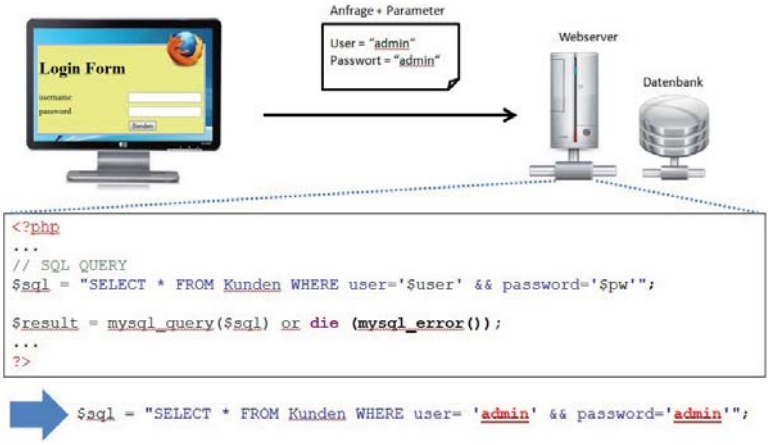


Abb. 3. Normaler Ablauf nach Senden eines ausgefüllten Webformulars

PHP Skript wie es auf dem Server gespeichert sein könnte. Man beachte hierbei die direkte Konkatenation der ausgelesenen Parameter, welche ohne weitere Überprüfung direkt zu einer SQL-Anweisung zusammengefügt und anschließend ausgeführt wird.

Abbildung 4 zeigt, wie der Vorgang der Authentifizierung mittels SQL-Injection umgangen werden kann.

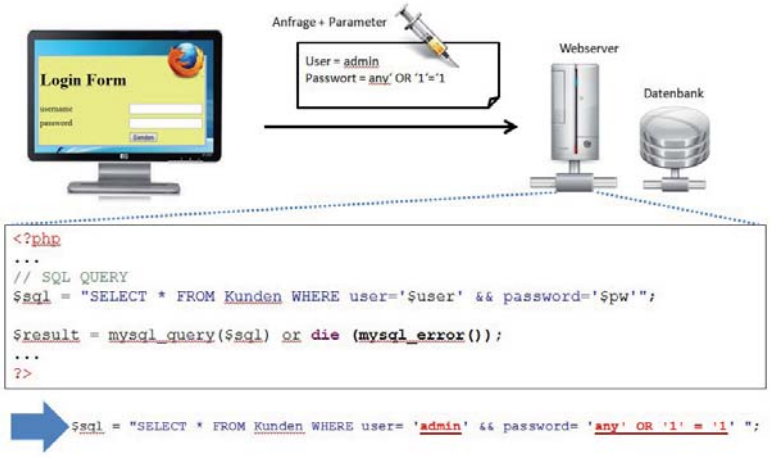


Abb. 4. Überlisten des Formulars mittels SQL-Injection

Im Gegensatz zum vorigen "normalen" Fall übergibt der Angreifer als Passwort nun einen präparierten String in Form von *any' OR '1'='1*. Wichtig hierbei ist die richtige Verwendung der Anführungsstriche, sodass diese sich nahtlos in die SQL-Anweisung integrieren, ohne einen Syntaxfehler zu ergeben (siehe Abbildung 4 unten). Wird nun diese SQL-Anweisung gegen die Datenbank ausgeführt, ergibt sich als Ergebnis, dass der Nutzer authentifiziert wird, obwohl eigentlich kein richtiges Passwort eingegeben wurde. Grund dafür ist, dass sich der Ausdruck password='any' OR '1'='1' immer zu einer wahren Aussage auswertet. Der Authentifizierungsmechanismus der Webseite wurde erfolgreich umgangen. An dieser Stelle wären weitere Möglichkeiten denkbar, wie bspw. Daten aus der Datenbank zu lesen oder gar ganze Tabellen zu löschen. Der interessierte Leser sei hier jedoch auf *SQL Injection Attacks and Defense von Justin Clarke* [10] verwiesen, welches SQL-Injection in weit größerer Tiefe betrachtet.

Strategien zur Vermeidung von SQL-Injection Um SQL-Injection erfolgreich verhindern zu können, müssen alle Eingaben vor ihrer Verwendung in einer SQL-Abfrage geprüft und gegebenenfalls bereinigt werden. Eine einfache und schnell umgesetzte Möglichkeit um bestimmte Sonderzeichen in den übergebenen Parametern zu maskieren, ist die Verwendung bestimmter Funktionen wie sie in den meisten Programmiersprachen heutzutage angeboten werden. In PHP sind das z.B. die Funktionen[2] addslashes() und mysql real escape string() [11, S. 27]. Sehr häufig ist die Bereinigung der Eingabedaten auch davon abhängig, welche Art von Daten oder Zeichen in der Eingaben jeweils zulässig sind. So können bspw. bei Telefonnummern alle nicht-nummerischen Zeichen über entsprechende reguläre Ausdrücke in Filterregeln herausgefiltert werden. Ein Nachteil solcher Filterregeln ist, dass sie häufig durch andere Darstellungsformen der zu filternden Zeichen (Kodierung) umgangen werden können. Ein stärkerer aber auch aufwändigerer Ansatz ist die Verwendung sogenannter Black- oder Whitelists als Kontrollmechanismus. Während Blacklists alle Eingaben zulässt, außer denen, welche auf der Liste stehen, verhält es sich bei der Whitelist genau umgekehrt. Hier wird generell allem misstraut und alles geblockt, es sei denn es steht auf der Whitelist. Für beide Ansätze gibt es Vor- und Nachteile und es verbleibt letztlich am Entwickler Aufwand der Administration und Nutzen abzuwägen und sich zu entscheiden. Eine weitere Möglichkeit SQL-Injection zu vermeiden liegt in der Verwendung von prepared statements. Abbildung 5 auf der Folgeseite zeigt einen alternativen PHP-Code unter Verwendung von Prepared Statements für das eingeführte Beispiel.

Weil sehr viele Fehler bei der Bereinigung von Eingabedaten gemacht werden können schlägt die OWASP die Verwendung von ESAPI vor. Die Enterprise Security API (ESAPI) ist eine freie API, welche von der OWASP mit dem Ziel entwickelt wurde, Entwicklern eine einfachere Möglichkeit zu bieten, sicherere

[2] Mehr Informationen zu den Funktionen finden sich unter `http://www.php.net/`

```php
<?php

$user = $_POST['username'];
$pw = $_POST['password'];

if (isset($user,$pw)){

    $db_con = new myscli("localhost","root","","appattackdb");


    // SQL QUERY
    $sql = $db_con->prepare("SELECT * FROM Kunden WHERE user= ? && password= ?");

    $sql->bind_param("ss",$user,$pw);   // Bindet Variablen an Prepared Statement
                                        // ss => 2 x string

    $sql->execute();

    if ($sql->fetch())
        die("connected");
    else
        die ("Access denied");
}
?>
```

Abb. 5. Beispielcode mit Prepared Statements in PHP

Webapplikationen zu programmieren. Da ESAPI von vielen Entwicklern als open source entwickelt wurde, gilt sie generell als sicherer als eigene Indivuallösungen[3].

Da die Vermeidung von SQL-Injection ein sehr komplexes Thema ist, und hier nur erste Ansätze vorgestellt werden konnten, sei zudem auch noch auf das *OWASP SQL-Injection Prevention Cheat Sheet*[4] hingewiesen, welches weitere Möglichkeiten zur Vermeidung von SQL-Injection aufführt.

4.3 Cross-Site-Scripting (XSS)

Definition Unter Cross-Site-Scripting (XSS) wird ein Angriff verstanden, bei welchem JavaScript- und HTML-Code in eine Webseite eingeschleust wird, um so dem Angreifer zu erlauben Skriptcode im Browser seines Opfers auszuführen [11, S. 33]. Man unterscheidet generell drei Arten von XSS:

- nicht-persistentes XSS,
- persistentes XSS und
- DOM-basiertes XSS

je nachdem, auf welchem Weg der Schadcode in die im Browser angezeigte Webseite gelangt. Alle drei Arten sollen im Anschluss näher betrachtet werden, wobei ein Beispiel lediglich für persistentes XSS gezeigt werden soll. Für ein tieferes Verständnis von XSS sei auf weiterführende Fachliteratur, wie *XSS Attacks von Jeremiah Grossmann et. al* [12] verwiesen.

[3] Nähere Informationen zur ESAPI finden sich unter `https://www.owasp.org/index.php/Category:OWASP_Enterprise_Security_API`

[4] Nähere Informationen unter `https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet`

XSS-Schwachstellen können je nach Art immer dann auftreten, wenn eine Anwendung nicht vertrauenswürdige Daten entgegennimmt und ohne entsprechende Validierung an einen Webbrowser (zurück-) sendet (oder in einer Datenbank zur spätere Auslieferung speichert) [2].

Die Gefahren, die von XSS ausgehen, können dabei sehr vielfältig sein. XSS kann es einem Angreifer beispielsweise erlauben Cookies (SessionIDs) eines Benutzers auszulesen und dadurch ganze Benutzersitzungen zu übernehmen oder ihn mittels ungewollter Redirects auf bösartige Seiten umzuleiten [2, S. 8]. Weiterhin können ganze Seiteninhalte verändert werden, und dadurch zum Beispiel Phisingseiten "über" bekannte Seiten des Nutzers (sogenannte Overlay-Pages) gelegt werden, wodurch es dem Angreifer ermöglicht wird Login-Daten abzufangen. Man stelle sich hier beispielsweise eine exakte Kopie der bekannten Bankseite des Benutzers als Phisingseite vor, die über die eigentlich echte Seite gelegt wird (sofern eine entsprechende XSS Lücke in der Bankwebseite vorhanden ist). Der ahnungslose Nutzer wird zum Opfer.

Nicht-persistentes (reflektiertes) XSS Bei nicht-persistentem (reflektiertem) XSS muss ein Angreifer sein Opfer zunächst dazu bringen eine vorher präparierte URL anzuklicken. In den Variablenparametern dieser URL versteckt er dabei Schadcode, welcher vom Server unverändert wieder zurück gesendet wird (deshalb auch der Name reflektiertes XSS). Dieser Schadcode kann daraufhin im Browser des Benutzers interpretiert und ausgeführt werden.[9]

Man spricht von nicht-persistent, weil der Schadcode nur temporär bei Generierung der Webseite eingeschleust wird, nicht jedoch gespeichert wird, wie beim persistenten XSS. Abbildung 6 [5] links veranschaulicht reflektiertes XSS.

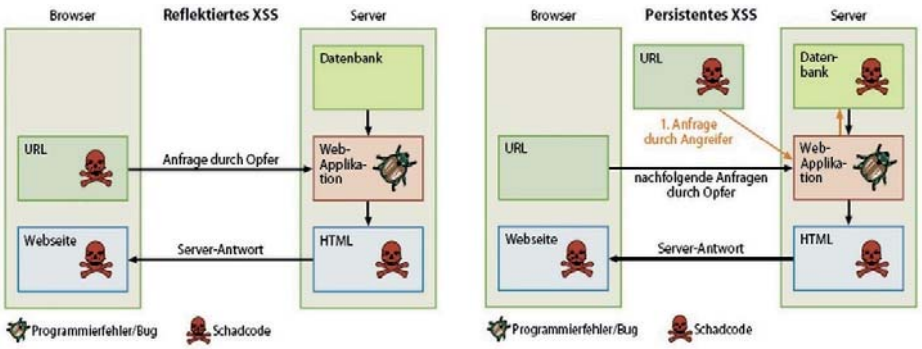


Abb. 6. Übersicht reflektiertes XSS links und persistentes XSS rechts

[5] Abbildungen entnommen aus [9]

Persistentes (stored) XSS Beim persistenten (stored) XSS sendet der Server ähnlich dem reflektierten XSS den Schadcode aus der URL als Webinhalt an den Browser zurück, mit dem entscheidenden Unterschied, dass dieser zuvor in der Datenbank gespeichert wird (deshalb der Name persistent). Bei jeder neuen Anfrage wird dieser Code nun mit ausgeliefert. Man beachte, dass durch Speicherung in der Datenbank auch Benutzer die keinen bösartigen Link angeklickt haben gefährdet sind, wenn sie die Webinhalte anfordern.[9] Diese Situation soll in einem Beispiel später näher betrachtet werden. Abbildung 6 rechts veranschaulicht persistentes XSS grafisch.

DOM basiertes (lokales) XSS Während der Programmcode, welcher letztlich den Schadcode in die Webseite eingebettet hat, bei reflektiertem und persistentem XSS serverseitig lief, spielt sich der gesamte Angriff bei DOM-basiertem (lokalen) XSS - vom Klicken einer manipulierten URL bis hin zum Einbetten des Schadcodes in die Webseite - auf dem Rechner des Anwenders ab. Ein fehlerhaftes browserseitiges Anwendungsskript kopiert den Schadcode direkt aus der angeklickten URL per DOM-Zugriff (daher der Name DOM-basiert) in die anzuzeigende Webseite. Ein Umweg über den Server entfällt. Diese Art von Angriff kann häufig bei Web 2.0 Anwendungen vorgefunden werden.[9] Abbildung 7 [6] veranschaulicht DOM-basiertes XSS grafisch.

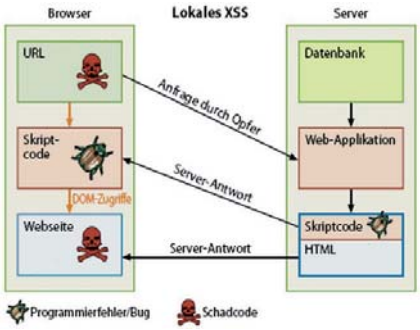


Abb. 7. Übersicht lokales XSS

Beispiel für Persistentes XSS Als Ausgangspunkt für das Beispiel soll eine Kommentarfunktion einer einfachen Blogging-Software dienen, bei welcher Besucher einen Kommentar zu einem Artikel hinzufügen können. Jeder kennt dieses Feature beispielsweise aus Wordpress. Im Normalfall können die Nutzer hier ihren Namen und ihren Kommentar eingeben, welche beide anschließend an entsprechender Stelle in der Datenbank des Webservers gespeichert werden.

[6] Abbildung entnommen aus [9]

Klicken nun weitere Besucher auf den Artikel wird dieser einschließlich Kommentar aus der Datenbank geladen.

Ein Angreifer kann nun versuchen einen speziellen String in eines der Felder einzuschleusen z.B. <script> javascript:alert("XSS-Attacke") </script>. Fehlen auf Serverseite entsprechende Filterregeln, bzw. Eingabevalidierungen wird dieser String unverändert in der Datenbank gespeichert und die XSS Attacke war erfolgreich (vgl. Abbildung 8).

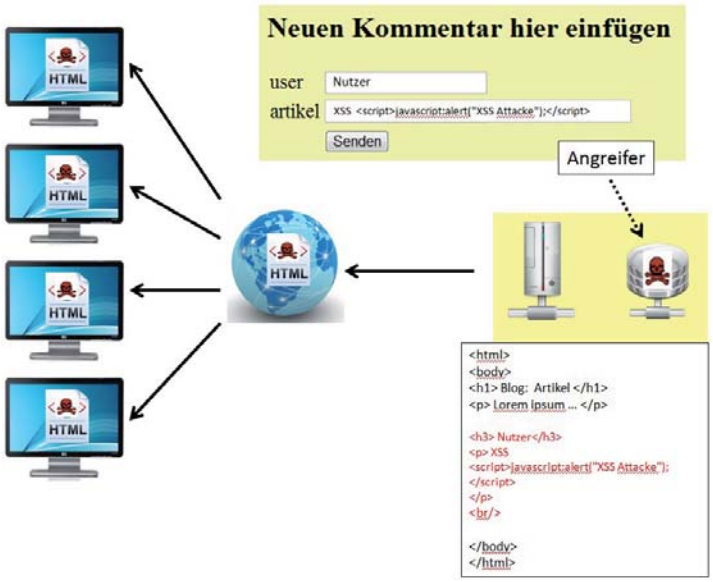


Abb. 8. Beispiel für persistentes XSS

Das verheerende an dieser Attacke ist, dass jeder Besucher, der zu einem späteren Zeitpunkt die infizierte Webseite aufruft, automatisch den entsprechenden Schadcode ausgeliefert bekommt. Dieser wird dann bei aktiviertem Javascript in seinem Browser ausgeführt auch wenn er ursprünglich auf keine bösartige URL geklickt hat. Eine vereinfachte HTML-Steie, wie sie vom Server dynamisch generiert und ausgeliefert werden könnte, ist in Abbildung 8 unten zu sehe (der Schadcode ist rot markiert). Im beschriebenen Beispiel besteht die XSS-Attacke lediglich darin, eine Alert-Box per Javascript aufzurufen. Hier wären jedoch, wie oben bereits erläutert, weit schlimmere Attacken denkbar.

Strategien zur Vermeidung von Cross-Site-Scripting Die Strategien zur Vermeidung von SQL-Injection können prinzipiell auch zur Verhinderung von XSS verwendet werden. So müssen zunächst alle Stellen identifiziert werden, in welchen Eingabeparameter in den Code integriert werden und die Parameter dann, wie oben bereits erläutert, über entsprechende Funktionen z.B. strip

tags() oder html entities() maskiert werden. Dies können "normale" Eingabeparameter sein, die über GET oder POST an den Server übertragen werden, aber auch Cookies, welche ja nichts anderes enthalten, wie bestimmte Daten und von einem Angreifer leicht manipuliert werden können, dürfen bei der Betrachtung nicht ausgeschlossen werden [11, S. 41]. Wiederum bietet sich die Verwendung einer sicheren API (ESAPI) an, sowie ein Vorgehen über Black- oder Whitelists. Für ausführlichere Strategien sei wiederum auf das von der OWASP erstellte *OWASP XSS Prevention Cheat Sheet*[7] hingewiesen.

Zusammenfassend lässt sich sagen, dass sich die Vermeidung von XSS oder auch SQL-Injection zunächst prinzipiell recht einfach anhört. Man muss lediglich das Einschleusen von Code sei es nun Javsscript oder SQL verhindern, indem man jede Benutzereingabe oder Eingabeparameter überprüft und gegebenenfalls bereinigt. Leider klingt dies oft einfacher als es in der Praxis ist. Häufig ergeben sich Konflikte zwischen, was dem Nutzer einerseits erlaubt sein soll - zum Beispiel Ausdruckskraft durch Verwendung von HTML sowie weiterer Tags in Kommentaren oder Artikeln - und andererseits was aus sicherheitsrelevanten Gründen sinnvoll erscheint. Hier gilt es die richtige Balance zu finden zwischen Komfort für den Nutzer einerseits und der Sicherheit der Webseite andererseits.

5 Fazit und Ausblick

Da es sich bei Webapplikationen, sowie anderen Informationssystemen, um komplexe Systeme handelt werden Sicherheitslücken mit hoher Wahrscheinlichkeit auch zukünftig noch zu finden sein. Ein System, welches einhundert prozentige Sicherheit garantieren kann, existiert nicht und wird es vermutlich auch nie geben. Dieser Grad an Sicherheit wäre theoretisch nur erreichbar, wenn man das System komplett isolieren und von jeglichem Netz (sei es Inter- oder Intranet) trennen würde, was jedoch häufig nicht im Sinne des Anwenders ist. Wichtig ist, dass die kritischsten Teile der Webapplikation identifiziert und geschützt werden. Es gilt das Risiko der Kompromittierung auf ein für das Unternehmen tragbares Maß zu reduzieren. Hierzu müssen zunächst Antworten auf die eingangs gestellte Frage "Wer darf was?" gefunden werden und Schutzziele wie Vertraulichkeit, Integrität, Verbindlichkeit und Verfügbarkeit der Daten und Systeme thematisiert werden. Nur wenn man genau weiß, was es zu schützen gilt, können entsprechende Maßnahmen getroffen und das Risiko entsprechend eingedämmt werden. Das vorgestellte Ebenenmodell soll dabei als Denkmodell helfen, verschiedene Perspektiven des Problems zu betrachten, um so eine bessere Sicherheit erreichen zu können.

Es ist essentiell, dass sich Entwickler, Designer, Software-Architekten aber auch Führungskräfte in sicherheitsrelevanten Fragen schulen und weiterbilden lassen. Security Awareness ist ein Buzzword in diesem Zusammenhang. So sollten sich Entwickler über aktuelle Sicherheitslücken in einschlägigen Foren und

[7] Nähere Informationen unter `https://www.owasp.org/index.php/XSS_Prevention_Cheat_Sheet`

Mailinglisten[8] informieren. Der *OWASP Top 10 Report* sollte hier lediglich als Einstieg dienen. Die betrachteten Angriffsarten, SQL-Injection und Cross-Site-Scripting, stellten dabei die zwei kritischsten Sicherheitslücken dar. Darüber hinaus existieren jedoch noch eine Vielzahl weiterer Angriffsvektoren und auch Varianten der vorgestellten Angriffe, auf die hier aber in Anbetracht der Seitenbeschränkung nicht weiter eingegangen werden konnte. Auch die Etablierung von Secure Coding Standards im Unternehmen sollte überlegt und eingeführt werden, um ein höheres Maß an Sicherheit bei der Entwicklung eigener Produkte, Informationssystemen bzw. der Onlinepräsenz gewährleisten zu können. Führungskräfte des Unternehmens hingegen müssen verstehen, dass IT-Sicherheit trotz anfallender Kosten und keinem unmittelbaren Ertrag dennoch notwendig ist, da ein verursachter Schaden, sei es durch Imageverlust des Unternehmens oder kompromittierten Benutzerdaten, letzten Endes häufig viel teurer ist. Schließlich gilt es IT-Sicherheit, als Prozess zu verstehen, nicht als Produkt.

References

1. dpa N24: 2011 wird zum Jahr der Hackerangriffe. Website (2011) Online verfügbar unter `http://www.n24.de/news/newsitem_7077061.html`; besucht: März, 2012.
2. Williams, J., Wichers, D.: OWASP Top 10 for 2010. Website (2010) Online verfügbar unter `https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project`; besucht: März, 2012.
3. Kappel, G., Pröll, B., Reich, S., Retschitzegger, W.: Web Engineering - Systematische Entwicklung von Web-Anwendungen. 1. aufl. edn. dpunkt-Verlag, Heidelberg (2004)
4. Eckert, C.: IT-Sicherheit - Konzepte, Verfahren, Protokolle. 5. überarbeitete und erweiterte auflage edn. Oldenbourg Verlag, München (2008)
5. Oberhaitzinger, B., Gerloni, H., Reiser, H., Plate, J.: Praxisbuch Sicherheit für Linux-Server und -Netze. 1. aufl. edn. Hanser, München (2004)
6. Schreiber, T., Veit, T.: Sicher auf allen Ebenen. Website (2005) Online verfügbar unter `http://www.securenet.de/fileadmin/papers/BSI-Sechs-Ebenen-Modell.pdf`; besucht: März, 2012.
7. Schreiber, T.: Web Application Security auf sechs Ebenen - Ein Klassifizierungsschema zur Sicherheit von Webanwendungen. Tagungsband des Deutschen IT-Sicherheitskongress (2005) Online verfügbar unter `http://www.securenet.de/fileadmin/papers/Klassifizierungsschema_Web_Application_Security.pdf`; besucht: März, 2012.
8. Schneier, B.: Secrets and lies - IT-Sicherheit in einer vernetzten Welt. dpunkt-Verl., Heidelberg (2004)
9. Rütten, C., Glemser, T.: Gesundes Misstrauen. Webseite (2007) Online verfügbar unter `http://www.heise.de/security/artikel/Sicherheit-von-Webanwendungen-270870.html`; besucht: März, 2012.
10. Clarke, J.: SQL Injection Attacks and Defense. Syngress, Burlington, MA (2009)
11. Eilers, C.: Ajax Security - Sichere Web-2.0 Anwendungen. entwickler.press, Unterhachingen (2008)
12. Fogie, S., Grossman, J., Hansen, R., Rager, A., Petkov, P.: XSS Attacks - Cross Site Scripting Exploits and Defense. Syngress (2007)

[8] Nähere Informationen bspw. unter `http://www.exploit-db.com/`

Herausforderungen im Web-Projektmanagement

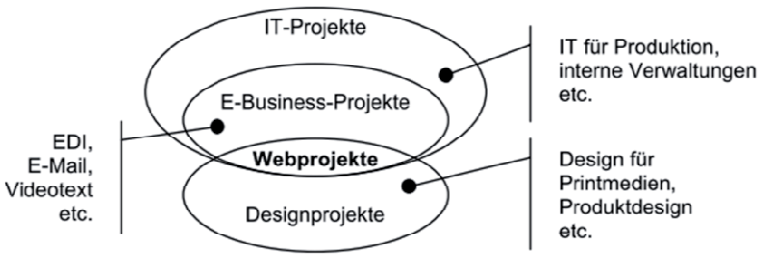
Robert Alexander Apfel

Zusammenfassung

Mit der Seminararbeit wird ein breiter Überblick über Herausforderungen von Web-Projekten geschaffen. Der Fokus liegt dabei auf mittelgroßen, bis großen Webprojekten. Durch zahlreiche Studien und eigenen praktischen Erfahrungen lässt sich bestätigen, dass Kommunikationsprobleme und eine mangelhafte Planung die Hauptursachen von scheiternden Projekten sind. Um dieser Entwicklung entgegenzuwirken, werden Planungsinstrumente zur Strukturierung und Standardisierung vorgestellt. Die Arbeit orientiert sich an den typischen Phasen eines Webprojekts. In der Planungs-, Durchführungs- und Nutzungsphase werden Denkansätze und Methoden vorgestellt, um den Herausforderungen zu begegnen. Die vorgestellten Ansätze müssen je nach Projektkomplexität individuell angepasst werden.

