

UNIVERSITÄT LEIPZIG

Fakultät für Mathematik und Informatik

Institut für Informatik

Entwicklung semantischer Webapplikationen auf Basis des agilen Vorgehensmodells Scrum

DIPLOMARBEIT

Leipzig, April 2008

Begutachter: Prof.Dr. Fähnrich, Dr. Auer

vorgelegt von

Aron Schneider

Matrikel: 8977198

geb. am: 24.07.1981

Studiengang Informatik

Zusammenfassung

Das Semantic Web stellt den nächsten Schritt zur Informationsgesellschaft dar. Durch das Hinzufügen von Semantik zum aktuellen Web, entstehen neue Optionen im Hinblick auf den Umgang mit Informationen und Daten. Um einer möglichen Web-Krise vorzubeugen, bietet es sich an auf die Erfahrung des Software-Engineering zuzugreifen und Analogien zu ziehen. Daraus entsteht das Web-Engineering. Es beinhaltet unter anderem die Verwendung eines Vorgehensmodells zur Entwicklung einer Webanwendung. In dieser Diplomarbeit soll das agile Vorgehensmodell Scrum vergleichend betrachtet und bewertet werden, im besonderen Hinblick auf seine Eignung zur Entwicklung einer Semantic Web Applikation.

Stichworte: Semantic Web, Scrum, agiles Vorgehensmodell, Entwicklung einer Web Applikation, Booking Application

Danksagung

Zu Beginn möchte ich mich bei Dr. Sören Auer bedanken, der mich im Projekt Booking Application aufgenommen hat und mir dann später immer wieder mit Rat zur Seite stand, als es um die Ausarbeitung dieser Diplomarbeit ging.

Mein nächster Dank geht an das Projektteam, das die Booking Application entwickelt hat, bei der ich mithelfen durfte. Beteiligt waren Jens Lehmann als Leiter, Michael Martin als Chefprogrammierer, Tom Weiland, Ruslan Masold und Martin Weise als Programmierer.

Bei der Erstellung des \LaTeX -Dokuments wurden mir durch die Hilfe von Thomas Kaminski viele Stunden Arbeit erspart.

Unverzichtbar waren auch die Lektoren, allen voran Michael Keutel, durch den ich den erweiterten Infinitiv mit zu, wieder zu schätzen weiß.

Mein größter Dank geht allerdings an meine bessere Hälfte Cindy. Sie hat sich während dieser Zeit immer wieder als mein hilfreiches schlechtes Gewissen entpuppt und hat mich, wo sie nur konnte, entlastet und unterstützt. Dafür möchte ich mich besonders bedanken.

Abkürzungsverzeichnis

- AJAX** Asynchronous JavaScript and XML
- API** Application Programming Interface
- ASP** Active Server Pages
- DAWG** RDF Data Access Working Group
- DBMS** Datenbankmanagementsystem
- DOAP** Description of a Project
- FOAF** Friend of a Friend
- GUI** Graphical User Interface
- HTTP** Hypertext Transfer Protocol
- IRI** Internationalized Resource Identifier
- JS** JavaScript
- JSON** JavaScript Object Notation
- JSP** JavaServer Pages
- MVC2** Model View Controller 2
- N3** Notation 3
- OWL** Web Ontology Language
- PHP** PHP: Hypertext Preprocessor
- RDF** Resource Description Framework
- RDFS** Resource Description Framework Schema
- REST** Representational State Transfer
- RSS** RDF Site Summary
- RUP** Rational Unified Process
- SIOC** Semantically-Interlinked Online Communities
- SPARQL** SPARQL Protocol and RDF Query Language
- SQL** Structured Query Language
- SVG** Scalable Vector Graphics

SVN Subversion
SWSE Semantic Web Search Engine
UML Unified Modeling Language
URI Uniform Resource Identifier
URL Uniform Resource Locator
UWE UML-based Web Engineering
WebML Web Modeling Language
WWW World Wide Web
W3C World Wide Web Consortium
XML Extensible Markup Language
XP eXtreme-Programming
YUI Yahoo! User Interface Library

Abbildungsverzeichnis

2.1	Schichten des Semantic Web	7
3.1	Hauptphasen des Wasserfallmodell	14
3.2	Hauptphasen des V-Modell	17
3.3	Phasen und Disziplinen des Rational Unified Process	19
3.4	Ablauf einer XP Iteration	23
4.1	Ablaufzyklus von Scrum	28
4.2	Beispiel einer Aufgabentafel	32
4.3	Burndown Diagramm	33
5.1	Aufbau der Oberfläche der Booking Application	38
5.2	Ablauf einer Buchung	41
5.3	Ablauf einer Änderung der Emailadresse des Kunden	42
5.4	Nachbarschaftsgraph der Ontologie der Booking Application	43
5.5	Vererbungsgraphsgraph der Ontologie der Booking Application	44

Tabellenverzeichnis

2.1	Übersicht verbreiteter Semantic Web Vokabulare	10
3.1	Gegenüberstellung der verschiedenen Arten von Vorgehensmodellen	25
4.1	Auszug aus dem Product Backlog der Booking Application	30
4.2	Sprint Backlog aus der Booking Application	31
6.1	Evaluation der Booking Application	59

Inhaltsverzeichnis

Abbildungsverzeichnis	v
Tabellenverzeichnis	vi
1 Einführung	1
1.1 Motivation	1
1.2 Thematische Abgrenzung und Ziel dieser Arbeit	2
1.3 Gliederung	3
2 Semantic Web und Semantic Web Applikationen	4
2.1 Idee des Semantic Web	4
2.2 Technologien für Semantic Web Applikationen	6
2.2.1 Wissensrepräsentationsstandards	8
2.2.2 Semantic Web Vokabulare	9
2.2.3 Datenspeicherung und Zugriff	9
2.3 Anwendungsbeispiele für Semantic Web Applikationen	11
3 Vorgehensmodelle	13
3.1 Klassische Vorgehensmodelle	14
3.1.1 Wasserfallmodell	14
3.1.2 Spiralmodell	16
3.2 Moderne Vorgehensmodelle	16
3.2.1 V-Modell	17
3.2.2 Rational Unified Process	17
3.3 Agile Vorgehensmodelle	20
3.3.1 eXtreme-Programming	21
3.3.2 Scrum	23

3.4	Gegenüberstellung der Vorgehensmodelle	25
4	Anwendung von Scrum zur Entwicklung einer Semantic Web Applikation	27
4.1	Scrum Ablauf bei einer Semantic Web Applikation	27
4.1.1	Vision	28
4.1.2	Rollen	28
4.1.3	Product Backlog	29
4.1.4	Sprint und Sprint Backlog	30
4.1.5	Evolutionäre Umsetzung	33
4.2	Auswahl und Integration von Scrum	34
4.3	Bewertung von Scrum zur Entwicklung einer Semantic Web Applikation . . .	35
5	Anwendungsfall Booking Application	37
5.1	Beschreibung der Booking Application	37
5.2	Anforderungen an die Booking Application	39
5.2.1	Anwendungsfälle innerhalb der Booking Application	41
5.3	Ontologie der Booking Application	43
5.3.1	Werkzeuge zur Modellierung einer Ontologie	46
5.3.2	Verifikation einer Ontologie	46
5.4	Eingesetzte Techniken und Technologien	47
5.4.1	Anfragemöglichkeiten	48
5.4.2	Zugriffskontrolle	50
5.5	Scrum Einsatz zur Entwicklung der Booking Application	51
6	Evaluation	53
6.1	Vorstellung der Evaluationskriterien	53
6.2	Bewertung der Semantic Web Applikation	55
6.3	Evaluation von Scrum zur Entwicklung einer Semantic Web Applikation . . .	56
6.4	Probleme beim Projekt Booking Application	57
7	Diskussion	60
7.1	Zusammenfassung	60
7.2	Ausblick	61
	Literaturverzeichnis	63

1 Einführung

Im ersten Abschnitt (vgl. Abschnitt 1.1, S. 1) wird die Motivation für diese Arbeit geliefert. Danach soll die Diplomarbeit thematisch eingrenzt werden (vgl. Abschnitt 1.2, S. 2). Abschließend folgt eine Gliederung, welche den Inhalt der einzelnen Kapitel kurz zusammenfasst (vgl. Abschnitt 1.3, S. 3).

1.1 Motivation

Das Internet ist die größte frei verfügbare und frei zugängliche Informationsquelle der menschlichen Spezies. Im Laufe der Jahre hat es sich stets weiter entwickelt. Anfang der neunziger Jahre war es üblich, dass einige wenige Menschen Informationen publizierten und diese einer breiten Masse zugänglich machten. Im Zuge der Web 2.0 Entwicklung änderte sich diese Form rasch. Denn inzwischen hat sich die Technik soweit fortentwickelt, dass eine Internetpublikation den Nutzer nicht mehr vor technische Herausforderungen stellt, wie 10 Jahre zuvor [23]. Durch technologische Neuerungen wie Wikis, Blogs und Content-Management-Systeme kann inzwischen jeder Internetnutzer¹ selbst Informationen veröffentlichen. Dieser Trend der Selbstverantwortung (bezüglich der Option Informationen und damit auch eine eigene Sichtweise zu publizieren) wurde ein großer Erfolg. Unzählige Communities basieren auf dem Prinzip der Selbstverantwortung [27]. Durch den rapiden Zuwachs an Autoren nimmt auch die Anzahl an verfügbaren Informationen im Internet zu [8]. Diese Informationen müssen navigierbar sein. Die Navigation durch Informationen stellt eine große technische Herausforderung dar. Durch den Umfang an Autoren kommt es vermehrt zu Missverständnissen oder Fehlinformationen, wie beispielsweise durch Spambots verbreitete Werbung. Die Suche nach relevanten, korrekten, sowie vollständigen Dokumenten kann durch die bloße Menge existierender Informationen nicht von menschlicher Hand geschehen, sondern muss maschinell passieren.

¹Unter der Voraussetzung, dass er über einen Internetzugang verfügt.

Eine maschinelle Suche benötigt Metainformationen, also Informationen über Informationen, um den vorgegebenen Suchkriterien zu entsprechen. Aktuell eingesetzte Suchalgorithmen bewerten meist nur die direkt sichtbaren Informationen. Diese sind nicht immer valide. Zur Auswertung, ob es sich bei *Bank* um ein Gebäude, eine Person, die diesen Nachnamen trägt oder ein Möbelstück handelt, werden Metainformationen benötigt. Sie helfen unter anderem bei der thematischen Klassifikation eines Dokuments. Darüber hinaus ermöglichen Metainformationen automatisches logisches Schlussfolgern. So wird die Maschinenlesbarkeit maßgeblich verbessert.

An dieser Stelle kommt das Semantic Web zum Einsatz. Eine Semantic Web Anwendung operiert mit Metainformationen, aber dazu muss sie erst einmal entwickelt werden. Je umfangreicher diese Entwicklung wird, desto besser muss ihre Planung sein. Wenn bei der Entwicklung einer Anwendung die Programmierung ohne Abschätzung der Problemstellen oder Definition von Kernkomponenten, Kernfunktionen und Schnittstellen begonnen wird, dann treten über kurz oder lang Probleme auf. Um den Entwicklungsprozess kontrollierbar zu gestalten, bietet sich ein geregeltes Vorgehen an. Softwaretechnisch wird dies als Vorgehensmodell bezeichnet. Es regelt alle Phasen der Softwareentwicklung. Vorgehensmodelle können je nach Projekttyp und Einsatzfeld unterschiedlich aussehen. So benötigt die Entwicklung eines Betriebssystems wie Windows Vista tausende von Programmierern, die über Jahre hinweg an einem Softwareprojekt arbeiten [13], während ein minimales elektronisches Einkaufssystem von einem kleinen Team binnen weniger Wochen entwickelt werden kann. Die Entwicklung von Vista benötigt eine umfangreiche Dokumentation, da auf einem anderen Wege eine Kommunikation unpraktikabel ist. Die Entwicklung des elektronischen Einkaufssystems funktioniert auch ohne vollständige Dokumentation, da sich die Teammitglieder direkt miteinander verständigen und so Probleme beseitigen können. Je stärker das Vorgehensmodell dem Softwareentwicklungsprozess angepasst ist, desto effektiver und wirtschaftlicher ist es.

1.2 Thematische Abgrenzung und Ziel dieser Arbeit

In dieser Diplomarbeit soll zu Beginn das Semantic Web und seine Daseinsberechtigung, sowie die zugrundeliegenden Techniken und Technologien erklärt werden. Anschließend soll ausführlich auf Vorgehensmodelle eingegangen werden. Dabei sollen nur die wichtigsten Vertreter betrachtet werden. In dieser Arbeit sollen keine Aspekte des Web-Projektmanagements

betrachtet werden, da nur die eigentliche Entwicklung bewertet werden soll. Nachdem ein Überblick über existierende Vorgehensmodelle erzeugt wurde, soll ein agiles Vorgehensmodell ausgewählt und hinsichtlich seines Einsatzfeldes, der Entwicklung einer Semantic Web Applikation, genauer beleuchtet werden. Hierbei wird die Wirtschaftlichkeit und Praktikabilität der Vorgehensweise vergleichend betrachtet und kritisch bewertet. Damit soll der Einführungs- und Theorieteil der Arbeit beendet sein.

Der Praxisteil der Arbeit beginnt mit einer umfangreichen Darstellung der Booking Application, als Beispiel einer Semantic Web Anwendung. Daran soll der Einsatz von Scrum aufgezeigt und die Motivation angeregt werden, weshalb dieses Vorgehensmodell besonders für die Entwicklung einer Semantic Web Anwendung geeignet ist. In der Evaluation soll der Einsatz von Scrum und die Qualität der geleisteten Arbeit bewertet werden. In dieses Kapitel sollen keine Performancetests einfließen, aufgrund des Prototypenstatus der zu bewertenden Anwendung.

In der abschließenden Konklusion sollen erreichte Ziele, sowie noch existierende Probleme aufgezeigt und zukünftige Schritte zur Optimierung der Herangehensweise geschlussfolgert werden.

1.3 Gliederung

Diese Diplomarbeit ist in sieben Kapitel unterteilt, wobei das Erste die Motivation und thematische Abgrenzung der Arbeit behandelt (vgl. Kapitel 1, S. 1). Im zweiten Kapitel, S. 4 wird das Semantic Web und seine Grundlagen erklärt. Das dritte Kapitel, S. 13 befasst sich mit Vorgehensmodellen, wobei diese kurz klassifiziert und abschließend verglichen werden. Das agile Vorgehensmodell Scrum tritt aus diesem Vergleich als geeignet hervor, zur Entwicklung einer Semantic Web Applikation. Aus diesem Grund wird Scrum im vierten Kapitel, S. 27, im Hinblick auf die Entwicklung von Semantic Web Applikationen, genau beleuchtet. Kapitel fünf, S. 37 beschreibt das Anwendungsbeispiel Booking Application. Dabei handelt es sich um eine Semantic Web Anwendung, die Mithilfe von Scrum entwickelt wurde. Diese Entwicklung wird in Kapitel sechs, S. 53 evaluiert. Kapitel sieben, S. 60 offeriert, als Abschluss dieser Arbeit, ein Fazit bezüglich erreichter Ziele und einen Ausblick hinsichtlich diverser Verbesserungsmöglichkeiten.

2 Semantic Web und Semantic Web Applikationen

Um über Vorgehensmodelle im Zusammenhang mit Semantic Web Applikationen sprechen zu können, ist es unabdingbar eingehend zu klären, was das Semantic Web und was eine Semantic Web Applikation ist. Um Definitionen für diese beiden Begriffe festzulegen, wird in diesem Kapitel als erstes die grundsätzliche Idee des Semantic Web betrachtet (vgl. Abschnitt 2.1, S. 4). Ausgehend von dieser Idee lässt sich der Begriff Semantic Web definieren. Anschließend werden Techniken und Technologien des Semantic Web und seiner Anwendungen betrachtet (vgl. Abschnitt 2.2, S. 6). Dieses Kapitel wird mit einigen Anwendungsbeispielen abgeschlossen (vgl. Abschnitt 2.3, S. 11).

2.1 Idee des Semantic Web

Das **World Wide Web** (WWW) ist binnen weniger Jahre zur größten, verfügbaren Informationsquelle gewachsen. Inzwischen ist es sowohl aus dem privaten Leben, als auch aus der Wirtschaft nicht mehr wegzudenken. Der Erfolg des WWW liegt begründet in seiner Informationsvielfalt, der Verfügbarkeit, der Zugriffsmöglichkeiten und der Aktualität dieser Informationen [18].

Die Organisation dieser Datenmenge ist eine große technische Herausforderung. Es müssen dabei Fragen gestellt werden wie:

- Wie wird die gesuchte Information gefunden?
- Ist die gesuchte Information korrekt, vollständig und aktuell?

- Wie sind Redundanzen und etwaige Abweichungen bei der Auswertung der Suchanfrage zu bewerten?

Durch den riesigen Umfang an Internetdokumenten ist es für einen Menschen unmöglich, diese Dokumente nach einer konkreten Information zu durchsuchen. Diese Aufgabe übernehmen Suchprogramme, im folgenden auch *Agenten* genannt. Diese funktionieren bisher ausreichend gut. Sie werden realisiert durch statistische Algorithmen, die das Auftreten und das Zusammenspiel der vorkommenden Wortgruppen im Suchdokument bewerten. In der (unzureichenden) Genauigkeit und der (Un-) Vollständigkeit der Ergebnisse von Suchanfragen ist das Problem begründet, dass immer mehr an Bedeutung gewinnt [42].

Informationen müssen nicht eindeutig sein, beziehungsweise können sie missverstanden werden [15]. Beispielsweise können aufgrund von Ambivalenzen Fehltreffer bei einer Suchanfrage ermittelt werden. Wenn ein Agent nach *Paris* sucht, dann enthält die Rückgabemenge nicht nur Dokumente über die französische Hauptstadt, sondern auch viele über die gleichnamige Hotelerbin. Solche Probleme werden auch durch orthografische Fehler und durch Synonyme verursacht. Ein *Pariser* kann ein Stadtbewohner sein, aber auch ein Kondom, oder eine spanische Person mit dem Namen *Pari* die fälschlicherweise mit dem Verb *ser* zusammengeschieden wird. Ein Agent stößt, in solchen Fällen, auf seine Grenzen.¹ Die eingesetzten Algorithmen lassen sich zwar in gewisser Weise noch syntaktisch optimieren, aber qualitativ deutlich bessere Ergebnisse sind mit dieser Herangehensweise nicht zu erwarten. Dieses Problem soll durch das Semantic Web behoben werden [39].

Es bieten sich zwei Lösungen für dieses Problem an. Die erste Lösung ist, dass ein Agent zum Einsatz kommt, der über fortschrittliche Methoden der künstlichen Intelligenz verfügt. Dieser Agent übernimmt die kognitiven Funktionen eines Menschen, womit er implizites Wissen folgert. Hierbei sei anzumerken, dass es zum heutigen Zeitpunkt (25.03.2008) noch keinen Agenten dieser Art gibt, der auch zufriedenstellende Genauigkeit, einhergehend mit akzeptabler Effizienz, zur Verfügung stellt.

Die zweite Lösung dieses Problems liegt in der Umgestaltung der zugrundeliegenden Daten. Diese müssen für den Agenten semantisch interpretierbar sein, so dass er selbstständig

¹Unter der Voraussetzung, dass der Agent *Heterogenität* (also unterschiedliche Dateiformate oder ein andere Wissensrepräsentationsvariante) der zu durchsuchenden Daten bewältigen kann. Da andernfalls bereits vorher Probleme auftreten.

schlussfolgern kann, wenn $\langle LAND \rangle \textit{Frankreich} \langle /LAND \rangle$, $\langle STADT \rangle \textit{Paris} \langle /STADT \rangle$ in einem Satz vorkommen, dass sicher nicht der Frankreichbesuch von Frau Hilton gemeint ist ². Dazu ist es notwendig, Informationen zu beschreiben und auszuzeichnen. Die Funktion der Auszeichnung eines Wissensraums im Semantic Web übernimmt die *Ontologie*, die als Wissensbasis zu verstehen ist. Anhand einer Ontologie lassen sich Inferenzen, durch Einsatz von formaler Logik, bilden.

Neben der Informationssuche unterstützt das Semantic Web zusätzlich weitere Bereiche [15]:

- Die **Informationsextraktion** folgert logisch neue Informationen aus bereits vorhandenen.
- Die **Wartung** sich regelmäßig ändernder Texte wird durch die zugrundeliegende Struktur beeinflusst. Geschieht sie semantisch, anstatt syntaktisch, wird sie vereinfacht, da keine umständlichen Wrapper mehr notwendig sind.
- Die **automatische Generierung von Dokumenten** geht mit der Informationsextraktion einher.

Zusammenfassend betrachtet sorgt das Semantic Web für eine Strukturierung der zugrundeliegenden Daten. Dadurch können Informationen genauer gesucht und miteinander verbunden werden.