1 Web-Projektmanagement

Viele große monolithische Anwendungen werden heutzutage durch eine Vielzahl kleiner Web-Anwendungen ersetzt. Dies führt dazu, dass Software nicht mehr anhand von spezifizierten Anforderungen von Grund auf entwickelt werden müssen, sondern einzelne Komponenten nach dem agilen Prinzip miteinander verbunden werden (vgl. [1]). Grundlage dieser Vorgehensweise ist die agile Entwicklung. Diese versucht mit geringem bürokratischen Aufwand und wenigen Regeln möglichst schnell ein adäquates Produkt zu erstellen.
Dadurch ergeben sich wesentliche Unterschiede bei der Betrachtung von Web-Projekten und klassischen Softwareprojekten.



›-Projekte und deren Schnittstellen Quelle: [2]

Web-Projekte dienen in erster Linie der Neuerstellung oder Änderung einer Webpräsenz. Sie bilden eine Schnittstelle von E-Business-, IT- und Designprojekten.

Zum E-Business zählen alle Unterstützungsfunktionen, die einen direkten Kundenkontakt ermöglichen (z.Bsp. Email, User-Interface,…). Ein IT-Projekt umfasst die interne Mitarbeiterverwaltung oder die Steuerung von Produktionssystemen. Designprojekte beschäftigen sich mit der Contenterstellung und deren Einbindung. Je nach Projekt kommt es zu Überschneidungen in den verschiedenen Bereichen. Die daraus entstehenden interdisziplinären Teams erhöhen den Kommunikationsbedarf. Um die Kommunikation zu strukturieren haben sich in der Praxis Kommunikationspläne (siehe Kapitel 2.3) durchgesetzt. Der Projektleiter nimmt hierbei eine tragende Rolle ein (siehe Kapitel 3.2).

Die Frage nach der Kostenverteilung auf die betroffenen Abteilungen trägt viel Konfliktpotenzial in sich. Zudem gibt es bei Web-Projekten kein etabliertes Preismodell zur Bewertung von Anforderungen. Ansätze zur Anforderungsmessung bieten die Function-Point-Analyse (siehe Kapitel 2.5.1) oder die Pert-Methode (siehe Kapitel 2.5.2).

Während herkömmliche Softwareprojekte (z.Bsp. Entwicklung eines neuen Betriebssystems) mehrere Jahre in Anspruch nehmen können, sind bei Web-Projekten erheblich kürzere Projektzyklen zu beobachten. Die durchschnittliche Projektdauer beträgt drei bis sechs Monate (vgl. [3]).

Ein weiterer Aspekt ist der Content (Texte, Bilder,…) einer Website. Dieser ist getrennt von der Struktur und muss nach dem Aufsetzen der Webpräsenz weiter gepflegt werden, da veralteter Content eine Website uninteressant macht. Deswegen ist dessen Erstellung und Pflege auch nach Projektende erforderlich. Zu Konzepten, bei denen Nutzer selbst ihren Content in Form von Videos einpflegen, zählt zum Beispiel Youtube.com.

Heutzutage besitzt annähernd jeder Haushalt eine Internetanbindung, wodurch eine breite Zielgruppe entsteht. Ziel ist es, die Webpräsenz so zu gestalten, dass sie für alle Nutzergruppen ansprechend und leicht zu bedienen ist. Möglichkeiten, dies zu gewährleisten, sind individuelle Templates oder die Schaffung von Barrierefreiheit.

Ein sich stetig änderndes Umfeld sorgt für weitere Herausforderungen. Änderungswünsche des Kunden während der Projektdurchführung sind keine Seltenheit. Eine Planungshilfe und Argumentationsbasis für Verhandlungen bietet dabei das Lasten- und Pflichtenheft (siehe Kapitel 2.1).

Um die Webpräsenz auf dem neusten technischen Stand zu halten, was bereits aus Sicherheitsaspekten zu empfehlen ist, sind regelmäßige Updates erforderlich. Hierbei treten häufig Kompatibilitätsprobleme auf. Der modulare Aufbau von Joomla bietet eine einfache Bedienbarkeit und Einbindung neuer Funktionen für die Webpräsenz.

Bei der Migration auf eine aktuellere bzw. sicherere Version kann es vorkommen, dass die Funktionen bestimmter Module eingeschränkt oder gar nicht mehr verfügbar sind. Um dies rechtzeitig zu erkennen und Workarounds durchzuführen, ist nach Launch der Website eine fachkundige Wartung unerlässlich.

2 Die Planungsphase

Ein Web-Projekt lässt sich grundsätzlich in drei Phasen aufteilen. Dabei unterscheidet man die Planungs-, Durchführungs- und Nutzungsphase. In der Planungsphase werden die Projektziele und Anforderungen festgelegt. In der Durchführungsphase findet die Umsetzung der Anforderungen und die Kontrolle und Steuerung des Projekts statt. In der Nutzungsphase stellt sich heraus, ob das Web-Projekt erfolgreich war und die Projektziele erreicht wurden. Zudem werden hier die Ergebnisse des Projekts zusammengefasst und dokumentiert.

2.1 Das Lasten- und Pflichtenheft

„Das Lastenheft des Auftraggebers beschreibt die Zielsetzungen, Aufgabenstellungen und Eckdaten des Webprojekts und bedient sich dabei der Dokumentation des Istzustands mit anschließender Erläuterung des Sollzustands." (vgl. [4]). Es dient in erster Linie dazu, dem Auftragnehmer eine detaillierte und ausführliche Aufgabenbeschreibung zu liefern.

In der Praxis kommt es oft vor, dass Kunden kein Lastenheft erstellen und diese Aufgabe dem Auftragnehmer zufällt. In diesem Fall muss ein Lastenheft in enger Zusammenarbeit mit dem Auftraggeber erstellt werden.

Ein gutes Lastenheft zeichnet sich durch die folgenden Eigenschaften aus:

1. Klar definierte Ziele (siehe Kapitel 2.2)
2. Vollständigkeit, die in der Praxis vom Auftragnehmer durch Fragebögen verifiziert werden sollte
3. Inhaltliche Relevanz für den Auftragnehmer

Wichtig in diesem Zusammenhang ist, dass die Kundenanforderungen exakt und schriftlich festgehalten werden. Will der Kunde mitten im Projekt plötzlich einen E-Shop auf seiner Site, die ursprünglich nur für eine einfache Unternehmensrepräsentation ausgelegt wurde, so sprengt dies schnell das dadurch veraltete Lastenheft und kann den bisherigen Planungsaufwand nichtig machen.

Das Lastenheft lässt sich folgendermaßen gliedern (vgl. [5]):

1. Zielbestimmung
2. Produkteinsatz
3. Produktübersicht
4. Produktfunktionen
5. Produktdaten
6. Produktleistung
7. Qualitätsanforderung
8. Ergänzungen

Sobald die Anforderungen im Lastenheft festgehalten sind, kann die Erstellung des Pflichtenhefts durch den Auftragnehmer erfolgen. Hier findet die Leistungsbeschreibung für das Web-Projekt statt. Wichtigster Aspekt ist die Berücksichtigung der Vollständigkeit. Alle Anforderungen die mit dem Kunden im Lastenheft definiert wurden, müssen im Pflichtenheft erscheinen.

2.2 Die richtige Zielsetzung

Oft scheitern Projekte bereits bei der Zielsetzung. Unklare, schwer verständliche, nicht terminierbare oder messbare Ziele sind hierbei die Hauptursachen. Abhilfe verschafft das SMART Modell, welches klare Zieleigenschaften vorgibt. Ziele gelten als SMART wenn sie folgende Eigenschaften aufweisen:

Spezifisch: Ziele müssen eindeutig definiert sein.

Messbar: Ziele müssen anhand von Messbarkeitskriterien messbar sein.

Ausführbar/Erreichbar: Ziele müssen von den Mitarbeitern akzeptiert werden und dürfen die Mitarbeiter weder über- noch unterfordern.

Realistisch: Ziele müssen erreichbar sein.

Terminierbar: Jedes Ziel hat eine klare Terminvorgabe, bis wann das Ziel erreicht werden muss (vgl. [6]).

2.3 Der Kommunikationsplan

Um sicherzustellen, dass die Zielsetzungen auch mit den Erwartungen des Kunden übereinstimmen, ist eine konstante Kommunikation zwischen den Parteien erforderlich. Viele Kunden können oft schlecht einschätzen, welche Ziele Konflikte bei einer Webpräsenz auslösen können. Ein Video beim Betreten der Site kann das Unternehmen medial gut präsentieren, aber auch manche Nutzer aufgrund von

langsamer Internetverbindung komplett vom Unternehmen fernhalten. Feinheiten dieser Art müssen vor Projektstart klar mit dem Kunden abgesprochen werden.
Um die Kommunikationsstruktur zu standardisieren und langfristige Kommunikationskanäle zu schaffen, haben sich Kommunikationspläne etabliert.

Kommunikation	Freigabe	Start	Häufigkeit	Format	Verantwortlich	Zielgruppe
Wöchentlicher Projektstatusreport	Projekt-manager	01.01.2004	wöchentlich	E-Mail	Hinz	T, S
Monatlicher Projektstatusreport	Projekt-manager	01.01.2004	monatlich	E-Mail	Hinz	T, S, B
Projekt-Information	Projekt-manager	01.03.2004	vierzehntägig	E-Mail	Hinz	T, S, B, I
Information zur Einführung	Entscheider	01.06.2004	einmal	E-Mail	Schmidt	S, B, I
Einführungs-präsentation	Entscheider	10.06.2004	einmal	Präsentation	Schmidt	B, I

Legende: T = Projektteam, S = Steuerungsausschuss, B = Business-Stakeholder, I = Interne Nutzer

Abbildung 2: Der Kommunikationsplan Quelle: [7]

Durch den standardisierten und strukturierten Aufbau wird sichergestellt, dass die Kommunikationskanäle aufrechterhalten werden. Außerdem hat der Projektmanager stets die Übersicht über die Kommunikationsstruktur und kann bei Bedarf Kommunikationstermine hinzufügen oder entfernen.

2.4 Methoden der Priorisierung

Durch die vorhergehenden Schritte stehen nun mehrere Aufgaben an. Alle Anforderungen auf einmal zu realisieren führt meistens zum Chaos und Scheitern des Projekts. Sinnvoller erscheint die Priorisierung wichtiger Aufgaben.

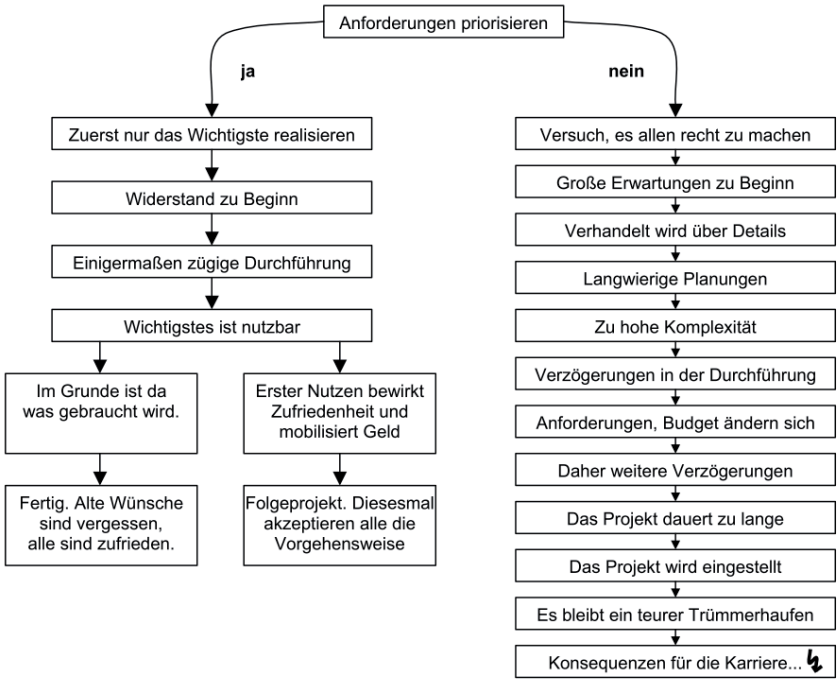


Abbildung 3: Priorisieren Ja oder Nein ? Quelle: [8]

2.4.1 Muss-Soll-Kann Kategorien

Bei dieser Priorisierungsform werden Anforderungen einer der drei Kategorien zugeordnet:

- MUSS: Ohne diese Anforderung ist das Vorhaben nicht sinnvoll.
- SOLL: Wichtige Anforderung, die wenn irgend möglich umgesetzt werden sollte.
- KANN: Nice-to-have-Anforderung, die keinen schweren Verlust für den Projekterfolg darstellt (vgl. [9]).

2.4.2 Das Pareto-Prinzip

Das wohl am bekanntesten und am häufigsten angewandte Prinzip ist das Paretto Prinzip. Auf Web-Projekte lässt sich der Grundgedanke ableiten, dass 20 % der Anforderungen 80 % des Projekterfolgs ausmachen und man die verfügbaren Ressourcen auf diese 20 % konzentrieren sollte. Das Lastenheft bildet hierzu die Planungsbasis.

2.5 Methoden zur Aufwandsschätzung von Web-Projekten

Die Aufwandsschätzung stellt gerade bei Web Projekten eine Herausforderung dar. Das änderungsfreudige Umfeld (Kundenwünsche, neue Technologien,…) und fehlende standardisierte Preismodelle erschweren die Planung.

2.5.1 Die PERT-Methode

Besonders bei neuartigen Projekten fällt es oft schwer den zeitlichen Aufwand für die Anforderungen abzuschätzen. Die PERT-Methode hilft dabei, einen ersten Überblick über den möglichen Arbeitsaufwand zu erlangen. Dabei werden die einzelnen Arbeitspakete in drei Varianten geschätzt:

1. Der Best Case stellt den Zeitbedarf dar, der bei einer problemlosen Projektabwicklung ohne Einarbeitungszeit zu erwarten ist.
2. Der Average Case beachtet normal auftretende Verzögerungen und eine gewisse Einarbeitungszeit.
3. Der Worst Case definiert den Fall, dass ein Projekt an jedem möglichen Problem hängen bleibt.

Sind die Werte geschätzt, kann im nächsten Schritt der Expectedcase ermittelt werden, welcher den Schätzwert darstellt.

$$expectedcase = \frac{bestcase + 4 * averagecase + worstcase}{6}$$

Abbildung 4: Berechnung des Expected Case Quelle: [10]

Schwer abschätzbare Faktoren können mithilfe von Pufferzeiten berücksichtigt werden.

Nr.	Arbeitspakete	Schätzung			
		Aufwand in Stunden			
		best case	average case	worst case	expected case
1	Einbinden des Web-Shops				
1.1	Kundenbedarf ermitteln	3	5	6	4,83
1.2	Bewertung und Auswahl des Shop-Systems	2	4	7	4,17
1.3	Implementierung des Shop-Systems	5	6	10	6,50
	Summe	10	15	23	15,5
	Puffer 25 %	2,5	3,75	5,75	3,875
	Summe mit Puffer	12,5	18,75	28,75	19,375

Abbildung 5: PERT-Schätzung Quelle: [11]

Die Pert-Schätzung ist einfach anzuwenden und gibt einen ersten Überblick über den möglichen Arbeitsaufwand. Die Berechnung ist leicht nachvollziehbar. Nachteilig ist die rein subjektive Bewertung des Arbeitsaufwands. Dies führt gerade bei neuen Projekten zu Fehlschätzungen. Durch die standardisierte Vorgehensweise können die Abweichungen durch zukünftige Erfahrungswerte minimiert werden und somit bildet die Pert-Methode eine gute und leicht anpassbare Planungsbasis.

2.5.2 Die Function-Point-Analyse (FPA)

Die Function-Point-Analyse wird für die kaufmännische Bewertung von IT-Projekten verwendet. Mit ihr soll eine möglichst frühe und realistische Aufwandsschätzung des Gesamtprojekts ermöglicht werden. Ausgangspunkt bei der FPA sind fachliche Anforderungen, die mehr oder weniger detailliert vorliegen (siehe Kapitel 2.1). Die Kosten der technischen Umsetzung werden mithilfe von Function Points (FP) ermittelt. Funktionen werden anhand der Komplexität (einfach, mittel, komplex) und Kategorie (Eingaben, Ausgaben, Abfragen, interne Datenbestände und Referenzdateien) unterschieden. Daraus ergeben sich 15 Fälle, in die Funktionen eingeteilt werden können (vgl. [12]).

Für die Bewertung ist zunächst die Identifikation der Elementarprozesse notwendig. Ein Elementarprozess stellt sinnvoll zerlegte Anforderungen aus Anwendersicht dar. In der Praxis wird ein funktionaler Hierarchiebaum erstellt, um Elementarprozesse zu identifizieren.

Anschließend wird die Bewertung anhand der 15 Fälle durchgeführt. Je nach Projektart und -umfang kann eine alternative Form der Bewertung gewählt werden.

Elementarprozess	Erläuterung	Typ	Punktwert
Veranstaltungen	Datenbestand, in dem die Veranstaltungen mit ihren Merkmalen abgelegt sind	ILF	7
Veranstaltungsliste anzeigen	beinhaltet auch die Möglichkeit, nach bestimmten Kriterien zu filtern und sortieren	EQ	4
Veranstaltungsdetails anzeigen	zusätzlich wird abgeleitet aus der Zahl der Anmeldungen und der verbleibenden Zeit ein Seminarstatus berechnet und in Form einer Ampel angezeigt	EO	5
Veranstaltung anlegen	Neuanlage einer Veranstaltung mit allen Merkmalen	EI	4
Veranstaltungsdetails ändern	nicht explizit gefordert, aber für den Geschäftsprozess notwendig	EI	4
Veranstaltung ohne Teilnehmer stornieren	eine relativ einfache Verarbeitungslogik	EI	4
Veranstaltung mit Teilnehmern stornieren	besondere Verarbeitungslogik erforderlich	EI	4
Veranstaltungstermin verlegen	besondere fachliche Verarbeitungslogik	EI	4
Summe der bewerteten Anforderungen			**36**

Abbildung 6: Bewertung von Elementarprozessen Quelle: [13]

Bei Verwendung eines einheitlichen Bewertungsverfahrens für die FPs ermöglicht dieses Verfahren die Vergleichbarkeit von Projekten und bietet insbesondere für das Projektcontrolling eine interessant Informationsquelle. In sehr frühen Projektphasen liefert die FPA eine Grobbeschreibung von Anforderungen. Die Nachvollziehbarkeit visualisiert dem Kunden klar, welche Kosten für welche Anforderungen entstehen. Nachteilig ist der hohe Aufwand für die Erfassung der Elementarprozesse und deren Bewertung.

2.5.3 Projektmanagementsoftware

Projektmanagementsoftware ermöglicht eine kostengünstige und schnelle Kommunikation mit Teammitgliedern und schafft Transparenz über vor- und nachgelagerte Prozesse. Gängige PM-Software ermöglicht neben der Früherkennung von Fehlentwicklungen und der Fortschrittskontrolle auch eine Ressourcenallokation.

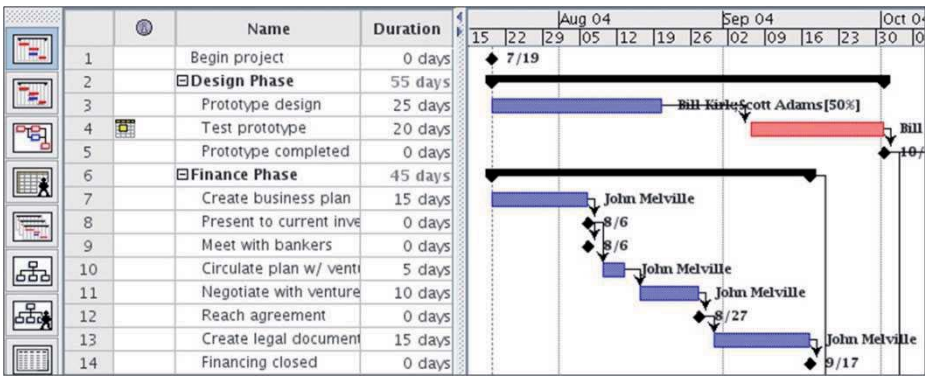


Abbildung 7: OpenProj Projektmanagementsoftware Quelle: [14]

Der Einsatz von Projektmanagementsoftware lohnt sich vor allem bei Teams, die standortunabhängig voneinander agieren.

3 Die Durchführungsphase

In der Durchführungsphase findet die Umsetzung der Anforderungen und die Kontrolle und Steuerung des Projekts statt. Gerade in dieser Phase treten bei Web-Projekten Probleme auf. Ursachen sind meistens Kommunikationsprobleme, die durch interdisziplinäre Teams, neue Technologien oder unklare Zielsetzungen ausgelöst werden.

3.1 Kommunikation

Zur effizienten Zielvermittlung und -erreichung ist eine konstante Kommunikation zwischen allen internen sowie externen Beteiligten unerlässlich. Kommunikationspläne (siehe Kapitel 2.3) und Meetings sind die am häufigsten gewählte Form der Kommunikationsförderung und Fortschrittsüberwachung.
Leider laufen viele Meetings ineffizient ab. Hauptgründe sind fehlende Ziele, Zeitrahmen und Kommunikationsregeln.

Kommunikationsregeln können sein:

- Jeden ausreden lassen und zuhören.
- Sich kurz zu fassen und Sachverhalte auf den Punkt bringen.
- Objektiv argumentieren ohne andere persönlich anzugreifen.

Die Kommunikation und Durchsetzung der oben genannten Basisregeln sind Aufgabe des Projektleiters.

3.2 Die Rolle des Projektleiters

Der Projektleiter nimmt beim Web-Projektmanagement eine tragende Rolle ein. Er ist für die projektweite Koordination und das Lösen von internen sowie externen Konflikten zuständig. Dies reicht von Unstimmigkeiten zwischen Teammitgliedern bis zur Anpassung der Prozesse an die sich verändernden Kundenbedürfnisse.

Viele Führungskräfte neigen dazu psychologische und soziale Aspekte beim Projektmanagement zu unterschätzen. Zu spät oder nicht erkannte Konflikte zwischen Teammitgliedern können ein Projekt ausbremsen und letztendlich zu dessen Scheitern führen. Dies tritt häufig bei Web-Projekten auf, bei denen Spezialwissen auf sehr wenige Teammitglieder verteilt ist und Kompromisslösungen nur selten möglich sind.

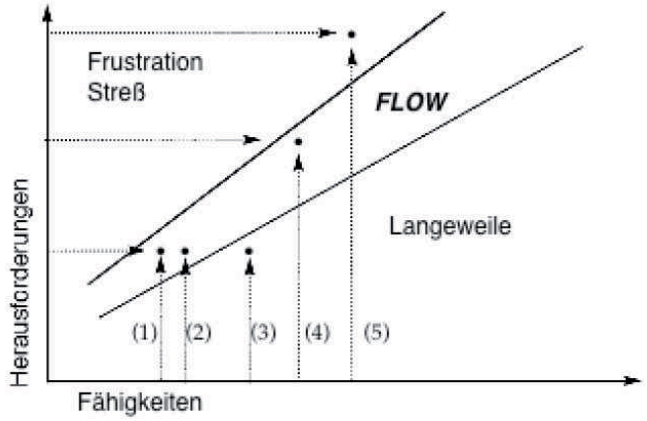
Neben der Sozialkompetenz ist die Fachkompetenz des Projektleiters entscheidend. Er muss in der Lage sein die groben technischen Zusammenhänge zu verstehen. Dies ist vor allem bei der Verteilung der Ressourcen und der Mitarbeiteranweisung unerlässlich. Ein Projektleiter, der unrealistische Vorgaben aufgrund von mangelndem Fachwissen gibt, verliert schnell den Rückhalt des Teams. Auf der anderen Seite muss er in der Lage sein abzuschätzen, wie lange beispielsweise das Implementieren eines Shop-Systems dauert.

Das frühzeitige Erkennen von Über- und Unterforderung stellt eine große Herausforderung dar, da jeder Mitarbeiter individuelle Stärken und Schwächen aufweist und ein simples Problem durch externe Faktoren, wie spezielle Kundenwünsche, sehr schnell, sehr komplex werden kann. Diesbezüglich muss der Projektleiter einschätzen können, inwieweit ein Kundenwunsch mit gegebenen Ressourcen realisierbar ist, aber auch wann eine Anfrage abgelehnt werden muss.

3.3 Mitarbeitermotivation

Braucht ein Mitarbeiter ungewöhnlich lange für eine Aufgabe oder erledigt diese nicht sorgfältig, so sind dies Anzeichen für eine Über- oder Unterforderung. Ziel ist es, eine Balance zwischen diesen Extremen zu finden. Das Erreichen dieser Balance wird als „Flow“ bezeichnet.

„Flow entsteht nur, wenn Fähigkeiten und Herausforderungen in Einklang miteinander sind. Sind die Fähigkeiten hoch, die Herausforderungen und Aufgaben dagegen zu einfach, so stellt sich rasch Langeweile ein. Übersteigen die Herausforderungen aber die Fähigkeiten, dann entsteht schnell Stress und Angst.“ (vgl. [15]).



(1) Die Fähigkeiten entsprechen der Herausforderung: **FLOW**
(2) Bei wiederholter Bewältigung der gleichen Herausforderung: **FLOW** und Zunahme der Fähigkeiten: Lernen
(3) Bei der Bewältigung der gleichen Herausforderung stellt sich nun **Langeweile** ein, da die Fähigkeiten zugenommen haben: Routine
(4) Wird eine neue, *größere* Herausforderung gesucht: erneut **FLOW**
(5) Nun wird, ermuntert durch die vergangenen FLOW-Erfahrungen eine wesentlich größere Herausforderunge gesucht, die im Beispielsfall die Fähigkeiten übersteigt: **Fehlschlag, Streß und Frust.**

Abbildung 8: Der Flow Kanal Quelle: [16]

4 Die Nutzungsphase

4.1 Abnahmetests

Bevor das Web-Projekt Online geht und dem Kunden übergeben wird, sind Tests erforderlich. Bereits in der Planungsphase festgelegte und an die Kundenanforderungen angepasste Use-Cases sollten spätestens hier überprüft werden. Ein Use-Case stellt einen Anwendungsfall dar, der alle möglichen eintretenden Szenarien bündelt. Dies kann die Bestellung eines Produkts im E-Shop oder das Aufrufen einer bestimmten Internetseite sein. Schlägt ein Anwendungsfall fehl, muss das Problem aufgedeckt und behoben werden, bevor das Endprodukt dem Kunden übergeben wird.

4.2 Abschlussevaluation

Jedes Projekt endet mit der Abschlussevaluation. Bei Web-Projekten findet diese beim Launch oder nach einer kurzen Nutzungsphase statt. Wartungsarbeiten und Contentpflege werden entweder vom Kunden selbst durchgeführt oder outgesourct.
Abschließend wird geprüft, ob alle gesetzten Ziele erreicht wurden. Die Dokumentation der Ergebnisse und der aufgetretenen Probleme kann zu einer nachhaltigen Verbesserung von zukünftigen Projekten führen. Um den Projekterfolg messbar zu machen, sollten zentrale Fragestellungen bei Projektende beantwortet werden. Diese können sein:

Welche Milestones wurden termingetreu abgeschlossen?
Wurde das Budget über- oder unterschritten?
Welcher Prozessschritt hat die größten Verzögerungen und Budgetüberschreitungen ausgelöst?
Waren die Ziele SMART?

Hintergedanke hierbei ist die stetige Verbesserung der internen Prozesse. Bei ähnlichen Projekten ist es sinnvoll, ein Wissensmanagement aufzubauen. Der Rückgriff auf Erfahrungswerte ermöglicht den Projektleitern, Prozesse von vorneherein effizienter zu gestalten. Außerdem können Vergleiche mit der PERT-Methode oder Function-Point-Analyse projektübergreifend durchgeführt und potenzielle Erfolgs-/Fehlerindikatoren ermittelt werden. Große Verzögerungen oder Probleme bei Vorprojekten ergeben eine exzellente Datenbasis für zukünftige Verbesserungen. Damit Fehler festgehalten und nicht unter den Tisch gekehrt werden, ist es sehr wichtig, eine tolerante Fehlerpolitik zu betreiben. Um diese Mentalität im Unternehmen durchzusetzen, ist es erforderlich, Fehler als natürlichen Entwicklungsprozess und nicht als das Versagen einer bestimmten Person zu betrachten.

Damit die Wissensbasis übersichtlich und transparent bleibt, ist eine Standardisierung unerlässlich. Eine einfache Methode dies umzusetzen ist die Vorgabe einer Vorlage, die von jedem Prozessabschnitt ausgefüllt und am Ende evaluiert wird.

Extern betrachtet ist der wichtigste Faktor die Kundenzufriedenheit mit dem neuen Produkt oder der Dienstleistung. Da diese qualitativer Natur ist, ist eine Erfassung nur schwer durchführbar. Die verbreiteteste Form ist die direkte Befragung des Kunden in Form eines Fragebogens. Gewisse Anreize, wie x % Nachlass, können die Rücklaufquoten dieser Befragungen erheblich erhöhen. Eine Standardisierung der Fragen auf essentielle Faktoren (Erfüllt Produkt/Dienstleistung den Zweck, Kompetenz des Projektleiters,…) ermöglichen die projektübergreifende, objektive Beurteilung der Kundenzufriedenheit.

5 Fazit

Zusammenfassend lässt sich festhalten, dass es unerlässlich ist, ein Webprojekt gründlich zu planen. Fehler, die in der Planungsphase gemacht werden, ziehen sich oft durch das ganze Projekt. Da diese erst später auftreten, werden oft nur die Symptome des Problems behandelt und nicht die Ursachen. Nur eine konsequente Umsetzung von standardisierter Projektplanung und -kontrolle kann dieser Entwicklung begegnen.
Die Planungsbasis bildet dabei das Lasten- und Pflichtenheft. Die im Lastenheft festgehaltenen Anforderungen sollten mit dem Kunden priorisiert werden. Um die Anforderungen zu bewerten, bietet sich die Pert-Methode an, die einfach anzuwenden und nachvollziehbar ist. Die Function-Point Analyse bietet eine standardisierte Bewertung von Anforderungen anhand von Elementarprozessen und ermöglicht einen projektübergreifenden Vergleich von Web-Projekten. Dies schafft vor allem Transparenz für den Kunden, der genau sehen kann welche Kosten für welche Anforderung entstehen.

Quellenangaben:

[1] Kappel, Gerti u.a.: Web Engineering - Systematische Entwicklung von Web-Anwendungen. 1 Auflage 2004. dpunkt.verlag GmbH. S 210
[2] Robert Stoyan, Management von Webprojekten, 2 Auflage Springer Verlag , S. 13
[3] Robert Stoyan, Management von Webprojekten, 2 Auflage Springer Verlag , S. 4 und Kappel, Gerti u.a.: Web Engineering - Systematische Entwicklung von Web-Anwendungen. 1 Auflage 2004. dpunkt.verlag GmbH. S 210
[4] Claudia Lettau, Das Web Pflichtenheft, 1 Auflage MITP Verlag, S.75
[5] Helmut Balzert, Lehrbuch der Software-Technik, 2000, S.62-63
[6] Doran, G. T. (1981). There's a S.M.A.R.T. way to write management's goals and objectives. Management Review, Volume 70, Issue 11(AMA FORUM), pp. 35-36.
[7] Robert Stoyan, Management von Webprojekten, 2 Auflage Springer Verlag , S. 249
[8] Robert Stoyan, Management von Webprojekten, 2 Auflage Springer Verlag , S. 53
[9] Robert Stoyan, Management von Webprojekten, 2 Auflage Springer Verlag , S. 52
[10] http://de.wikipedia.org/wiki/Program_Evaluation_and_Review_Technique, abgerufen am 25.03.2012 15:57
[11] eigene Darstellung
[12] http://www.enzyklopaedie-der-wirtschaftsinformatik.de/wi-enzyklopaedie/lexikon/is-management/Software-Projektmanagement/Aufwandschatzverfahren/Function-Point-Methode, aufgerufen am 26.03.2012 11:44
[13] http://de.wikipedia.org/w/index.php?title=Datei:Beispiel_Funktionaler_Baum.png, aufgerufen am 26.03.2012 12:19
[14] http://www.fit-for-bit.de/Projektmanagement-Software-OpenProj.php aufgerufen am 25.03.2012 16:41
[15] Das Flow-Erleben als Schlüssel für Lernen, Wachstum und Motivation Aus: Evolutionäres Management, Herausgegeben von Sonja Radatz, Verlag für Systemisches Management, Wien 2004, S. 248 ff.
[16] Das Flow-Erleben als Schlüssel für Lernen, Wachstum und Motivation Aus: Evolutionäres Management, Herausgegeben von Sonja Radatz, Verlag für Systemisches Management, Wien 2004, S. 248 ff.

Semantic Web - Das Netz der Bedeutungen im Netz der Dokumente

Hans-Christian Sperker

Otto-Friedrich-Universität Bamberg, Lehrstuhl für Medieninformatik
hans-christian.sperker@stud.uni-bamberg.de

Zusammenfassung Ziel dieser Arbeit ist der Versuch dem IT-versierten Leser einen ganzheitlichen Einblick in das Thema Semantic Web zu vermitteln. Zunächst wird hierfür die Vision Tim Berners-Lee's bezüglich des Semantic Web dargestellt, bevor dann dessen technologischen Grundlagen vorgestellt werden. Anschließend wird auf die Entwicklung von Anwendungen und Websites eingegangen und die Suche im Semantic Web beschrieben. Zur Komplettierung wird dann das Browsen von Websites und Inhalten und dessen Besonderheiten im Vergleich zum Browsen im World Wide Web angerissen. Abschließend versucht diese Arbeit dann die Reife des Semantic Web einzuschätzen und zeigt noch einmalmal bestehende Probleme auf.

Schlagworte: Linked Data, OWL, RIF, RDF, RDFa, RDFS, Semantic Web, SPARQL, Web of Data

1 Einleitung

Das Informationsangebot im World Wide Web wächst im Laufe der Jahre weiter rasant an. Gründe hierfür sind nicht zuletzt eine immer weitere Akzeptanz sowie Verfügbarkeit. Mit diesem vorhandenen Potential des Web, Informationen zu jedem erdenklichen Sachverhalt zugreifbar zu machen wächst zugleich der Wunsch auch komplexere Informationsbedürfnisse zu befriedigen.

Ein Beispiel für ein komplexes Informationsbedürfnis könnte die Suchanfrage, auch Query genannt ‚welcher Radiosender spielt Titel von schwedischen Interpreten' darstellen. Solche Queries stellen herkömmliche Suchmaschinen allerdings vor ein Problem [1]. Sollte es ein einzelnes Dokument geben welches eine Antwort auf die Anfrage und möglicherweise eine Variation dieser selbst[1] beinhaltet könnte die Suchmaschine dieses Informationsbedürfnis noch decken. Ein wesentlich realistischerer Fall besteht allerdings darin, dass Antworten auf komplexere Queries aus verschiedenen Dokumenten generiert werden müssen [1]. Dies setzt den Einsatz von intelligenter Software voraus, die auf einem für Maschinen verständlichen Netz aus Daten operiert, aus denen wiederum das Wissen zur Beantwortung der Queries abgeleitet werden kann [1][2].

[1] Eine Variante der Frage in einem gefundenen Dokument stellt für die Suchmaschine eine Möglichkeit da, dieses für die Suchanfrage als relevant zu betrachten.

So verlangt die Befriedigung komplexer Informationsbedürfnisse einen Wandel in der Struktur des World Wide Web. Das Web, welches bis jetzt noch zum größten Teil aus unstrukturierten und unabhängigen Dokumenten besteht, die weitestgehend nur von Menschen lesbar sind, wandelt sich zu einem Netz aus strukturierten und untereinander in Verbindung stehenden Daten, die auch für Maschinen verständlich sein sollen [3].

Dieser Gedanke, ein Netz der Bedeutungen innerhalb des Netzes der Dokumenten zu schaffen, ist allerdings nichts Neues. Geprägt wurde er 2001 von Tim Berners-Lee unter dem Begriff *Semantic Web* [4], fand allerdings schon Erwähnung auf der ersten World Wide Web Conference[2] im Jahre 1994 [3].

Hinter dem Begriff *Semantic Web* verbirgt sich allerdings mehr als nur das reine Erzeugen neuen Wissens aus bekannten Daten mit der Hilfe von logischem Schließen. Folgendes Beispiel, angelehnt an den Scientifice American Artikel *The Semantic Web* von Tim Berners-Lee et al. [4] soll einen Einblick in die Vision eines Semantischen Netz geben, und damit dessen gesamtes Potential veranschaulichen.

Example 1. Ein Patient möchte aufgrund einer ärztlichen Diagnose einen Termin bei einem Spezialisten vereinbaren. Dazu vertraut diese Person auf die Unterstützung eines privaten Software-Agenten der unter Berücksichtigung aller für den Besitzer wichtigen Aspekte den bestmöglichen Spezialisten findet und einen Termin festlegt. Hierzu aktiviert die Person den Agenten und gibt diesem den Auftrag. Der Agent verbindet sich zunächst über das Internet mit dem Agenten das Arztes, der die Diagnose gestellt hat, um heraus zu finden welche Art von Spezialist benötigt wird. Dann sucht der Agent im Netz die Spezialisten heraus, die von der Versicherung des Patienten abgedeckt sind und bewertet diese mit Hilfe von vertrauenswürdigen Rating-Services. Schließlich versucht der Agent des Patienten in Verbindung mit dem Agenten des bestmöglichen Spezialisten und unter Berücksichtigung des Terminplans der beiden einen Termin zu vereinbaren. Kommt es zu einer Übereinkunft wird der Patient über einen neuen Termin in dessen Kalender informiert und der Agent hat die Aufgabe erfolgreich beendet.

Dieses Beispiel verdeutlicht gut das Ziel und den Nutzen eines semantischen Netzes, zeigt allerdings auch, dass es eine ehrgeizige Vision ist, mit verschiedensten komplexen Herausforderungen für die Forschung [5].

Die vorliegende Arbeit soll nun einen detaillierten Einblick in die technologischen Voraussetzungen für das *Semantic Web* und dessen mögliche Nutzbarkeit und Reife geben. Hierzu gliedert sich diese Arbeit wie folgt. In Abschnitt 2 wird näher auf die Grundbausteine die für die Entwicklung eines semantischen Netzes benötigt werden, eingegangen. Abschnitt 3 beschäftigt sich hingegen mit dem *Semantic Web Engineering* und geht demnach auf die Entwicklung von Anwendungen für ein semantisches Netzwerk ein. Abschnitt 4 wechselt die Sichtweise von der Anwendungsentwicklung hin zum Suchen im semantischen Netz. Um die

[2] Die World Wide Web Conference ist eine jährliche internationale Konferenz, in der die weitere Entwicklung des World Wide Web thematisiert wird.

Arbeit zu vervollständigen geht Abschnitt 5 kurz auf das Browsen im *Semantic Web* ein, bevor dann Abschnitt 6 einen Ausblick gibt, in dem die Nutzbarkeit sowie Reife des *Semantic Web* einzuschätzen versucht wird.

2 Grundbausteine des Semantic Web

Voraussetzung für die Realisierung der in Beispiel 1 beschriebenen Vision eines semantischen Netzes ist eine standardisierte Architektur. Diese muss es Maschinen ermöglichen die Bedeutung von Daten im Netz zu verstehen und darüber hinaus aus den Daten mit Hilfe logischen Schließens neues Wissen abzuleiten. Zudem führt eine Standardisierung der Architektur dazu, dass heterogene Softwarelösungen, wie etwa Agentensysteme für das semantische Netz, untereinander interagieren können, sowie, dass die Komplexität solcher Software reduziert werden kann. Dies wiederum führt zu einer besseren Akzeptanz.

Allerdings gibt es bis jetzt noch keinen eindeutigen Architekturstandard für das *Semantic Web*. So legen sich Horrocks et al. in [6] zwar auf eine Architektur fest, die auch an nicht ganz offizieller Stelle des World Wide Web Consortium (W3C)[3] Erwähnung findet [7], stellen darüber hinaus aber gleich drei weitere schlüssige Architekturen vor. Diese Unstimmigkeiten auf der untersten Ebene des semantischen Netzes ergeben sich vor allem daraus, dass noch nicht alle Technologien die in der Architektur beschrieben sind standardisiert sind und sich immer noch in der Entwicklung befinden [8].

Um nun trotz der herrschenden Unstimmigkeiten einen Einblick in das Fundament des *Semantic Web* geben zu können, legt sich diese Arbeit auf eine Erweiterung [7][8][9][10] der in [6] genannten Architektur fest, dargestellt in Abbildung 1.

Nachfolgend werden nun kurz die wichtigsten Technologien der *Semantic Web* Architektur vorgestellt. Dabei wird auf eine Vorstellung der Schichten *User interface and applications*, *Trust*, *Proof*, *Unifying Logic* sowie auf die vertikal verlaufende Schicht *Cryptography* verzichtet, da diese noch keine Umsetzung gefunden haben [8]. Die Technologien *Unicode*, *URI* sowie *XML* werden zusammengefasst und nur kurz vorgestellt, da diese als bekannt vorausgesetzt werden. Der Bereich *RIF/SWRL* findet auch nur eine kurze Erwähnung, da auch diese Technologien erst in einer sehr rudimentären Version vorliegen [8].

Rundum sollte dem Leser klar sein, dass die hier vorgestellten höheren Technologien alle noch immer einer ständigen Entwicklung unterliegen wodurch die Gültigkeitsdauer der hier getroffenen Aussagen möglicherweise begrenzt ist. Daher wird versucht einen Fokus auf die basalen Eigenschaften der jeweiligen technologischen Ebenen zu richten.

[3] Das World Wide Web Consortium ist die internationale Organisation zur Verwaltung der Standards des World Wide Web. Die Abkürzung W3C ist nicht zu verwechseln mit der World Wide Web Conference, deren Partner aber nicht deren Organisator, das W3C ist.

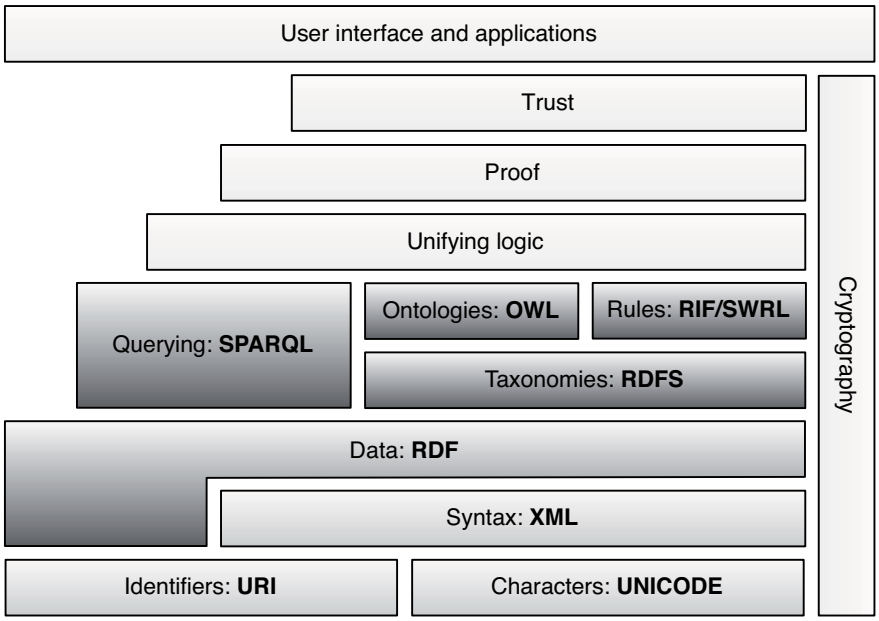


Abbildung 1. Semantic Web Architektur

2.1 Unicode / URI / XML

Unicode[4] bezeichnet den ISO Standard für den internationalen Zeichensatz der Buchstaben und Zeichen aller bekannten Schriftkulturen vereinigt. Dies führt zu einer Erleichterung der Lesbarkeit und damit Austauschbarkeit von Dokumenten [9].

Uniform Resource Identifier[5] (URI) wird verwendet, um Ressourcen im Web zu identifizieren. Dabei sind zwei Klassen zu unterscheiden. Während die *Uniform Resource Name* (URN) die Identität eines Gegenstandes definiert, bietet die *Uniform Resource Locator* (URL) eine Möglichkeit diesen aufzufinden [9]. Verwendung findet die URI im *Semantic Web* bei der eindeutigen Identifizierung von Ressourcen sowie deren Erreichbarkeit.

Extensible Markup Language[6] (XML) ermöglicht die Erzeugung benutzerdefinierter, strukturierter sowie semistrukturierter Dokumente, die mittels XML-Tags für das *Semantic Web* die Grundlage für maschinenlesbare Daten darstellen [9].

Diese Technologien stellen allesamt das Fundament des semantischen Netzes und sorgen dafür, dass maschinell global lesbare Daten auffindbar und aus-

[4] `http://de.wikipedia.org/wiki/Unicode`
[5] `http://de.wikipedia.org/wiki/Uniform_Resource_Identifier`
[6] `http://de.wikipedia.org/wiki/XML`

tauschbar sind. Im Gegensatz zu den anderen im weiteren Verlauf vorgestellten Technologien handelt es sich hierbei um gut getestete und in vielen anderen Anwendungsgebieten verwendete und akzeptierte Standards.

2.2 RDF

Das *Resource Description Framework*[7] (RDF) definiert sowohl ein Datenmodell als auch eine Sprache zur Beschreibung von Ressourcen und damit Daten für das semantische Netz. Die Beschreibungen liegen dabei in Tripeln der Form *(Subjekt, Prädikat, Objekt)* vor. Eine Subjekt-Ressource, identifiziert durch eine URI ist ein beliebiger Gegenstand, über den eine Aussage getroffen werden soll. Das Prädikat beschreibt eine Beziehung zwischen Subjekt- und Objekt-Ressourcen und muss auch über eine URI eindeutig identifiziert werden können. Das Objekt schließt die Aussage ab und kann entweder als eine Ressource beschrieben durch eine URI oder als Literal[8] vorliegen. Handelt es sich bei dem Objekt um ein Literal, so wird empfohlen dies anhand der XML Schema Datentypen wie *string*, *decimal* oder *date* zu typisieren [9][2]. So könnte die Aussage *Sokrates ist ein Mensch* in RDF zum Beispiel wie in Abbildung 2 dargestellt werden. So beinhaltet die erste Zeile zwei Namensräume die zur Definition der genutzten Ressourcen benötigt werden. In Zeile zwei startet dann die Definition des Subjekts Sokrates, dessen Prädikat die Aussage ‚ist vom Typ' ist (Zeile 3), der dann auf das Objekt Person (Zeile 4) festgelegt ist.

```
<rdf:RDF ↩
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" ↩
    xmlns:foaf="http://xmlns.com/foaf/0.1/">
  <rdf:Description ↩
      rdf:about="http://dbpedia.org/page/Socrates">
    <rdf:type>
      <foaf:Person/>
    </rdf:type>
  </rdf:Description>
</rdf:RDF>
```

Abbildung 2. Exemplarisches RDF Tripel *Sokrates ist eine Person*

Durch das RDF Modell können somit große Graphen aus vielen Tripeln über deren Subjekte beziehungsweise Objekte zusammengesetzt werden, und so als eine Sammlung von Fakten angesehen werden. Durch die Wiederverwendung von schon bestehenden Konzepten wie im gezeigten Beispiel *foaf:Person* wird sichergestellt, dass neue Informationen Verbindungen zu schon bestehenden Daten

[7] http://www.w3.org/TR/rdf-concepts/

[8] Ein Literal ist dabei eine Zeichenkette die Werte in Form von Basisdatentypen darstellt.

haben, wodurch eine Erweiterung des verwendbaren Datengraphen gewährleistet ist.