2.2 Technologien für Semantic Web Applikationen

Um eine Semantic Web Applikation zu erstellen, werden verschiedene Technologien und Techniken benötigt. Die wichtigsten sollen in diesem Abschnitt vorgestellt werden. Dazu wird als Erstes gezeigt, wie das eigene Wissen semantisch ausgezeichnet wird (vgl. 2.2.1, S. 8). Unterstützend zu dieser Auszeichnung wirken die Semantic Web Vokabulare (vgl. 2.2.2, S. 9), die Grundlagendefinitionen und die Beziehungen in einer konkreten Domäne beinhalten. Anschließend werden Methoden zur Datenspeicherung und zum Zugriff vorgestellt (vgl. 2.2.3, S. 9). Abschließend werden im Abschnitt 2.3, S. 11 einige Semantic Web Anwendungen präsentiert.

²Ausser dieser Satz steht im Zusammenhang mit der Annotation: $\langle \textit{Person} \rangle \textit{Paris Hilton} \langle /\textit{Person} \rangle$.

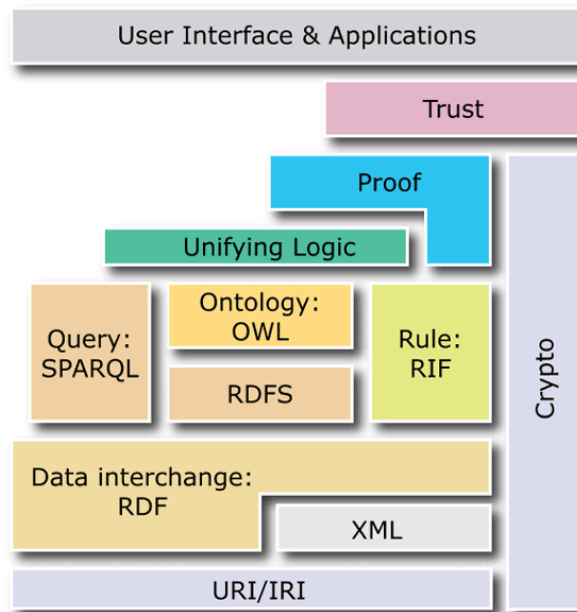


Abbildung 2.1: Schichten des Semantic Web

Einen Überblick zu den wichtigsten Semantic Web Technologien und deren Einordnung vermittelt die Abbildung 2.1, S. 7³. Die eindeutige Auszeichnung einer Ressource geschieht durch den **Uniform Resource Identifier (URI)**, beziehungsweise seinem internationalisiertem Pendant, dem **Internationalized Resource Identifier (IRI)**. Zum Datenaustausch wird meist RDF (vgl. Abschnitt 2.2.1, S. 8), XML und in letzter Zeit auch JSON verwendet. OWL (vgl. Abschnitt 2.2.1, S. 9) und RDFS (vgl. Abschnitt 2.2.1, S. 8) stellen als Ontologiebeschreibungssprachen entsprechende Techniken zur Modellierung einer Wissensdomäne zur Verfügung. Abgefragt werden können Informationen aus einer Wissensdomäne mit der Anfragesprache SPARQL (vgl. Abschnitt 2.2.3, S. 11). Durch die Hilfe von Inferenzmaschinen, die unter anderem unifizierte Logik einsetzen, lassen sich neue Informationen folgern. Der gesamte Prozess lässt sich eindeutig zuweisen, beziehungsweise kryptologisch verarbeiten. Später wird durch hinzufügen einer sozialen Komponente ein Web des Vertrauens etabliert, bei dem Nutzer den Informationen ihnen bekannter Nutzer indirekt mitvertrauen, worauf in dieser Arbeit allerdings nicht weiter eingegangen wird.

³Quelle: <http://www.semanticfocus.com/media/insets/semantic-web-layer-cake.png> [Stand: 21.03.2008]

2.2.1 Wissensrepräsentationsstandards

Der folgende Abschnitt befasst sich mit den Wissensrepräsentationsstandards des Semantic Web, die im Zuge dieser Arbeit eingesetzt wurden. Kurz dargestellt werden dabei RDF, RDFS und OWL.

RDF Das **R**esource **D**escription **F**ramework (RDF)⁴ ist eine formale Sprache, um strukturierte Informationen zu beschreiben. Mit dieser Sprache werden Daten im Web ausgetauscht, ohne dass ihre ursprüngliche Bedeutung verloren geht. Dargestellt werden Informationen entweder per Graph, oder in serialisierter Form als Tripel. Letzteres ist vorzuziehen, da Tripel maschinell besser zu verarbeiten sind. Eine Aussage in Form eines Tripel, besteht aus einem Subjekt, einem Prädikat und einem Objekt, beschrieben durch eine URI. Das Subjekt ist die zu beschreibende Ressource. Mit dem Prädikat werden Eigenschaften definiert. Das Objekt kann eine Ressource, oder ein Literal sein. [28] Die syntaktische Annotation von RDF kann auf unterschiedliche Weisen erfolgen. Die älteste Syntax zur Beschreibung von RDF Tripeln, ist die von Tim Berners-Lee geschaffene Notation **3** (N3). Das W3C schlägt hingegen die Syntax *N-Triples* vor. In der Praxis wird, wegen seiner Kurzschreibweisen, oft *Turtle* verwendet. In RDF werden Datenwerte durch Literale angegeben. Ist kein Datentyp mitangegeben, dann wird davon ausgegangen, dass es sich bei der Ressource um eine Zeichenkette handelt. Anderenfalls wird er entsprechend seines Typs interpretiert.

RDFS Das **R**esource **D**escription **F**ramework **S**chema (RDFS)⁵ ist eine Ontologiesprache, die die Syntax von RDF dahingehend erweitert, dass semantische Abhängigkeiten innerhalb einer Domäne dargestellt werden [28]. Sie bietet die Option, terminologisches Wissen zu spezifizieren. RDFS stellt mit den Annotationen *rdfs:subClassOf* und *rdfs:subPropertyOf* eine Möglichkeit zur Verfügung, ein Vererbungskonzept innerhalb eines Modells zu realisieren. Mithilfe von *rdfs:range* wird der Wertebereich eines Ausdrucks und mit *rdfs:domain* der Definitionsbereich desselbigen festgelegt. Weitere Konstrukte (*rdfs:label*, *rdfs:seeAlso*, *isDefined-By*, *rdfs:comment*) erweitern die Lesbarkeit für den Modellierer [18].

⁴<http://www.w3.org/RDF/> [Stand: 12.03.08]

⁵<http://www.w3.org/TR/rdf-schema/> [Stand: 12.03.08]

OWL Die Web Ontology Language (OWL)⁶ erweitert als Sprache für das Semantic Web, die Ausdrucksmächtigkeit von RDFS. Sie basiert auf der Prädikatenlogik erster Stufe, wodurch automatisches logisches Schlussfolgern⁷ ermöglicht wird. OWL gibt es in drei verschiedenen Teilsprachen: *OWL Full*, *OWL DL* und *OWL Lite*. Diese stehen in folgender Teilmengenbeziehung zueinander: $OWL\ Lite \subset OWL\ DL \subset OWL\ Full$. OWL Full ist unentscheidbar und wird deshalb nur bedingt eingesetzt. OWL DL ist entscheidbar und besitzt im schlechtesten Fall als Komplexität *NExpTime*. Dadurch eignet es sich besonders zur Abbildung komplexen Wissens, bei dem viel Inferenz benötigt wird. Wird mehr Wert auf Laufzeit, als auf Ausdrucksstärke gelegt, bietet sich OWL Lite mit der Komplexität *ExpTime* an. [18]

2.2.2 Semantic Web Vokabulare

Semantic Web Vokabulare definieren die Begrifflichkeiten einer Domäne. Das Ergebnis dieser Definition ist durch die Verwendung von URIs eindeutig. Darüber hinaus werden die Beziehungen zwischen diesen Begrifflichkeiten dargestellt. Tabelle 2.1, S. 10 veranschaulicht einige wichtige Vokabulare (RSS, FOAF⁸, SIOC⁹, DOAP¹⁰, vCard)¹¹.

2.2.3 Datenspeicherung und Zugriff

Nachdem im vorangegangenen Abschnitt die Wissensrepräsentationsstandards und Vokabulare für Semantic Web Applikationen vorgestellt wurden, werden im kommenden Abschnitt Möglichkeiten zur Datenspeicherung und zum Zugriff der entsprechenden Daten vorgestellt. In Triple Stores und Knowledge Stores werden die Daten gespeichert und mittels Linked Data und SPARQL abgefragt. Triplify hilft bei der Umwandlung der Daten in einer relationalen Datenbank, so dass diese semantisch genutzt werden können.

⁶<http://www.w3.org/2004/OWL/> [Stand: 12.03.08]

⁷Am gebräuchlichsten sind das Tableaukalkül und axiomatisches Schließen, unter den Konklusionsalgorithmen für OWL.

⁸<http://www.foaf-project.org/> [Stand: 12.03.08]

⁹<http://sioc-project.org/> [Stand: 12.03.08]

¹⁰<http://usefulinc.com/doap/> [Stand: 12.03.08]

¹¹Weitere Vokabulare und Informationen befinden sich auf: <http://schemaweb.info> [Stand: 12.03.08]

Tabelle 2.1: Übersicht verbreiteter Semantic Web Vokabulare

Vokabular	Erklärung
RSS 1.0	Das zum aktuellen Zeitpunkt (11.03.2008) gängigste Semantic Web Vokabular ist RDF Sity Summary (RSS). Es wird benutzt um eine Kurzzusammenfassung (oder auch nur ein paar Stichworte) einer Quelle zu einem konkreten Thema zu erhalten. Dadurch ist es, beispielsweise mit einem RSS-Reader, möglich bestimmte Informationen zu abonnieren und somit die Kurzzusammenfassungen an einer zentralen Stelle festzuhalten, obwohl die <u>Quellinformationen</u> verteilt sind.
FOAF	Friend of a Friend (FOAF) modelliert soziale Netzwerke in maschinenlesbarer Form. Mit FOAF lassen sich Menschen mit ihren persönlichen Werten, wie Name, Geburtstag oder Beruf, u.a. und ihren Beziehungen zu anderen Menschen darstellen.
SIOC	Semantically-Interlinked Online Communities (SIOC) stellt Methoden zur Verfügung, um Diskussionsplattformen (wie Blogs, Foren oder Mailing Listen) zu beschreiben und zu verbinden.
DOAP	Description of a Project (DOAP) wird benutzt um Open Source Projekte zu beschreiben.
vCard	vCard ist eine digitale Visitenkarte einer Person. Sie enthält Kontaktinformationen und allgemeine Informationen über die Person.

Triple Stores Tripel bestehen aus Subjekt, Prädikat und Objekt. Triple Stores kennzeichnen Speicherungsmöglichkeiten von Tripeln. Ein Beispiel dafür ist RDFlib¹² [28].

Knowledge Stores verwalten die Daten der Semantic Web Applikation als Menge von Tripeln. Sie werden in relationalen DBMS gespeichert und bieten APIs, Anfrageschnittstellen wie SPARQL, sowie integrierte Reasoning-Engines an [32]. Verbreitete Knowledge Stores sind Virtuoso¹³, RAP¹⁴, Sesame¹⁵, Jena¹⁶, Redland¹⁷, OWLIM¹⁸ und Oracle 11g¹⁹ [44].

¹²<http://rdflib.net/> [Stand: 20.04.2008]

¹³<http://virtuoso.openlinksw.com/> [Stand: 30.03.2008]

¹⁴<http://www4.wiwiss.fu-berlin.de/bizer/rdfapi/> [Stand: 30.03.2008]

¹⁵<http://www.openrdf.org/> [Stand: 30.03.2008]

¹⁶<http://jena.sourceforge.net/> [Stand: 30.03.2008]

¹⁷<http://librdf.org/> [Stand: 30.03.2008]

¹⁸<http://www.ontotext.com/owlim/> [Stand: 20.04.2008]

¹⁹<http://www.oracle.com> [Stand: 30.03.2008]

Linked Data benutzt das WWW, um Daten, Informationen und Wissen zu verbinden, die inhaltlich zusammengehören²⁰ und noch nicht verbunden sind. Dazu werden HTTP URIs und RDF benutzt [7].

SPARQL ist ein rekursives Akronym und steht für **SPARQL Protocol and RDF Query Language** (SPARQL). Verantwortlich für die Standardisierung und Entwicklung dieser Sprache ist das W3C²¹ und die RDF Data Access Working Group (DAWG)²². Die Sprache wurde am 15. Januar 2008 eine **World Wide Web Consortium (W3C) Recommendation**.

Eingesetzt wird SPARQL, um konkrete Informationen aus einer Ontologie abzufragen, analog zur Art und Weise, wie SQL bei relationalen Datenbanken als Anfragesprache eingesetzt wird. Als Ergebnismengen werden Ressourcen, Literale oder auch Untermodelle geliefert [17]. Der große Unterschied zwischen SPARQL und SQL ist, dass SPARQL bisher nur lesenden Zugriff auf die Ontologie erhält. SQL kann sowohl lesen, als auch schreiben. Der Schreibzugriff mit SPARQL wird durch andere Methoden, wie beispielsweise durch die Erfurt-API, realisiert.

Triplify²³ ist ein sehr leicht konfigurierbares Werkzeug mit dem Daten, welche in relationalen Datenbanken gespeichert sind, in RDF, JSON oder Linked Data umgewandelt werden können. Entwickelt wird Triplify an der Universität Leipzig seit 2008.

2.3 Anwendungsbeispiele für Semantic Web Applikationen

Eine Semantic Web Applikation setzt Technologien und Techniken des Semantic Web ein, mit dem Ziel, die Maschinenlesbarkeit und die Wartbarkeit der Webanwendung zu steigern [21]. Die Semantic Web Anwendung soll komplexe Anfrageausdrücke unterstützen und nicht mehr nur simple Schlagworte. Dabei soll das System, anhand von festgestellten Beziehungen zwischen Begrifflichkeiten, durch die Verwendung logischer Schlußfolgerung, neues Wissen extrahieren können²⁴.

²⁰Eine inhaltliche Zusammengehörigkeit folgt aus neuen Betrachtungsweisen und spezifischen Anfragen.

²¹<http://www.w3.org/> [Stand: 13.02.2008]

²²<http://www.w3.org/2001/sw/DataAccess/> [Stand: 13.02.2008]

²³<http://triplify.org/Overview> [Stand: 20.04.2008]

²⁴Eine umfassende Definition des Begriffes ist zu finden in der Masterarbeit von Michael Martin [24].

Im abschließenden Abschnitt dieses Kapitels werden kurz einige Semantic Web Anwendungen vorgestellt.

Semantic Web Suchmaschinen setzen die WWW-Suche semantisch um. Sie benutzen die Möglichkeit der logischen Schlußfolgerung zur Verbesserung der Suche. Swoogle²⁵ ist die älteste Semantic Web Suchmaschine. Weitere bekannte Varianten sind SWSE²⁶ und Sindice²⁷.

Ping the Semantic Web²⁸ ist ein Web-Service, der die Verfügbarkeit einer URI überprüft. Zur Identifizierung einer URI nutzt diese Anwendung aktuelle Versionen verbreiteter Vokabulare, wie SIOC, FOAF, DOAP, RDFS und OWL. Zusätzlich lassen sich RDF-Daten überwachen.

Joost²⁹ überträgt Fernsehkanäle im Internet via Peer-to-Peer-TV und setzt dabei auf Open-Source-Komponenten, wie RDF oder SVG. Joost setzt Semantic Web Technologien dazu ein, individualisierte Programme, durch eigenverantwortliche Verschlagwortung durch den Nutzer, zu erstellen [40].

OntoWiki³⁰ ist eine Semantic Web Anwendung, die in einem verteilten, agilen Wissensrepräsentationsumfeld unterstützend wirkt. Es handelt sich bei OntoWiki um ein RDF-verarbeitendes Wiki. Im Gegensatz zu anderen Semantic Wikis, beispielsweise das Semantic MediaWiki³¹, werden die Seiten im OntoWiki nicht um eine semantische Annotation ergänzt, sondern hier werden der Wissensbasis Wikitechnologien zur Verfügung gestellt. OntoWiki stellt Editierfunktionen und Visualisierungsoptionen zur Veranschaulichung von verschiedenartigen Daten samt ihrer Metadaten zur Verfügung. Darüber hinaus lässt es sich durch visuelle Themen und Plugins erweitern [37].

²⁵<http://swoogle.umbc.edu/> [Stand: 20.04.2008]

²⁶<http://swse.deri.org/> [Stand: 20.04.2008]

²⁷<http://sindice.com/query/keyword> [Stand: 20.04.2008]

²⁸<http://pingthesemanticweb.com/> [Stand: 20.04.2008]

²⁹<http://www.joost.com/> [Stand: 20.04.2008]

³⁰<http://ontowiki.net/Projects/OntoWiki> [Stand: 22.01.2008]

³¹http://semantic-mediawiki.org/wiki/Semantic_MediaWiki [Stand: 23.04.2008]

3 Vorgehensmodelle

Ein Vorgehensmodell, oder auch Prozessmodell, oder Prozessmanagement genannt, soll die Komplexität der Softwareentstehungsphase, also der Zeit, die zwischen der ersten Anforderungsanalyse und der Endabgabe verstreicht, handhabbar gestalten (quasi den gesamten Prozess der Softwareentwicklung regeln). Das Prozessmodell definiert Aktivitäten in einer gewissen Reihenfolge und legt Artefakte (Produkte) fest, die aus den Aktivitäten resultieren.

„Ein Vorgehensmodell ist eine Beschreibung einer koordinierten Vorgehensweise bei der Abwicklung eines Vorhabens. Es definiert sowohl den Input, der zur Abwicklung der Aktivität notwendig ist, als auch den Output, der als Ergebnis der Aktivität produziert wird. Dabei wird eine feste Zuordnung von Rollen vorgenommen, die die jeweilige Aktivität ausüben.“ [41]

Unter Vorgehen oder Prozess versteht man eine konkrete Vorgehensweise bei einer bestimmten Problemstellung oder bei einer Phase im Prozess der Softwareentstehung. Auswahlkriterien für ein bestimmtes Vorgehensmodells sind vor allem die Größe und Verteilung des Projektteams, sowie die Projektart und der Projektumfang. Da es sich bei Software um ein immaterielles Produkt handelt, lassen sich einige ingenieurspezifische Erfahrungen (Planungsstrategien) nicht übertragen. Die Begründung für diesen Punkt spiegelt sich in Meßbarkeitsproblemen wider. In anderen Fällen, wie dem Testen eines Prototypen hinsichtlich diverser Kriterien wie Funktionsfähigkeit, Belastbarkeit, Fehleranfälligkeit, der Verwendung von Architekturmuster, ect. profitiert die Informatik bei der Gestaltung von Vorgehensmodellen von den Erfahrungen der etablierten Ingenieurkünste (wie dem Bauwesen, der Elektrotechnik usw.).

Die Ziele des Einsatzes von Prozessmodellen sind im Allgemeinen:

- einen besseren Einsatz des Personals durch festgelegte Rollen erreichen
- eine effizientere Kommunikation mit den Auftraggebern umsetzen

- die Erhöhung der Qualität des Produktes verwirklichen
- eine bessere Werkzeugunterstützung realisieren

3.1 Klassische Vorgehensmodelle

Unter klassischen Vorgehensmodellen werden die ersten praxisrelevanten Verfahren zur systematischen Erstellung von Software verstanden. Ihre Existenz begründet sich durch das erste Aufkommen der Softwarekrise. Im folgenden soll auf die zwei wichtigsten klassischen Vorgehensmodelle eingegangen werden, das Wasserfallmodell und das Spiralmodell, wobei Letzteres durch ein paar Erweiterungen schon zu den modernen Vorgehensmodellen zu zählen ist. Die Erweiterungen sollen in dieser Arbeit jedoch nicht betrachtet werden.

3.1.1 Wasserfallmodell

Das Wasserfallmodell ist das älteste Vorgehensmodell [30]. Es wird häufig als eine der wesentlichen Ursachen der Softwarekrise bezeichnet. Dennoch existiert weiterhin ein Nutzerkreis, welcher dieses veraltete Vorgehensmodell einsetzt [6].

Im Wasserfallmodell laufen die Aktivitäten sequentiell ab, veranschaulicht in Abbildung 3.1, S. 14.

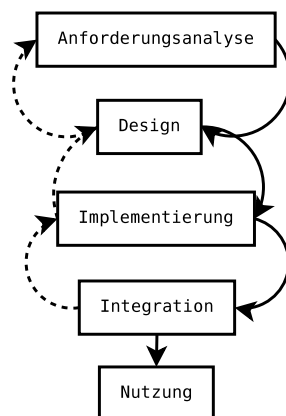


Abbildung 3.1: Hauptphasen des Wasserfallmodell

Aus dieser Abbildung gehen folgende Hauptphasen des Softwareentwicklungsprozesses hervor [33]:

Anforderungsmanagement erzeugt eine möglichst exakte Spezifikation des zu entwickelnden Systems.

Design trifft eine Auswahl bezüglich Ausstattung, einzusetzender Softwaremodule und dem Aufbau von Benutzerschnittstellen. Im Detail werden Datenfluss, Algorithmen und Benutzeroberflächen beschrieben.

Implementierung und Tests setzen das Entwurfsmodell in einer gewählten Programmiersprache um.

Integration steht für den Einsatz in einem Produktivsystem. Also wird hier das System beim Kunden installiert und die Mitarbeiter des Kunden erhalten eine entsprechende Schulung.