Während RDF-Graphen für gewöhnlich in *Repositories*, vergleichbar mit relationalen Datenbank-Management-Systemen, abgelegt werden, bietet das W3C darüber hinaus weitere Möglichkeiten, RDF Daten verfügbar zu machen. So ermöglicht *Resource Description Framework - in - attributes*[9] (RDFa) RDF in XHTML zu integrieren, was besonders reizvoll für Laien ist, die sich nicht all zu tief mit der Materie auseinander setzen möchten.

2.3 RDFS

Das RDF-Schema ist eine Ontologiesprache zur Erstellung taxonomieähnlicher Ontologien aufbauend auf Daten in RDF. Als Ontologie bezeichnet man dabei eine Art der Wissensrepräsentation durch eine Menge an Begrifflichkeiten und Beziehungen zwischen diesen, verwendet in einem gewissen Anwendungsgebiet [1]. RDFS nimmt also eine hierarchische Gliederung von Aussagen durch Beschreibung von verwandten Ressourcen und Relationen zwischen diesen vor [10]. Durch diese Strukturierung werden in RDF vorliegende Daten verständlicher.

RDFS bietet dazu strukturbildende Sprachkonstrukte wie unter anderem *rdfs:Resource*, *rdfs:Class*, *rdfs:subClassOf* oder *rdfs:subPropertyOf* sowie eine auf Mengen und Mengenoperatoren wie basierende Semantik an, wodurch eine große Flexibilität gewährleistet werden kann [9][2].

Die Flexibilität von RDFS führt allerdings zu potentieller Instabilität, die zu Problemen in den oberen Schichten führen kann, wenn es darum geht logische Sprachen auf Basis RDFS zu entwickeln. Ursache hierfür ist, dass RDFS ermöglicht, beliebige Aussagen über Aussagen zu treffen. So ist es beispielsweise möglich mit einer Aussage der Art *(rdfs:Class, rdf:type, rdfs:Class)* befindlich in Tripelform die Bedeutung der verschiedenen basalen Konstrukte der Sprache selbst zu verändern [10].

2.4 OWL

Die zur Familie der Beschreibungslogik gehörende Sprache *Web Ontology Language* (OWL), ein Nachfolger von DAML-OIL, ist die vom W3C empfohlene Ontologiesprache [10]. Wie RDFS basiert die Syntax von OWL auf XML und integriert sich über erweiterte Sprachkonstrukte wie *owl:Class*, *owl:Individual* oder *owl:complementOf* in RDF im Gegensatz zu RDFS ist OWL jedoch wesentlich besser für das Zusammenführen von heterogenen Daten geeignet, und ermöglicht zudem wesentlich detailliertere Relationen abzubilden [9][10]. Auch die Stabilität der Sprache selbst ist gewährleistet. So unterscheidet OWL strikt zwischen *Aussagen in der Sprache* und *Aussagen über die Sprache*, wodurch es zu keiner ungewollten Änderung der Semantik der Sprache selbst kommen kann [10].

[9] `http://en.wikipedia.org/wiki/RDFa`

Da mit der gesteigerten Ausdrucksstärke von OWL eine Steigerung der Komplexität einhergeht, verfügt OWL über drei unterschiedliche aufeinander aufbauende Sprachprofile, die von *OWL Lite* über *OWL DL* zu *OWL Full* sowohl von der Syntax als auch von der Semantik mächtiger werden.

Ende des Jahres 2009 ist die überarbeitete Version OWL 2 erschienen. Im Vergleich zu OWL 1 bringt diese einige neue Sprachkonstrukte sowie Verbesserungen mit sich[10]. So war das Lite Profil in OWL 1 zu komplex, was die Entwicklung von auf dieser Version basierenden Software zum logischen Schließen hinderte. Wie OWL 1 bringt auch OWL 2 drei überarbeitete Sprachprofile mit sich *OWL2EL*, *OWL2QL* sowie *OWL2RL* [9]. Die Tatsache, dass OWL zu den auf Beschreibungslogik basierenden Sprachen gehört, führt dazu, dass sich OWL besonders gut zum Ziehen logischer Schlüsse eignet.

2.5 SPARQL

SPARQL Protocol and RDF Query Language (SPARQL) bezeichnet die Standardsprache des W3C zur Formulierung von RDF Queries. Trotz der XML Struktur von RDF Daten und der Existenz von XML Anfragesprachen wie XPath oder XQuery, wird für RDF eine gesonderte Anfragesprache benötigt. Dies liegt daran, dass herkömmliche XML Anfragesprachen die Datenstruktur der XML Dateien verwenden, wohingegen SPARQL auf die durch XML beschriebenen Daten zugreift [1][9].

SPARQL ist eine graph-basierte Anfragesprache, wobei der simpelste Graph einfach ein Tripel der Form *(Subjekt, Prädikat, Objekt)* ist und anstelle eines der drei Elemente eine Variable aufweist. Ein so erstellter Graph wird auch *Pattern* bezeichnet. Abbildung 3 veranschaulicht solch eine simple Anfrage, die in Verbindung mit dem RDF Beispiel aus Abbildung 2 (referenziert in Zeile 4) die Ressource *Sokrates* als Antwort ausgeben würde [9].

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?a_person -- Variablen beginnen in SPARQL mit ?/$
FROM <http:/path/to/RDF/repository>
WHERE {
  ?a_person rdf:type foaf:Person
}
```

Abbildung 3. Eine exemplarische Anfrage in SPARQL.

Neben einfach gehaltenen Queries in schlichter Tripel-Form bietet SPARQL die Möglichkeit komplexere Anfragen unter Verwendung verschiedener Pattern, wie *Group Graph Patterns* oder *Optional Graph Patterns* zu bilden. Vier

[10] http://www.w3.org/TR/owl2-overview/

unterschiedliche Arten der Anfragen geben dem Entwickler einer Query die Möglichkeit Ergebnisse in unterschiedlicher Form zu erhalten. Während *SELECT* ein oder mehrere Bindungen für Variablen gefunden innerhalb eines übereinstimmenden Graphen zurück gibt, liefert *CONSTRUCT* einen neuen Graphen in RDF erzeugt durch Ersetzung von Variablen. *ASK* liefert boolesche Werte in Abhängigkeit von der Übereinstimmung eines gefunden Graphen mit dem Suchpattern und *DESCRIBE* liefert einen RDF Graphen, der das gefundene Ergebnis beschreibt [9].

Auch SPARQL steckt noch in den Kinderschuhen. So fehlt es der Sprache zum Beispiel noch an Aggregationsfunktionen wie *count()*. Auch eine Unterstützung für *Subqueries*, also Anfragen innerhalb einer Anfrage ist noch nicht vorhanden.

2.6 RIF / SWRL

Sowohl beim *Rule Interchange Format* (RIF) als auch bei der *Semantic Web Rule Language* (SWRL) handelt es sich um Frameworks zur Definition von logischen Regeln. Im Nachfolgenden werden die beiden Technologien nun nur kurz vorgestellt da die Frameworks noch nicht ausgereift sind.

RIF ist in erster Linie ein Austauschformat für Regeln, das von heterogenen regelbasierten Systemen verwendet werden soll damit diese untereinander Regeln austauschen können, und ist für diesen Anwendungsfall eine Empfehlung des W3C [10]. Um Kompromisse bei der Vereinheitlichung unterschiedlicher regelbasierter Sprachen zu finden gibt es in RIF drei unterschiedliche Versionen, auch Dialekte genannt. Diese sind der *RIF Core*, *RIF Basic Logic Dialect* sowie *RIF Production Rule Dialect*[11]. *RIF Core* beinhaltet dabei Konstrukte die die meisten regelbasierten Systeme untereinander gemeinsam haben. *RIF BLD* sowie *RIF PRD* erweitern *RIF Core*. *BLD* ergänzt den *Core* dabei zum Beispiel mit logischen Funktionen und hält sich dabei an die Horn-Klausel Form. *PRD* erweitert den *Core* durch Aspekte für Produktionsregeln und ermöglicht es in Abhängigkeit vom ‚feuern' gewisser Regeln Aktionen auszuführen [9].

SWRL hingegen ist kein Format für den Austausch von Regeln heterogener Regelsysteme, sondern ein Vorschlag für eine standardisierte Regelsprache für das *Semantic Web*, eingereicht beim W3C. Dabei basiert SWRL auf einer Kombination aus OWL sowie RuleML, einer XML Auszeichnungssprache für Regeln, geeignet zur Deduktion sowie zum logischen Schließen. Da SWRL keine Lücke zwischen schon bestehenden Regelsprachen schließen möchte, ist es im Vergleich zu RIF weniger komplex und liegt daher auch nur in einer Version vor.

3 Semantic Web Engineering

Nachdem im vorangehenden Abschnitt die Architektur des *Semantic Web* vorgestellt wurde, die es ermöglicht maschinenlesbar Daten und Fakten zu erzeugen (RDF), diese Anzufragen (SPARQL) und für das logische Ziehen von Schlüssen

[11] `http://www.w3.org/2005/rules/wiki/RIF_FAQ`

zu Gruppieren (RDFS / OWL / RIF / SWRL), soll nun in diesem Abschnitt gezeigt werden, wie Webanwendungen im bis jetzt bestehenden Umfeld des semantischen Netzes entwickelt werden könnten.

Will man lediglich eine Website erstellen, die Daten in RDF beinhaltet, so kann man dies über RDFa Auszeichnungen in den XHTML-Dokumenten der Seite oder durch die Verwendung und Einbindung von RDF-Repositories erreichen. Für die durchaus interessantere Entwicklung einer größeren Webanwendung, die unter Verwendung mehrere, heterogener Datenquellen Informationen ableitet und präsentiert, gestaltet sich diese Entwicklung schwieriger. Im weiteren Verlauf dieses Abschnitts wird daher ein grobes Vorgehen für die Entwicklung von Anwendungen des *Semantic Web* beschrieben und zudem kurz auf mögliche Probleme während dieser eingegangen werden. Grundlage hierfür bildet die Arbeit von Omitola et al. in der eine datenintensive Webanwendung unter Verwendung unterschiedlicher, heterogener Datenquellen gebaut wird [11].

Beschaffung angemessener Daten. Zunächst muss festgelegt werden, welche Daten benötigt werden. Der Bedarf wird dabei aus den Zielen der Webanwendung abgeleitet und variiert auch mit diesen. Eines der größten Probleme hierbei besteht darin, dass benötigte Daten möglicherweise in unstrukturierter oder allenfalls in semistrukturierter Form vorliegen. So sind die meisten Daten im Netz immer noch in HTML, PDF oder XML Form vor. Diese müssen dann zunächst über Transformation in eine verwertbare Struktur gebracht werden.

Festlegung einer Darstellungsform. Damit die Daten untereinander in Verbindung gebracht werden können und darüber hinaus aussagekräftig für Maschinen werden, müssen sie möglicherweise noch auf eine einheitliche Form, zum Beispiel RDF, gebracht werden.

Anknüpfpunkte für Daten bestimmen. Eine Voraussetzung für die erfolgreiche Zusammenführung von unterschiedlichen Daten ist die Bestimmung von Verknüpfungen unter bestimmten Eigenschaften der Daten. Dies kann möglicherweise auch bedeuten, dass zwei Datensätze erst mit weiteren Daten angereichert werden müssen, um diese dann zum Beispiel über geographische Eigenschaften zu verbinden.

Persistierung der Daten. Hat man die benötigten Daten soweit aufbereitet und in eine Struktur gebracht, müssen diese noch gespeichert werden. Im wahrscheinlichen Fall der Verwendung von RDF kann dies durch RDF-Repositories, wie zum Beispiel den *Triplestore* 4store[12], realisiert werden. Zum wieder auffinden der Daten können dann SPARQL Anfragen an das Repository gestellt werden.

Darstellung der Daten. Schließlich benötigt man noch einen Konsumenten für die Daten. Also eine Anwendung, die die persistierten Daten verwendet, anschaulich aufbereitet und dem Nutzer darstellt. Wie in [11] beschrieben, stellt dies ein größeres Problem dar, weil es zur Zeit noch keine Frameworks für die Erstellung von grafischen Benutzeroberflächen gibt, die auf RDF-Daten arbeiten. So müssen über SPARQL Anfragen gelieferte Ergebnisse

[12] `http://4store.org`

konvertiert und umformuliert werden, um sie anschließend in einer Benutzeroberfläche zu präsentieren. Hat man die aus dem RDF-Repository gelieferten Daten dann konvertiert, befindet man sich wieder auf festem Boden und kann mit Werkzeugen wie Exhibit[13] eine schnelle Darstellung der Daten erreichen. Exhibit ist ein vom MIT entwickeltes Framework zur Entwicklung von datenlastigen Webseiten, für die dann Interaktionen wie facettiertes Suchen sowie unterschiedliche Darstellungsformen wie Tabellen, Zeitachsen, Karten oder Vorschaubilder bereitgestellt werden.

Die Probleme hervorgerufen durch die Nutzung extrahierter unstrukturierter oder semistrukturierter Daten können durch die Verwendung von Daten, die bereits in RDF vorliegen, umgangen werden. Dort ist dann allerdings das Problem zu lösen, dass solchen Daten, falls diese aus unterschiedlichen Anwendungsbereichen kommen, auch verschiedenen Ontologien zugrunde liegen, wodurch erst nach passenden Anknüpfpunkten gesucht werden muss.

Aus diesem Problem erwächst die Frage, warum es dann keine Einigung auf eine global gültige Ontologie gibt, bei der nicht extra nach Anknüpfpunkten gesucht werden muss. Betrachtet man allein die Komplexität für die Koordinierung dieser Aufgabe, eine solche globale Ontologie zu erstellen, wird schnell klar warum dies nicht möglich ist. Daher überlässt man die Aufgabe, Ontologien zu einzelnen Anwendungsgebiete zu entwickeln, den Domänenexperten und führt diese später zusammen, was dann zu einem Netzwerk aus unter einander verbundenen Ontologien führt. So fällt es leicht Daten innerhalb des entstehenden Ontologie-Graphen zusammenzuführen.

Ein Beispiel für solche Arbeit ist die *Linked Data Community*[14], die durch das Veröffentlichen von Daten und deren Verknüpfungen untereinander, nicht nur Daten zum Entwickeln von *Semantic Web* Anwendungen bereit stellt, sondern auch noch die Entwicklung von verwendbaren Ontologien beschleunigt. Eines der größten Projekte innerhalb dieser Community und somit auch ein Hauptknoten im Ontologie-Graphen[15], ausschnittsweise dargestellt in Abbildung 4 ist das *DBpedia* Projekt[16]. Dessen Zielsetzung ist es, Informationen aus Wikipedia zu extrahieren um diese dann in From von RDF/XML online verfügbar zu machen.

Die hier vorgestellte Herangehensweise ist hierbei allerdings ähnlich zu der Entwicklung von Mashup-Anwendungen, also Anwendungen deren Bestandteil das Kombinieren schon bestehender entfernter Inhalte für das Web 2.0 ist. Die entstehenden Probleme sind, wie sich zuvor gezeigt hat, natürlich technologiespezifisch. Omitola et al. beschreiben in [11] einige für Mashups bekannte Probleme. So kann zum Beispiel die Reinheit der Daten aus entfernten, nicht unter der Kontrolle des Entwicklers stehenden Datenquellen nicht gewährleistet werden. Einige dieser Probleme stellen auch die Entwickler von semantischen Webanwendungen vor Probleme.

[13] `http://simile-widgets.org/exhibit/`
[14] `http://linkeddata.org/`
[15] `http://richard.cyganiak.de/2007/10/lod/imagemap.html`
[16] `http://dbpedia.org/About`

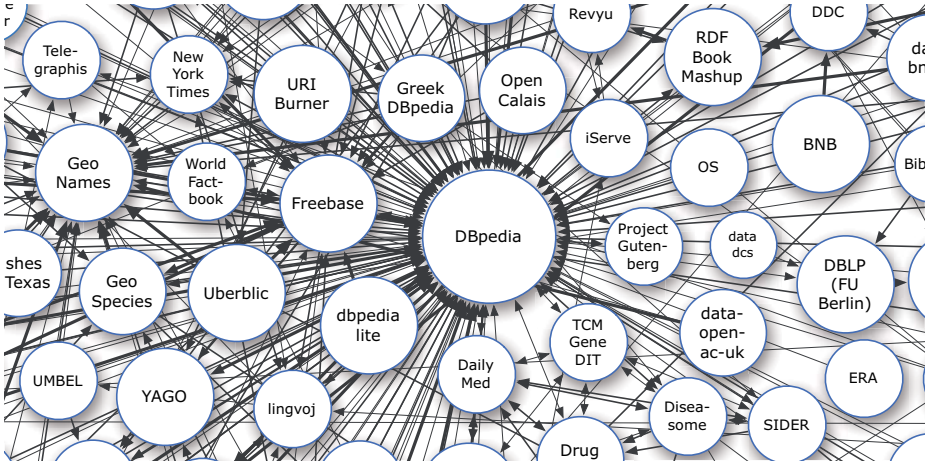


Abbildung 4. Ein Ausschnitt des Linked Data Graphen.

Der darüber hinaus von Omitola et al. genannte Punkt, Mashup-Anwendungen des Web 2.0 seien gegen eine fixe Anzahl von Datenquellen entwickelt, ist in jedem Fall zwar richtig, trifft aber auch auf den bisherigen Stand der Entwicklung im *Semantic Web* zu. Dies liegt daran, dass es bis jetzt noch keine *Verzeichnisdienste* zum dynamischen Auffinden von Anbietern für Datenquellen die sich an gewissen Vereinbarungen halten gibt.

4 Semantic Web Search

Ein wesentlicher Bestandteil des Navigierens durch das Netz ist mittlerweile das Suchen geworden, weshalb sich dieser Abschnitt der *Semantic Web Search* widmet.

In den letzten Jahren wurden einige Suchmaschinen entwickelt, die *Linked Data* crawlen, indem sie RDF Links folgen und Anfragemöglichkeiten auf aggregierte Daten anbieten. Dabei darf allerdings nicht vergessen werden, dass das *Semantic Web* und das *World Wide Web* keine zwei separiert von einander betrachtbaren Netze sind. Vielmehr sind beide mit einander verknüpft, weshalb ein Navigieren anhand von Links auch durch beide Netzteile führen kann [12].

Bestehende Suchmaschinen für das *Semantic Web* kann man grob anhand ihrer Nutzer in zwei Bereiche, nämlich in personenorientiert sowie maschinenorientiert aufteilen.

Personenorientierte Suchmaschinen wie Falcons[17] bieten eine schlagwortbasierte Suche mit ähnlichen Interaktionsmöglichkeiten wie beispielsweise Google an. So ist es einem Suchenden bei Falcons möglich nach Objekten, Konzepten, Ontologien oder Dokumenten zu suchen, die sich wie folgt unterscheiden.

[17] `http://ws.nju.edu.cn/falcons/`

Während eine Such nach Objekten sich mehr für konkrete Dinge wie Personen oder Orte eignet, versucht die Suche nach Konzepten Klassen oder Eigenschaften in Ontologien zu finden. Die Suche nach Ontologien oder Dokumenten hingegen ähnelt mehr der gewöhnlichen Suche im Internet die auf Dokumente abzielt. Überraschend ist, das die meisten personenorientierten Suchmaschinen wie Falcons wider Erwarten keine Möglichkeit anbieten, die Struktur potentieller Daten bei der Query mit einzubeziehen, sprich, Anfragen beispielsweise in RDF zu formulieren [12], was möglicherweise an der Komplexität und dem beim Nutzer vorhandenen Vorwissen liegt.

Für die Darstellung der Suchergebnisse nutzt Falcons die bekannte und für die Suchmaschine verständliche Form der gefundenen Daten, und gibt so beispielsweise an, ob es sich bei einem Suchtreffer etwa um eine Person einen Ort oder ein Dokument handelt. So ist es einem Suchenden möglich, die gefundenen Suchergebnisse anhand bestimmter Konzepte zu filtern. Siehe hierzu exemplarisch Abbildung 5.

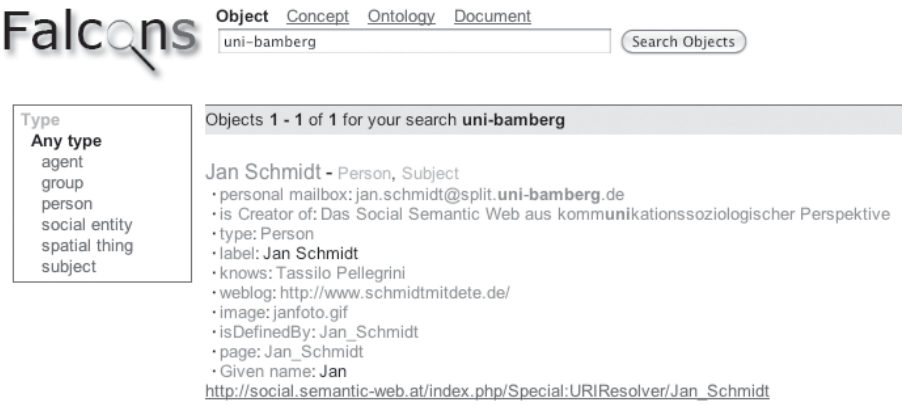


Abbildung 5. Darstellung der Suchergebnisse bei Falcons.

Maschinenorientierte Suchmaschinen wie Watson[18] hingegen bieten APIs an, wodurch Anwendungen für das *Semantic Web Linked Data* nach RDF Dokumenten durchsuchen können [12]. Somit ist es möglich, Anwendungen für das semantische Netz zu entwickeln, ohne selbst einen anwendungsspezifischen Teil für das Suchen von Dokumenten zu erstellen.

Anhand des On2broker, entwickelt von Fensel et al. [13], wird nun ein Überblick über eine mögliche Architektur einer Suchmaschine für das *Semantic Web* dargestellt. On2broker besteht aus vier wesentlichen Bestandteilen einer *Query Engine*, einem *Info Agent*, einer *Inference Engine* sowie einem *Database Manager* und verwendet Ontologien als globales Strukturprinzip.

[18] `http://kmi-web05.open.ac.uk/`

Query Engine Die *Query Engine* versucht unter Verwendung der über den *Database Manager* zugreifbaren Fakten Anfragen zu beantworten. Dazu werden die von den Benutzern gestellten Anfragen in SPARQL übersetzt. Damit der Nutzer allerdings kein Vorwissen über die Interna des On2brokers benötigt, erstellt er seine Anfragen in tabellarischer Form.

Info Agent Der *Info Agent* extrahiert Fakten aus gecrawlten Dokumenten und übergibt diese dem *Database Manager*, damit dieser die Datenbasis aktualisiert. Dazu werden verschiedene Vorgehensweisen verwendet. Bereits in RDF vorliegende oder mit RDF annotierte Dokumente werden in das interne Schema übersetzt. Für andere Dokumente wurden Wrapper entwickelt, die versuchen Fakten aus dem jeweiligen Dokument zu extrahieren, die dann in das interne Schema überführt werden.

Inference Engine Die *Inference Engine* dient dem On2broker dazu, aus vorhandenem Wissen neues Wissen abzuleiten. Da dies ein eher aufwändiger Schritt ist, ist die *Inference Engine* auch von keinem weiteren Teilsystem außer dem *Database Manager* abhängig. Über diesen werden Fakten bezogen, sowie generiertes Wissen gespeichert.

Database Manager Der *Database Manager* dient dem System zur Abstraktion weg vom darunter liegenden Repository. Er hilft darüber hinaus die *Query Engine* von der *Inference Engine* zu trennen.

Die wesentlichen Bestandteile von Suchmaschinen für das World Wide Web sind also auch bei denen für *Linked Data* gegeben. Natürlich erweitert um Operationen, ermöglicht durch die von Maschinen verständliche Semantik. Fraglich ist allerdings, inwiefern Konzepte wie der Aufbau von speziellen Indizes zur besseren Beantwortung der Anfragen benötigt sind und übernommen werden können.

5 Semantic Web Browsing

Nachdem bereits das Entwickeln semantischer Webanwendungen sowie das Suchen innerhalb des semantischen Netzes beschrieben wurde, gibt dieser Abschnitt nun einen kurzen Einblick in das Browsen innerhalb von *Linked Data* zu geben.

Wie gewöhnliche Web-Browser geben Browser für das *Semantic Web* den Nutzern die Möglichkeit, anhand von Links, dargestellt in Form von RDF Tripeln, innerhalb verknüpfter Daten zu navigieren. Dabei können verschiedene RDF Repositories nacheinander besucht werden. Als ein Beispiel für solch einen Browser kann *Disco*[19], eine Entwicklung der Freien Universität Berlin genannt werden.

Wie schon bei der Darstellung von Suchergebnissen bei Falcons kurz angerissen wurde, bieten maschinell verarbeitbare Daten einige Besonderheiten im Vergleich zu herkömmlichen Daten [12]. So sollten Nutzer die Verknüpfungen zwischen Daten weiter erforschen können, sowie die Möglichkeit haben, eine besuchte Seite über die initial angezeigten Daten mit Inhalten aus entfernten Repositories anzureichern. Dies stellt eine Herausforderung für die Darstellung dar,

[19] `http://www4.wiwiss.fu-berlin.de/bizer/ng4j/disco/`

kann aber mit Konzepten wie dem facettierten Anzeigen großer Datenmengen oder Zeitachsen[20] bewältigt werden.

6 Ausblick

Trotz der guten Fortschritte und einem wachsenden Forschungsinteresse im Bereich des *Semantic Web* ist die Kluft zwischen der in der Einleitung genannten Vision aus dem Jahr 2001 und der Realität nur wenig kleiner geworden. Ideen wie private Agenten, die untereinander Wissen austauschen und autonom Entscheidungen treffen sind zunächst aus dem Fokus der Forschung verschwunden. Ursache für diese Stagnation in der Entwicklung ist unter anderem, dass Grundlagentechnologien teilweise noch immer in den Kinderschuhen stecken oder aber noch nicht einmal eine Einigung über den standardisierten Einsatz von Technologien herrscht.

Ein guter Antrieb für die weitere Entwicklung ist die immer größere Menge an vernetztem Faktenwissen, bereitgestellt von der *Linked Data* Community. Denn diese stellt eine hervorragende Testumgebung für die Grundlagentechnologien des *Semantic Web* dar [12]. Auch dass die Suchmaschinenbetreiber Yahoo sowie Google beide mittlerweile teilweise semantische Daten in ihren Suchmaschinen verwenden[21], könnte das *Semantic Web* weiter beflügeln [1].

Aber auch in höheren Ebenen des *Semantic Web* gibt es noch viele Probleme, die es noch zu lösen gilt. So lassen sich zum Beispiel folgende Probleme bestimmen [12]. Die Mehrdimensionalität der Daten stellt Entwickler von grafischen Oberflächen vor Probleme. Das beim Ableiten neuen Wissens benötigte *on-the-fly* Durchlaufen von vernetzen Fakten geht mit einer hohen Laufzeit einher, welches ein intelligentes Caching voraussetzt. Das Mapping zwischen Fakten aus verschiedenen Ontologien sowie das Zusammenführen verschiedener Datensätze mit Fakten über das gleiche Objekt, stellen Probleme dar. Häufig ist die Aktualität von Links innerhalb von RDF-Dokumenten nicht gewährleistet. Oft verweisen RDF Dokumente auf fremde Ressourcen und können nicht sicherstellen, dass diese zu einem späteren Zeitpunkt noch erreichbar sind. Die Frage, wie Korrektheit von Fakten sichergestellt werden kann, ist noch ungeklärt. Ein speziell für Suchmaschinenanbieter bestehendes Problem ist die Frage, ob eine Suchmaschine alle gefundenen Ontologien unterstützen oder eine interne umfassende Ontologie verwenden soll. Im einen Fall ist mit einer schlecht schätzbaren aber großen Menge von Ontoligien zu rechnen, im andern Fall steht man vor dem Problem einer globalen Ontologie.

Eine Parallele, die zusätzlich zu einer Synergie von Technologien führen könnte, besteht zwischen den Bereichen *Semantic Web* sowie *Service Oriented*

[20] Gerade eine Aggregation von Daten anhand der Zeit kann zu Problemen führen, da unterschiedliche Datenquellen verschiedene Zeitabstände verwenden können.

[21] Yahoo hat SearchMonkey entwickelt, eine Plattform die unter Anderem semantische Daten für das Web-Retrieval verwendet, Google hat mit der Entwicklung begonnen semantische Daten im e-Commerce bereich zu Nutzen.

Architecture. So können *Web-Services* beispielsweise eine standardisierte Umsetzung von Agenten für das *Semantic Web* darstellen. Auch das dynamische Auffinden sowie Anbinden von Diensten, ist ein Problemfeld der *Service Oriented Architecture*, dessen Lösung Nutzen im Bereich des *Semantic Web* findet.

Rund um kann man das *Semantic Web* aber immer noch als nicht reif für die breite Masse einschätzen. So ist es einem Leien aufgrund mangelnder Toolunterstützung noch nicht einmal möglich, private Seiten im Netz zu annotieren, ohne dafür vertieftes Wissen über die zu nutzenden Technologien zu haben.

Literatur

1. Janik, M., Scherp, A., Staab, S.: The semantic web: Collective intelligence on the web. Informatik-Spektrum **34** (2011) 469–483 10.1007/s00287-011-0535-x.
2. Kappel, G., Pröll, B., Reich, S., Retschitzegger, W.: Web Engineering - The Discipline of Systematic Development of Web Applications. John Wiley & Sons, Ltd (2006)
3. Shadbolt, N., Berners-Lee, T., Hall, W.: The semantic web revisited. IEEE Intelligent Systems **21**(3) (July 2006) 96–101
4. Berners-Lee, T., Hendler, J., Lassila, O.: The semantic web. Scientific American **284**(5) (2001) 34–43
5. Doan, A., Madhavan, J., Domingos, P., Halevy, A.: Learning to map between ontologies on the semantic web. In: Proceedings of the 11th international conference on World Wide Web. WWW '02, New York, NY, USA, ACM (2002) 662–673
6. Horrocks, I., Parsia, B., Patel-Schneider, P., Hendler, J.: Semantic web architecture: Stack or two towers? In Fages, F., Soliman, S., eds.: Principles and Practice of Semantic Web Reasoning. Volume 3703 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg (2005) 37–41
7. Bratt, S.: Semantic web, and other technologies to watch (January 2007) `http://www.w3.org/2007/Talks/0130-sb-W3CTechSemWeb` Abgerufen 03.01.2012.
8. Wikipedia: Semantic web stack (November 2011) `http://en.wikipedia.org/wiki/Semantic_Web_Stack` Abgerufen 03.01.2012.
9. Fensel, D., Facca, F.M., Simperl, E., Toma, I., Fensel, D., Facca, F.M., Simperl, E., Toma, I.: Semantic web. In: Semantic Web Services. Springer Berlin Heidelberg (2011) 87–104
10. Domingue, J., Fensel, D., Hendler, J.A., eds.: Handbook of Semantic Web Technologies. Springer, Berlin (2011)
11. Omitola, T., Koumenides, C., Popov, I., Yang, Y., Salvadores, M., Szomszor, M., Berners-Lee, T., Gibbins, N., Hall, W., schraefel, m., Shadbolt, N.: Put in your postcode, out comes the data: A case study. In Aroyo, L., Antoniou, G., Hyvönen, E., ten Teije, A., Stuckenschmidt, H., Cabral, L., Tudorache, T., eds.: The Semantic Web: Research and Applications. Volume 6088 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg (2010) 318–332
12. Bizer, C., Heath, T., Berners-Lee, T.: Linked Data - The Story So Far. International Journal on Semantic Web and Information Systems (IJSWIS) (2009)
13. Fensel, D., Angele, J., Decker, S., Erdmann, M., Schnurr, H.P., Staab, S., Studer, R., Witt, A.: On2broker semantic-based access to information sources at the www. In: In Proceedings of the World Conference on the WWW and Internet (WebNet 99), Honolulu, Hawaii, USA. (1999)

Usability von Web-Anwendungen: Psychologische Aspekte

Matthias J. Linhardt

Die Bergner 37, 96049 Bamberg
joerg-matthias.linhardt@uni-bamberg.de

Zusammenfassung. Usability, der Grad an Gebrauchstauglichkeit, ist ein wichtiges Akzeptanzkriterium für interaktive Anwendungen im Allgemeinen und besonders für Web-Anwendungen im Speziellen. Um die Usability einer Web-Anwendung für den Nutzer zu erhöhen, kann die Anwendung an die Fähigkeiten und Bedürfnisse des jeweiligen Nutzers angepasst werden. Da über den individuellen Nutzer einer Web-Anwendung wenig bekannt ist, stellt die vorliegende Arbeit grundlegende psychologische Aspekte der Wahrnehmung, des Gedächtnisses und der Aufmerksamkeit vor, die für alle Nutzer gleichermaßen gelten. Darauf aufbauend werden Implikationen für die Gestaltung von Web-Anwendungen abgeleitet.

Schlagworte: Usability, Web-Anwendung, Psychologie.

1 Einleitung

Die anhaltende Weiterentwicklung der Computertechnologie in den letzten Jahrzehnten ist kaum zu übersehen: Computer werden immer kleiner, günstiger und vor allem leistungsfähiger. Bei all dem Fortschritt der zugrundeliegenden Technologie und dem stetigen Einzug von Computern in unsere Alltagswelt erscheint es umso erstaunlicher, dass ein Aspekt lange Zeit nahezu unverändert blieb: Die Schwierigkeiten, denen die menschlichen Nutzer im Umgang mit dem Computer begegnen.