Jede einzelne Phase erreicht durch festgelegte Transitionen ihren Nachfolger, beziehungsweise auch den Vorgänger. Im gesamten Lebenszyklus des Wasserfallmodells wird Dokumentation betrieben. Diese wird benutzt um das Ergebnis einer Phase festzuhalten, womit verbindliche Dokumente gegenüber einem Kunden erzeugt werden.

Gründe für das Scheitern dieses Vorgehensmodells im praktischen Einsatz sind [35]:

- Die Anforderungen werden nicht vollständig vor Projektbeginn erfasst.
- Die Benutzer wissen meist erst, was sie wirklich wollen, wenn sie einen Prototypen direkt sehen.
- Anforderungen ändern sich während der Entwicklungsphase.
- Neue Werkzeuge und Technologien machen die Implementierung schwer einschätzbar.

Fazit Da in diesem Ablauf keine Phase für das Änderungsmanagement vorgesehen und geeignet ist, erweist sich das Wasserfallmodell als unflexibel und nur für kleine Projekte¹ geeignet [5], bei denen schon im Vorfeld klar definierte Anforderungen, Probleme und Lösungsalgorithmen existieren und somit das Auftreten von Änderungen selten ist.

¹Entwicklung von Prototypen, Kleinprogramme, Programmierung von Hardware im Embedded-Bereich

3.1.2 Spiralmodell

Das Spiralmodell ist eine Weiterentwicklung des Wasserfallmodells, in dem die besten Teile desselbigen zusammengefügt werden mit Prototyping und Risikoanalyse. Dieses Vorgehensmodell enthält folgende Aktivitäten:

- Festlegung der Ziele und Rahmenbedingungen
- Evaluierung von Alternativen, Reduktion möglicher Risiken
- Realisierung und Überprüfung des Zwischenergebnisses
- Projektplanung
- Spirale beenden oder neustarten

Fazit Charakteristisch für klassische Vorgehensmodelle ist ihre Unflexibilität gegenüber auftretenden, unerwarteten Änderungen. Das Spiralmodell legt mehr Wert auf die Projektplanung, als auf die eigentliche Umsetzung, wodurch wiederum Problemfälle aufgezeigt werden können. Die möglichen Einsatzgebiete sind im Vergleich zum Wasserfallmodell etwas komplexer, da durch die Risikoanalyse auch größere, aber dennoch klar definierte Projekte realisierbar sind.

3.2 Moderne Vorgehensmodelle

Nachfolgend wird auch von schwergewichtigen Vorgehensmodellen die Rede sein. Das kostenlose V-Modell (XT) und das kommerzielle RUP werden in den kommenden Unterabschnitten dazu verwendet Merkmale, sowie Vor- und auch Nachteile dieser Prozessmodelle zu beleuchten.

3.2.1 V-Modell

Das V-Modell ist eine Projektmanagement-Struktur. In einer v-förmigen Darstellung gliedert es Projektelemente nach etwaigem Zeitbedarf und Detailtiefe, veranschaulicht in folgender Abbildung ²:

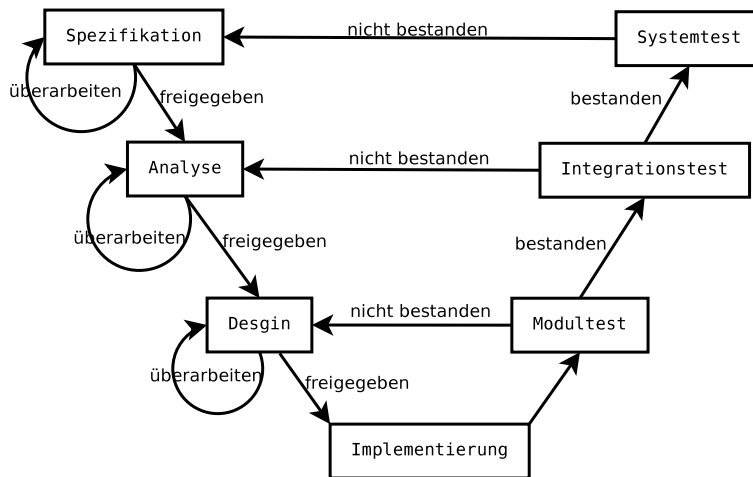


Abbildung 3.2: Hauptphasen des V-Modell

Fazit Das V-Modell wurde im Auftrag der Bundesregierung entwickelt. Diese sieht für alle auftretenden Prozesse eine Dokumentation vor. Durch den großen Umfang des Berichtwesens ist bei den meisten Entwicklungsvorgängen von diesem Vorgehensmodell abzuraten. Das V-Modell XT ³ welches 2005 veröffentlicht wurde versucht dieses Problem zu beseitigen, durch zusätzliche Modularisierung und einer Orientierung an agilen Ansätzen.

3.2.2 Rational Unified Process

Im Jahre 1999 kam der **Rational Unified Process (RUP)** erstmals auf den Markt [41]. Als Basis für RUP dienten verschiedene Vorgehensmodelle, wie das Spiralmodell, die Object Management Technologie und das Object-oriented Software Engineering. Somit ist dieses Prozess-

²Quelle: Grafik adaptiert von: <http://www.entwickler-software.com/glossar/images/vmodell14.jpg> [Stand: 14.01.2008]

³<http://www.v-modell-xt.de/> [Stand: 11.04.2008]

modell hybrid. RUP besteht aus Aktivitäten und Artefakten, die von bestimmten Rollen durchgeführt werden. Die Aktivitäten werden unterschiedlichen Disziplinen zugeordnet:

Geschäftsprozessmodellierung stellt den Ablauf des Geschäftsprozesses meist grafisch dar. Berücksichtigt werden dabei auch Datenflüsse und Organisation.

Anforderungsmanagement hält die Anforderungen die der Kunde stellt genau fest. Außerdem werden die Auswirkungen der Änderung auf das System analysiert, bevor sie akzeptiert werden.

Analyse bedeutet das Einsatzszenario, samt möglicher auftretender Risiken, für die Entwicklung zu erschließen.

Design trifft Entscheidungen im Detail, bezüglich eingesetzter Algorithmen, Programmiersprachen und zu verwendenden Werkzeugen.

Implementierung ist die programmiertechnische Umsetzung des Projekts.

Test wird verschiedenartig durchgeführt, um sicherzugehen, dass die entwickelte Software fehlerfrei lauffähig an der Kunden ausgeliefert werden kann.

Verteilung beinhaltet die Einbettung in das Endanwendersystem.

Darüber hinaus werden diese Disziplinen durch folgende Vorgänge unterstützt:

Konfigurations- und Änderungsmanagement ist für die Erweiterbarkeit und Änderbarkeit des Projektes zuständig.

Projektmanagement stellt Methoden zum Durchführen der Projekte zur Verfügung.

Umgebungsdisziplin stellt zusätzlich erforderte Ressourcen und zugehöriges erforderetes Wissen bereit, um die Projektrealisierung zu verbessern.

Diese Disziplinen werden über den gesamten Ablauf der Softwareentwicklung verfolgt. Im RUP wird zwischen folgenden vier Phasen differenziert [33]:

Konzeptionsphase Das Ziel dieser Phase ist es einen Anwendungsfall zu etablieren. Hierzu müssen alle Entitäten (Rollen und Systeme), die mit dem System interagieren, identifiziert und ihre Interaktionen definiert werden.

Entwurfsphase In dieser Phase soll die Entwurfsdomäne kritisch betrachtet und hinsichtlich möglicher zukünftig auftretender Risiken untersucht werden. Am Ende der Entwurfsphase verfügt das Projekt über ein Systemanforderungsmodell, eine architektonische Beschreibung und einen Entwicklungsplan für diese Software.

Konstruktionsphase Die Implementierung, das Design und die Tests werden in der Konstruktionsphase durchgeführt. Ein lauffähiges System mitsamt der zugehörigen Dokumentation ist das Ergebnis dieser Phase.

Übergangsphase Abschließend wird das erzeugte System dem Kunden übergeben. In der Übergangsphase wird gesichert, dass das System auch beim Kunden funktioniert, also in einem realen Umfeld.

Die Phasen werden in Iterationen unterteilt, veranschaulicht in Abbildung 3.3, S.19 [41].

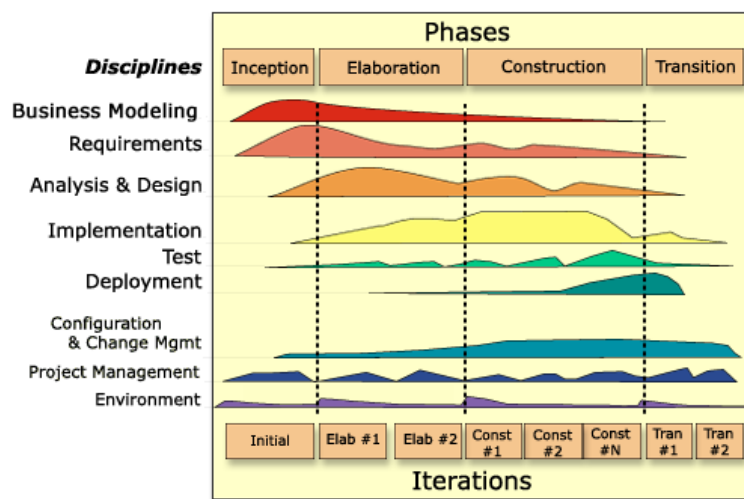


Abbildung 3.3: Phasen und Disziplinen des Rational Unified Process

Fazit Die wichtigste Innovation dieses Vorgehensmodells ist die Trennung zwischen Phasen und Disziplinen und die Berücksichtigung, dass die Einbettung, der entwickelten Software in das Kundensystem, ein Teil des Prozesses ist. Phasen sind dynamisch und besitzen Ziele, während Disziplinen statisch (technische Aktivitäten) und keiner konkreten Phase zugeordnet sind. [33]

RUP ist ein Vorgehensmodell, das für die Entwicklung objektorientierter Software vorgesehen ist. Es passt aber nicht zu allen Entwicklungstypen. Vor allem bei kleineren Projekten wären sowohl Lizenzkosten, als auch Einarbeitungs- und Nutzungsaufwand zu hoch.

3.3 Agile Vorgehensmodelle

Im Jahr 2001 fanden sich die Vordenker von XP (Kent Beck), Scrum (Ken Schwaber und Jeff Sutherland), Crystal ⁴ (Alistair Cockburn) und Vordenker anderer Ansätze zusammen. Das Ergebnis dieses Treffens war das Agile Manifest. [22]

Inhaltlich zusammengefasst besagt das Agile Manifest, dass projektbezogene Personen zu bevorzugen sind. Im Zusammenhang mit diesem Punkt steht die Zusammenarbeit mit dem Kunden. Der Kunde wird direkt in das Projekt miteinbezogen (als Tester oder jemand der regelmäßig Rückmeldung über den Projektstand erstattet) und ist somit nicht mehr nur der Geldgeber und der Vertragspartner. Weiterhin ist funktionsfähige Software wichtiger, als eine umfassende Dokumentation. Es wird mehr Wert auf Rückmeldung, als auf Einhaltung eines Plans gelegt.

Agile Vorgehensmodelle können flexibel auf wechselnde Anforderungen reagieren. Sie stellen die Entwicklung und die Menschen, die daran beteiligt sind, in den Vordergrund. Dadurch werden Risiken schneller erkannt und bekämpft. Die Systementwicklung erfolgt mit agilen Prozessen iterativ, wodurch sich die Gesamtkomplexität des Systems verringert, da immer in Teilaufgaben vorgegangen wird. Durch den evolutionären Ansatz werden frühstmöglich Ergebnisse erzielt, die für Rückmeldungen zum Kunden eingesetzt werden, was wiederum eine rapide Reaktion auf Änderungswünsche bewirkt.

Agile Softwareentwicklung besteht aus kurzen Entwicklungszyklen. Im Gegensatz zu einem sequentiellen Vorgehensmodell, wie dem Wasserfallmodell, werden bei der agilen Herangehensweise Implementierung, Tests, Design und Analyse parallel durchgeführt.

Zur Veranschaulichung dieser Charakteristika dienen die agilen Vorgehensmodelle **eXtreme Programming (XP)** und **Scrum**.

⁴ist ein verbreitetes agiles Vorgehensmodell, auf das in dieser Arbeit nicht weiter eingegangen wird. Informationen zu Crystal: http://alistair.cockburn.us/index.php/Crystal_methodologies_main_foyer [Stand: 01.03.2008]

3.3.1 eXtreme-Programming

XP wurde dazu geschaffen, kleinen Entwicklergruppen ein Vorgehensmodell bereitzustellen, um Projekte mit wechselnden Anforderungen zu realisieren. Hierbei werden die Entwickler in den Vordergrund gestellt. Berücksichtigt wird bei der Entwicklung auch die Haltbarkeit der Software. Ziel ist es, eine Langlebigkeit dieser zu erwirken, um einen maximalen Nutzen-Kosten Gewinn zu erzielen. Dabei wird das Programmieren die Hauptaktivität der gesamten Entwicklungsphase [2].

Programmiert wird immer paarweise, wobei die Zweier-Teams regelmäßig vermischt werden. Hierbei wird testgetrieben vorgegangen. Das heisst, es werden zuerst Tests geschrieben, bevor konkret geforderte Funktionen umgesetzt werden. Um erfolgreich mit XP zu arbeiten sind vier Werte vonnöten:

Kommunikation muss stetig und in ausreichendem Umfang erfolgen, sowohl unter den Entwicklern, als auch mit den Auftraggebern.

Einfachheit zieht für gewöhnlich eine geringere Fehlerrate nach sich, als komplexe Lösungsansätze. Weiterhin hat es sich erwiesen, dass es effektiver ist erst in einer späteren Projektphase komplexe Ansätze umzusetzen [2].

Rückmeldung beinhaltet die Reaktion des Systems auf einen Testfall oder auf mehrere.

Mut ist erforderlich, um als Programmierer seiner eigenen Arbeit kritisch gegenüber zu stehen und auch Code verwerfen zu können, wenn dieser nicht optimal ist. Ein Entwickler muss Veränderungen wollen.

Diese Werte sind noch zu vage, deswegen wird zusätzlich nach Grundprinzipien differenziert [3]:

Unmittelbare Rückmeldung erspart ein erneutes Rekapitulieren des Sachverhalts.

Inkrementelle Veränderung bedeutet in Teilschritten bei der Systemerweiterung vorzugehen und nicht Alles auf einmal umzusetzen. Die *Teile und Herrsche* Strategie beweist den Vorteil dieser Vorgehensweise.

Qualitätsarbeit läßt sich schlecht messen. Hierzu gehört unter Anderem ein sauberer Programmierstil inklusive Dokumentation.

Da Personenbezogen entwickelt wird bietet sich die Verwendung eines Rollensystems an. In XP wird zwischen folgenden Rollen unterschieden:

Der Programmierer ist der Hauptakteur des Entwicklungsprozesses. Abweichend, von der klassischen Auffassung über einen Programmierer, muss ein XP Programmierer vor allem kommunikative Aufgaben mit übernehmen. Durch das vorgeschriebene paarweise Programmieren ist ein Einzelkämpferdasein damit ausgeschlossen.

Der Kunde ist ein fester Bestandteil des Entwicklerteams. Er gibt regelmäßig Rückmeldung zum aktuellen Projektstand, stößt Änderungen an und schreibt darüber hinaus selbst Tests für die Anwendung.

Der Tester ist verantwortlich für die Ausführung der geschriebenen Tests und für die Bekanntgabe der Ergebnisse der Tests.

Der Terminmanager hat die Aufgabe dem Team Rückmeldung zu geben, über die Aufwandschätzung der letzten Iteration (war diese zu hoch oder zu niedrig). Darüber hinaus trifft er Voraussagen, ob die Anforderungen für eine Iteration erfüllt werden können oder auch nicht.

Der Coach ist für den Gesamtprozess verantwortlich. Er überwacht alle Mitglieder und versucht in Problemfällen zu helfen und bei Konflikten durchzugreifen.

Der Berater ist ein externer Experte, der bei technischen Fragen hinzugezogen wird.

Der Boss ist der Verwalter des Teams. Er wird über auftretende Flaschenhälse, also etwaige Probleme bei der Umsetzung des Projekts informiert. Darüber hinaus zeichnet er sich für Personalfragen verantwortlich.

In Abbildung 3.4⁵ wird der Ablauf einer Iteration innerhalb des Prozessmodells XP grafisch veranschaulicht.

Fazit Extremprogrammierung ist besonders geeignet für kleine bis mittlere Projekte, mit wechselnden Anforderungen. Wird eine gewisse Projektgröße überschritten, ist die Kommunikation zu teuer, sowohl zeitlich, als auch aufwandstechnisch. Der Zwang des paarweisen Programmierens und der testgetriebenen Entwicklung ist für manche Projekte ungeeignet. Frameworks sollten nicht mit XP entwickelt werden, da hier sowohl Schnittstellen, als auch

⁵Quelle: Grafik adaptiert von: <http://www.extremeprogramming.org/> [Stand: 01.03.2008]

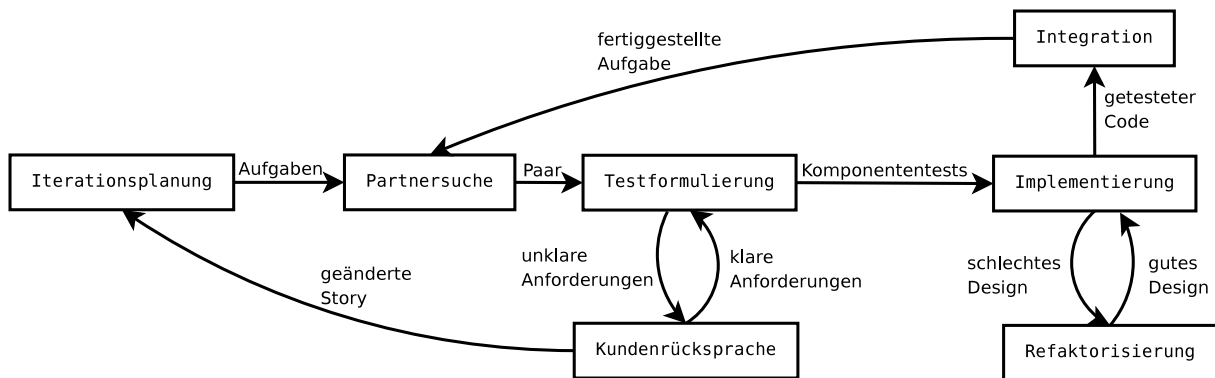


Abbildung 3.4: Ablauf einer XP Iteration

das Design von Anfang an feststehen müssen. Außerdem ist es bei einigen Projekten einfach unmöglich, nicht regelmäßig Überstunden zu machen, was mit XP strikt vermieden werden soll [14].

3.3.2 Scrum

Die Idee ein Team mit Entscheidungsfreiraum und einer globalen Sicht auszustatten, sowie täglich einen Meilenstein zu setzen, wurde bereits erfolgreich in der Industrie vorgemacht, wie Honda, Canon und Fujitsu zeigen. 1993 wurde Scrum zum ersten Mal in einem Softwareentwicklungsprozess eingesetzt [35]. Die Produktivitätssteigerung, die aus täglichen Besprechungen resultiert, wurde bereits 1994 von Borland festgestellt. Ein Team aus vier und später acht Entwicklern hat in 31 Monaten 1.000.000 Zeilen Quellcode erzeugt. Das entspricht 200 Zeilen Quellcode pro Werktag pro Person und ist somit einer der produktivsten Softwareentwicklungsprozesse, die je dokumentiert wurden [11].

Begonnen wird bei diesem Vorgehensmodell mit einer ausführlichen Planungssitzung, die das Gesamtziel für die nächsten 6 Monate beschreibt. Anschließend wird diese Zeit in 6 Sprints (vgl. Abbildung 4.1, S. 28) unterteilt, bei denen zu implementierende Funktionalitäten im Detail beschrieben werden. Tägliche Sitzungen bewirken, dass jedes Teammitglied einen Überblick auf alle Aspekte des Projekts erhält. Diese werden kurz gehalten (für gewöhnlich weniger als 30 Minuten). Dabei werden inhaltlich folgende Fragen geklärt:

- Was hat man gestern gemacht?
- Was wird man heute tun?

- Welche Hindernisse traten bisher auf und welche könnten noch auftreten?

Diese Treffen erzeugen bei den Teammitgliedern ein besseres Gesamtverständnis. Dadurch, dass jedes Mitglied jeden Tag den Fortschritt und auch die Probleme aller anderen Mitarbeiter sieht, lassen sich Engpässe und Problemstellen leichter behandeln. Oft pflanzen sich Fehler in einer Kette unterschiedlicher Entwicklungsschritte einfach fort. So kann ein eingangs simpler Fehler in einem Folgeschritt drastische Konsequenzen nach sich ziehen. Wenn ein solches Problem bereits von Anfang an bekämpft wird, werden viele Arbeitsstunden eingespart. Nicht nur für die Beseitigung des eigentlichen Fehlers, sondern auch für die Mitarbeiter, die für das Testen und für die Dokumentation der Anwendung verantwortlich sind.

An jedem Monatsende bekommt der Auftraggeber eine lauffähige Demoversion, die die aktuell hinzugefügten Funktionalitäten enthält. So kann er schnell Rückmeldung bezüglich auftretender Änderungswünsche geben.

Fazit Durch die aufgelockerte Struktur des Prozessmodells wird dem Entwickler zugearbeitet. Der Entwicklungsprozess verläuft in seinem Sinne und wird nicht durch schwergewichtige Komponenten, wie Pflichten- oder Lastenheft oder sonstiges starres Berichtswesen aufgehalten. Nachteilig ist nur, dass Scrum für maximal mittelgroße Projekte (ca. 10 Personen) eingesetzt werden kann (zumindest in dieser Weise), da ansonsten eine Kommunikation zwischen den Teammitgliedern zu umständlich wird. Vor Allem aber sollte es bei Anwendungen eingesetzt werden, bei denen vorher noch nicht alle Anforderungen klar sind.

Da dies der Fall bei einer Semantic Web Applikation ist, widmen sich Kapitel 4, S. 27 und Kapitel 5, S. 37 der Umsetzung von Scrum in einer Produktivumgebung.

Bei dem Vergleich dieser beiden agilen Methoden fällt auf, dass der Schwerpunkt von XP hauptsächlich auf dem Programmieren liegt, wohingegen Scrum sich zusätzlich um das Projektmanagement kümmert.

3.4 Gegenüberstellung der Vorgehensmodelle

Ausgehend von den vorgestellten Vorgehensmodellen, lassen sich folgende allgemeine Merkmale extrahieren (vgl. Tabelle 3.1, S. 25) ⁶.

Tabelle 3.1: Gegenüberstellung der verschiedenen Arten von Vorgehensmodellen

	<i>Agile</i>	<i>Klassische und Moderne</i>
Vorgehensweise	empirisch, evolutionär	vorhersehbar
Genauigkeit von Anforderungen	gering	hoch
Betonung auf	rapide Entwicklung	Vollständigkeit ursprünglicher Anforderungen
Erfolgsmaß	Gewinn	Planeinhaltung
Projektgröße	klein	groß
Budget	gering	hoch
Erfahrung der Teammitglieder	überdurchschnittlich	projektabhängig
Management	dezentralisiert	autokratisch
Dokumentation	gering	unabdingbar
Verhältnis zum Kunden	vertrauensvoll	geschäftlich
Schwerpunkt	Personen	Prozesse
Treibende Kraft	Personen	Dokumente + Plan

Klassische Prozessmodelle gehen bei der Entwicklung planmäßig vor und sind daher vorhersehbar. Sie benötigen eine umfassende Dokumentation. Agile Methoden beruhen auf Erfahrungswissen und auf evolutionärer Entwicklung. Dadurch wird auf eine genauere Dokumentation (und dem damit verbundenen Kosten- und Arbeitsaufwand zur Erstellung derselbigen) verzichtet.

Bei klassischen Vorgehensmodellen obliegen kritische Entscheidungsfragen (Designänderung, Architekturänderung u.a.) lediglich dem Management. Wohingegen bei agilen Methoden die Entwickler selbst im Mittelpunkt stehen und das Management mehr Hintergrund- und Verwaltungsarbeit, als Entscheidungsarbeit erledigt.

Die Entwicklungsstrategie wird mit schwergewichtigen Vorgehensmodellen durch Prozesse

⁶Quelle adaptiert von: <http://agile-cologne.de:8080/snipsnap-0.5.2a/space/Vorgehensmodelle> [Stand: 11.04.2008]

(die durchaus fehlerhaft oder nicht optimal sein können) umgesetzt. Agile Ansätze vertrauen auf die Kompetenz und die Erfahrung der eigenen Mitarbeiter.

Agile Prozesse eignen sich besonders bei kleinen und mittleren Projekten, bei denen die Anforderungen unklar sind. Mit zunehmender Projektgröße wird die Kommunikation erschwert. Das hat zur Folge, dass eine Steigerung der Durchführung administrativer Prozesse und zusätzliches Berichtswesen unabdingbar zur erfolgreichen Beendigung eines Projekts sind.

Die agile Vorgehensweise begünstigt evolutionäres Vorgehen von Beginn an. So kann ein Auftraggeber durch Prototypen, in regelmäßigen Abständen, die Erfüllung der gestellten Anforderungen überprüfen. Dadurch bietet es sich an, je nach Bedarf, zu einem späteren Projektzeitpunkt größere Investitionen vorzunehmen, wodurch mehr Ressourcen benutzt werden können, um die Realisierung der Software zu beschleunigen. Auf der anderen Seite kann ein erfolgloses Projekt, unter einem geringeren finanziellen Verlust, abgebrochen werden. Die schwergewichtigen Prozesse erschweren dem Auftraggeber dieses Vorgehen, denn hier steht das Team normalerweise von Anfang an in vollem Umfang fest. Durch Spezialistenrollen lassen sich schlecht Kräfte einsparen, ohne einen qualitativen Verlust hinnehmen zu müssen.

Agile Prozesse kennen Spezialisten nur in begrenztem Maße. Jeder Mitarbeiter kennt das ganze Projekt mit allen Stärken und Schwächen. Das erfordert mehr Einarbeitungszeit und Fachwissen, als bei den schwergewichtigen Methoden vonnöten ist.

Die Realisierung der Software erfolgt mit klassischen Prozessen strikt nach Berichtswesen, wie Pflichtenheft und Lastenheft. Hier wird eine eindeutige Umsetzung der darin gestellten Anforderungen angestrebt. Wohingegen agile Methoden sich primär auf Hauptfunktionalitäten konzentrieren und alle anderen Zusatzfunktionalitäten je nach Bedarf entwickeln, was eine schnelle Prototypisierung nach sich zieht. Dadurch kann der Kunde schneller Änderungswünsche äußern oder bei Zeitdruck einfach Zusatzfunktionalitäten streichen.

Ein weiterer wichtiger Punkt für agile Prozesse ist die Verfügbarkeit des Kunden. Ohne eine direkte Teilnahme des Kunden am Projekt, ist ein Erfolg schwer vorstellbar. Bei schwergewichtigen Methoden ist die Teilnahme des Kunden nicht zwangsläufig notwendig, da man sich hier auf die erfassten Anforderungen, zur Umsetzung der Software, stützen kann. Das beinhaltet, dass Anforderung klar mit langwierigen Prozessen erfasst werden müssen. Beim Einsatz agiler Methoden können Anforderungen ungenauer gestellt werden, da sie regelmäßig anhand von Prototypen überprüft werden können.

4 Anwendung von Scrum zur Entwicklung einer Semantic Web Applikation

Dieses Kapitel befasst sich mit der Umsetzung von Scrum zur Entwicklung einer Semantic Web Applikation. Im ersten Abschnitt soll allgemein über den Aufbau von Scrum referiert werden. Im darauf folgenden Abschnitt wird auf die Integration in eine Produktivumgebung eingegangen. Abschließend folgt die Bewertung von Scrum zur Entwicklung einer Semantic Web Anwendung.

4.1 Scrum Ablauf bei einer Semantic Web Applikation

Wenn die Entscheidung auf Scrum als Vorgehensmodell zur Entwicklung einer Semantic Web Applikation fällt, wird als erstes die Vision (vgl. 4.1.1, S. 28) festgelegt. Daraus wird anschließend der Product Backlog (vgl. 4.1.3, S. 29) extrahiert. Beteiligte Rollen (vgl. 4.1.2, S. 28) werden festgelegt. Danach wird der Product Backlog in Etappen zerlegt, den sogenannten Sprint Backlogs (vgl. 4.1.4, S. 30). Dargestellt wird der Ablauf in der Abbildung 4.1¹, S. 28. Nach der Festlegung dieser Werte beginnt die eigentliche Entwicklung. Da eine Webanwendung entwickelt wird, kann der Auftraggeber jederzeit problemlos² auf den aktuellen Stand zugreifen und eine Rückmeldung dazu liefern. So wird iterativ vorgegangen, bis der Product Backlog erreicht wurde und die Anwendung ausgeliefert werden kann.

¹Quelle: http://jeffsutherland.com/scrum/uploaded_images/scrum-overview-741006.gif [Stand: 07.03.2008]

²Bezogen auf die Option die Webanwendung in einem beliebigen Browser zu testen, wodurch keine zusätzliche Software oder Programmierverständnis benötigt wird.

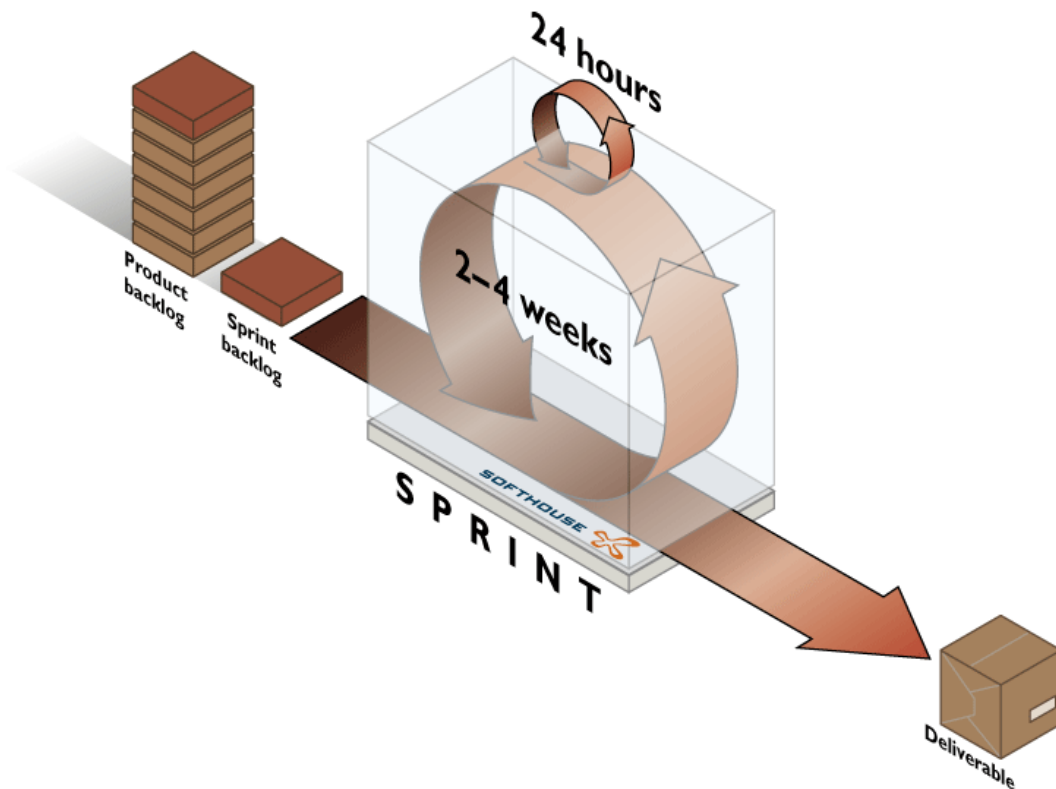


Abbildung 4.1: Ablaufzyklus von Scrum

4.1.1 Vision

Ein Projekt, bei dem Scrum eingesetzt wird, beginnt mit einer Vision des Systems. Sie enthält die allgemeinen Anforderungen an das Projekt, sowie die Zeit- und Kostenabschätzung, die zur Realisierung benötigt werden. Die Vision dient als Entscheidungsgrundlage zur Projektdurchführung. Die Anforderungsanalyse für eine Semantic Web Applikation kann beispielsweise mit SWORE [38] erfolgen.

4.1.2 Rollen

Rollen erzeugen Spezialisten, welche zur erfolgreichen Bewältigung komplexer Projekte benötigt werden. Allerdings bedeutet dies auch ein Problem für den Fall, dass ein Spezialist ausfällt. Zur Absicherung gegenüber dem Ausfall eines Spezialisten verzichtet Scrum auf ein

umfangreiches Rollenkonzept. Das Ziel ist, dass jedes Mitglied ein Experte für das ganze System ist. Deshalb unterscheidet Scrum nur zwischen drei verschiedenen Rollen:

Product Owner ist dafür verantwortlich die Interessen aller beteiligter Personen zu repräsentieren. Ausserdem akquiriert er die finanziellen Mittel und legt zu Projektbeginn die allgemeinen Anforderungen (Product Backlog, S. 29) fest. Darüber hinaus kümmert er sich darum, dass in jeder Iteration die wichtigsten Anforderungen berücksichtigt werden. Bei der Entwicklung von Semantic Web Anwendungen muss der Product Owner über einen aktuellen Überblick, bezüglich Techniken und Technologien, in diesem Bereich verfügen.

Team entwickelt die Funktionalitäten. Das Team ist selbstorganisierend. Es legt fest, wie der Product Backlog iterativ umgesetzt wird und was darüber hinaus benötigt wird, um das Sprint Ziel zu erreichen.

ScrumMaster ist der Prozessverantwortliche. Er unterrichtet die Projektmitglieder und überwacht die Umsetzung der Konzepte und der Rollen, sowie die Umsetzung der Vision.

4.1.3 Product Backlog

Der Product Backlog ist eine Liste aller funktionalen und nichtfunktionalen Anforderungen. Sie wird vom Product Owner kontrolliert. Jedoch tragen weitere Personen Punkte zum Product Backlog bei. [1] Sie ordnet die Produktanforderungen nach Geschäftswert an. Ein Auszug des Product Backlog der Booking Application dient in diesem Fall als Beispiel (vgl. Tabelle 4.1, S. 30). Hierbei werden Funktionen unterschiedliche Prioritäten zugeordnet. Je kleiner der Wert der Priorität ist, desto wichtiger ist die Funktion für das Gesamtprojekt. Weiterhin geht aus dieser Übersicht der oder die Bearbeiter der Funktion, sowie eine Aufwandsabschätzung in Stunden hervor. Der Auftraggeber erhält so eine kompakte Sicht auf die wichtigsten Anforderungen und kann, bei Bedarf, weitere hinzufügen oder ändern. Der Product Backlog ist nicht vollständig. Er entwickelt sich dynamisch mit dem Produkt. Das bedeutet in diesem Fall, dass sich Prioritäten verschieben und neue Funktionen hinzukommen können. Nach einigen Sprints könnte beispielsweise der Bank Konnektor im Product Backlog ganz nach oben gelangen, da die Anforderung gestellt wird, bestimmte Kaufmannsfunktionen automatisiert durch das Buchungssystem ablaufen zu lassen.

Tabelle 4.1: Auszug aus dem Product Backlog der Booking Application

<i>Priorität</i>	<i>Funktion</i>	<i>Beschreibung</i>	<i>Aufwand</i>	<i>Bearbeiter</i>
sehr hoch	1	Modell erzeugen	30	AS, MM, JL
	2	Web Frontend Prototyp	25	MM
	3	Controller Struktur festlegen	20	TW, MM
	4	Framework Auswahl	10	JL, MM, RM
	5	Tests: Frameworks	30	RM
	6	Erzeugung von Testdaten	15	AS
hoch	7	Anlegen von Nutzern	20	AS
	8	Modelldaten in tabellenform anzeigen	24	RM
	9	Controller implementieren	80	TW
	10	Testfälle schreiben	30	JL, MM
	11	Session	10	TW
	12	JS Validierung	8	RM, TW
	13	Suchanfragen per JSON	20	AS, MM
mittel	14	Themeschalter	10	TW
	15	Bank Konnektor	20	AS
	16	JS freie Seiten erstellen	50	TW, RM
	17	Automatischer Emailversand	10	JL
niedrig	18	Grafischer Drag & Drop Planer	250	AS
	19	Analysewerkzeuge	120	MM, AS
	20	Plugin Schnittstelle	50	TW

4.1.4 Sprint und Sprint Backlog

Ein *Sprint* bezeichnet eine Iteration und dauert einen Monat. Zu Beginn eines Sprints legt das Team fest was getan werden muss. Das geschieht im sogenannten *Sprint Meeting*. Es wird in zwei Teile getrennt. In den ersten vier Stunden präsentiert der *Product Owner* den *Product Backlog*. Hierbei stellen die Mitarbeiter Fragen zu dieser Anforderungsliste, bezüglich möglicherweise auftretender Ambivalenzen, konkretem Inhalt und Intension der Applikation. Anhand der daraus resultierenden Punkte werden für diesen Iterationsschritt zu implementierende Funktionalitäten extrahiert. Diese werden im *Sprint Backlog* (vgl. Tabelle 4.2, S. 31) festgehalten. Der Sprint Backlog stellt Aufgaben und Zeit in Stunden dar, die schätzungsweise für die Umsetzung benötigt wird. Dabei werden die Aufgaben so unterteilt, dass sie in 4 bis maximal 16 Stunden zu bewältigen sind. Das Team erstellt den Sprint Backlog selbstständig. Im zweiten Teil des Scrum Meetings plant das Team den Ablauf des kommenden Sprints. Das Ergebnis einer Iteration muss immer ausführbar, getestet und problemlos erweiterbar sein.

Tabelle 4.2: Sprint Backlog aus der Booking Application

<i>Aufgabe</i>	<i>Mo</i>	<i>Di</i>	<i>Mi</i>	<i>Do</i>	<i>Fr</i>
Implementierung der Benutzerschnittstelle	6	6	8	2	5
Implementierung von Nutzer registrieren			3	8	
Implementierung von Passwort vergessen				4	2
Implementierung von ändere Wert			8	8	7
Einrichtung eines Bug Tracker	3				
Schreiben der Dokumentation	1	1	1	1	1

Aus Abbildung 4.1, S. 28 geht die Zerteilung eines Sprints in Tagesetappen hervor. Das Projektteam trifft sich täglich zu einer festgelegten Zeit und geht auf die im Abschnitt 3.3.2, S. 23 erwähnten Fragen ein. Es wird innerhalb von maximal 10 Minuten geklärt, was seit dem letzten Treffen erreicht wurde, welche Probleme auftraten und was als Nächstes geplant ist. So hat jeder Beteiligte eine aktuelle und vollständige Übersicht, bezüglich des Projektstands. Außerdem lassen sich Probleme besser eingrenzen, weil der Erfahrungsschatz (zur Problemanalyse) einer gesamten Gruppe zur Verfügung steht. Zur zusätzlichen Unterstützung der aktuellen Übersicht wird eine *Aufgabentafel*³ (vgl. Abbildung 4.2, S. 32) verwendet. Hier werden die Aufgaben auf Zettel festgehalten und können, je nach Bedarf, in andere Bereiche umgehängt werden.

Am Ende eines Sprints findet eine *Rezension* statt. Das Projektteam stellt dem Product Owner (und eventuell den Auftraggebern) vor, was seit dem letzten Sprint entwickelt wurde. So wird die Möglichkeit geschaffen, Fehlentwicklungen frühzeitig zu erkennen. Dadurch erhalten die Auftraggeber die Option, ihre Anforderungen regelmäßig ändern, erweitern oder aktualisieren zu können.

Nach der Rezension findet die *Retrospektive* mit dem ScrumMaster statt. Ihre Funktion ist es, das Vorgehen des Teams zu analysieren, wodurch Verbesserungsvorschläge offeriert werden, zur Optimierung des Scrum Prozesses und damit zur Steigerung der Produktivität des Teams. Dazu wird das *Burndown Diagramm* verwendet [26].

³Quelle: http://www.mountaingoatsoftware.com/system/hidden_asset/file/29/MockedTaskBoard.jpg [Stand: 10.03.2008]

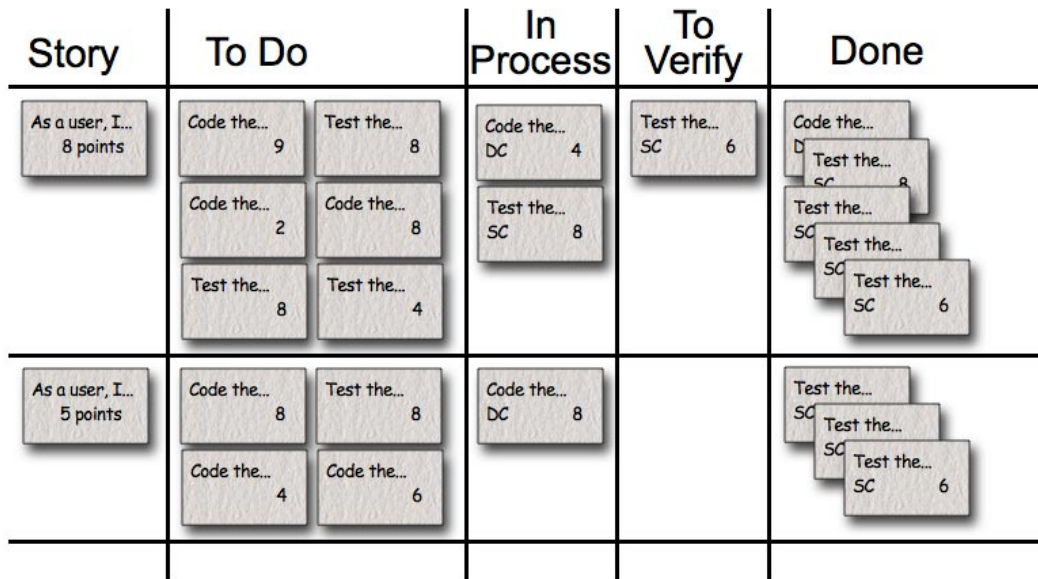


Abbildung 4.2: Beispiel einer Aufgabentafel

Burndown Diagramm

Ein Burndown Diagramm⁴ (vgl. Abbildung 4.3, S. 33) spiegelt den Arbeitsumfang in Relation zur verbleibenden Zeit wider. Durch die Visualisierung der Korrelation zwischen Arbeitsaufwand und tatsächlich erledigter Arbeit lassen sich Problemstellen schneller erkennen. Die horizontale Achse beziffert die eingeplanten Sprints bis zum Projektende. Die vertikale Achse stellt den noch verbliebenen Arbeitsumfang (gemessen in verbliebenen Anforderungen) dar.