Der renommierte Informatiker und Experte im Fachbereich Mensch-Maschine-Interaktion Paul Dourish sieht die Ursache hierfür in der Entstehungsgeschichte des Computers. Da die ersten Computersysteme außerordentlich teure Geräte waren, galt Rechenzeit als wesentlich wertvoller als die Arbeitszeit der menschlichen Nutzer. Diese streng ökonomische Sichtweise, die durch den meist militärischen Anwendungskontext der ersten Computersysteme verstärkt wurde, setzte das Hauptaugenmerk auf den Computer und seine Leistung, nicht auf den Menschen und sein Wohlbefinden. Seit diesen Anfängen ist mehr als ein halbes Jahrhundert vergangen. Computer sind heute so günstig und leistungsfähig, dass der Vergleich von Rechenzeit gegenüber Personalkosten überdeutlich zu Gunsten des Menschen ausfällt. Die ursprüngliche Haltung, Leistung über Nutzerfreundlichkeit zu stellen, hat ihre Existenzberechtigung verloren und Raum für eine neue Sichtweise freigegeben, in der der Mensch samt seiner Fähigkeiten und Bedürfnisse im Mittelpunkt steht [1].

Die Zielsetzung, die Computernutzung für den Menschen möglichst angenehm, effizient und intuitiv zu gestalten, entspringt eben dieser noch vergleichsweise jungen Perspektive und hat durch das nicht mehr wegzudenkende Schlagwort Usability Einzug in die Fachsprache der Informatik gehalten. Diese Zielsetzung, dem menschlichen Nutzer entgegenzukommen, beruht jedoch nicht auf Altruismus, sondern hat – ähnlich wie in der eingangs geschilderten Abwägung von Rechenzeit und Personalkosten – seine Grundlage in wirtschaftlichem Pragmatismus: In einer Konsumwelt, in der der Nutzer zugleich Verbraucher ist und die freie Wahl hat, bei welchem Anbieter er seine Nachfrage stillt, muss es dem Nutzer so angenehm wie möglich gemacht werden, damit er nicht zu einem konkurrierenden Anbieter wechselt. Der Wechsel des Nutzers zum Produkt der Konkurrenz wird umso bedrohlicher, je schneller, einfacher und freier von Nachteilen der Wechsel aus Nutzersicht ist. Diese Charakteristika finden sich gerade bei dem vermehrten Angebot von Dienstleistungen im Web wieder, weswegen Usability für Web-Anwendungen von ganz besonderer Bedeutung ist.

Eine Anwendung im Sinne der Usability auf den menschlichen Nutzer zuzuschneiden, setzt grundlegendes Wissen und Verständnis der relevanten Eigenschaften und Fähigkeiten des Nutzers voraus. Die vorliegende Arbeit möchte daher – nach einer kurzen Einführung in Usability und ihre besondere Bedeutung für Web-Anwendungen – einen Einblick in ausgewählte Bereiche der Informationsverarbeitung des Menschen gewähren und daraus resultierende Implikationen für die Gestaltung von Web-Anwendungen aufzeigen.

2 Usability für Web-Anwendungen

Usability wird umgangssprachlich meist mit *Benutzerfreundlichkeit* übersetzt, ein Schlagwort, das durch seinen inflationären Gebrauch an Bedeutungsinhalt und Trennschärfe verloren hat. Etwas sperriger, aber dafür inhaltlich unmissverständlicher ist der offizielle deutsche Begriff *Gebrauchstauglichkeit*. Gebrauchstauglichkeit ist gemäß der internationalen Norm ISO/IEC 9241-11 definiert als „Das Ausmaß, in dem ein Produkt durch bestimmte Benutzer in einem bestimmten Nutzungskontext genutzt werden kann, um bestimmte Ziele effektiv, effizient und zufrieden stellend zu erreichen" [2, S. 267]. Relevant sind hier die Fragen nach dem Benutzer des Produkts, den Zielen und dem Kontext, in welchem das Produkt genutzt wird. Lediglich der letzte Abschnitt der Definition, in dem die gewünschten Eigenschaften der Zielerreichung hervorgehoben werden, entspricht in etwa dem umgangssprachlichen Verständnis von Benutzerfreundlichkeit: Die Zielerreichung soll vollständig und genau (effektiv) sein, in vertretbarem Aufwand in Relation zum zu erreichenden Ziel stehen (effizient) sowie frei von Beeinträchtigungen oder sogar zu positiver Einstellung des Benutzers führend sein (zufrieden stellend) [2].

Usability ist ein wichtiges Akzeptanzkriterium für jegliche interaktive Anwendung. Wie im Folgenden noch ausgeführt werden wird, ist Usability aber für Web-Anwendungen sogar von noch größerer Bedeutung als für traditionelle Programme. Die Abwesenheit oder zu geringe Ausprägung von Usability im Softwarebereich kann schwerwiegende negative Konsequenzen mit sich bringen: Die Folgen reichen von

Frustration des Nutzers über innerer Auflehnung gegenüber dem Produkt hin zur kompletten Nutzungsverweigerung. Vor allem die Nutzungsverweigerung ist ein Problem, das umso bedrohlicher wird, je mehr Wahlfreiheit der Nutzer besitzt. Gerade im Web besteht eben diese Wahlfreiheit in großem Maße, und der Wechsel zur Konkurrenz geht schneller und einfacher vonstatten als bei herkömmlicher Software – nahezu sprichwörtlich mit nur einem Klick. Dies wiederum bleibt nicht ohne finanzielle Tragweite, denn Bereiche wie E-Commerce und E-Government haben echte Geschäftsprozesse in die virtuellen Gefilde des Web eingeführt. Eine Nutzungsverweigerung im Geschäftsfeld des E-Commerce bedeutet zwangsläufig Umsatzeinbußen durch ungetätigte Verkaufsabschlüsse. Ähnliches gilt für E-Government: Im Web angebotene Behördendienste werden nach wie vor größtenteils über konventionelle (für den Staat vergleichsweise kostspielige) Wege in Anspruch genommen, weil das Vertrauen des Nutzers fehlt oder Unklarheiten vorherrschen [2].

Die besondere Bedeutung von Usability speziell für Web-Anwendungen gegenüber herkömmlichen Software-Produkten ergibt sich aber auch aus der weiten – je nach Sprache womöglich weltweiten – Verbreitung sowie besonderen Charakteristika von Web-Anwendungen, die im Folgenden erläutert werden:

- *Keine Software im Karton:* Gegenüber einem in einer Verpackung gekauften Programm, das zu Hause erst ausgepackt und installiert wird, sieht der Nutzer bei einer Webseite sprichwörtlich auf den ersten Blick, was ihm geboten wird.
- *Kein personal investment:* Der Nutzer hat noch keinerlei personal investment getätigt, das ihn dazu verleiten könnte, etwaige Unannehmlichkeiten des Produkts zu übersehen oder zu akzeptieren, um die Güte seiner getätigten Entscheidung für eben dieses Produkt nicht in Frage stellen zu müssen.
- *Große Wahlfreiheit:* Der Nutzer ist in keinster Weise an das Produkt oder den Anbieter gebunden und kann sich binnen Sekunden umentscheiden und einer anderen, ansprechenderen Web-Anwendung den Vorzug geben, da er im Web weitestgehend grenzenlose Wahlfreiheit genießt.
- *Kein Handbuch, keine Schulung:* Einer Webseite liegt kein Handbuch bei, sie ist entweder selbsterklärend strukturiert oder wird mit Nicht-Nutzung gestraft. Auch die gerade für Unternehmenssoftware verbreiteten Software-Schulungen sind nicht auf Web-Anwendungen übertragbar. Etwaige Unzulänglichkeiten des Produkts können nicht durch Anpassung des Nutzers auf das Produkt durch spezielle Schulungen kompensiert werden.
- *Kaum Wissen über das Endgerät:* Die vom Nutzer verwendeten Endgeräte können dem modernsten Stand entsprechen oder auch deutlich betagter sein. So sind zum Beispiel trotz wachsenden Ausbaus von Breitbandinternet immer noch viele Nutzer mit Modems unterwegs, aber auch alte Rechnerkomponenten oder Betriebssysteme sind mögliche Fallstricke, die leicht übersehen werden können. Nicht zuletzt durch die rasante Verbreitung mobiler internetfähiger Endgeräte bleiben jahrzehntealte Faustregeln zur Auflösung oder zum Ladevolumen von Webseiten auch weiterhin gültig.
- *Kaum Wissen über den Nutzer:* Es fällt schwer, sich an den Nutzer der Web-Anwendung anzupassen, denn die Kommunikation im Web ist größtenteils

einseitig. Während der Schuhverkäufer durch geschickte Nachfragen das für den Kunden passende Produkt vorschlagen kann, muss sich der Web-Shop hauptsächlich mit statistischen Maßen begnügen, denn der Nutzer am anderen Ende der Leitung ist für die Web-Anwendung weitestgehend anonym.

Für die Usability von Web-Anwendungen ist gerade der letzte Punkt, das mangelnde Wissen über den Nutzer, von entscheidender Bedeutung. Denn ohne Wissen um die Eigenschaften und Fähigkeiten des Nutzers fällt es schwer, die Web-Anwendung auf seine Bedürfnisse zuzuschneiden, so dass sie von ihm effektiv, effizient und zufrieden stellend genutzt werden kann [2].

Um der Vielzahl an möglichen Nutzern, unabhängig von Nationalität, Kultur, Bildung, Geschlecht oder Alter, gerecht zu werden, muss auf den kleinsten gemeinsamen Nenner aller möglicher Nutzer zurückgegriffen werden: Sie alle sind Menschen und verfügen über bestimmte, interindividuell vergleichbare Charakteristika der menschlichen Informationsverarbeitung. Die vorliegende Arbeit wird im Folgenden drei Aspekte der menschlichen Informationsverarbeitung hinsichtlich ihrer Implikationen zur Usability von Web-Anwendungen beleuchten: Wahrnehmung, Gedächtnis und Aufmerksamkeit.

3 Wahrnehmung

Wie der Inhalt einer Webseite wahrgenommen wird, unterscheidet sich von Mensch zu Mensch weniger, als man annehmen könnte. Denn Wahrnehmung beruht größtenteils auf unbewussten und automatischen Prozessen, die interindividuell und mit geringen Ausnahmen auch interkulturell vergleichbar sind. Im allgemeinen Sprachgebrauch bezieht sich der Begriff „Wahrnehmung“ auf den Gesamtprozess des Erfahrbarmachens von Gegenständen und Ereignissen. Den Vorgang der Wahrnehmung kann man in drei Stufen (siehe Abbildung 1) unterteilen: Empfinden, Organisieren, Identifizieren/Einordnen [3].

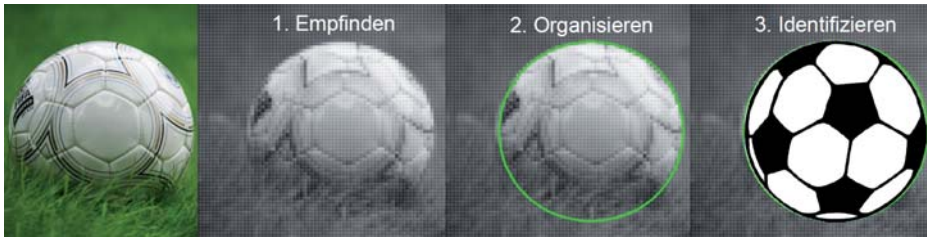


Abb. 1. Vereinfachte Illustration des Ablaufs der Wahrnehmung gemäß dem 3-Stufen-Modell. Von links nach rechts: Der vorliegende Umweltreiz, die durch Stäbchen-Rezeptoren registrierte Empfindung von Helligkeitsunterschieden, die Organisation zusammenhängender Reizempfindungen, die Identifikation des integrierten Objekts gemäß vorhandenem Wissen über die Welt.

Auf der ersten Stufe, der Empfindung, wird physikalische Energie durch die Sinnesorgane in neuronal kodierte Information umgewandelt, die vom Gehirn weiterverarbeitet werden kann. Die Empfindung ist dabei aber nur die allererste Repräsentation von grundlegenden Reizen, also zum Beispiel Hell-/Dunkel-Unterschiede auf den Stäbchen-Rezeptoren der Netzhaut. Die zweite Stufe, die Organisation der Wahrnehmung, baut eine innere Repräsentation des äußeren Objekts oder Ereignisses auf, indem die einzelnen wahrgenommenen Reize gemeinsam integriert und kombiniert werden. So entsteht zum Beispiel aus einzelnen Kanten und Flächen die Repräsentation eines kreisförmigen Objekts. Erst die dritte Stufe, das Identifizieren und Einordnen, erlaubt die Bedeutungszuweisung. Hier wird aus dem kreisförmigen Objekt ein Fußball, eine Münze oder eine Orange, mit entsprechenden verknüpften Eigenschaften. Dies wird nur möglich durch Zuhilfenahme von Erinnerungen oder Theorien über das wahrgenommene Objekt [3].

3.1 Gestaltpsychologie

Dies macht deutlich, dass selbst banalste Alltagserfahrungen komplexen Verarbeitungsschritten unterliegen. Gerade die Fähigkeit, relevante sensorische Informationen zu organisieren und zu integrieren ist Gegenstand eines einflussreichen theoretischen Ansatzes der Wahrnehmungsforschung – der Gestaltpsychologie. Sie entwickelte sich zu Beginn des 20. Jahrhunderts unter Wolfgang Köhler und Max Wertheimer, die überzeugt waren, dass „psychologische Phänomene nur zu verstehen seien, wenn man sie als organisierte, strukturierte Ganzheiten betrachte" [3, S. 113]. Gemäß dem Grundsatz „Das Ganze ist mehr als die Summe seiner Teile" beschreibt die Gestaltpsychologie die menschliche Wahrnehmung als Fähigkeit, Struktur und Ordnung in Sinneseindrücken auszumachen. Dieses Zusammenführen der Einzelteile in der Wahrnehmung findet sich in den sogenannten Gestaltgesetzen wieder. Die Gestaltgesetze beschreiben basale Organisationsprozesse der Wahrnehmung, die es dem Gehirn erleichtern, Ordnung in das Chaos aus einströmenden Sinneseindrücken zu bringen. Erst diese Prozesse ermöglichen eine weitere, sinnvolle Verarbeitung; ohne diese automatisch ablaufenden Prozesse der Informationsverdichtung könnten wir aus der Informationsflut von mehreren Millionen Rezeptoren alleine auf der Netzhaut des Auges nicht die Umrisse eines Fußballs herausfiltern [3].

Bevor die Gestaltgesetze Anwendung finden, macht jedoch zunächst die Gliederung in Bereiche den Anfang der Informationsorganisation. Hier werden die neuronalen Reaktionen der Sinneszellen auf der Netzhaut zu sinnvollen Einheiten zusammengefasst. Wie wichtig das Zusammenführen des sensorischen Inputs zu zusammenhängenden Einheiten ist, soll Abbildung 2.a illustrieren, die sinngemäß die durch die Dichte der Rezeptoren auf der Retina begrenzt aufgelöste Anordnung von Hell-/Dunkel-Wahrnehmungen einzelner Stäbchen-Rezeptoren darstellt. Erst durch Verstärkung von Kontrasten an Kanten ergeben sich eindeutig abgegrenzte Regionen. Der visuelle Kortex hält eine Reihe von informationsverarbeitenden Zellen bereit, die unter anderem auf Kantendetektion spezialisiert sind. Der zweite Schritt beinhaltet die Trennung von Figur und Grund. Durch heuristische Annahme, dass einzelne kleine Regionen innerhalb großer Regionen Figuren vor dem Hintergrund darstellen, werden

Regionen als Objekte identifiziert (siehe Abb. 2.b). Diese Präferenz, Figuren vor und nicht hinter dem Grund zu sehen, lässt sich nur schwer aushebeln (siehe Abb. 2.c) [3].

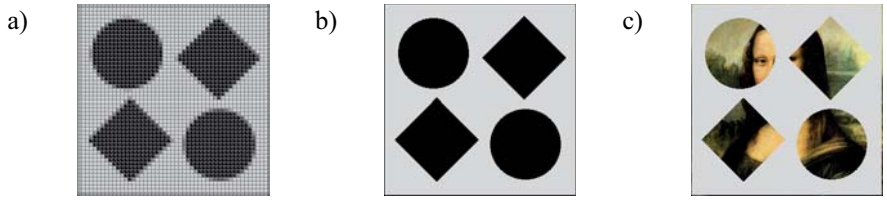


Abb. 2. Organisationsprozesse der Gestaltpsychologie: 2.a) sensorischer Input, 2.b) Kantendetektion und Bereichsgliederung, 2.c) forciertes Aushebeln der Figur-vor-Grund-Präferenz.

Erst jetzt folgen die eigentlichen Gestaltgesetze, die einzelne Objekte in Gruppierungen zusammenfassen. Abbildung 3 veranschaulicht die Aussage „Das Ganze ist mehr als die Summe seiner Teile“. In der Gestaltpsychologie ist sie gleichbedeutend mit „Die Gestalt ist mehr als die Summe der Einzelteile“. Die Abbildung illustriert das Gesetz der Nähe, das besagt, dass nähere Objekte als zusammengehörend aufgefasst werden, das Gesetz der Ähnlichkeit, das ähnliche Objekte gruppiert, und das Gesetz des gemeinsamen Schicksals, das besagt, das Objekte, die sich zum Beispiel gemeinsam bewegen, als zusammengehörend wahrgenommen werden [3].

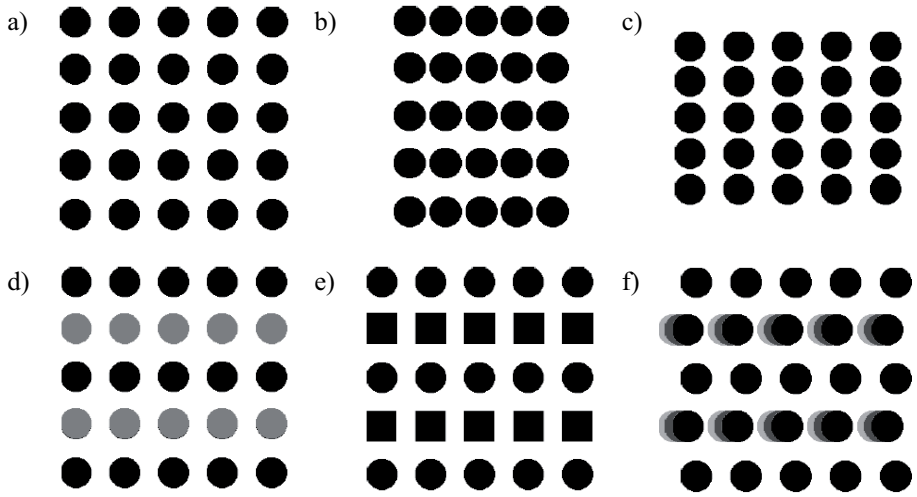


Abb. 3. Illustration der Gestaltgesetze: 3.a) neutraler Ausgangsreiz, 3.b) und 3.c) Gesetz der Nähe, 3.d) und 3.e) Gesetz der Ähnlichkeit, 3.f) Gesetz des gemeinsamen Schicksals.

3.2 Top-Down Einflüsse und der Halo-Effekt

Wahrnehmung ist kein ausschließlich datengeleiteter, reizgetriebener Prozess (*bottom-up*), sondern enthält viele *top-down* Einflüsse, d.h. höhere kognitive Funktionen haben Einfluss auf unsere Wahrnehmung. Man spricht auch von konzept- oder hypothesengeleiteter Verarbeitung. Wir können zum Beispiel problemlos „MEDIENINFORMATIK“ in Abbildung 4 lesen, obwohl die rein physikalischen Reize keine Unterscheidung der Zeichen E und F erlauben. Allein der Kontext erlaubt eine hypothesengeleitete Interpretation der uneindeutigen physikalischen Reize [3].

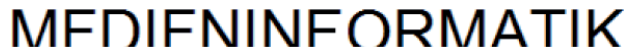


Abb. 4. Illustration hypothesengeleiteter Interpretation mehrdeutiger physikalischer Reize.

Diese top-down Verarbeitung findet unbewusst in vielen Situationen Anwendung, und nicht immer auf eine objektiv-rational vertretbare Art und Weise. So kann der Kontext unzureichend bestimmter Reize auch fälschlich interpretiert werden, wenn man zum Beispiel unbekannte Personen einschätzen muss. So kann zum Beispiel die Gesichtsform (ein offensichtlich irrationales und ungeeignetes Kriterium) einer unbekannten Person die Einschätzung der Selbstständigkeit und geistigen Reife dieser Person beeinflussen. Eine solche Verzerrung der Wahrnehmung, in der von bekannten Eigenschaften auf unbekannte geschlossen wird, nennt man *Halo-Effekt* [4].

3.3 Implikationen

Die selbstverständlich anmutende Wahrnehmung beruht auf einer Reihe von komplexen Prozessen, von der ersten sensorischen Wahrnehmung über das Organisieren der sensorischen Wahrnehmung mittels Gestaltgesetzen bis zum Interpretieren durch top-down Einflüsse. Daraus lassen sich bestimmte Implikationen für die Usability von Web-Anwendungen ableiten. Wenn Usability effizientes, effektives und zufrieden stellendes Zielerreichen meint, muss die Webseite eine möglichst effiziente und möglichst wenig anstrengende Wahrnehmung erlauben. Je unpassender die Webseite für die menschliche Wahrnehmung ist, desto mehr kognitive Kapazitäten werden beansprucht, da automatische Organisationsprozesse nicht mehr ausreichen. Umso weniger stehen dann diese abgezogenen kognitiven Kapazitäten der eigentlichen Zielverfolgung zur Verfügung; es wird anstrengender für den Nutzer, sein Ziel effektiv und effizient zu verfolgen.

Die visuelle Gestaltung der Webseite sollte daher eine möglichst einfache Exzerption von relevanten Objekten und deren Zusammenhang erlauben. So ist es zum Beispiel förderlich, wenn inhaltlich zusammengehörende Elemente durch Nähe oder Ähnlichkeit als zusammengehörend dargestellt werden. Genauso sollten Objekte leicht vom Hintergrund oder anderen irrelevanten Reizen zu unterscheiden sein; gemusterte Hintergründe oder Fotos als Hintergrund eignen sich hierfür denkbar

schlecht, weil sie den Kantendetektoren des visuellen Kortex die Abgrenzung von Objekt und Grund sowie die vorhergehende Bereichsgliederung erschweren. Die Gestaltgesetze können auch das Finden relevanter Informationen erschweren, wenn zur Zielerreichung relevante Inhalte durch Nähe oder Ähnlichkeit in Bezug zu irrelevanten Inhalten gesehen werden. Gesuchte Objekte auf der Webseite werden schwerer gefunden, wenn sie genauso bunt wie das Werbebanner am Rand sind. Aber auch Text nahe einer Werbung wird eher missachtet und erst später wahrgenommen, weil die Nähe zum irrelevanten Reiz Werbung ebenfalls Irrelevanz suggeriert [5, S. 103f.].

Der Tenor der Gestaltpsychologie, die Gestalt sei mehr als die Summe ihre Einzelteile, zeigt sich auch auf Webseiten. Der Nutzer bewertet nicht einzelne Textbausteine und Buttons, sondern hat auf einen Blick einen Gesamteindruck der Webseite. Ist dieser erste Eindruck negativ, wirkt sich das dank Halo-Effekt auf alle weiteren Interaktionen mit dem Inhalt der Webseite ungünstig aus. Lindgaard und Kollegen haben sowohl zeigen können, dass der erste Eindruck innerhalb von nur 50ms aufgebaut wird [6], als auch, dass dieser erste visuelle Eindruck mittels Halo-Effekt einen Einfluss auf die Nutzereinschätzung von Usability und Trustworthiness der Webseite hat [7]. Betrachtet man die Kürze dieser Zeitspanne, so wird deutlich, dass die Webseite sich von Anfang an im besten (und gut überschaubaren) Licht darstellen muss. Komplizierte Strukturen, die auf den zweiten Blick sinnvoll erscheinen mögen, werden evtl. keines zweiten Blickes gewürdigt. Auch einleitende Animationen bei erstmaligem Betreten der Webseite sollten unter diesem Blickwinkel kritisch hinterfragt werden.

4 Gedächtnis

Nach dem Informationsverarbeitungsansatz bezeichnet Gedächtnis die Fähigkeit, Informationen aufzunehmen, zu speichern und bei Bedarf wieder abzurufen. Gedächtnisleistung kann explizit, aber auch implizit erfolgen. Sogar sehr viele alltägliche Dinge beruhen auf impliziter Gedächtnisleistung, d.h. sie benötigen keine bewusste Gedächtnisabfrage. Vor allem prozedurale Gedächtnisleistungen, also solche, die mit der Ausführung einer Tätigkeit einhergehen, sind meist implizit. Einen Schuh binden oder Fahrradfahren sind Beispiele für Dinge, die wir ohne bewusstes Erinnern der einzelnen benötigten Handgriffe vollführen. Im Gegenteil, es kann sogar schwierig sein, die einzelnen Handgriffe richtig aufzuzählen, wenn man es einer fremden Person erklären muss. Das implizite Gedächtnis unbewusster oder nicht-bewusstseinsfähiger Erinnerungen nimmt daher eine wesentliche Rolle in unserem Alltagserleben ein [3].

Nicht desto trotz sind die expliziten Gedächtnisprozesse nicht weniger wichtig, sie machen nur einen kleineren Anteil aus, denn für bewusste Gedächtnisprozesse stehen nur vergleichsweise kleine Kapazitäten zur Verfügung. Dies ist kein Zufall oder einem schlechten Bauplan der Natur geschuldet, sondern durchaus sinnvoll, wenn man sich vor Augen führt, was für eine riesige Menge die auf den Menschen einströmenden Reize darstellen. Die Beschränktheit des Kurzzeitgedächtnisses auf einige wenige Informationseinheiten stellt die Fokussierung des Bewusstseins auf die

für die Zielerreichung relevanten Informationen sicher und erleichtert die Ausrichtung der kognitiven Ressourcen [3].

Nichts desto trotz ist die begrenzte Kapazität des Kurzzeitgedächtnisses ein zentraler Faktor, den es bei der Planung von Interaktion zwischen Mensch und Web-Anwendung zu bedenken gilt. Die Kapazität des Kurzzeitgedächtnisses wird häufig mit 7(+/-2) Elementen angegeben. Die wahre Kapazität liegt vermutlich sogar deutlich darunter, bei etwa 2 bis 4 Einheiten. Grund für diese systematische Überschätzung der ursprünglichen Experimente ist die Nichtbeachtung des sensorischen Gedächtnisses, das einen nicht unerheblichen Beitrag zur Aufgabenlösung (dem freien Reproduzieren einer kurz dargebotenen Zahlenreihe) leistet. Das sensorische Gedächtnis besteht aus einem visuellen Speicher, der ein Nachbild optischer Reize über etwa 0,5 Sekunden aufrechterhält, sowie einem auditiven Speicher, der echoartig bis zu 2 Sekunden in die Vergangenheit reicht. Das sensorische Gedächtnis leistet also dem Kurzzeitgedächtnis Hilfestellung beim Erinnern an die Zahlenfolge, wobei die zuletztgelesenen Zahlen zum Beispiel als Echo der eigenen inneren Stimme noch im auditiven Gedächtnis vorliegen [3].

Eine weitere Hilfestellung, die dem Kurzzeitgedächtnis zu Gute kommt, ist die Methode des *Chunking*. Dabei werden mehrere Informationseinheiten zu einer einzelnen Informationseinheit (einem sogenannten *Chunk*) verschmolzen, wodurch der Bedarf an Kapazität schrumpft (zum Beispiel kann die Ziffernfolge 2, 0, 1 und 2 zu einem einzigen Chunk, der Jahreszahl 2012, zusammengefasst werden). Dies zeigt, wie flexibel der Begriff Informationseinheit zu verstehen ist und verdeutlicht, dass das Kurzzeitgedächtnis anders als der Arbeitsspeicher des Computers nicht durch Größe der Datenmenge bedingt limitiert ist, sondern durch die Anzahl simultaner Aufmerksamkeitsausrichtung. Chunking ist eine effiziente Strategie, um die Menge zu enkodierender Information zu verringern. Allerdings erfordert diese Neuanordnung der einzelnen Informationseinheiten zu einem zusammengehörenden Chunk Zeit, die unter Umständen nicht zur Verfügung steht, und läuft in der Regel nicht automatisch, sondern unter Zuhilfenahme bewusster Prozesse ab. Außerdem ist die Anwendbarkeit von Chunking stark vom präsentierten Reizmaterial und der Reihenfolge einzelner Informationseinheiten abhängig. Es ist daher schwer vorherzusagen, in welchen Situationen Chunking zur Schonung der begrenzten Kapazität des Kurzzeitgedächtnis wirklich angewendet werden kann [3].

Im Gegensatz zur begrenzten Kapazität von Kurzzeit- und sensorischem Gedächtnis verfügt das Langzeitgedächtnis über keinerlei bekannte Begrenzungen hinsichtlich Speicherdauer und -volumen. Der für permanenten Informationserhalt notwendige Transfer in das Langzeitgedächtnis geschieht teils implizit direkt aus dem sensorischen Gedächtnis oder explizit aus dem Kurzzeitgedächtnis [3].

4.1 Gedächtniseffekte

Für den erneuten Abruf von Eingespeichertem aus dem Langzeitgedächtnis gibt es zwei Szenarien: die freie Reproduktion (*recall*) und das reizgetriebene Wiedererkennen (*recognition*). In beiden Fällen helfen sogenannte Abrufhilfen oder Hinweisreize (*cues*) beim Abruf aus dem Langzeitgedächtnis. Im Falle der

Reproduktion müssen diese jedoch selber generiert werden, während sie beim Wiedererkennen bereits vorliegen. Daher ist das Wiedererkennen wesentlich häufiger von erfolgreichem Gedächtnisabruf gekrönt als die Reproduktion [3].