Innerhalb einer Iteration agiert das Projektteam autark. So werden den Mitgliedern Aufgaben nach persönlichen Kompetenzen zugewiesen. Treten Probleme auf, werden sie in der Gruppe diskutiert und unmittelbar behandelt. Dadurch ergeben sich deutlich weniger Folgefehler im Vergleich zu stark rollenbasierten Herangehensweisen, bei denen sich die Projektmitglieder infolge ihrer Spezialisierungen schlechter helfen können [31].

⁴Quelle: <http://www.mountaingoatsoftware.com/images/releaseburndown.png> [Stand: 10.03.2008]

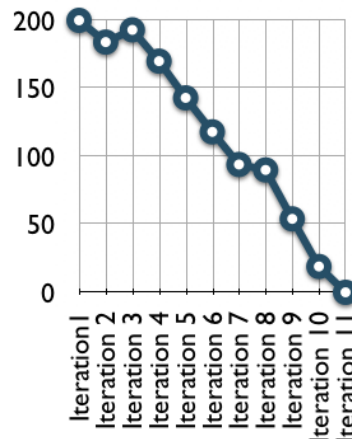


Abbildung 4.3: Burndown Diagramm

4.1.5 Evolutionäre Umsetzung

Unter genauer Betrachtung fällt auf, dass die eigentliche Entwicklung in Scrum evolutionär abläuft. Aus diesem Grund befasst sich dieser Abschnitt mit evolutionärer Entwicklung und seiner Umsetzung.

Evolutionäre Entwicklung beruht auf der Idee, eine initiale Implementation als Grundlage für alle notwendigen Erweiterungen zu nutzen, welche phasenweise folgen, solange bis ein endgültiges System produziert wird. Dabei wird in regelmäßigen Abständen der Auftraggeber hinzugezogen, um Änderungs- und Erweiterungswünsche zu äußern. [33] Im Folgenden wird nicht mehr von initialer Implementation die Rede sein, sondern vielmehr von einem Prototypen.

*„In der Technik stellt ein **Prototyp** ein für die jeweiligen Zwecke funktionsfähiges, oft aber auch vereinfachtes Versuchsmodell eines geplanten Produktes oder Bauteils dar. Es kann dabei nur rein äußerlich oder auch technisch dem Endprodukt entsprechen. Ein Prototyp dient oft als Vorbereitung einer Serienproduktion, kann aber auch als Einzelstück geplant sein, das nur ein bestimmtes Konzept illustrieren soll.“⁵*

Es wird zwischen zwei grundsätzlichen Arten bei der evolutionären Entwicklung unterschieden:

⁵Quelle: <http://de.wikipedia.org/wiki/Prototyp> [Stand: 12.02.2008]

Exploratives Prototyping stellt dem Auftraggeber Stück für Stück neue Funktionalitäten zur Verfügung, die er dann testen und absegnen oder ändern lassen kann. Es entsteht nach einer bestimmten Anzahl von Schritten, durch ständige Erweiterung, eine finale Softwareversion.

Wegwerf Prototyping wird sehr unsauber umgesetzt und dient nur der Verdeutlichung der Anforderungen, besonders solcher die noch unklar sind.

Der evolutionäre Ansatz ist effektiv, da beispielsweise der Umfang und die Genauigkeit des Berichtwesens, im Vergleich zu konventionellen Vorgehensmodellen, erheblich reduziert wird. Diese Reduktion ist ein Produkt der gesteigerten Kommunikation der Projektbeteiligten untereinander. Es ist nicht notwendig zu Projektbeginn jede Funktion zu beschreiben. Da sich heutzutage immer mehr Anforderungen zu Beginn des Projekts schlecht oder gar nicht abschätzen lassen, können schwergewichtige Softwareentwicklungsansätze fehlschlagen.

4.2 Auswahl und Integration von Scrum

Ein Vorgehensmodell muss dem Projekteinsatz im Unternehmen angepasst sein. Ausgehend von den Werten der Tabelle 3.1, S. 25, werden die wichtigsten Auswahlkriterien berücksichtigt. Dazu wird zuerst der Projekttyp und die Projektgröße betrachtet. Bei einer Semantic Web Applikation, wie sie in Kapitel 5, S. 37 vorgestellt wird, ist davon auszugehen, dass die Projektgröße klein bis mittel ist.

Anschließend werden bereits existierende und funktionierende Prozesse im Unternehmen bewertet, da nicht jeder Prozess innerhalb des Vorgehensmodells für ein Projekt zwingend optimal ist. Es ist vorstellbar, dass die existierenden Prozesse bereits maximal effizient sind. Dadurch, dass Scrum ein lockeres Rollensystem hat, lassen sich zusätzliche Rollen einführen, die dann für die existierenden (zu übernehmenden) Prozesse verantwortlich sind. In diesem Fall ist ein Experte unumgänglich, da es zu zeitintensiv wäre die Softwareentwickler in die firmeneigenen Prozesse einzuführen.

Im nächsten Schritt sind die Anforderungen des Kunden zu berücksichtigen. In sicherheitskritischen Projekten werden die Anforderungen seitens des Kunden als Erweiterung des Vorgehensmodells, in Form gesteigerter Qualitätssicherung festgehalten. Scrum ist gerade wegen

seiner Flexibilität besonders für den Einsatz als Vorgehensmodell zur Entwicklung von Semantic Web Anwendungen geeignet. Charakteristisch für diesen Anwendungstypen ist, dass sich die Anforderungen besonders häufig ändern oder weiterentwickeln.

Unternehmensvorlagen, die das Berichtswesen betreffen, müssen den Firmenvorgaben entsprechend angepasst werden. Dies betrifft vor Allem die schwergewichtigen Softwareentwicklungsprozesse. Der *Product Backlog*, sowie der *Sprint Backlog* in Scrum sind problemlos erweiterbar. Darüber hinaus sind heutige Semantic Web Anwendungen noch keine riesigen Projekte⁶, wodurch die Bedeutung eines umfangreichen Berichtwesens in den Hintergrund rückt.

Abschließend sollen die Artefakte bewertet werden. Ein Vorgehensmodell erzeugt unter Umständen unerwünschte Nebenprodukte oder sogar nicht ausreichende Endprodukte, durch eine fehlende Dokumentation und andere Versäumnisse. In einer Semantic Web Anwendung könnte beispielsweise der Einsatz einer neuen Technologie fehlen oder Schnittstellen, um eine solche problemlos zu integrieren. Scrum beugt diesem Fehler durch frühzeitige Rückmeldungen seitens des Auftraggebers und durch regelmäßige Rezensionen vor.

Scrum in einem Unternehmen zu integrieren ist, in Folge der bereits erwähnten Punkte, leicht realisierbar.

4.3 Bewertung von Scrum zur Entwicklung einer Semantic Web Applikation

Ein Vorgehensmodell zur Entwicklung einer Semantic Web Anwendung muss sich schnell auf Änderungen einstellen können. In den vorherigen Abschnitten wurde bereits gezeigt, dass dieser Punkt auf Scrum zutrifft. Das wichtigste Kriterium ist jedoch die Effektivität (hinsichtlich der Wirtschaftlichkeit) des Vorgehensmodells. Scrum kostet nichts in der Anschaffung. Es sind keine Lizenzgebühren dafür notwendig. Lediglich der ScrumMaster sollte entsprechend kostenpflichtig geschult sein. Scrum fokussiert sich auf den eigentlichen Implementierungsprozess und spart dadurch Zeit und Geld für unnötige Fehlentwürfe, die informell festgehalten werden.

⁶Allerdings hat Jeff Sutherland 2007 nachgewiesen, dass Scrum, mit wenigen Erweiterungen, auch bei großen Projekten funktioniert. [19]

Der klare Vorteil einer evolutionären Entwicklungsweise ist, dass die Anforderungen und Spezifikationen inkrementell weiterentwickelt werden. Prototypen offerieren schnellstmögliche Rückmeldungen seitens des Auftraggebers.

Allerdings soll an dieser Stelle nicht unerwähnt bleiben, dass auch Probleme im Zusammenhang mit der Verwendung und Entwicklung von Prototypen existieren:

- Der Prozess ist nicht sichtbar für Manager. Diese benötigen regelmäßigen, meßbaren Fortschritt. Beim Einsatz zur Erstellung schnell entwickelter Systeme ist eine, entsprechend der Industrienorm, geforderte Dokumentation notwendig, die jede Version widerspiegelt. Die Erstellung dieser Dokumentation ist nicht kosteneffektiv, da wertvolle Arbeitsstunden für eine (im Sinn der Industrienorm) ordnungsgemäße Dokumentation benötigt werden.⁷
- Systeme, die so produziert werden, sind oft schlecht strukturiert. Dadurch steigert sich die Schwierigkeit der Erstellung von Schnittstellen zu anderen Softwaresystemen.

Fazit Eine Semantic Web Anwendung ist ein Projekt kleineren bis mittleren Umfangs, welches von regelmäßigen Änderungen betroffen ist. Diese Werte und der Fakt, dass bei agilen Vorgehensmodellen der Schwerpunkt auf der eigentlichen Entwicklung und nicht auf dem sonstigen benötigten Berichtswesen und anderen administrativen Prozessen liegt, lassen auf Scrum als Idealkandidaten eines Vorgehensmodells zur Erzeugung einer Semantic Web Anwendung schließen. Andere agile Vorgehensmodelle, wie XP, sind in ihrer Struktur zu starr (keine Überstunden, testgetriebene Entwicklungsweise) oder nicht restriktiv genug, wie Chrystal (da es sehr auf Projektmanagement bezogen ist). Ein konkretes Beispiel der Anwendung von Scrum zur Entwicklung einer Semantic Web Anwendung folgt in Kapitel 5.5, S. 51.

Scrum hebt sich von anderen agilen Ansätzen durch seine größere Flexibilität ab, die besonders in der Phase der Neuerschaffung von Software wichtig ist. Überdies existieren bereits Ansätze, wie Scrum bei verteilten, großen Projekten erfolgreich eingesetzt werden kann, mit einem geringen Mehraufwand [36].

⁷Mike Cohn löst dieses Problem im Vorfeld durch anders aufgebaute Verträge zwischen Auftrager und Entwickler. [9]

5 Anwendungsfall Booking Application

Einleitend wird die Booking Application erklärt, bezüglich Anwendungsart, Funktionalitäten und technischer Umsetzung. Im zweiten Abschnitt dieses Kapitels geht es um die konkreten Anforderungen, die an die Buchungsanwendung gestellt sind. Basierend auf den Anforderungen soll anschließend ein Modell extrahiert werden. Danach werden Technologien, die bei der Umsetzung der Booking Application zum Einsatz kommen vorgestellt. Am Ende dieses Kapitels wird beschrieben, wie Scrum zur Entwicklung der Buchungsanwendung eingesetzt wird.

5.1 Beschreibung der Booking Application

Die Booking Application ist eine semantische Webanwendung, die dem Endbenutzer einen komfortablen, sicheren, schnellen und intuitiven Weg offeriert, um diverse Objekte, wie eine Busfahrt, eine Schiffsreise, einen Restaurantbesuch oder auch eine Kombination aus unterschiedlichen Objekten buchen zu können. Abbildung 5.1, S. 38 zeigt die Oberfläche dieser Anwendung.

Diese Anwendung stellt Funktionen bereit, die es ermöglichen, die Buchungsdaten effektiv zu verwalten. Zu diesen gehören ein GUI-basierter Buchungsplaner¹, sowie ein Benutzerverwaltungssystem, welches verschiedene Sichten und Rollen ermöglicht. Das verwendete Benutzerverwaltungssystem wird im Abschnitt 5.4.2, S. 50 genauer vorgestellt.

Der Endbenutzer nimmt keinen Unterschied zu einer konventionellen Web 2.0 Applikation, beispielsweise einem Blog oder einem sozialen Netzwerk, in Hinblick auf die Nutzbarkeit

¹Weitere Informationen dazu befinden sich in den beiden Bachelorarbeiten von Ruslan Masold [25] und von Tom Wieland [43]

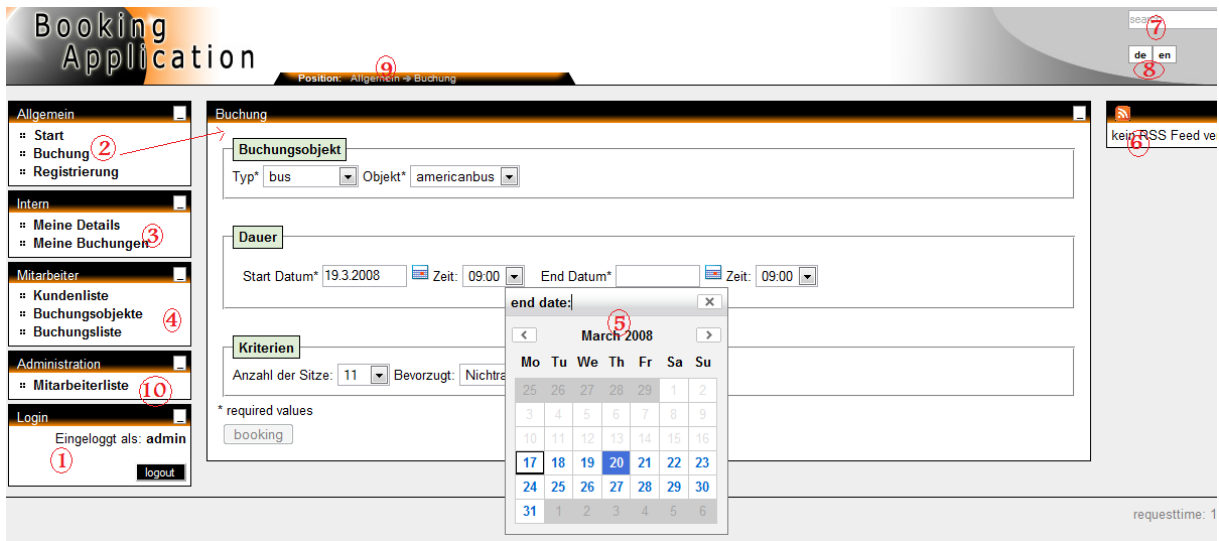


Abbildung 5.1: Aufbau der Oberfläche der Booking Application

und die Form der Darstellung wahr. Um diesen äußeren Anschein zu erzeugen stellt die Booking Application interaktive Elemente zur Verfügung, wie einen Feed (vgl. Abbildung 5.1, (6)), den bereits erwähnten GUI-basierten Buchungsplaner (2), durch AJAX erzeugte Tabellen (5) oder die Positionsleiste (9). Zusätzlich erhält der Nutzer immer die Funktionen Sprache wechseln (8), anmelden und abmelden (1), suchen (7), buchen und Benutzer registrieren (2). Die Sichtbarkeit und damit auch die Benutzbarkeit der Boxen (3), (4) und (10) ist an die Rolle des angemeldeten Nutzers gebunden. In diesem Fall ist der Administrator angemeldet und darf somit alles sehen und benutzen.

Entscheidend bei einer semantischen Webapplikation ist, was sich im Kern der Anwendung abspielt. Das Hauptaugenmerk bei der Entwicklung der Booking Application, im Folgenden auch *Buchungsanwendung* genannt, liegt auf der Erweiterbarkeit, der Bedienbarkeit, der Wartbarkeit und vor allem der Maschinenlesbarkeit. Die ersten beiden Punkte lassen sich auch ohne ein semantisches Gefüge um die Anwendung entwickeln. Der Schwerpunkt liegt auf der Maschinenlesbarkeit. Das Ziel der Entwicklung der Buchungsanwendung ist es, über ein simples Auszeichnen von Metainformationen (welche zur besseren maschinellen Verarbeitung von Informationen eingefügt wird) hinauszugehen. Stattdessen wird die von RDF, RDFS und OWL zur Verfügung gestellte Syntax zur merklichen Verbesserung dieses Punktes genutzt. In der Motivation wurde die Bedeutung der Maschinenlesbarkeit bereits ausführlich dargestellt. Durch die erreichte Steigerung der Maschinenlesbarkeit wird es Agenten, wie Webcrawlern

oder Bots und anderen Programmen erleichtert, mit den entsprechenden Daten zu arbeiten. Ein positiver Effekt ist weiterhin, dass sich ausgehend von der zugrundeliegenden Ontologie, Änderungen oder Erweiterungen bezüglich des Modells sehr schnell umsetzen lassen. Dieser Punkt wird im Abschnitt 5.3, S. 43, am konkreten Beispiel ersichtlich. Weiterhin lässt sich eine Semantic Web, wie die Booking Application, sehr gut mit anderen Webanwendungen vernetzen. Dazu werden gemeinsame Vokabulare genutzt (vgl. Abschnitt 2.2.2, S. 9).

5.2 Anforderungen an die Booking Application

Die Booking Application ist eine Semantic Web Anwendung. Als solche verwendet sie Web 2.0 Technologien, wie AJAX, JSON und REST. AJAX stellt dem Benutzer interaktive Elemente zur Verfügung. Mit JSON werden benötigte Daten codiert und dann mit REST transportiert. Weitere wichtige, nichtfunktionale Anforderungen (neben der intuitiven Benutzbarkeit der Buchungsanwendung) sind:

- Sicherheit zu gewährleisten (im Hinblick auf gespeicherte Kundendaten und durchgeführte Transaktionen)
- Problemlose Erweiterbarkeit bereitzustellen und dennoch performant zu bleiben

Die Buchungsanwendung verwaltet buchbare Objekte verschiedener Firmen. Diese verschiedenen Firmen bieten unterschiedliche Dienste an. So ist es möglich, bei der Buchung einer Schifffahrt ein ganzes Deck zu mieten, wenn die Kosten dafür im Voraus beglichen werden. Ein anderes Beispiel ist, dass eine Funktion geplant ist, die Informationen bereitstellt, wie die Tische und Stühle eines Restaurants zu einem besonderen Anlass (geschlossene Gesellschaft, Hochzeit usw.) aufgestellt werden. Diese Funktion wäre bei einem Schiff noch vorstellbar, für einen Bus jedoch nicht sinnvoll. Aus diesem Grund müssen Regeln zur Verwendung der Modellinformationen und der, in der Buchungsanwendung angebotenen Funktionen, festgelegt werden.

Diese buchbaren Objekte müssen in einem gewissen Rahmen editierbar sein. Die Buchungsanwendung muss entscheiden können, ob man beispielsweise einen Bus aus seiner Ontologie löschen darf oder nicht und was dann mit den zugehörigen Daten geschieht. Unter zugehörige Daten fallen dem Bus zugeordnete Mitarbeiter, sowie transaktionsrelevante Informationen, also welche Buchungsvorgänge, von wem und wie, noch offen sind. Wenn das System feststellt,

dass der Bus X keine zugeordneten Transaktionen und Mitarbeiter hat, so wird er aus der Buchungsobjektsicht (nicht aus der Datenbank) gelöscht. In dem Fall, dass eine dieser Restriktionen verletzt wird, soll die Buchungsanwendung dem Benutzer helfen, indem es eine genaue Fehlermeldung und einen Problemlösungsansatz zurückgibt.

Gleiches gilt auch für alle anderen, vom System verwalteten Daten. Wenn ein Mitarbeiter einen Kunden löschen möchte, so ist dies grundsätzlich möglich, insofern keine Zahlungen vom Kunden ausstehen. Ein Kunde soll dieses Recht nicht erhalten. Ihm wird die Sicht auf die Kundendaten verwehrt. Er kann nur seine eigenen Informationen, wie Name, Anschrift, Kontaktoptionen, Zahlungswege und Buchungsvorgänge bearbeiten. Auch hierbei soll das System dem Kunden helfen, indem es alle Eingaben prüft und bestimmte Informationen automatisch ergänzt. Wenn eine Postleitzahl nur genau zu einem Ort gehört, so kann dieser nach der Eingabe der Postleitzahl durch eine AJAX-Anfrage ermittelt und in das entsprechende Feld eingetragen werden. Zur Steigerung der Sicherheit wird die Richtigkeit aller Eingaben, so weit wie möglich, automatisch getestet. Wenn eine Postleitzahl nicht fünfstellig ist, der Kunde aber Deutschland als Heimatland angegeben hat, dann ist sie falsch und das System muss dies umgehend melden. Ein weiteres Beispiel ist die automatische Verifikation der Emailadresse des Kunden. Sowie eine Emailadresse neu eingegeben oder geändert wird, prüft eine Funktion die Erreichbarkeit dieser Adresse und gibt bei Nichterreichbarkeit eine Fehlermeldung zurück.

Die Hauptfunktion der Booking Application ist das Buchen von Objekten. Hier kommen automatisierte Regeln zum Einsatz, die die Mitarbeiter der Tourismusfirmen bei ihrer täglichen Arbeit unterstützen. Das Semantic Web wirkt bei der Modellierung der Regeln unterstützend. Beispiele für Regeln der Booking Application:

- Wird eine Ausnahme gebucht (andere Tischaufstellung in einem Restaurant, anderes Essen, oder ein Bus mit Programm für die Gäste einer Hochzeit), so muss diese vollständig im Vorraus, bis 4 Wochen vor dem eigentlichen gebuchten Termin, bezahlt werden.
- Wenn eine solche Ausnahme nicht mindestens zur Hälfte ausgebucht ist, dann soll die Booking Application den zuständigen Mitarbeiter informieren, der dann über eine mögliche Stornierung entscheiden kann.
- Wenn Urlaubszeiten, Reparaturen oder sonstige Ausfälle anstehen, so muss die Buchung für das betreffende Objekt verboten sein.

- Wenn die Buchungssumme 1.000 Euro übersteigt, dann muss 50 % davon im Voraus bezahlt werden.
- Zwei Tage vor dem Termin sollen keinerlei Änderungen an der Buchung mehr zulässig sein.
- Stehen zu einem Termin nicht ausreichend Kapazitäten zur Verfügung, so sollen Alternativen angeboten werden.
- Ändern sich die Nahrungsbestände des Lagers, so soll das System Mangel oder Überkapazitäten aufzeigen und in Einzelfällen auch selbstständig Nachschub bestellen können.

5.2.1 Anwendungsfälle innerhalb der Booking Application

Im vorangegangenen Abschnitt wurden bereits mehrere Anwendungsfälle erwähnt. In diesem Abschnitt sollen zwei Anwendungsfälle genauer erklärt werden. Zum einen die Hauptfunktion, das Buchen eines Objektes und zum anderen das Ändern der Emailadresse durch einen Kunden.

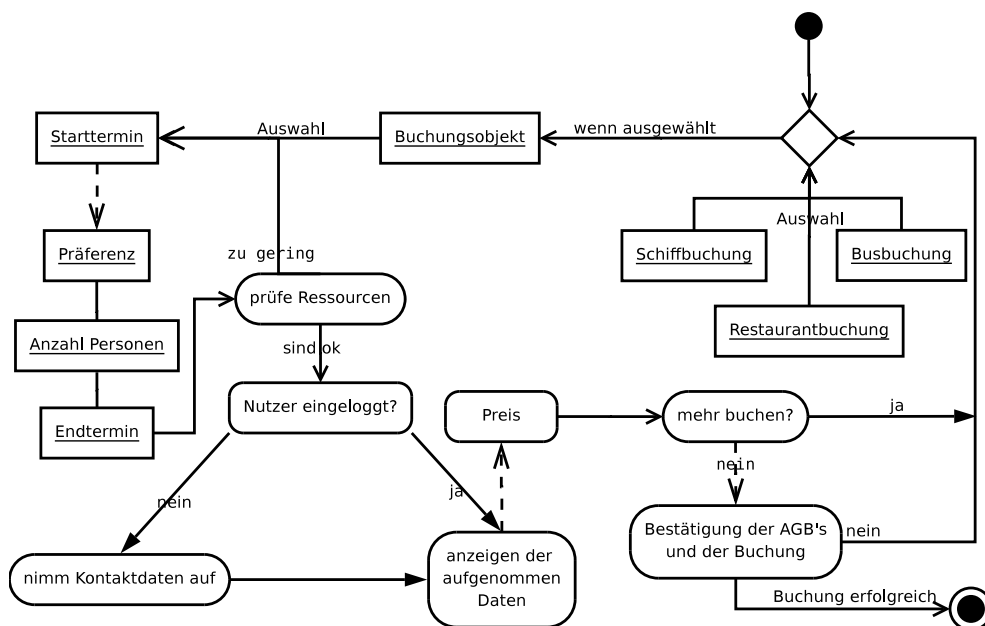


Abbildung 5.2: Ablauf einer Buchung

Anwendungsfall Buchung Eine Buchung beginnt mit der Auswahl eines Objektes, mit den möglichen Fällen Bus, Restaurant oder Schiff, wie in der Abbildung 5.2, S. 41 ersichtlich ist. Daraufhin wird diese Auswahl spezifiziert, ob der Kunde nun den amerikanischen Schulbus oder den Londonbus wählen möchte (analog für Restaurant und Schiff). Damit steht das Buchungsobjekt fest. Danach wird ein Starttermin und Endtermin ausgewählt und dabei die Verfügbarkeit des Objektes zu diesem Zeitraum berücksichtigt². Optional ist die Angabe der Präferenz und der Anzahl der Personen. Werden diese Werte geändert, so prüft das System im Hintergrund die Verfügbarkeit des gewünschten Buchungsobjekts zum gewünschten Termin. Wenn alle benötigten Werte ausgewählt wurden, testet die Buchungsanwendung ob der Kunde angemeldet ist oder nicht. Für den Fall, dass er nicht angemeldet ist, werden die angegebenen Daten in einer Session zwischengespeichert und er wird direkt zum Anmeldeformular weitergeleitet. Ist dieses Formular nicht korrekt ausfüllt, wird die Aktion abgebrochen. Ist der Kunde korrekt angemeldet, sollen die Allgemeinen Geschäftsbedingungen und der Preis der Buchung angezeigt werden. Nun kann der Kunde bestätigen, ob er die Buchung unter diesen Konditionen durchführen will oder nicht. Wenn er das nicht will, wird die Buchung abgebrochen. Will der Kunde die Buchung so durchführen, wird sie erfolgreich verbucht. Anschließend kann er weitere Buchungen durchführen, die dann zu einer Gesamtrechnung zusammengefügt werden. Dabei werden der Gesamtpreis und mögliche Änderungen der Allgemeinen Geschäftsbedingungen berücksichtigt und angezeigt. Der Kunde erhält eine Auftragsbestätigung per Email.

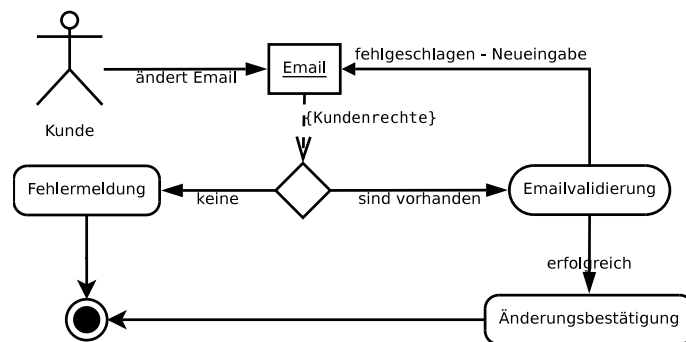


Abbildung 5.3: Ablauf einer Änderung der Emailadresse des Kunden

²Der Übersichtlichkeit halber, wurden die zusätzlich benötigten Überprüfungen und Auswahlen nicht mit in die Abbildung 5.2 aufgenommen.

Anwendungsfall Emailadresse ändern Ein Kunde ändert seine Emailadresse. Dazu muss überprüft werden, ob er die nötigen Rechte besitzt (vgl. Abbildung 5.3), um diese Aktion durchzuführen. Wenn er diese nicht besitzt, dann wird eine Fehlermeldung angezeigt und die Aktion wird abgebrochen. Im Regelfall besitzt eine Kunde dieses Recht. Also wird nun die Emailadresse validiert. Ist die Änderung korrekt, wird sie gespeichert. Ansonsten wird der Nutzer zur Neueingabe, mit Hinweis auf den möglichen Fehler, aufgefordert.

5.3 Ontologie der Booking Application

Ausgehend vom Hauptanwendungsfall Buchung und unter Berücksichtigung maximaler Erweiterbarkeit zur Verfügung zu stellen, wurde die folgende Ontologie (vgl. Abbildung 5.4, S. 43), unter Verwendung von OWL, für die Booking Application erstellt.

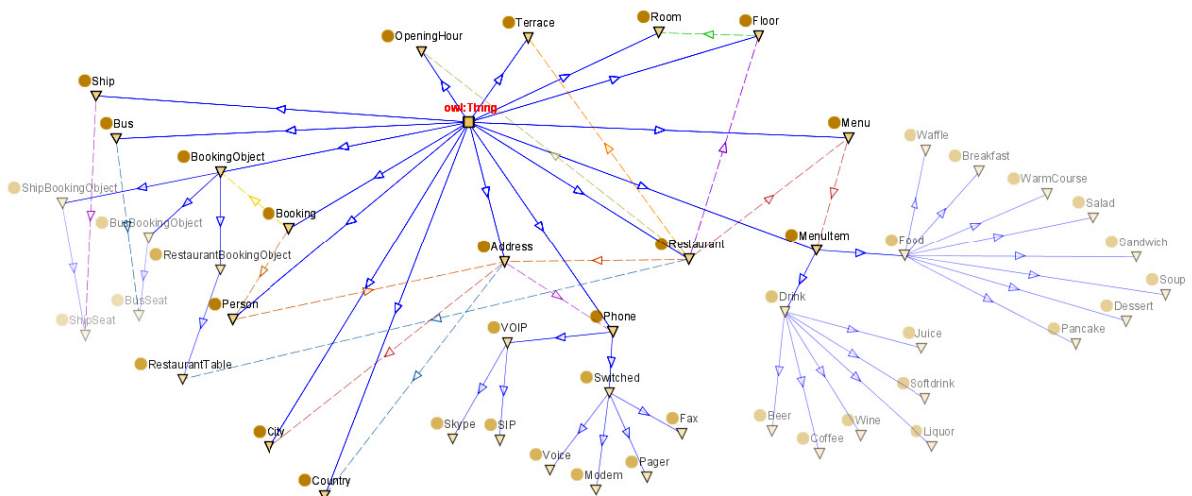


Abbildung 5.4: Nachbarschaftsgraph der Ontologie der Booking Application

Alle Klassen erben von der allgemeinen Klasse *owl:Thing*, ersichtlich in Abbildung 5.5, S. 44. Aus dieser Abbildung geht außerdem hervor, dass Vererbung nur bedingt eingesetzt wird. Begründet wird dieser Punkt zum Einen durch unterschiedliche Konzepte, die verschiedene

Klassen fordern und zum Anderen wird die Verifikation der Ontologie durch eine tiefe Hierarchie nicht begünstigt. Es sind mehr Iterationen vonnöten, um die Ontologie zu überprüfen. Diese Performanceeinbuße ist jedoch weniger gewichtig, da die Ontologie nur bei einer Modelländerung überprüft werden muss, was in diesem Fall kaum vorkommen wird, da es sich um eine abgeschlossene Wissensdomäne handelt.

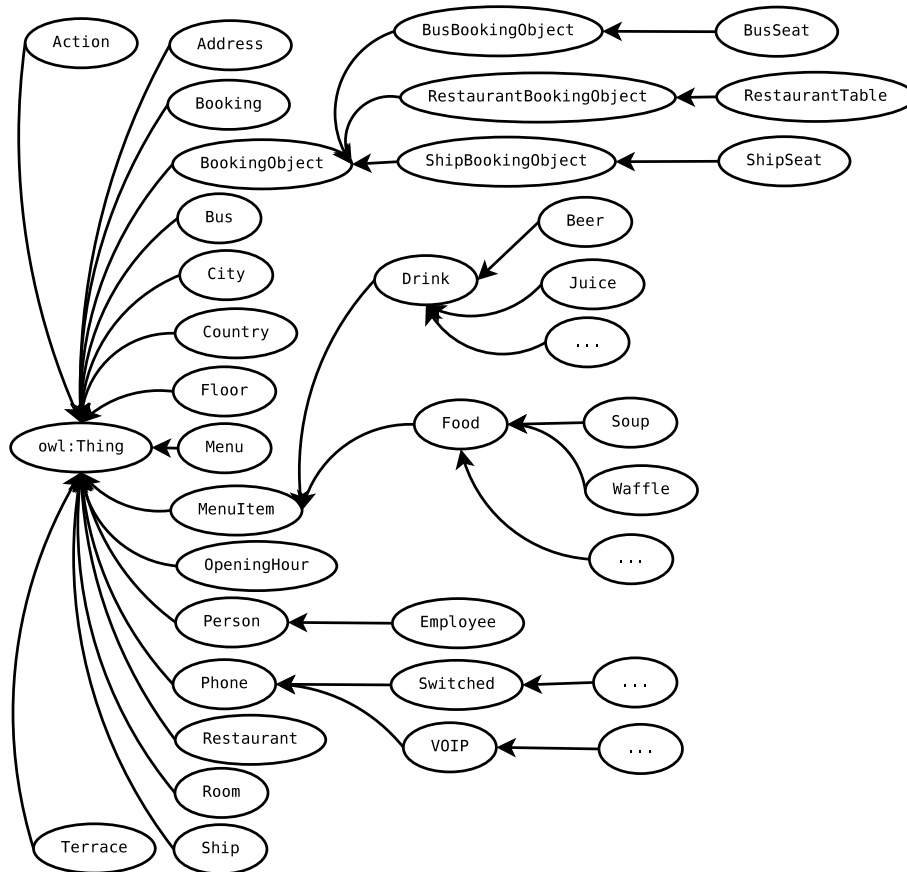


Abbildung 5.5: Vererbungsgraph der Ontologie der Booking Application

Eine der wichtigsten Klassen ist *BookingObject*. Diese Klasse besitzt die Eigenschaften *startDate*, *endDate*, *preference* und *bookingState*. Alle Kindklassen von *BookingObject* erben ebendiese Eigenschaften. Darüber hinaus ist eine tiefgreifendere Spezifikation zur Differenzierung der unterschiedlichen Buchungsmöglichkeiten notwendig. Angenommen ein Kunde bucht eine Busfahrt, dann muss das System so beschaffen sein, dass es alle dazu notwendigen Informationen (und nicht alle möglichen) aus dem Modell abfragt. In diesem Fall ist die Frage nach der Anzahl der Sitze interessant, wobei diese nach keinen weiteren Kriterien unterschieden

werden. Dagegen ist bei einem Restaurant, wenn ein Tisch gebucht wird, die Frage wichtig, ob der Tisch ausreichend Stühle besitzt. Durch die Differenzierung der Buchungsobjekte lässt sich das Modell problemlos erweitern, ohne das bereits bestehende Daten inkonsistent werden. Wenn ein Restaurant beispielsweise eine ganze Terrasse, unabhängig von der Anzahl der Tische und Stühle, verbuchen will, dann wird im Modell ein neue Kindklasse von *RestaurantBookingObject* abgeleitet, die als Eigenschaft einen Verweis auf die zu buchende Terrasse, sowie zusätzlich benötigte Informationen enthält. Diese Erweiterungsmöglichkeit ist nicht nur für Klassen und Eigenschaften, sondern auch für Beziehungen gegeben. In einer klassischen Webanwendung gestaltet es sich schwierig, bestehende Daten konsistent zu halten und dennoch Beziehungsänderungen umzusetzen, die meist mit einer Tabellenänderung des Datenbankschema einhergehen. Durch die Speicherung des Modells per Triple Store wird die Konsistenz der eigenen Daten abgesichert, unter der Voraussetzung, dass das Modell weiterhin konsistent ist. Mit der Frage nach der Konsistenz des Modells befasst sich der Abschnitt 5.3.2, S. 46.

Der folgende Quelltextauszug dient als Beispiel für die Notation einer Klasse innerhalb des Modells der Booking Application. In Listing 5.1 wird der Quelltext für die Klasse *Employee* aus der Ontologie der Booking Application gezeigt. *owl:Class* besagt, dass es sich um eine OWL Klasse handelt, mit dem Namen *Employee*. Sie besitzt Anzeichnungen für die Sprachen Englisch und Deutsch. Annotiert sind die Sprachen in diesem Fall per *xml:lang*. *rdfs:subClassOf* bezogen auf *Person* besagt, dass *Employee* eine Unterklasse von *Person* ist. Somit werden alle Eigenschaften von *Person*, wie Vorname, Nachname, Anschrift, Geburtsdatum usw. auf *Employee* übertragen.

```
1 <owl:Class rdf:about="#Employee">
2   <rdfs:label rdf:datatype="xsd:string">Employee</rdfs:label>
3   <rdfs:label xml:lang="en">employee</rdfs:label>
4   <rdfs:label xml:lang="de">Angestellter</rdfs:label>
5   <rdfs:subClassOf rdf:resource="#Person"/>
6 </owl:Class>
```

Listing 5.1: OWL Klasse Employee aus der Ontologie der Booking Application

5.3.1 Werkzeuge zur Modellierung einer Ontologie

Bei einfachen Ontologien mit nur wenigen Klassen, Eigenschaften und simplen Beziehungen genügt ein einfacher Texteditor zur Beschreibung der Ontologie. Mit zunehmender Komplexität geht die Übersicht bei einer solchen Modellierungsweise verloren. Um diese zu wahren, bietet sich die Verwendung eines grafischen Editors an.

Protégé³ ist der verbreitetste und bekannteste Ontologieeditor. Er wurde mit Java entwickelt. Darüber hinaus finden sich viele nützliche Erweiterungen für diesen Editor. So kam beispielsweise zur Erzeugung der Nachbarschaftsgraphen 5.4, S. 43 das Jambalaya-Plugin⁴ zum Einsatz.

SWOOP⁵ wurde ursprünglich von Mindswap entwickelt. Inzwischen wurde es an Google weitergereicht. Es ist ebenfalls eine Java Anwendung. SWOOP zeichnet sich durch hohe Benutzer- und Einsteigerfreundlichkeit, sowie gute Übersicht aus. Es wurde zur Erstellung der Ontologie der Booking Application eingesetzt und mit den integrierten Inferenzmaschinen wurde diese dann getestet.

OntoWiki (vgl. Abschnitt 2.3, S. 12)⁶ wird in der Booking Application eingesetzt, um nach Daten zu suchen und fehlende Informationen zu vervollständigen. Außerdem wurde es bei der Entwicklung der Buchungsanwendung dazu verwendet, noch nicht geschriebene Funktionen, hinsichtlich des Informationsflusses und -bedarfs zu testen. Weiterhin ermöglicht dieses Werkzeug eine umfassende Nutzerorganisation, wie in 5.4.2, S. 50 erwähnt wird.

5.3.2 Verifikation einer Ontologie

Nachdem eine Ontologie erstellt, verändert oder erweitert wurde, muss sie auf logische Konsistenz, respektive Validität überprüft werden. Neu eingeführte Beziehungen können unter

³<http://protege.stanford.edu/> [Stand: 21.03.2008]

⁴<http://webhome.cs.uvic.ca/~chisel/projects/jambalaya/jambalaya.html> [Stand: 21.03.2008]

⁵<http://code.google.com/p/swoop/> [Stand: 21.03.2008]

⁶<http://ontowiki.net/Projects/OntoWiki/> [Stand: 21.03.2008]

Umständen alte Beziehungen ungültig machen. Neue Klassen und Eigenschaften könnten bei bereits existierenden Abhängigkeiten andere Probleme hervorrufen. Aus diesem Grund muss eine Ontologie nach einer Veränderung auf logische Gültigkeit geprüft werden. Diese Verifikation übernimmt ein Algorithmus, der verantwortlich ist für die Kategorisierung, die Erfüllbarkeitsprüfung, die Konsistenzprüfung und die Instanzprüfung. Die meisten dieser Algorithmen basieren intern auf dem Tableau-Kalkül. Einige bekannte Algorithmen zur Verifikation einer Ontologie sind [18]:

Pellet⁷ ist die einzige Inferenzmaschine, die ganz OWL DL unterstützt. Dieser Reasoner wurde bei der Entwicklung der Booking Application eingesetzt.

KAON2⁸ bietet ausser der Inferenzmaschine noch weitere Funktionen an, wie ein Modul zur Extraktion von Ontologie Instanzen aus einer relationalen Datenbank.

FaCT++⁹ legt Wert auf Performance. Dazu wird FaCT++ mit C++ entwickelt.

RacerPro¹⁰ ist ein performanter kommerzieller Reasoner.

5.4 Eingesetzte Techniken und Technologien

Der folgende Abschnitt stellt die Techniken und die Technologien vor, die zur Entwicklung der Booking Application verwendet werden. Aus der Vielzahl verfügbarer Webentwicklungssprachen (.NET ASP¹¹, JSP¹², Perl¹³ und Weitere) fiel die Wahl auf **PHP: Hypertext Preprocessor (PHP)**¹⁴. Begründet wird diese Wahl mit der weiten Verbreitung von PHP, sowie seiner einfachen Programmierweise und der großen, aktiven Gemeinschaft, die für zahlreiche Erweiterungen sorgt. Zu diesen technologischen Erweiterungen zählt, dass in der Booking Application verwendete Zend-Framework¹⁵, welches im Paragraph 5.4.1, S. 50 kurz vorgestellt

⁷<http://pellet.owldl.com/> [Stand: 21.03.2008]

⁸<http://kaon2.semanticweb.org/> [Stand: 21.03.2008]

⁹<http://owl.man.ac.uk/factplusplus/> [Stand: 21.03.2008]

¹⁰<http://www.racer-systems.com/> [Stand: 21.03.2008]

¹¹<http://www.asp.net/> [Stand: 05.04.2008]

¹²<http://java.sun.com/products/jsp/index.jsp> [Stand: 05.04.2008]

¹³<http://www.perl.org/> [Stand: 05.04.2008]

¹⁴<http://www.php.net> [Stand: 05.04.2008]

¹⁵<http://framework.zend.com/> [Stand: 05.04.2008]

wird. Das Entwicklerteam hat sich für dieses Framework entschieden, da es in der ebenfalls verwendeten Erfurt-API zum Einsatz kommt. Darüberhinaus ist bei den Konkurrenten (CakePHP¹⁶, Symfony¹⁷, eZ Components¹⁸) von Zend die Integration und die Dokumentation schlechter [16].