Damit ein Abrufversuch überhaupt erfolgreich sein kann, muss die Information auch vorher ins Langzeitgedächtnis aufgenommen worden sein. *Primacy*- und *Recency-Effekt* beschreiben, wann Erinnerungen besonders gut ins Langzeitgedächtnis enkodiert werden. Der Primacy-Effekt besagt, dass von einer Reihe von Reizen diejenigen besonders gut eingespeichert werden, die besonders früh erscheinen, weil sie nicht mit anderen Reizen gleichzeitig um die Aufnahme in das Langzeitgedächtnis konkurrieren müssen. Der Recency-Effekt tritt auf, wenn von einer Reihe von Reizen die zuletzt erschienen besonders gut ins Langzeitgedächtnis eingespeichert werden, weil sie nicht durch nachfolgende Reize im Kurzzeitgedächtnis überschrieben werden und durch die längere Verweildauer im Kurzzeitgedächtnis wiederum mehr Zeit für den Übergang ins Langzeitgedächtnis zur Verfügung steht. Eine mögliche Illustration des Zusammenwirkens von Primacy- und Recency-Effekt ist die Darstellung der Behaltensleistung als eine U-förmige Kurve, die einen Abfall der Gedächtnisleistung im mittleren Bereich der Zeitachse aufzeigt [3].

Die vorhin genannten impliziten Gedächtnisprozesse können unbewusst die Verarbeitung von Reizen beeinflussen. Der Effekt des *Priming* bezeichnet den Umstand, dass ein vorangegangener Reiz die Interpretation nachfolgender Reize beeinflusst, ohne notwendigerweise bewusst wahrgenommen zu werden. Dies geschieht, indem der vorangegangene Reiz (zum Beispiel das Bild eines lachenden Kindes) bottom-up Gedächtnisinhalte aktiviert, die mit dem Reiz verknüpft sind (zum Beispiel Freude). Diese aktivierten Gedächtnisinhalte beeinflussen dann die Verarbeitung nachfolgender Reize, da die stattgefunden Vor-Aktivierung eine erneute Aktivierung begünstigt. Priming setzt keine bewusste Verarbeitung des vorangegangenen Reizes voraus, der primende Reiz muss noch nicht einmal bewusstseinsfähig sein. Dies konnte in zahlreichen Experimenten nachgewiesen werden, die den primenden Reiz nur so kurz auf dem Bildschirm aufflackern ließen, dass eine bewusste visuelle Wahrnehmung nicht stattfand ($t < 20ms$). Die Versuchspersonen gaben dann bei Befragung an, nichts gesehen zu haben, obwohl ihr Verhalten auf die danach bewusst wahrgenommen Reize durch den primenden Reiz beeinflusst worden war. So wird zum Beispiel das mehrdeutige Verhalten einer fiktiven Person von Versuchspersonen als negativer bewertet, wenn zuvor Worte wie „unkind“ und „hostile“ eingeblendet wurden – auch wenn die Darbietung dieser primenden Reize unter der bewussten Wahrnehmungsschwelle blieb [8].

Der *Mere-Exposure-Effekt* beruht ähnlich wie der Priming-Effekt ebenfalls auf unbewussten Prozessen. Er beschreibt, dass die Einstellung zu Personen oder Dingen durch mehrfache Darbietung positiv beeinflusst werden kann – oder anders ausgedrückt ein höherer Grad an Vertrautheit durch implizite Gedächtnisprozesse mit einer positiveren Bewertung korreliert [3, 8].

4.2 Implikationen

Der Nutzer einer Web-Anwendung muss während seiner Zielverfolgung oft eine ganze Agenda an einzelnen Schritten im Kopf behalten. Soll die Web-Anwendung so gestaltet werden, dass der Gedächtnisleistung des Nutzers entgegengekommen wird, bieten sich zwei mögliche Ansatzpunkte an. Zum einen sollte der Anteil irrelevanter Informationen so weit wie möglich reduziert werden, damit diese nicht unnötig mit relevanten Informationen um die begrenzte Kapazität des Kurzzeitgedächtnisses konkurrieren. Zum anderen sollte der Abruf aus dem Langzeitgedächtnis durch zusätzliche Abrufreize erleichtert werden, indem relevante Informationen zur Abarbeitung der Agenda jederzeit einsehbar zur Verfügung gestellt werden. Dies kann zum Beispiel als Einblendung der bereits im Warenkorb abgelegten Objekte umgesetzt werden, oder durch eine Navigationsstruktur am Seitenrand, die Aufschluss über die momentane Position des Nutzers in der Ebenenstruktur stark verschachtelter Seiten gibt.

Der Primacy-Effekt verdeutlicht, wie wichtig der erste Eindruck ist, da er besonders gut in Erinnerung bleiben wird. Eine unangenehme Erfahrung zu Beginn der Webseite-Nutzung (wie zum Beispiel durch ein nicht abbrechbares Introvideo) sollte vermieden werden. Gleichzeitig zeigt der Recency-Effekt, dass auch das Ende der Zielverfolgung des Nutzers besonders gut in der Erinnerung haften bleiben wird. Daher sollten wichtige Informationen, die nicht vergessen werden dürfen, am Schluss noch einmal erneut hervorgehoben werden (zum Beispiel das Erinnern an die vom Kunden zunächst zu tätigende Vorkassenzahlung auf der Bestätigungsseite nach Abschluss einer Bestellung). Zugleich bietet der Abschluss der Zielerreichung eine günstige Gelegenheit, dem Nutzer mit einem guten Bild (zum Beispiel einer besonders freundlichen Verabschiedung) in Erinnerung zu bleiben, da ja die Eindrücke zum Abschluss besonders haften bleiben.

Auch der Priming-Effekt und sein Einfluss auf die Einstellung des Nutzers zur Web-Anwendung sollten besonders bei Illustrationen und der Gestaltung des Logos bedacht werden. Diese sollten so gewählt werden, dass sie die Wahrnehmung der Web-Anwendung aus Nutzersicht hinsichtlich der zu vermittelnden Werte positiv unterstützt. Auf einer FAQ-Seite zur Sicherheit finanzieller Transaktionen erzielt die Illustration des Begriffs Sicherheit durch das Bild eines Sicherheitsschlosses sicherlich mehr subjektive Trustworthiness beim Nutzer als durch das Bild eines maskierten Einbrechers. Auch das Logo der Webseite kann (zum Beispiel durch einen angedeuteten lächelnden Mund) einen unbewussten positiven Übertrag auf die Wahrnehmung der ganzen Webseite induzieren.

Der Mere Exposure-Effekt hingegen erscheint zunächst schwer für die eigene Web-Anwendung auszunutzen, da es nicht direkt möglich ist, die Vertrautheit des Nutzers mit der Web-Anwendung durch wiederholte Darbietung zu erhöhen. Es gibt jedoch zwei weniger offensichtliche Möglichkeiten, das Prinzip der Vertrautheit indirekt zu nutzen. Erstens kann eine einheitliche Gestaltung aller Einzelseiten einer Webseite sowie aller im Portfolio befindlichen Web-Anwendungen (Stichwort Corporate Design) eine positive Einstellung gegenüber dem einheitlichen Bild auslösen, indem Vertrautheit durch wiederholte Darbietung einzelner Eigenschaften (wie zum Beispiel Logo, Anordnung, Farbwahl) erreicht wird. Durch Priming kann

dann ein irrationaler und unbewusster Übertrag von der durch Vertrautheit ausgelösten positiven Einstellung gegenüber der Web-Anwendung auf unbekannte Eigenschaften, wie zum Beispiel Trustworthiness, erfolgen. Zweitens kann das Befolgen von gängigen Standards hinsichtlich dem Aufbau und der Navigationsstruktur der Webseite sowie der Anordnung bestimmter archetypischer Elemente das Gefühl der Vertrautheit erhöhen. Wenn zu vermuten ist, dass der Nutzer Erfahrung mit vergleichbaren Web-Anwendungen besitzt (zum Beispiel mit anderen Onlineshops, die alle den Warenkorb rechts oben angeordnet haben), sollte die eigene Anwendung den dort gebräuchlichen Organisationsprinzipien entsprechen, um beim Nutzer einen Mere Exposure-Effekt durch die Vertrautheit mit gewohnten Strukturschemata zu ermöglichen – auch wenn er genau diese Web-Anwendung zum ersten Mal nutzt.

5 Aufmerksamkeit

Wie in den vorangegangenen Abschnitten zu Wahrnehmung und Gedächtnis aufgezeigt wurde, strömt einerseits kontinuierlich eine gewaltige Datenflut aus Umweltreizen durch die Sinnesorgane auf den Menschen ein, während andererseits das Kurzzeitgedächtnis, das ähnlich dem Arbeitsspeicher des Computers Ort bewusster Informationsverarbeitung ist, nur über vergleichsweise stark begrenzte Kapazitäten verfügt. Diese offensichtliche Divergenz und die daraus resultierende Tatsache, dass unmöglich alle einströmenden Informationen der bewussten Verarbeitung zur Verfügung gestellt werden können, macht einen Mechanismus notwendig, der einzelne Umweltreize gezielt für die Weiterleitung an das Bewusstsein selektiert.

Diese Schlüsselfunktion im Zusammenspiel von aufgenommenen Sinnesreizen und ihrer Bewusstheit übernimmt die Aufmerksamkeit. Sie fungiert als Nadelöhr zwischen der riesigen äußeren und der begrenzten inneren Welt – man spricht von der Filterfunktion der Aufmerksamkeit. Im visuellen Bereich lässt sich Aufmerksamkeit gut durch die Analogie eines sehr kleinen Suchscheinwerfers illustrieren, der zwar äußerst schnell hin und her springen kann, aber immer nur wenig gleichzeitig beleuchten kann. Diese Filterfunktion bildet die Grundlage für zielgerichtetes Planen und Handeln, indem eine Fokussierung der beschränkten Ressourcen auf aktuell handlungsrelevante Reize durch das Ausblenden momentan irrelevanter Reize sichergestellt wird. Ähnlich wie bei einem Radio, bei dem nur das gerade eingestellte, aber jederzeit veränderbare Frequenzband zu hören ist, obwohl alle anderen Radiowellen gleichzeitig als Umweltreiz vorliegen, lässt die Aufmerksamkeit eben nur das gerade Relevante ins Bewusstsein [3].

5.1 Ausrichtung der Aufmerksamkeit

Wie bereits erwähnt wurde kann die Aufmerksamkeit schnell zwischen verschiedenen Umweltreizen wechseln, bis sie auf einem relevanten Stimulus verweilt. Diese Ausrichtung der Aufmerksamkeit kann sowohl willentlich als auch automatisch

gelenkt erfolgen. Letzterer Fall dient in der Funktionsweise vor allem dem Überleben das Organismus und wurzelt daher tief im menschlichen Verhaltensrepertoire. So können potentiell Gefahren ankündigende Reize automatisch und entgegen der willentlichen Ausrichtung die Aufmerksamkeit auf sich ziehen. Dazu zählen intensive Farbreize [9, 10] und die unerwartete Wahrnehmung von Bewegung, die auf die Belebtheit eines Objekts schließen lässt und damit ein möglicher Hinweis auf ein Raubtier sein kann [11]. Da das Entdecken von Bewegung in der visuellen Perzeption eine wichtige Funktion einnimmt, existiert hierfür eine eigene physiologische Struktur: Für die Sensitivität gegenüber Bewegung existieren im primären visuellen Kortex Gruppen komplexer Kortex-Zellen, die allein auf die Detektion von Bewegungen spezialisiert sind [12].

5.2 Orientierungsreaktion und Yerkes-Dodson-Gesetz

Die Ausrichtung der Aufmerksamkeit ist nur ein Teil der automatischen Reaktion des Körpers auf unerwartete Umweltreize wie plötzlich wahrgenommene Bewegung. Die Aktivierung des Organismus (die einhergeht mit einer Erhöhung von Hautleitfähigkeit, Blutdruck und Muskeltonus) auf unerwartet auftretende Stimuli wird als Orientierungsreaktion bezeichnet [13]. Während eine erhöhte Aktivierung eine passende Vorbereitung auf fight-or-flight Situationen im Sinne der Evolution bietet, kann sie jedoch für die Ausführung kognitiver Leistungen nachteilig sein.

Der Einfluss von erhöhter Aktivierung auf die Leistungsfähigkeit wird von dem sogenannten Yerkes-Dodson-Gesetz beschrieben. Es postuliert einen umgekehrt U-förmigen Zusammenhang zwischen Erregungsniveau auf der horizontalen und Leistung auf der vertikalen Achse eines Koordinatensystems. Dieser Zusammenhang wird von der Schwierigkeit der zu bearbeitenden Aufgabe moderiert. Gemäß Yerkes-Dodson-Gesetz ist die Leistungsfähigkeit grundsätzlich bei moderater Aktivierung am besten, wohingegen hohe und niedrige Aktivierung in schlechterer Leistung resultieren. Je leichter die Aufgabe ist, desto mehr verschiebt sich dieses Optimum in Richtung höherer Aktivierung. Mit zunehmender Aufgabenschwierigkeit verschiebt sich das Optimum in Richtung niedrige Aktivierung [3].

5.3 Implikationen

Wie bereits erläutert ist die Aufmerksamkeit begrenzt. Auch wenn in der subjektiven Erfahrung durch die schnelle Ausrichtung der Aufmerksamkeit der Eindruck entstehen mag, man könne sich auf alles oder zumindest vieles gleichzeitig konzentrieren, wird die visuelle Umgebung doch vielmehr wie mit einem winzigen Suchscheinwerfer sukzessive nach relevanten Stimuli abgesucht. Da die Relevanz eines Reizes nicht eine dem Reiz inhärente Eigenschaft darstellt, sondern immer gemäß der momentanen Zielsetzung neu bewertet werden muss (so wird zum Beispiel ein Briefkasten, den man sonst beim Vorbeilaufen ignoriert, zum relevanten Objekt, sobald man einen Brief absenden möchte), kostet die Suche nach relevanten Reizen

umso mehr Zeit, je mehr irrelevante Reize zeitgleich verfügbar sind und ebenfalls auf Relevanz überprüft werden müssen.

Die Zielerreichung des Nutzers einer Web-Anwendung büßt demnach umso mehr an Effizienz ein, je mehr Sucharbeit notwendig ist, um relevante Umweltreize zu entdecken. Eine übersichtliche und schlanke Aufmachung der Web-Anwendung kann diese Sucharbeit deutlich reduzieren, weshalb eine klare Struktur mit deutlicher visueller Hervorhebung von aufgabenspezifisch relevanten Informationen unterstützend wirkt. Gleichzeitig wird die Sichtbarkeit relevanter Informationen natürlich umso mehr erhöht, je weniger sie mit irrelevanten Informationen um Aufmerksamkeit konkurrieren müssen, weshalb die Anzahl an für die Zielerreichung des Nutzers irrelevanten Reizen minimiert werden sollte.

Dies wird gestützt durch den empirischen Fund, dass das Maß an Desorientierung für Nutzer einer Web-Anwendung mit zunehmender Anzahl an dargestellten Objekten steigt. Dabei besteht der gefundene Zusammenhang zwischen Objektanzahl und Desorientierung bemerkenswerterweise unabhängig von Interneterfahrung, Nutzungshäufigkeit und Geschlecht des Nutzers. Lediglich das Alter hat einen nachgewiesenen Einfluss, wobei ältere Probanden schneller desorientiert sind als jüngere. Dieser moderierende Einfluss des Alters auf die Desorientierung ist jedoch umso geringer, je mehr Objekte in der Web-Anwendung dargestellt werden. Demnach können auch junge Nutzer ab einem bestimmten Grad an Informationsüberladung die negativen Auswirkungen hoher Objektdichte nicht mehr kompensieren [10, S. 239].

Des weiteren wurde erläutert, wie bestimmte Reize eine Orientierungsreaktion samt automatischer Aufmerksamkeitsausrichtung auslösen können. Da die willentliche Ausrichtung der Aufmerksamkeit auf relevante Informationen für die Zielverfolgung elementar ist und jede unwillentliche, automatische Ausrichtung auf andere, für die Zielverfolgung irrelevante Reize unnötig Zeit kostet, wird die Zielerreichung durch solche Stimuli hinsichtlich Effizienz empfindlich beeinträchtigt. Aus diesem Grund sollte die Web-Anwendung wo immer möglich auf Objekte und Gestaltungshilfsmittel verzichten, die die Aufmerksamkeit des Nutzers vom eigentlich relevanten Inhalt ablenken und somit die effiziente Zielverfolgung erschweren. Dazu gehören bewegte Objekte wie zum Beispiel animierte Werbebanner genauso wie kräftige Signalfarben – eben alles, was dem Nutzer sprichwörtlich „ins Auge springt“.

Das solche Objekte nicht nur durch die automatische Ausrichtung der Aufmerksamkeit die Effizienz, sondern durch die Orientierungsreaktion auch die Effektivität der Zielerreichung beeinträchtigen können, wird bei Betrachtung des vom Yerkes-Dodson-Gesetz postulierten Zusammenhangs von Erregungsniveau und Leistung ersichtlich. Da durch die erwähnte Orientierungsreaktion eine Erhöhung der Aktivierung erfolgt, muss ein umso höheres Erregungsniveau des Nutzers angenommen werden, je mehr Objekte von der Web-Anwendung dargestellt werden, die die bereits genannten aufmerksamkeitserheischenden Merkmale besitzen. So wird eine puristische Darstellung mit einigen wenigen Objekten weniger Aktivierung beim Nutzer induzieren als eine mit einer Vielzahl an farbigen und animierten Text- und Bildobjekten überfrachtete Oberfläche. Gemäß Yerkes-Dodson-Gesetz ist ein erhöhtes Erregungsniveau des Nutzers jedoch lediglich für die Leistung bei sehr einfachen Aufgaben von Vorteil, für mittelschwere oder schwierige Aufgaben würde sich dies aber negativ auf die Leistung auswirken. Je weniger über die (für jeden Nutzer subjektive) Schwierigkeit der Zielerreichung mittels Web-Anwendung

bekannt ist, desto mehr sollte daher eine weniger hohe Aktivierung des Nutzers angestrebt werden, da ein zu hohes Erregungsniveau immer nachteilig ist, ein zu niedriges jedoch lediglich die Gefahr des Nutzungsabbruchs auf Grund von Langeweile birgt. Gerade diese Gefahr dürfte aber in der Regel vor allem bei kommerziellen Web-Anwendungen, die der Nutzer aktiv zur Zielerreichung verwendet, vernachlässigbar sein. Somit wiegt – auch wenn Extreme grundsätzlich zu vermeiden sind – die Gefahr einer zu hohen Aktivierung im Kontext wirtschaftlich relevanter Web-Anwendungen schwerer als das Risiko zu geringer Aktivierung. So ist zum Beispiel bei einer Web-Anwendung zum Onlinebanking ein durch erhöhte Aktivierung begründeter Fehler des Nutzers beim Ausfüllen einer Onlineüberweisung ein größeres, folgenschwereres Risiko als die Gefahr, dass der Nutzer wegen zu geringer Aktivierung den Vorgang und sein Vorhaben, eine Überweisung zu tätigen, ganz abbricht.

6. Fazit

Usability ist ein elementares Gütekriterium für interaktive Anwendungen, dem im Kontext von Web-Anwendungen zusätzliche Bedeutung zufällt. Die vorliegende Arbeit hat aufgezeigt, dass sich aus interindividuell und interkulturell vergleichbaren Aspekten menschlicher Informationsverarbeitung Empfehlungen zur Gestaltung von Web-Anwendungen ableiten lassen, die auch ohne explizites Wissen über individuelle Nutzergruppen positives Einwirken auf den Grad der Gebrauchstauglichkeit erlauben. Dies bedeutet jedoch weder, dass zusätzliches Wissen um den Nutzer der Anwendung dadurch obsolet werden oder an Nützlichkeit verlieren würde, noch kann die vorliegende Arbeit einen Anspruch auf Vollständigkeit bezüglich der für Usability von Web-Anwendungen relevanten psychologischen Aspekte erheben. Vielmehr möchte sie dem Leser als verständlicher Einstieg in elementare psychologische Phänomene dienen und grundlegendes Interesse sowie Verständnis für die Nützlichkeit der Psychologie für konkrete Anwendung im Feld der Informatik wecken.

Quellen

1. Dourish, P.: Where the Action Is: The Foundations of Embodied Interaction. The MIT Press, Cambridge, MA (2004)
2. Hitz, M., Leitner, G.: Usability von Web-Anwendungen. In: Kappel. G., Pröll, B., Reich, S., Retschnitzegger, W. (Hrsg.): Web-Engineering: Systematische Entwicklung von Web-Anwendungen, S. 265--295. dpunkt.verlag, Heidelberg (2004)
3. Zimbardo, Ph., Gerrig, R.: Psychologie. Springer, Berlin u.a. (1999)
4. Zebrowitz, L., Montepare, J.: Impressions of Babyfaced Individuals Across the Life Span. Developmental Psychology 28(6), S. 1143--1152 (1992)
5. Abbey, B.: Instructional and cognitive impacts of Web-based education. Idea Group Inc, Hershey, PA (2000)

6. Lindgaard, G., Fernandes, G., Dudek, C., Brown, J.: Attention web designers: You have 50 milliseconds to make a good first impression!. Behaviour & Information Technology 25(2), S. 115--126 (2006)
7. Lindgaard, G., Dudek, C., Sen, D., Sumegi, L., Noonan, P.: An Exploration of Relations Between Visual Appeal, Trustworthiness and Perceived Usability of Homepages. ACM Transactions on Computer-Human Interaction 18(1), Artikel Nr. 1 (2011)
8. Aronson, E., Wilson, T., Akert, R.: Social Psychology. Pearson Education, Upper Saddle River (2005)
9. Snowden, R.: Visual Attention to Color: Parvocellular Guidance of Attentional Resources?. Psychological Science 13(2), S. 180--184 (2002)
10. Jukl, G.: Psychologische Aspekte des Webdesigns unter besonderer Berücksichtigung der Gestaltpsychologie. In: Vitouch, P.: Psychologie des Internet: Empirische Arbeiten zu Phänomenen der digitalen Kommunikation, S. 206--241. WUV, Wien (2001)
11. Christ, S., Abrams, R.: Motion Onset Captures Attention. Psychological Science 14(5), S. 427--432 (2003)
12. Schandry, R.: Biologische Psychologie: Ein Lehrbuch. Beltz, Weinheim u.a. (2003)
13. Schandry, R.: Lehrbuch der Psychophysiologie. Beltz, Weinheim u.a. (1998)

Usability von Web-Anwendungen - mobile Endgeräte

Phillipp Freiherr von Rotenhan

Otto-Friedrich-Universität Bamberg, Lehrstuhl für Medieninformatik

Zusammenfassung Diese Arbeit stellt eine Übersicht der Anforderungen an die Usability von Web-Anwendungen, für mobile Endgeräte, dar. Hierbei wird speziell auf die Besonderheiten von mobilen Endgeräten, im Zusammenhang mit Usability, auf die verschiedenen Typen von mobilen Web-Anwendungen und Hauptfaktoren für gute Usability einer mobilen Webanwendung eingegangen. In dieser Arbeit wird das Wissen über die Begriffsdefinition der Usability und zugehörige psychologische Aspekte vorausgesetzt.

Schlagworte: UI Pattern, Faktoren mobiler Usability

1 Einführung

1.1 Verbreitung von mobilen Endgeräten

Seit der Vorstellung des ersten iPhones im Jahr 2007 und des ersten iPads im Jahr 2009 hat sich ein großer der Markt für mobile Endgeräte entwickelt. Derartige Geräte waren vorher meist Geschäftsleuten vorbehalten, da für Privatanwender die hohen Geräteanschaffungskosten und die teuren Mobilfunkverträge für mobiles Internet nicht lohnend schienen.
Insgesamt nimmt die Anzahl der Personen, die mobiles Internet auf ihren Handys nutzten von Jahr zu Jahr leicht zu. So nutzen im Jahr 2008 13% der Deutschen[1, S. 4] mit ihrem Handy mobiles Internet, während es zwei Jahre später bereits 17%[2, S. 4] waren. Zudem stieg die Anzahl der Nutzer, die täglich mit dem Handy auf das Internet zugreifen von 22% auf 43%.[2, S. 4] Im Zusammenhang mit diesem Trend steht auch, dass die „Verbreitung von Geräten mit bedienerfreundlichen Benutzeroberflächen und Navigation deutlich zu nimmt“[2, S. 4]. Die Nutzer gewöhnen sich somit an neue Oberflächenkonzepte, erwarten im Gegenzug aber auch, dass die von ihnen genutzten Anwendungen diese neuen Bedienkonzepte auch umsetzten, sonst sehen sie sich meist nach einfacheren, eleganteren oder leichter zu bedienenden Applikationen um, was durch die zentralen Application Stores meist recht einfach ist.
Zu den internetfähiger Handys kam 2010, durch die Vorstellung des ersten iPads und ähnlicher Geräte auf Android-Basis, die Gerätekategorie der Tablets. Der Tabletmarkt entwickelte sich in Deutschland rasant, so wurden 2010 ca. 800.000 Geräte verkauft, während für 2011 2,1 Millionen Verkäufe geschätzt wurden[3].

Durch die erweiterten Möglichkeiten und die zunehmende Verbreitung dieser Geräte entsteht bei den Verbrauchern Bedarf nach Anwendungen, die die Möglichkeiten dieser neuen Gerätekategorien, unter anderem im Bezug auf Mobilität und Interaktion, nutzen. So haben bereits 60% der mobilen Internetnutzer auf ihren Endgeräten Apps[2, S. 4] installiert und auch dieser Tendenz wird mit zunehmender Verbreitung von Smartphones und Tablets weiter steigen, allerdings zeigt eine Untersuchung, dass im Durchschnitt nur etwa 1% der Nutzer eine App auch über langen Zeitraum einsetzten[4, S. 15]. Um Nutzer für einen langen Zeitraum zu behalten muss eine Applikation für den Nutzer einen Mehrwert bieten, sei es durch eine Funktionalität, die so nur von dieser Applikation angeboten wird, durch eine gute Usability, oder idealerweise durch beides.

1.2 Arten mobiler Web-Anwendungen

Mobile Web-Anwendungen lassen sich grundsätzlich in drei unterschiedliche Typen einteilen.