Als Semantic Web Bibliothek kommt die Erfurt-API zum Einsatz. Sie wird im Abschnitt 5.4.1, S. 49 vorgestellt. Diese API wird vom Lehrstuhl für Betriebliche Informationssysteme am Institut für Informatik der Universität Leipzig entwickelt. Die Erfurt-API ist eine der ersten Semantic Web APIs für PHP.

5.4.1 Anfragemöglichkeiten

Durch RDF, RDFS und OWL werden Informationen maschinenlesbar strukturiert und miteinander verknüpft. Um von dieser Modellierung zu profitieren, folgen nun Anfrage- und Zugriffsmöglichkeiten, denn ohne diese wäre das Modell im Praxisfall nicht verwendbar.

In der Praxis stellt sich fest, dass durch die formale Semantik von RDF nur schwache Formalismen für Anfragen zur Verfügung gestellt werden. Damit werden nur simple Anfragen ermöglicht. So fehlen gänzlich Mengeneinschränkungen und Filteroptionen. Beispielsweise ist es nicht möglich, alle Kunden die eine Busreise am 24.07.2008 gebucht haben und Nichtraucher sind zurückgeben zu lassen (mit den bisher zur Verfügung gestellten Mitteln). Bisher lässt sich die Anfrage nur so stellen, dass alle Antworten zurückgegeben werden, also auch nicht Notwendige. Da sich dies als nicht praktikabel erweist werden Erweiterungen benötigt.

Die kommenden beiden Unterabschnitten klären, wie Anfragen und Datenänderungen innerhalb der Buchungsanwendung umgesetzt werden. Die Erfurt-API wird eingesetzt, um einfache Anfragen, sowie sämtliche Schreibzugriffe zu realisieren. Alle komplexeren Anfragen¹⁹ werden mit SPARQL ausgeführt.

¹⁶<http://www.cakephp.org/> [Stand: 06.04.2008]

¹⁷<http://www.symfony-project.org/> [Stand: 06.04.2008]

¹⁸<http://ez.no/ezcomponents> [Stand: 06.04.2008]

¹⁹Unter komplexe Anfrage wird verstanden, dass die erwartete Ergebnismenge größer als eins ist und eventuell gefiltert wurde.

SPARQL Einsatz in der Booking Application

```
1 PREFIX ns1: <http://booking.aksw.org#>
2 PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
3
4 SELECT DISTINCT ?personUri
5 WHERE ?personUri rdf:type ns1:Employee
6 LIMIT = 10
7 OFFSET = 20
```

Listing 5.2: SPARQL Beispiel

Aus diesem Quellcodeauszug gehen die grundlegenden Sprachkonstrukte von SPARQL hervor. Der *PREFIX* legt die verwendeten Namensräume fest. Hierbei dient *ns1* als Abkürzung für die URI, die sich hinter dem Doppelpunkt befindet. Der PREFIX *rdfs* stellt die Operationen von RDFS zur Verfügung.

SELECT definiert das Ausgabeformat der Ergebnismenge in Form konkreter Werte. Alternative Ausgabeformate werden entweder mit *ASK*, mit *CONSTRUCT* oder mit *DESCRIBE* erzeugt.

Variablen werden mit *?* zu Beginn der Zeichenfolge gekennzeichnet. Die *WHERE* Klausel beinhaltet die eigentliche Anfrage in Form eines oder mehrerer Tripel Statements, die in der bereits erwähnten Subjekt-Prädikat-Objekt Notation festgehalten wird.

Zur Einschränkung der Ausgabemenge wird *LIMIT* angegeben, was die maximal wiederzugebende Trefferanzahl widerspiegelt. *OFFSET* setzt eine Sprungmarke, ab wann die eigentliche Ergebnismenge ausgegeben werden soll.

In diesem Beispiel wird also nach allen Personen gefragt, die zur Klasse Employee gehören. Dabei sollen maximal 10 Personen, begonnen ab dem 20. Angestellten, zurückgegeben werden.

Erfurt-API

Unter diesem Begriff verbirgt sich eine Semantic Web Bibliothek, die Funktionen, Zugriffsmöglichkeiten und Bearbeitungsmethoden zur Gestaltung einer semantischen Webanwendung

anbietet. Die API ist geschrieben in der Skriptsprache PHP und somit konzipiert für PHP Anwendungen. In die Erfurt-API wurden das Zend Framework und RAP ²⁰ (RDF API for PHP) integriert. Darüber hinaus implementiert die Erfurt-API einen SPARQL-Endpoint ²¹.

Das Zend Framework beinhaltet eine Sammlung von Helferklassen und -methoden zur Gestaltung einer PHP-basierten Webanwendung. Darüber lässt sich mit diesem Framework eine MVC2 Struktur innerhalb eines Projekts festlegen. Die von Zend verwendete MVC2 Struktur wurde bei der Erfurt-API und bei der Booking Application verwendet. Weiterhin werden die Module *Zend_Session*, *Zend_Config*, *Zend_Controller*, *Zend_Exception*, *Zend_Feed*, *Zend_Json* und *Zend_Loader* innerhalb der Booking Application verwendet [43].

RAP stellt, als Semantic Web Open-Source Projekt, Programmierschnittstellen zur Manipulation von RDF-Graphen bereit.

5.4.2 Zugriffskontrolle

Die Booking Application stellt unterschiedlichen Benutzergruppen verschiedene Funktionen zur Verfügung. Zur Sicherung der Integrität und der Vertraulichkeit verwendeter Informationen ist es notwendig, ein Konzept zur Zugriffskontrolle umzusetzen. Zugriffskontrolle lässt sich mit folgender Frage erfassen:

„**Wer** darf in einem IT-System auf **welche** Ressourcen **wie** zugreifen?“[4]

Zugriffskontrolle innerhalb der Buchungsanwendung wird nicht über ein Rollenkonzept realisiert, sondern über eine Benutzerrechtenliste. Diese Liste ist ein Bestandteil von OntoWiki und beinhaltet Informationen über die Zugriffsoptionen der einzelnen Nutzer. Mögliche Zugriffsoptionen, die zugesichert oder verweigert werden sind:

- lesender Zugriff - Statements aus dem Modell können gelesen werden
- schreibender Zugriff - schreiben, löschen und hinzufügen wird zusätzlich ermöglicht

²⁰<http://sites.wiwiwiss.fu-berlin.de/suhl/bizer/rdfapi/> [Stand: 13.02.2008]

²¹<http://esw.w3.org/topic/SparqlEndpointDescription> [Stand: 13.02.2008]

Alle ausführbaren Aktionen der Booking Application sind im Modell verankert. Dadurch wird aktionsbasierte Zugriffskontrolle ermöglicht. Vor der Durchführung einer Aktion wird für den aktuellen Nutzer also immer geprüft, ob er die benötigten Rechte zur Ausführung der Aktion besitzt. Die Funktion *register new user* darf jeder Benutzer ausführen. Diese Funktion erhält also lesenden und schreibenden Zugriff. Dagegen muss bei der Funktion *delete booking* geprüft werden ob entsprechende Rechte existieren. Ein normaler Nutzer der nicht dieser Buchung zugeordnet ist, wird der lesende Zugriff darauf untersagt. Der Nutzer, der der Buchung zugeordnet ist erhält den lesenden Zugriff, nicht aber den schreibenden. Diesen bekommt erst ein Administrator, der dem Kunden zugeordnet ist und mögliche ausstehende Forderungen prüfen muss bevor er die Aktion durchführt.

5.5 Scrum Einsatz zur Entwicklung der Booking Application

Der Product Backlog für die Booking Application wurde vom Auftraggeber und Product Owner Cerial Jacobs erstellt. Das Team besteht aus sechs Mitgliedern: Jens Lehmann, Michael Martin, Aron Schneider, Ruslan Masold, Tom Weiland und Martin Weise. Die Rolle des ScrumMasters wurde nicht explizit vergeben, sondern indirekt durch Jens Lehmann besetzt. Die Umsetzung von Scrum sah in der Booking Application wie folgt aus:

- Es fanden im 14-tägigen Abstand Treffen statt, die einem Sprint entsprechen. Hierbei wurden Fragen hinsichtlich erreichter Punkte und existierender Probleme gestellt. Jedes Mitglied diskutierte Punkte, die es bis zum nächsten Treffen zu bewältigen galt.
- Cerial Jacobs hat permanenten Zugriff auf das SVN Projektarchiv. Darüber hinaus besucht er die Entwicklergruppe einmal im Quartal, um Rückmeldung vor Ort zu geben. Zusätzliche Rückmeldung liefert er an Jens Lehmann per Email oder Skype zur Sicherstellung der Einhaltung der Anforderungen. Da Cerial Jacobs selbst Informatiker ist, verfügt er über das benötigte Wissen, um eindeutige Rückmeldungen zu liefern.
- Ab November 2007 kam ein wöchentliches Treffen hinzu bei dem Ruslan Masold, Tom Weiland und Aron Schneider teamweise programmierten. Diese Treffen erwiesen sich

als sehr produktiv. Ursprünglich wurden Rollen für die Backend Programmierung, Controller Design und Frontend Erstellung festgelegt. Durch das gemeinsame Programmieren wurde der Spezialisierung entgegengewirkt. Verzögerungen durch Quelltextfehler eines anderen Mitarbeiters konnten maßgeblich reduziert werden.

- Es existiert kein starres Berichtswesen, kein Pflichten- oder Lastenheft. Cerial Jacobs dokumentierte Anforderungen auf wenigen Seiten in einem Wiki. Durch die Verwendung des Wikis konnten direkt Fragen, Änderungen und zusätzliche Informationen eingebunden werden.
- Die Verwendung eines Sprint Backlog, einer Arbeitstafel oder der Einsatz des Burn-down Diagramms fehlte bei der Entwicklung. Dadurch wurden Problemstellen schlechter erkannt und behandelt.
- Unter der Berücksichtigung, dass es sich bei dem Entwicklerteam fast ausschließlich um Studenten handelt, ist es plausibel, dass tägliche Treffen zwar produktiver gewesen wären, aber über einen solche Zeitraum (April 2007 - März 2008) nicht möglich waren.
- Da der Auftraggeber Cerial Jacobs aus den Niederlanden kommt, war eine bessere Projekteinbindung seinerseits schlecht möglich.

Fazit Zu Beginn des Projektes wurde sich nur auf eine generelle agile Vorgehensweise geeinigt, da sie für die Entwicklung der Booking Application am angemessensten ist (vgl. Abschnitt 3.3, S. 20). Die agile Vorgehensweise ermöglichte die zügige Entwicklung des Prototypen. Die Scrum spezifischen Eigenschaften wurden erst im Projektverlauf eingeführt. Um der Evaluation vorrauszugreifen, sei an dieser Stelle angemerkt, dass sich Scrum zur Erstellung einer Semantic Web Anwendung eignet. Ein zu überprüfender Erwartungswert wäre noch, in welchem Umfang eine Produktivitätssteigerung durch die komplette Umsetzung des Vorgehensmodells erreicht wird.

6 Evaluation

In diesem Kapitel soll, sowohl der Einsatz von Scrum, zur Entwicklung einer Semantic Web Applikation (vgl. Abschnitt 6.3, S. 56), als auch die Qualität der erzeugten Semantic Web Anwendung, unter verschiedenen Aspekten bewertet werden (vgl. Abschnitt 6.2, S. 55), die im ersten Abschnitt dieses Kapitels näher vorgestellt werden (vgl. Abschnitt 6.1, S. 53). Anschließend folgt, im letzten Abschnitt (vgl. 6.4, S. 57), eine Auseinandersetzung mit den Problemen, die während der Entwicklung der Booking Application auftraten.

6.1 Vorstellung der Evaluationskriterien

Das Hauptevaluationskriterium der Vorgehensmodelle ist die Wirtschaftlichkeit [12]. Dabei muss auf die Frage eingegangen werden, inwiefern das erzeugte Softwareprodukt, unter Benutzung aller zur Verfügung stehenden Ressourcen kostenoptimal und dennoch korrekt und vollständig, sowie qualitativ hochwertig erzeugt wurde.

Zur Feststellung der Wirtschaftlichkeit, wird eine Evaluation der Semantic Web Applikation benötigt. Dazu werden folgende Werte benutzt [20]:

Funktionalität beinhaltet *Angemessenheit, Richtigkeit, Interoperabilität, Ordnungsmäßigkeit* und *Sicherheit*.

Zuverlässigkeit beinhaltet *Fehlertoleranz* und *Wiederherstellbarkeit*.

Benutzbarkeit beinhaltet *Verständlichkeit, Erlernbarkeit, Bedienbarkeit* und *Attraktivität*.

Effizienz beinhaltet *Zeitverhalten* und *Verbrauchsverhalten*.

Die **Angemessenheit** einer Funktionalität spiegelt das Aufwand-Nutzen-Verhältnis wider. Mithilfe einer Inspektion lässt sich feststellen, ob eine Funktionalität ihre Aufgabe angemessen erfüllt oder ob sie zuviel oder zuwenig leistet.

Die **Richtigkeit** einer Funktionalität lässt sich testgetrieben oder durch statische Analyse validieren.

Interoperabilität beinhaltet Browser- und Plattform-Kompatibilitätstests. Darüber hinaus werden hier verschiedene Seiten der Anwendung auf unterschiedlichen Browsern und Systemen gedruckt und verglichen.

Ordnungsmäßigkeit beschreibt die Richtigkeit unter dem Aspekt der Interoperabilität. Sie wird testgetrieben und durch Browser- und Plattform-Kompatibilitätstests überprüft.

Die **Sicherheit** einer Webanwendung kann zum entscheidenden Erfolgsfaktor werden. Eine Webanwendung, die ihren Nutzern nicht die gewünschte Sicherheit offeriert, wird scheitern. Um die Sicherheit zu testen, werden verschiedene Attacken auf das System durchgeführt und analysiert.

Die **Fehlertoleranz**, im Hinblick auf Zuverlässigkeit wird durch Stresstests, Ausfalltests und Wiederanlaufstests überprüft.

Die **Wiederherstellbarkeit** wird ebenso getestet. Sollte die Hardware einmal ausfallen, dann muss sichergestellt werden, dass das System nach einem Neustart wieder funktioniert und das möglichst ohne einen Datenverlust erlitten zu haben.

Die Werte **Verständlichkeit**, **Bedienbarkeit** und **Erlernbarkeit** einer Webanwendung sind klare softwareergonomische Herausforderungen, die sowohl von Laien, als auch von Experten getestet werden sollten. Im Gegensatz dazu ist die **Attraktivität** einer Webanwendung subjektiv. Aus diesem Grund wird sie in dieser Ausarbeitung nicht weiter von Bedeutung sein.

Das **Zeitverhalten** einer Webanwendung wird durch Last- und Stresstests überprüft.

Das **Verbrauchsverhalten** berücksichtigt die Ladezeiten der Webanwendung. Diese können durch eine permanente Überwachung geprüft werden.

6.2 Bewertung der Semantic Web Applikation

Unter Berücksichtigung der gerade eingeführten Evaluationskriterien folgt, in diesem Abschnitt, die Bewertung der Semantic Web Applikation am Beispiel der Booking Application (vgl. Tabelle 6.1, S. 59).

Da sich diese Anwendung noch im Prototypenstatus befindet, macht es an dieser Stelle wenig Sinn, semantische Aspekte in die Evaluation mitaufzunehmen, da die zu testenden Elemente noch nicht weit genug entwickelt wurden.

Die Hauptbewertungskriterien für eine Semantic Web Applikation sind Erweiterbarkeit und Maschinenlesbarkeit. Bezüglich des ersten Kriteriums erweist sich das Modell der Booking Application als extrem flexibel. Neu eingeführte Klassen oder Eigenschaften, sowie sonstige Änderungen am Modell werden durch eine Inferenzmaschine verifiziert. Dadurch sind keine weiteren Tests (hinsichtlich der Überprüfung der Datenkonsistenz und -integrität) vonnöten, die sicherstellen, dass die Daten, mitsamt dem zugrundeliegenden Datenbankschema, weiterhin valide sind. Die Verwaltung des Datenbankschemas erfolgt durch den Triple Store. So wird Arbeitsaufwand bei Änderungen und Erweiterungen gespart. Während der Entwicklungsphase der Booking Application kamen immer wieder neue Anforderungen hinzu, die eine Modelländerung nach sich zogen. Beispielsweise, wurde in der letzten Entwicklungsphase die Klasse *Action* eingeführt, die alle Aktionen der Booking Application so definiert, dass sie für eine Zugriffskontrollliste benutzbar sind. Diese Anforderung war im Vorfeld unklar, da keine konkrete Anforderung an einen solchen Mechanismus gestellt wurde. Der erste Ansatz dazu ging sogar von einem möglichen Rollenkonzept aus. Der Arbeitsaufwand, um die Klasse *Action* einzuführen, zu testen und die Aktionen zu definieren betrug acht Stunden, wobei die meiste Zeit davon für die Aktionsdefinition verwendet wurde. Bei einer konventionellen Webanwendung hätte eine solche Änderung ein vielfaches dieser Zeit gekostet, da zusätzlich die Datenbank hätte angepasst werden müssen und überdies wären sowohl Konsistenz-, als auch Integritätstests unabdingbar gewesen.

Das nachfolgende Testbeispiel soll die Überprüfung der Maschinenlesbarkeit darstellen. Eine Anfrage wie: *Gib mir alle Kunden zurück*, wäre auch ohne eine Verwendung von Semantic Web Technologien problemlos mit einer einfachen SQL-Anfrage möglich. Bedeutend komplexer wäre folgende Anfrage: *Gib mir die Freunde aller Kunden zurück, bei denen die Kunden über 10.000 Euro verbucht haben*. In diesem Beispiel wird davon ausgegangen, dass sich

zumindest einige Freunde innerhalb von FOAF und ähnlichen Diensten oder Communities auffinden lassen. Da die Booking Application keine Definition für Freunde enthält, muss entweder ein externer Dienst eingesetzt werden, der die benötigten Informationen liefert oder ein entsprechendes Vokabular (wie FOAF) wird dazu benutzt, das Modell dahingehend zu erweitern, dass sich solche Informationen pflegen und von Fremdquellen abfragen lassen. Das erspart die zeit- und kostenintensive Entwicklung eines Programms, welches die externe Datenstruktur, zur Weiterbearbeitung der Informationen, an die interne Datenstruktur anpasst. Die Booking Application verfolgt den zweiten Weg und verwendet verbreitete Vokabulare. Theoretisch werden durch diese Vorgehensweise Agenten bei der Informationsfindung und -extraktion unterstützt. Der Praxisfall muss für die Booking Application noch untersucht werden.

Fazit Die direkten Qualitätskriterien, für eine Webanwendung, werden von der Booking Application zum größten Teil erfüllt. Die kritische Problemstelle liegt in den langen Zugriffszeiten. Diese sollen zukünftig durch eine Weiterentwicklung der verwendeten APIs, sowie durch bessere SPARQL Anfragen optimiert werden.

Bei der Bewertung der semantischen Komponenten stellt sich heraus, dass das Modell die wichtigste Komponente einer Semantic Web Applikation ist. Es beeinflusst die beiden Hauptbewertungskriterien maßgeblich. Die Erweiterbarkeit der Anwendung wurde anhand von Modelländerungen nachgewiesen. Die Maschinenlesbarkeit kann aufgrund fehlender Überprüfung nicht bewertet werden.

6.3 Evaluation von Scrum zur Entwicklung einer Semantic Web Applikation

Ausgehend von den Ergebnissen aus Kapitel 4.3, S. 35 soll in diesem Abschnitt die Aussage überprüft werden, ob Scrum das passenste Vorgehensmodell zur Entwicklung einer Semantic Web Anwendung ist.

Dazu muss die Wirtschaftlichkeit des Verfahrens gemessen werden. Da sich die Booking Application noch im Prototypenstatus befindet, kann ihre Entwicklung noch nicht mit anderen

Entwicklungen verglichen werden. Aus diesem Grund soll dieser Abschnitt nur eine Vorstellung zu möglichen Testkriterien verschaffen.

Bei der Bewertung von Scrum sollte nicht nur die Teamgröße, sondern auch die technische Kompetenz seiner einzelnen Mitglieder berücksichtigt werden. Ein entscheidender Erfolgsfaktor ist die Anzahl der entwickelten Funktionspunkte (gemessen in Arbeitsstunden, die zur Umsetzung notwendig sind). Dabei sind jedoch die Qualität und Priorität der entwickelten Funktionen zu berücksichtigen. An dieser Stelle wirkt sich Scrum durch seine täglichen Sitzungen besonders positiv aus, denn somit sind Entwicklungen unnützer Funktionalitäten unmöglich. Desweiteren sind personelle Ausfälle, durch das Fehlen von Spezialisten, besser auszugleichen.