Browserbasierte Anwendungen Dieser Anwendungstyp wird innerhalb des Browsers des jeweiligen Endgerätes ausgeführt, erfordern keine spezielle Installation und unterscheidet sich somit kaum von klassischen Web-Anwendungen, wie sie für PCs entwickelt werden. Oftmals wird lediglich die Oberfläche für mobile Geräte angepasst, so dass sich mit vergleichsweise geringem Aufwand eine breite Masse an mobilen Geräten unabhängig vom jeweiligen Betriebssystem ansprechen lässt. Durch das Auslesen des User-Agent-Headers des Geräts lässt sich eine angepasste Oberflächen beispielsweise für den jeweiligen Gerätetype, Smartphone oder Tablet, ausliefern, um den Nutzwert der Anwendung auf dem jeweiligen Gerätetyp zu maximieren. Mit der Einführung des HTML5 Standards[1] werden die Möglichkeiten für browserbasierte Anwendungen weiter ausgebaut, da dieser zum Beispiel Möglichkeiten zur die Nutzung der aktuellen Geräteposition[2] oder auch zur Speicherung von Daten auf dem Gerät bietet, um so die Nutzung auch ohne Internetverbindung zu ermöglichen[3]. Über das W3C ist außerdem eine Zusammenstellung von „Best Practises“ für mobile Web-Anwendungen verfügbar[5]. Zudem bieten die jeweiligen Plattformen Browser APIs an, um beispielsweise die Browsersteuerung und Adressleiste auszublenden[4].
Allerdings sind die Rechte, die browserbasierte Anwendungen eingeräumt werden von Plattform zu Plattform verscheiden[5].

[1] http://www.w3.org/TR/html5/
[2] http://dev.w3.org/geo/api/spec-source.html
[3] http://www.w3.org/TR/html5/offline.html
[4] http://developer.apple.com/library/ios/#documentation/AppleApplications/Reference/SafariWebContent/ConfiguringWebApplications/ConfiguringWebApplications.html#//apple_ref/doc/uid/TP40002051-CH3-SW3
[5] Siehe Abschnitt 2.4

Vertreter dieses Typs sind beispielsweise die Anwendungen für den Google Calendar[6], Facebook[7] oder der englischsprachigen Financial Times[8].

Native Anwendungen mit Webviews sind Anwendungen, die der Nutzer über den Application Store der jeweiligen Plattform auf dem Gerät installieren kann, die Eigentliche Anwendung wird aber in HTML, JavaScript und ähnlichen Web-Technologien implementiert und in einer WebView dargestellt. Vorteil hierbei ist die Kombination aus den umfassenden Rechten, die eine native Anwendung besitzt, und der meist leichteren Entwicklung mit Web-Technologien. Zudem führen Änderungen des webbasierten Teils nicht zu einem erneuten Prüfung durch die Zulassungesinstanz des jeweiligen Application Stores, was wiederum zu einem schnelleren Roll-Out von Updates führt. Nachteilig ist aber, wie auch bei den browserbasierte Anwendungen, dass bei jedem Start fast die gesamte Benutzeroberfläche über die Datenverbindung geladen werden muss[9]. Auch müssen für Verschiedene Zielsysteme verschiedene Stile der Benutzeroberfläche angeboten werden, wenn sich die der nachgeladene Teil aus der WebView nahtlos in die Oberfläche des Zielsystems integrieren soll.

Native Anwendung mit nachgeladenen Inhalten unterscheiden sich von den vorher genannten nativen Anwendungen mit Webviews in sofern, dass die gesamte Benutzeroberfläche auch mit der Anwendung mit geladen wird und somit nur noch der anzuzeigenden Inhalt über die Datenverbindung geladen wird. Vorteil dieser Lösung ist zum einen, dass durch die Nutzung der nativen Oberflächenelemente der Nutzer auch innerhalb der Anwendung mit den Vertrauten Eingabemöglichkeiten umgeht und somit die Eingewöhnungszeit verkürzt wird, zum anderen, dass die für die Anwendung notwendige Datenmenge nochmal sinkt, was unter anderem zu einem flüssigerem Programmablauf führt[10].
Allerdings sind die Kosten zur Entwicklung von vollständig nativen Anwendungen für mehrere Zielsystem bzw. Plattformen deutlich höher als bei den anderen Anwendungstypen, da mehr geschultes Personal benötigt wird, dass sich mit der jeweiligen Programmiersprache der Zielplattform auskennt.

2 Besonderheiten mobiler Endgeräte

2.1 Interaktion

Die Bedienung von modernen Smartphones und Tablets erfolgt, nicht wie vom PC bekannt per Maus und Tastatur, sondern zum Großteil über einen berührungsempfindlichen Bildschirm, über den die gewünschten Aktionen mit dem Finger

[6] `http://www.calendar.google.com`
[7] `http://facebook.com`
[8] `http://apps.ft.com/`
[9] Siehe auch Abschnitt 2.3
[10] Siehe auch Abschnitt 3.1

ausgewählt wird oder eine entsprechende Geste[6] mit einem oder mehreren Fingern ausgeführt wird. Manche Geräte bieten zusätzlich noch eine vollständig QWERTZ-Tastatur an, dieses Ausstattungsmerkmal findet aber im Vergleich zu einer reinen Touchbedienung keine so starke Verbreitung. RIM mit seinen Blackberry Smartphones, die im Geschäftsumfeld nach wie vor sehr verbreitet sind, ist hier der einzige Hersteller, der bei fast allen Modellen noch auf eine hardwareseitige QWERTZ-Tastatur setzt.

Abbildung 1. Blackberry (links) mit hardwareseitiger und iPhone mit virtueller Tastatur

2.2 Bildschirmauflösung und -orientierung

Ein zentrales Thema der Usability ist die Darstellung von Informationen, dies ist bei mobilen Geräten um so wichtiger, da die verbauten Bildschirme deutlich kleiner als ihre Desktop Pendants sind, aber inzwischen die selbe bzw. teils sogar höhere Auflösungen bieten. Dies führt dazu, dass das eine Unterscheidung, ob ein Zugriff von einem mobilem Gerät oder einem PC erfolgt anhand der Auflösung zwangsläufig fehlschlägt. Hinzu kommt, dass die meisten mobilen Endgeräte inzwischen über die Möglichkeit verfügen Inhalte sowohl im portrait als auch im landscape Format darzustellen, was wiederum neue Anforderungen an die Oberflächengestaltung stellt. Abbildung 2 zeigt die unterschiedlichen Auflösungen bei aktuellen Smartphones, die sehr stark voneinander abweichen und ein statisches Oberflächendesign für alle Geräte nahezu unmöglich machen. Erschwerend kommt hinzu, dass sich die Auflösungen nicht nur innerhalb der Gerätekategorie Smartphone unterscheiden, sondern Im Vergleich von Smartphone zu Tablets die

Auflösungen annähernd identisch bleiben, während die Displaygröße zunimmt. Das Samsung Galaxy Nexus zum Beispiel verfügt über ein Samsung HD Bildschirm mit einer Auflösung von 720 x 1280 Pixel bei einer Displaygröße von 4,65", die Auflösung ist somit annähernd identisch mit der eines 10,1"großen Galaxy Tab 10.1 N, dass 800 x 1280 Pixel aufweist. Um mit der rasanten Entwicklung dieser neuen Gerätekategorien mithalten zu können müssen sich Benutzeroberflächen für Webapplikationen dynamisch auf die Geräteauflösung, Bildschirmorientierung und nicht zuletzt auch auf die zur Verfügung stehende Displaygröße anpassen können, um dem Anwender ein optimales Nutzererlebnis zu ermöglichen und dem Entwickler auf längere Sicht Arbeit zu ersparen.

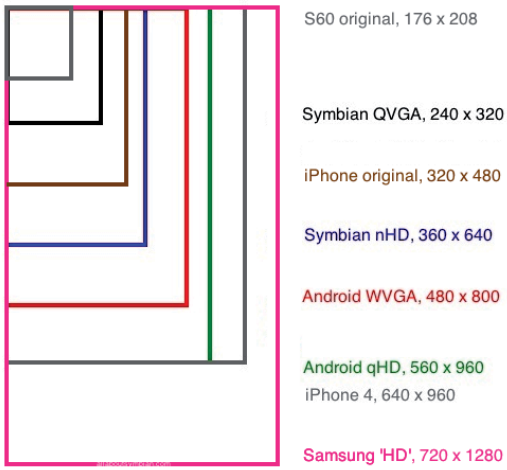


Abbildung 2. Verschieden Displayauflösungen von Smartphones im Vergleich

2.3 Wechselnde Verbindungsgeschwindigkeiten

Eine weitere Besonderheit mobiler Endgeräte sind die sehr unterschiedlichen Datenübertragungsraten, mit denen die Anwender im Alltag zu tun haben. Die Übertragungsraten reichen bei den Verschiedenen Mobilfunkstandards von ca. 220 kbit/s, bei Verwendung von EDGE[7], bis etwa 21 Mbit/s, unter Verwendung von UMTS mit HSPA+[8]. Zusätzlich zu der Verbindung per Mobilfunk haben inzwischen alle modernen Smartphones und Tablets die Möglichkeit sich über WLAN drahtlos mit einem Netzwerk zu verbinden und darüber eine Internetverbindung aufzubauen. Daher ist es bei mobilen Webapplikationen noch wichtiger auf die Menge der an das Endgerät zu übertragenden Daten zu achten und sie so gering wie möglich zu halten, als es bei den nicht mobilen Applikationen schon der Fall ist. Diese Thematik ist, aufgrund der Beeinflussung der Reaktionszeit einer Applikation durch die Verbindungsgeschwindigkeit, für Punkt 3.1 von großer Bedeutung.

2.4 Einschränkungen mobiler Betriebssysteme

Zu guter Letzt unterscheiden sich Post-PC Geräte durch die verwendeten Betriebssysteme von normalen PCs. Dies hat zum einen mit den veränderten Bedienungskonzepten, als auch mit der Optimierung des Betriebssystems für den mobilen Einsatz zu tun, da die meisten Smartphones und Tablets auf eine ARM-Architektur setzten, die der herkömmlichen X86-Architektur von Desktop-PCs nicht kompatibel ist. Die beiden populärsten mobilen Betriebssysteme, Android und iOS, setzten verfolgen hierbei zusätzlich noch sehr unterschiedliche Ansätze. So wird Android von vielen Firmen für ihre mobilen Endgeräte eingesetzt, da die Nutzung von Android für die Hersteller an keine Lizenzabgaben an Google geknüpft ist und die Hersteller die Oberfläche von Android nach ihren Vorstellungen anpassen können, da der Android Quelltext frei verfügbar ist, iOS hingegen wird lediglich von Apple für die eigenen Geräte entwickelt. Die große Anzahl an verschiedenen Android Geräten mit unterschiedlich gestalteten Benutzeroberflächen führt zu einer starken Fragmentierung, da nicht aller Geräte eins Herstellers sofort nach Veröffentlichung einer neuen Android Version ein Update auf selbige erhalten und so eine Großteil der Android Geräte mit älteren Android Versionen betrieben wird. Dies wiederum führt dazu, dass die Android Version, die mindestens für eine Web-Applikation vorausgesetzt wird gut gewählt werden sollte, gerade wenn es sich um einen Anwendungstyp handelt, nativ auf dem Gerät laufen soll. Zur Systemfragmentierung kommt aber auch noch die sehr unterschiedlichen Hardwareeigenschaften der einzelnen Geräte, was Displaygröße, Auflösung und Pixeldichte[11], aber auch die Prozessorgeschwindigkeit anbelangt. Durch seine starke Kopplung von Hard- und Software finden sich keine so starke Fragmentierung bei den iOS-Geräten, allerdings ist Apple bei iOS sehr restriktiv, was beispielsweise den Zulassungsprozess für den App Store anbelangt. Es kann jedoch auch ein Vorteil sein, wenn die Anwendung über eine Application Store vertrieben wird, denn so kann sie von den Nutzer schnell und einfach gefunden werden.
Ein weitere Einschränkung von mobilen Betriebssystemen, im speziellen iOS, sind die teilweise seht beschränkten Zugriffsrechte auf die lokalen Daten des Geräts. So lässt sich beispielsweise auf einem Android-Smartphone über die Web-Anwendung von Facebook ein Photo vom Gerät des Nutzers hochladen, während dies auf einem Gerät mit iOS, auf Grund des restriktiven Rechteverwaltung, nicht möglich ist[12].

3 Hauptfaktoren der mobile Usability

Im folgenden werde die zwei Hauptfaktoren für gute Usability einer mobilen Web-Anwendung näher betrachtet, Probleme aufgezeigt und Möglichkeiten, eine gute Usability zu erreichen, vorgestellt.

[11] Siehe Abschnitt 2.2
[12] Vgl. Abbildung 3

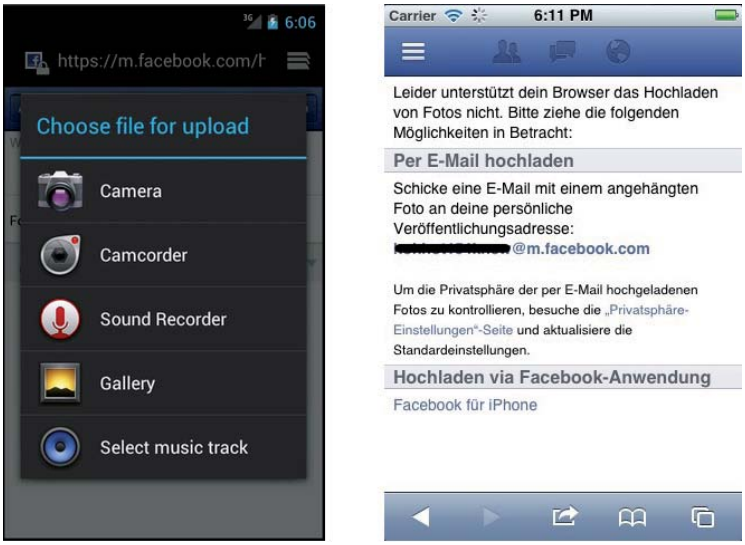


Abbildung 3. Foto-Upload der Facebook Web-App unter Android (links) und iOS

3.1 Flüssiger Programmablauf

Reaktionszeit bezeichnet die Zeit, die eine Anwendung benötigt, um auf eine Eingabe des Nutzers zu reagieren. Eine kurze Reaktionszeit ist wichtig um dem Nutzer ein Gefühl der direkten Interaktion zu vermitteln und den Nutzungsfluss nicht zu unterbrechen. Reaktionszeiten im Bereich von 0,1 Sekunden werden als unmittelbar wahrgenommen und Zeiten bis 3 Sekunden vom Nutzer toleriert, da sie den Gedankenfluss nicht unterbrechen[9, S. 225]. Zeiten im höheren einstelligen Bereich werden von den Nutzern dann noch in Kauf genommen, wenn für den Nutzer ersichtlich ist, dass sich das Warten lohnt. Generell gilt für diese längeren Wartezeiten aber, dass der Gedankenfluss des Nutzers unterbrochen wird und er damit beginnen wird sich mit anderen Dingen zu befassen[9, S. 225]. Reaktionszeiten von mehr als 3 Sekunden sind aber beispielsweise aufgrund der schwankenden Verbindungsgeschwindigkeiten[13] nicht außergewöhnlich, die Nutzer sind auch durchaus bereit bei webbasierten Anwendungen etwas geduldiger zu sein, aber das meist nur dann, wenn sie über den aktuellen Zustand des Systems, etwa den Grund und den Fortschritt des Ladevorgangs, informiert werden.

Interaktionseffizienz bezeichnet die Schritte, die ein Nutzer durchführt um eine bestimmte Aufgabe mit Hilfe der Anwendung zu erledigen. Die Hauptaufgaben einer Anwendung sollten möglichst einfach zu erreichen und schnell durchzuführen sein. Um eine gute Interaktionseffizienz zu erreichen sollte unter anderem die Größe von Oberflächenelementen im Verhältnis zur Displaygröße

[13] Siehe Abschnitt 2.3

und das Verdecken durch die virtuelle Tastatur berücksichtigt werden.
In Abbildung 4 ist erkennbar, dass durch die Tastatur der „Sign-In“-Knopf verdeckt wird, wodurch den Nutzer nach der Eingabe seiner Nutzerdaten erst die Tastatur ausblenden muss bevor er den Login-Vorgang vorsetzten kann. Ein wichtiger Punkt, der die Interaktionseffizienz von mobilen Anwendungen wesentlich beeinflusst ist die andere Umgebung in der sich der Anwender bei der Nutzung der Anwendung befindet. Nutzer von PCs verweilen meist während der gesamten Nutzungsdauer an einem Ort und sind somit lediglich den dortigen störenden Einflüssen ausgesetzt, wohingegen Smartphones häufig im Freien[10] genutzt werden. Der Zeitraum in dem der Nutzer mit einer mobilen Anwendung interagiert ist folglich kürzer als bei klassischen Anwendungen.
Diese Aspekte führen dazu, dass eine mobile Web-Anwendung nicht alle Funktionalitäten einer klassischen Web-Anwendung bieten muss, sondern sich auf die Funktionen beschränken sollte, die für den Nutzer im mobilen Kontext von größtem Nutzen sind und gegebenenfalls neue Funktionalitäten anbieten sollte, die gerade im mobilen Nutzungskontext sinnvoll sind. Ein Beispiel hierfür wäre etwa die automatische Erfassung des aktuellen Standorts in der Anwendung einer Bank, um den Nutzer auf die nächstgelegenen Geldautomaten hinweisen zu können. Zusätzlich zu dem bereits genannten kann die Interaktionseffizienz auch durch Beachtung der in Abschnitt 3.2 aufgeführten Punkte gesteigert werden.

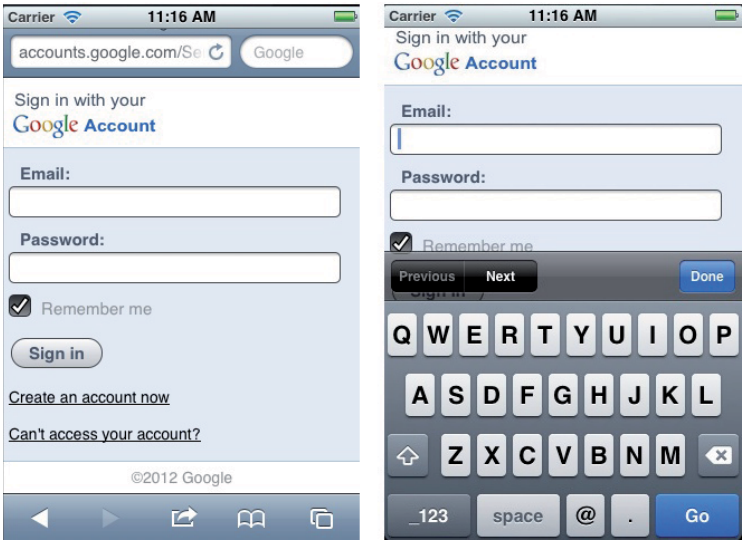


Abbildung 4. Bad Practise: Die eingeblendete Tastatur des Smartphones überdeckt den „Sign-In“-Knopf

3.2 Optimierte Oberflächengestaltung

Textgestaltung Da der Informationsaustausch zwischen Anwendung und Nutzer in den meisten Fällen in schriftlicher Form geschieht ist eine saubere und sinnvolle Gestaltung der in der Anwendung enthaltenen Texte sehr wichtig. Speziell für browserbasierte Web-Anwendungen, die für eine Vielzahl von Endgeräten ausgelegt sein müssen ist es wichtig keine fixen Textlayouts zu verwenden, sondern den Text dynamisch umlaufen zu lassen. Dies hat zwei Vorteile:

1. Die Bildschirmorientierung ist unerheblich, da immer die gesamte Bildschirmbreite ausgenutzt wird.
2. Man kann dem Nutzer die Möglichkeit zur Anpassung der Schriftgröße geben, was gerade bei textlastigen Anwendungen, wie die von Nachrichtenportalen, die Nutzerfreundlichkeit erhöht.

Generell sollten die Texte in der Anwendung kurz, aussagekräftig und einfach gehalten sein, da so wichtige Informationen leichter ins Auge fallen und es für den Nutzer leichter ist die zu angezeigten Informationen zu erfassen und zu verstehen[11, S. 100f].

Seitenstruktur Bei der Seitenstruktur geht es vor allem um die Frage, wie die Informationen für den Nutzer gegliedert werden sollen, damit er das Gesuchte möglichst einfach findet. Die hierfür relevanten psychologischen Gesetzt für die Gestaltung von Nutzeroberflächen werden an dieser Stelle vorausgesetzt. Zusätzlich dazu gilt es hierbei aber auch zu beachten, dass der Nutzer möglichst nur in eine Richtung, also vertikal oder horizontal, scrollen sollte, da das Scrollen selbst sonst eine zu große Präzision erfordert, die etwa bei der Nutzung in öffentlichen Verkehrsmitteln nur schwer zu erreichen ist. Gerade für das Problem der Strukturierung und Layouts von browersbasierten Web-Anwendungen in Verbindung mit mehreren unterschiedlichen Endgeräten wurden inzwischen einige Pattern entworfen[12].
Das „Mostly Fluid"-Pattern[14] zeichnet sich durch die Nutzung breiterer Ränder auf großen Bildschirmen aus und skaliert durch die Nutzung von flexiblen Gittern auch gut auf kleine Gerätebildschirme, wie Smartphones. Das „Layout Shifting"-Pattern[15] dagegen ist deutlich aufwändiger in der Implementierung, da das Layout hier spezieller auf die Bildschirmgrößen angepasst wird, was zwar mit größerem Aufwand verbunden ist, aber gegenüber dem „Mostly Fluid"-Pattern den Vorteil bietet, dass die vorhandene Anzeigefläche optimaler genutzt werden kann. Bei der Verwendung diese Patterns gibt es jedoch viel Spielraum, wann Ränder eingeblendet und wann die einzelnen Elemente übereinander angeordnet werden sollen und . Bei der Nutzung solcher Patterns ist aber trotzdem immer auf die Optimierung der Oberflächenelemente für Touchgeräte, etwa durch angepasste CSS-Styles, zu achten.

[14] Siehe Abbildung 5
[15] Siehe Abbildung 5

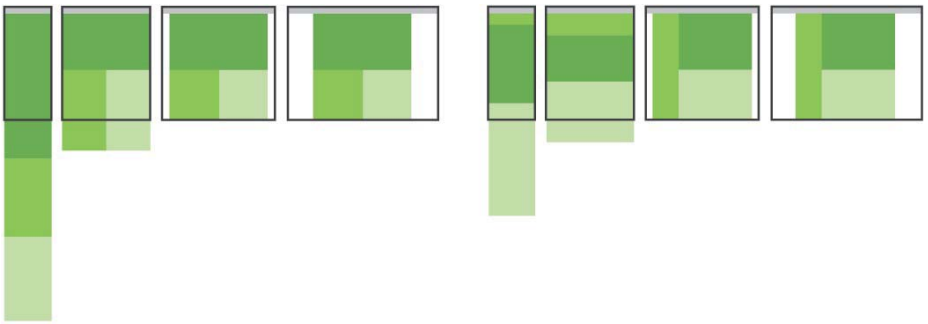


Abbildung 5. Layout-Pattern für mehrere Geräte: „Mostly Fluid“ (links) und „Layout Shifting“[12]. Die Farbsättigung kann direkt auf die Priorität der Informationen für den Nutzer umgelegt werden.

Navigationsstruktur Um dem Anwendern die Nutzung so einfach wie möglich zu gestalten ist es sinnvoll sich bei der Strukturierung der eigenen Anwendung an den jeweiligen Design-Richtlinien der Betriebssysteme[13][14] zu orientieren. Auch ist es sinnvoll, wenn bereits eine klassische Web-Anwendung für PCs vorhanden ist, bei den verwendeten Farben, Kategorie- und Funktionsbezeichnungen, soweit möglich und sinnvoll, konsistent zu bleiben. Dies führt dazu, dass Anwender, die bereits mit der PC-Version vertraut sind, schnell bekannten Funktion wieder finden und somit die bereits vorhandene emotionale Bindung aufrecht erhalten wird. Bei der Benennung von Buttons und Funktionen sollte aber auch beachtet werden, dass Anwendungsfunktionen, die gleiche Bezeichnungen, wie systemeigene Funktionen tragen, auch den gleichen Zweck dienen sollten. Sind die Einstellungen von iOS im Englischen unter „Settings“ zu finden sollte auch in der eigenen Anwendung „Settings“ den Vorzug vor „Preferences“ genutzt werden. Der Effekt der Wiedererkennung kann und sollte auch bei Icons benutzt werden. Die Displays von modernen Smartphones sind, bis auf einige Ausreißer, recht klein, wodurch sehr genau überlegt und getestet werden, welche Funktionen direkt aus der Hauptansicht der Anwendung erreichbar sein sollen und für welche es ausreichend ist sie durch sekundäre Ansichten oder Untermenüs zu erreichen. Android bietet hier mit der „Menü“-Taste eine Möglichkeit dem Nutzer in nativen Anwendungen kontextspezifische Aktionen zu ermöglichen, ohne sie direkt in die Hauptansicht platzieren zu müssen.
Auch im im Bereich der Navigation haben sich inzwischen einige Pattern herausgebildet, die teilweise in allen drei Anwendungstypen genutzt werden können[15]. Das „Off the canvas“-Pattern[12][16] wurde von Facebook als erstes für eine breiten Nutzermasse eingesetzt und wird inzwischen sowohl für die browserbasierte Anwendung, für die nativen iPhone und iPad Apps, als auch unter Android genutzt. Die Vorteile dieses Patterns liegen zum einen in der Flexibilität, mit der sowohl Smartphones, als auch Tablets angesprochen werden können, denn auf

[16] Bei Frost[15] auch als „left nav flyout“ bezeichnet.

einem querformatig gehaltenem Tablet kann ohne Probleme am linken oder rechten Bildschirmrand ein zusätzlicher Screen eingeblendet werden, der vorher nicht sichtbar war. Zum anderen kann man durch die Vorreiterrolle von Facebook, die dieses Pattern schon länger einsetzten, bei vielen Nutzern mobiler Endgeräte davon ausgehen, dass sie mit der Nutzung dieses Patterns zur Navigation vertraut sind.

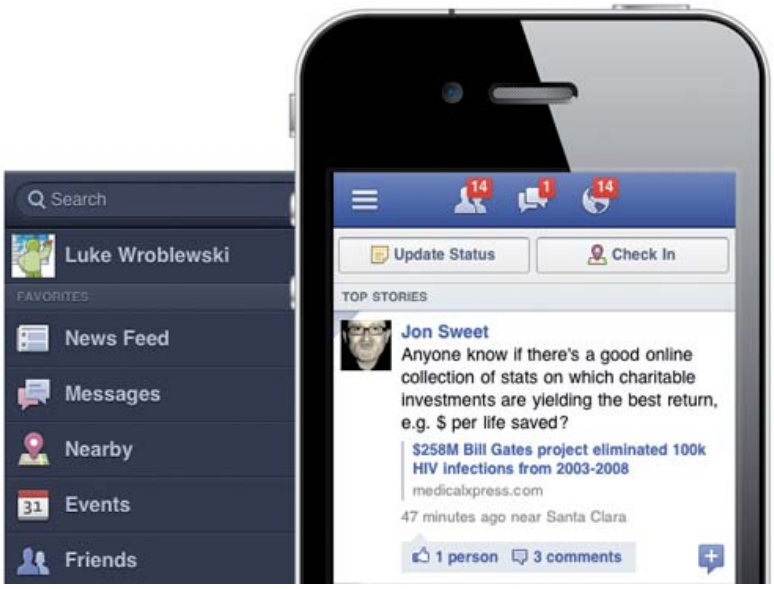


Abbildung 6. Beim „Off the canvas"-Pattern verbirgt sich die Navigation außerhalb des gerade angezeigten Screens und schieben sich per Knopfdruck in das Blickfeld des Anwenders[15]

4 Fazit

In den letzten Jahren hat sich die Usability von mobilen Anwendungen sehr weiterentwickelt, was vor allem neuen Interaktionskonzepten und Gerätekategorien zu verdanken ist. Dies ist in sofern wichtig, da die gute Usability einer Anwendungen zu einer insgesamt guten User Experience beiträgt, welche wiederum den Nutzer dazu verleitet die Anwendung gerne weiter zu nutzten und weiterzuempfehlen.
Die Nutzung von dem Anwender bekannten Patterns und Plattformstandards erleichtert es Entwicklern inzwischen, deutlich eine Anwendung mit guter Usability für mobile Endgeräte zu verwirklichen. Die Wahl des Anwendungstyps ist eine der wichtigsten Entscheidungen, die im Vorfeld des Entwicklungsstarts getroffen

wird. So fühlen sich native Anwendungen für den Nutzer meist flüssiger und insgesamt schneller an, sind aber auf das jeweilige Betriebssystem beschränkt. Auch ist es teilweise nicht einfach die Anwendung für die Installation auf der jeweiligen Plattform verfügbar zu machen. Für die Bereitstellung von Anwendungen über den App Store von Apple ist beispielsweise eine vorhergehende genaue Prüfung der Anwendung von Apple notwendig, was etwa das Bereitstellen von sicherheitskritischen Fehlerbehebungen verzögert.
Browserbasierte Web-Anwendungen hingegen lassen sich, eine entsprechende Popularität des Anbieters vorausgesetzt, einfach für viele Nutzer zugänglich machen und leicht aktualisieren. Von den möglichen Funktionen, die das Endgerät des Nutzers bietet kann von diesen Anwendungen aber nur ein kleiner Teil angesprochen werden, da Webtechnologien als möglichst breiter und geräteunspezifischer Standard konzipiert sind. Allerdings zeigt eine Anwenderstudie[16], dass die Nutzer bei nativen Anwendungen häufiger zum gewünschten Ziel kommen.
Alles in allem ist es immer sinnvoll langfristig in die Usability der eigenen Anwendung zu investieren und immer wieder zu überarbeiten, denn nur zufriedene Nutzer sind bereit für eine Anwendung zu bezahlen und sie weiterzuempfehlen, allerdings geht Usability mit anderen wichtigen Gesichtspunkten, wie etwa Performanz oder Sicherheit, Hand in Hand.

Literatur

1. Accenture (Kronberg): Mobile Web Watch 2008[17] (2008)
2. Accenture (Kronberg): Mobile Web Watch 2010[18] (2010)
3. BITKOM: Tablet Computer erobern den Massenmarkt[19] (2011)
4. Pinch Media: iPhone App Store Secrets[20] (2009)
5. Connors, A., Sullivan, B.: Mobile Web Application Best Practices[21] (14.12.2010)
6. Google Inc.: Android Design: Gestures[22] (2012)
7. Wikipedia: Enhanced Data Rates for GSM Evolution[23] (22.02.2012)
8. Wikipedia: Universal Mobile Telecommunications System[24] (28.02.2012)
9. Kappel, G., Pröll, B., Reich, S., Retschitzegger, W.: Web engineering: The discipline of systematic development of web applications. John Wiley & Sons, Hoboken and NJ (2006)
10. Johnson, D.: Where Are Smartphones Being Used?[25] (2011)
11. Colborne, G.: Simple and usable: Web, mobile, and interaction design. New Riders and Pearson Education, Berkeley and Calif and London (2011)

[17] http://www.accenture.com/de-de/company/newsroom-germany/pages/mobile-web-nutze-studie.aspx
[18] http://www.accenture.com/de-de/pages/insight-mobile-web-2010-summary.aspx
[19] http://www.bitkom.org/de/presse/64050_70631.aspx
[20] http://www.slideshare.net/pinchmedia/iphone-appstore-secrets-pinch-media
[21] http://www.w3.org/tr/2010/rec-mwabp-20101214/
[22] http://developer.android.com/design/patterns/gestures.html
[23] http://de.wikipedia.org/wiki/enhanced_data_rates_for_gsm_evolution
[24] http://de.wikipedia.org/wiki/universal_mobile_telecommunications_system
[25] http://www.tatango.com/blog/where-are-smartphones-being-used-infographic/

12. Wroblewski, L.: Multi-Device Layout Patterns[26] (14.03.2012)
13. Apple Inc. (Cupertino): iOS Human Interface Guidelines[27] (2012)
14. Google Inc.: Android Design[28] (2012)
15. Frost, B.: Responsive Navigation Patterns[29] (24.02.2012)
16. Nielsen, J.: Mobile Usability Update[30] (26.9.2011)

[26] http://www.lukew.com/ff/entry.asp?1514
[27] https://developer.apple.com/library/ios/#documentation/userexperience/conceptual/mobilehig/introduction/introduction.html#//apple_ref/doc/uid/tp40006556
[28] http://developer.android.com/design/index.html
[29] http://bradfrostweb.com/blog/web/responsive-nav-patterns/
[30] http://www.useit.com/alertbox/mobile-usability.html