In einer zukünftigen Arbeit sollten vergleichbare Projekte mit den wichtigsten Vorgehensmodellen entwickelt und unter den gerade genannten Aspekten verglichen werden, um so ein objektives Bild, über die tatsächliche Wirtschaftlichkeit zu erhalten.

6.4 Probleme beim Projekt Booking Application

In diesem Abschnitt werden die Probleme aufgelistet, die während der Entwicklung der Booking Application zu Verzögerungen oder zu Folgeproblemen geführt haben. Sie sollen dabei helfen, die zukünftige Entwicklung einer Semantic Web Anwendung produktiver zu realisieren.

- Der allgemeine Erfahrungsstand der Programmierer war zu gering. Dadurch war die Produktivität des Teams unzureichend.
- Semantic Web Anwendungen erfordern ein größeres technisches Wissen und Verständnis, im Vergleich zu konventionellen Web Anwendungen. Dadurch ging viel Zeit für die Einarbeitung verloren, die nicht ausreichend eingeplant wurde.
- Der Chefentwickler hätte sich zu Projektbeginn mehr beteiligen müssen, um eine anfängliche Fehlentwicklung zu vermeiden. Dieser war jedoch zu sehr in andere Projekte involviert.
- Der Zeitaufwand der Programmierer war in der Anfangsphase zu gering. In den letzten Monaten des Projekts, bis zum Ausscheiden von zwei Entwicklern, wurde zumindest ein

Programmiertreffen pro Woche über sechs Stunden durchgeführt (zusätzlich zur eigenen Programmierarbeit). Diese Programmiertreffen, die zu Projektbeginn unterschätzt wurden, stellten sich als sehr produktiv heraus.

- Aufgrund der mangelhaften Kommunikation, während der Anfangsphase des Projekts, kam es zu Verzögerungen durch fehlende Rückmeldungen.
- Die Rollenbildung führte zu zusätzlichen Verzögerungen, da sie für diese geringe Projektgröße zu ausgeprägt war. Traten Fehler in fremden Programmbereichen auf, war die Fehlerbeseitigung umständlich, so dass meistens auf die Fehlerbehebung durch den Verursacher gewartet werden musste.
- Die Projektmitglieder waren unzureichend mit dem Projekt als Ganzes vertraut.
- Die Kernkomponenten hätten testgetrieben entwickelt werden sollen, um vor allem die häufigen Änderungen der Frameworks schneller und besser analysieren und mögliche Probleme stärker eingrenzen zu können.
- Der Bug Tracker kam zu spät zum Einsatz. Dadurch wurde zuviel Zeit für die Fehleranalyse verwendet.
- Die Rolle des Kunden war zu klassisch ausgeprägt, d.h. er wurde nicht ausreichend in das Projekt einbezogen. Die Abstände der Kundenrückmeldung waren zu groß. Darüber hinaus wurden anfangs falsche Prioritäten, der zu verwirklichenden Funktionalitäten gesetzt.
- In den regelmäßig stattfindenden Teamsitzungen wurde nicht immer ausführlich genug auf die aufgetretenen Probleme und ihre Lösungsansätze eingegangen.
- Der Einsatz der Erfurt-API zog Probleme und damit auch einen Zeitverlust nach sich. Denn dadurch, dass sich diese API noch in der Entwicklung befand, änderten sich Kernkomponenten und Schnittstellen noch. Diese Änderungen waren nicht direkt für die Entwickler der Booking Application dokumentiert, wodurch Zeit für Fehler- und Problemsuche geopfert werden musste.

Tabelle 6.1: Evaluation der Booking Application

<i>Kriterium</i>	<i>Unterkriterium</i>	<i>Testverfahren und Ergebnis</i>
Funktionalität	<i>Angemessenheit</i>	wurde regelmäßig vom Auftraggeber überprüft - im Fall eines Fehlers gab es eine entsprechende Rückmeldung
	<i>Richtigkeit</i>	wurde durch das Team getestet - Funktionen, die Richtigkeit bewiesen haben wurden protokolliert
	<i>Interoperabilität</i>	Mozilla Firefox und Microsoft Internet Explorer wurde zum testen benutzt - manche YUI-Skripte funktionierten mit dem MSIE nicht
	<i>Ordnungsmäßigkeit</i>	wurde regelmäßig vom Auftraggeber überprüft - im Fall eines Fehlers gab es eine entsprechende Rückmeldung
	<i>Sicherheit</i>	URL-Rewriting, Sessions, Logging und Zugriffskotrollliste wurden überprüft - hier müssen mehr Tests erfolgen
Zuverlässigkeit	<i>Fehlertoleranz</i>	Wiederanlauf und Ausfall wurden durch erzwungene Neustarts getestet - für den Praxiseinsatz fehlt noch der Stresstest
	<i>Wiederherstellbarkeit</i>	erzwunger Neustart - bestanden
Benutzbarkeit	<i>Verständlichkeit</i>	Mehrsprachigkeit, Verwendung eindeutiger Bezeichner - bestanden bei Befragung von potentiellen Nutzer, die über keine Fachkompetenz verfügten
	<i>Erlernbarkeit</i>	bisher ungetestet
	<i>Bedienbarkeit</i>	Hilfefunktion, Icons, alternative Kontextmenüs per Rechtsklick in den Tabellen, zur Beschleunigung der Navigation - wurde bei Befragung von Nutzern als gut empfunden
	<i>Attraktivität</i>	ungetestet, da subjektiv - Themes und Anpassungsoptionen waren in Planung
Effizienz	<i>Zeitverhalten</i>	Lasttests - bisher ungenügend
	<i>Verbrauchsverhalten</i>	Testanfragen - bisher ungenügend

7 Diskussion

Im abschließenden Kapitel dieser Arbeit wird eine Zusammenfassung gezogen (vgl. Abschnitt 7.1, S. 60) und ein Ausblick (vgl. Abschnitt 7.2, S. 61) geliefert.

7.1 Zusammenfassung

Diese Diplomarbeit wurde begonnen mit der Motivation der Verwendung von Vorgehensmodellen im Zusammenhang mit der Entwicklung einer Semantic Web Applikation, wobei die Daseinsberechtigung derselbigen ebenfalls dargestellt wurde (vgl. Kapitel 1, S. 1). Hierbei wurde festgestellt, dass das Semantic Web zur weiteren Informationsvernetzung unserer Gesellschaft entscheidend beitragen kann. Dazu müssen Anwendungen systematisch entwickelt werden, welche die Techniken und Technologien des Semantic Web nutzen.

Die aktuellen Technologien und bereits existierenden Anwendungen wurden in Kapitel 2, S. 4 kurz vorgestellt.

Die bereits erwähnte systematische Entwicklung führt zum Web-Engineering, welches die Verwendung eines Vorgehensmodells vorsieht. Im dritten Kapitel, S. 13 wurden verschiedene Arten von Vorgehensmodellen charakterisiert und anschließend vergleichend betrachtet.

Aus diesem Vergleich trat das agile Vorgehensmodell Scrum, unter dem Kriterium ein Vorgehensmodell für eine Semantic Web Applikation zu sein, hervor. Aus diesem Grund wurde Scrum im vierten Kapitel, S. 27 ausführlich, unter der Berücksichtigung der Besonderheiten, welche zur Entwicklung einer Semantic Web Anwendung notwendig sind, behandelt.

Der Anwendungsfall war die Entwicklung der Booking Application, welche in Kapitel 5, S. 37 vorgestellt wurde. Hierbei wurden, neben den eingesetzten Werkzeugen und Technologien, auch die zugrundeliegenden Anforderungen an das Endprodukt, sowie die Besonder-

heiten der Modellierung der Ontologie vorgestellt. Abgeschlossen wurde dieses Kapitel mit der Beschreibung des Einsatzes von Scrum, während der Entwicklung der Booking Application.

Kapitel 6, S. 53 setzte sich mit einer Evaluation der erzeugten Software, sowie der verwendeten Vorgehensweise, auseinander. Diese konnte jedoch nur im Ansatz, aufgrund fehlender Vergleichsmöglichkeiten, bewertet werden. Dieses Kapitel wurde durch eine Auflistung der Probleme, welche bei der Entwicklung der Booking Application auftraten, abgeschlossen. Aus den gewonnenen Erfahrungen wurden erste Ausblicke erzeugt.

7.2 Ausblick

Wenn zukünftige Entwicklungen geplant werden, ist zu berücksichtigen, dass der durchschnittliche Projektumfang entscheidend zunehmen wird¹. Um Scrum unter dieser Voraussetzung einzusetzen, muss es erweitert werden. Es funktioniert auch bei großen verteilten Teams. Europäische, US-amerikanische und japanische Unternehmen neigen dazu, Entwicklungsprozesse nach Osteuropa auszulagern. Dabei treten vorwiegend Kommunikationsprobleme auf, welche die Produktivität beeinträchtigen. Das Scrum Meeting muss alle verteilten Entwickler miteinbeziehen. Dazu treffen sich die beteiligten ScrumMaster in regelmäßigen Abständen, beispielsweise per Telekonferenz. Weiterhin wird ein Product Owner Team gebildet, das sich regelmäßig trifft und ein MetaScrum durchführt, um multiple, simultane Produktveröffentlichungen zu managen. Außerdem existiert ein zentrales Repository, welches automatisch stündlich aktualisiert wird. Der Einsatz eines Bug Tracker und eines Reporting Tool (zum Aufzeigen des aktuellen Projektstandes) ist unverzichtbar. [19] Die parallele Bearbeitung von Sprints ist eine Möglichkeit, die Produktivität eines größeren Teams zu steigern. Dazu überlappen sich die Sprints. Weiterhin existiert eine Liste von Anforderungen, die von einem Online-System verwaltet wird und jedem Mitarbeiter die Gelegenheit bietet einzusehen, was gerade in Bearbeitung ist und was als nächstes erledigt werden muss. [36] Ein Untersuchungspunkt einer zukünftigen Arbeit wäre, inwiefern sich diese Erfahrungen auch auf die Entwicklung von Semantic Web Applikationen übertragen lassen.

¹Diese Entwicklung war bereits bei nicht webbasierter Software zu beobachten. Begründet wird diese Umfangssteigerung durch das Entstehen neuer Anforderungen und der zur Realisierung benötigter Technologien.

Mit zunehmender Projektgröße gewinnt der Aspekt des Einsatzes von Web-Modellierungssprachen und -werkzeugen an Bedeutung. Im Gegensatz zu konventionellen Softwareentwicklungsbeschreibungssprachen berücksichtigen sie verschiedene Ebenen. So lassen sich die Ebenen Inhalt, Hypertext und Präsentation hinsichtlich ihrer Struktur und ihrem Verhalten getrennt voneinander modellieren. Hypertextmodellierung verfolgt das Ziel, die Erreichbarkeit der Seiten sicherzustellen und zu verbessern. Dazu wird geprüft, dass jede Seite mindestens eine weitere Seite referenziert und dass diese Referenz auch wirklich existiert. Eine Präsentationsebene ist wichtig, um eine Flexibilität des Erscheinungsbildes der Webapplikation zu gewährleisten. Gerade im heterogenen Umfeld der Webbrowser ist diese Flexibilität unabdingbar. Überdies ermöglicht eine Präsentationsebene eine schnelle Anpassung an kulturelle Gegebenheiten. Die rechtsbündige Ausrichtung islamischer Texte wäre, bei einer Expansion der Auftraggeberfirma in den arabischen Raum, leicht umzusetzen. Die Web-Modellierungssprachen basieren meist auf der UML und erweitern diese Sprache. Sie bieten Optionen zur Modellierung von Web-spezifischen Elementen und Ebenen an. Zu den Web-Modellierungssprachen gehören WebML und UML-based Web Engineering (UWE). Ein Werkzeug für WebML ist WebRatio². Für UWE bietet sich das Werkzeug OpenUWE³ an [29]. Ein Untersuchungspunkt in einer fortführenden Arbeit wäre also die Fragestellung, inwiefern sich die Verwendung einer Web-Modellierungssprache auf die Effizienz und Effektivität, sowie auf die Steigerung der Qualität der erzeugten Arbeit niederschlägt.

Weiterhin sollten Verfahren und Techniken zur Qualitätssicherung stärker berücksichtigt werden, da die Qualität einer Anwendung entscheidenden Einfluß auf ihren Erfolg hat. An dieser Stelle müssen Semantic Web spezifische Kriterien definiert und untersucht werden. Wie müssen diese bewertet werden? Wie gewichtig ist ihr Einfluß auf die Qualitätssicherung?

In diesem Zusammenhang sind Modellierungsweisen zur Erzeugung der Ontologie einer Semantic Web Applikation mit unterschiedlichen Herangehensweisen zu bewerten. Die größte Schwachstelle der Booking Application ist ihre erzielte Performance. Aus diesem Grund sollten Richtlinien erstellt werden, wie ein Modell vollständig, aber dennoch maximal performant gestaltet werden kann.

²<http://www.webratio.com/sv1.do> [Stand: 22.04.2008]

³<http://www.pst.informatik.uni-muenchen.de/projekte/uwe/openuwe.shtml>
[Stand: 22.04.2008]

Literaturverzeichnis

- [1] ANGELA MARTIN, ROBERT BIDDLE, J. N., Ed. *How do XP, Scrum and ASD build the right software?* (2003).
- [2] BECK, K. *Extreme Programming - Das Manifest*, 2 ed. Addison-Wesley, 2000.
- [3] BECK, K. *Extreme Programming Explained - Embrace Change*, 2 ed. Addison-Wesley, Februar 2005.
- [4] BERGER, S. Konzept und Implementierung eines Access Control Layers für RDF Triple Stores. Master's thesis, Uni Leipzig, Januar 2008.
- [5] BREU, R. *Software-Engineering - Objektorientierte Techniken, Methoden und Prozesse in der Praxis*, 1 ed. Oldenburg Verlag München Wien, 2005, pp. 192–278.
- [6] BULLINGER, H.-J. *Betriebliche Informationssysteme*. Springer, 1997.
- [7] CHRIS BIZER, TOM HEALTH, D. A. Y. R. Interlinking Open Data on the Web. URL: <http://sites.wiwiss.fu-berlin.de/suhl/bizer/pub/LinkingOpenData.pdf>, [Stand: 30.03.2008].
- [8] CHRISTOPH GSCHIEDLE, M. F. Online 2007: Das Mitmach-Netz im Breitbandzeitalter. URL: http://www.ard-zdf-onlinestudie.de/fileadmin/Online07/Online07_Multimedia.pdf, [Stand: 06.03.2008].
- [9] COHN, M. Writing Contracts for Agile Development. URL: http://www.mountangoatsoftware.com/article_view/5-writing-contracts-for-agile-development, [Stand: 10.03.2008].
- [10] COHN, M. SUCCEEDING WITH AGILE. URL: <http://www.mountangoatsoftware.com/>, [Stand: 28.04.2008].

- [11] COPLIEN, J. O., Ed. *Borland Software Craftmanship: A New Look at Process, Quality and Productivity* (Juni 1994), Proceedings of the 5th Annual Borland International Conference.
- [12] DUMKE, R. *Software Engineering*, 4 ed. Vieweg Verlag, 2003, pp. 169–206.
- [13] ECKERT, M. Die Entwicklung von Vista hat mehr als 10 Milliarden US-Dollar verschlungen. URL: <http://www.tecchannel.de/news/themen/windows/457070/>, [Stand: 28.04.2008].
- [14] EMERY, P. THE DANGERS OF EXTREME PROGRAMMING. URL: <http://members.cox.net/cobbler/XPDangers.htm>, [Stand: 26.02.2008].
- [15] FENZEL, D. *Spinning the Semantic Web - Bringing the World Wide Web to its Full Potential*, 1 ed. The MIT Press, 2003, pp. 1–25.
- [16] FRANZ, M. PHP-Frameworks: Das nächste Jahrtausend. *iX 9-2007* (September 2007), 56–58.
- [17] HASCHKE, M. Semantic Personal Knowledge Management. Master's thesis, Uni Leipzig, Februar 2008.
- [18] HITZLER, P. *Semantic Web - Grundlagen*, 1 ed. Springer-Verlag Berlin Heidelberg, 2008.
- [19] JEFF SUTHERLAND, ANTON VIKTOROV, J. B. N. P., Ed. *Distributed Scrum: Agile Project Management with Outsourced Development Teams* (2007), Proceedings of the 40th Hawaii International Conference on System Science - 2007.
- [20] KAPPEL, G. *Web Engineering - Systematische Entwicklung von Web Anwendungen*, 1 ed. dpunkt.verlag GmbH, 2004.
- [21] KARUN BAKSHI, D. R. K. Semantic Web Applications. URL: <http://people.csail.mit.edu/kbakshi/EndUserInteraction-final.pdf>, [Stand: 21.03.2008].
- [22] KENT BECK, E. A. Manifesto for Agile Software Development. URL: <http://agilemanifesto.org/>, [Stand: 19.02.2008].

- [23] KÖHLER, A. Web 2.0: Jetzt auch als Enterprise Edition? URL: http://lpzradar.informatik.uni-leipzig.de/uploads/reports/5_2008.pdf, [Stand: 21.03.2008].
- [24] MARTIN, M. Semantic Web Applications. Master's thesis, Uni Leipzig, April 2008.
- [25] MASOLD, R. Verwendung von Benutzeroberflächen mit Web 2.0 Technologien in einer Semantischen Webapplikation. Bachelorarbeit, Februar 2008.
- [26] MIKE COHN, D. F. Introducing an Agile Process to an Organization. URL: <http://www.mountaingoatsoftware.com/article/access/10-introducing-an-agile-process-to-an-organization>, [Stand: 10.03.2008].
- [27] O'REILLY, T. What is Web 2.0 - Design Patterns and Business Models for the Next Generation of Software. URL: <http://www.oreillynet.com/lpt/a/6228>, [Stand: 08.03.2008].
- [28] POWERS, S. *Practical RDF*, 1 ed. O'Reilly, 2003.
- [29] PRINZ, M. Modellgetriebene Entwicklung ubiquitärer Web-Anwendungen. Master's thesis, TU Wien, April 2006.
- [30] ROYCE, W. W. Managing the development of large software systems. *IEEE Computer Society Press* (1970).
- [31] SCHWABER, K. What is Scrum? URL: <http://www.scrumalliance.org/system/resource/file/275/whatIsScrum.pdf>, [Stand: 03.03.2008].
- [32] SEBASTIAN DIETZOLD, S. A., Ed. *Access Control on RDF Triple Stores from a Semantic Wiki Perspective* (Juni 2006), Proceedings of Scripting for the Semantic Web Workshop at the ESWC, Budva, Montenegro.
- [33] SOMMERVILLE, I. *Software Engineering* 8, 8 ed. Addison-Wesley, 2007, pp. 63–91.
- [34] SUTHERLAND, J. Agile Can Scale: Inventing and Reinventing SCRUM in Five Companies. *CUTTER IT JOURNAL* 14, 12 (2001), 7.
- [35] SUTHERLAND, J. agile development: lessons learned from the first scrum. Tech. rep., PatientKeeper, Oktober 2004.

- [36] SUTHERLAND, J., Ed. *Future of Scrum: Parallel Pipelining of Sprints in Complex Projects* (Juli 2005), Agile 2005 Conference.
- [37] SÖREN AUER, SEBASTIAN DIETZOLD, J. L. T. R., Ed. *OntoWiki - A Tool for Social, Semantic Collaboration* (August 2007), Proceedings of the WWW-07 Workshop on Social and Collaborative Construction of Structured Knowledge Neural-Symbolic Learning and Reasoning (CKC), 2007, Uni Leipzig.
- [38] THOMAS RIECHERT, KIM LAUENROTH, J. L. S. A., Ed. *Towards Semantic based Requirements Engineering* (September 2007), Proceedings of the 1th SABRE Conference on Social Semantic Web (CSSW).
- [39] TIM BERNERS-LEE, JAMES HENDLER, O. L. The semantic web. Tech. rep., Scientific American Magazine, Mai 2001.
- [40] VAN GULIK, D.-W. Believe It: The Semantic Web is Coming to Web Video and Joost is Leading the Way. URL: <http://www.beet.tv/2007/12/believe-it-sema.html>, [Stand: 28.04.2008].
- [41] VERSTEEGEN, G. *Softwaremanagement*. Springer, 2002, pp. 29–61.
- [42] VIMAL MAYANK, NATALYA KOSITSYNA, M. A. Requirements Engineering and the Semantic Web: Part II. Representation, Management, and Validation of Requirements and System-Level Architectures. Tech. rep., University of Maryland, Februar 2004.
- [43] WEILAND, T. Controller Design für Semantische Webanwendungen. Bachelorarbeit, März 2008.
- [44] WU, Z. Oracle Semantic Technologies Inference Best Practices with RDF-S/OWL. URL: http://www.oracle.com/technology/tech/semantic_technologies/pdf/semantic_infer_bestprac_wp.pdf, [Stand: 21.04.2008].

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe, insbesondere sind wörtliche oder sinn-gemäße Zitate als solche gekennzeichnet. Mir ist bekannt, dass Zuwiderhandlung auch nach-träglich zur Aberkennung des Abschlusses führen kann.

Leipzig, 29. April 2008

Aron Schneider