Rheinische Friedrich-Wilhelms-Universität Bonn
Institut für Informatik III

# Semantisches Mapping von Microsoft SharePoint

**Bachelorarbeit**

betreut von Prof. Dr. Sören Auer

vorgelegt von Daniel Weber
Matrikelnummer 2234867

Bonn, den 29. September 2014

# Eidesstattliche Erklärung

Hiermit versichere ich, die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie die Zitate deutlich kenntlich gemacht zu haben.

Ort, Datum

Unterschrift

# Inhaltsverzeichnis

# 1 Einleitung

Viele Unternehmen haben ein großes Interesse daran, ihre organisatorischen Prozesse und verschiedenste Inhalte in so genannten Enterprise-Content-Management (kurz ECM) Systemen zu bündeln. Allem voran das Ziel, das vorhandene Wissen im gesamten Unternehmen verfügbar zu machen. Mit zunehmender Größe wächst die Menge an Informationen. 70% der Unternehmen mit mehr als 5000 Mitarbeitern setzen zu diesem Zweck bereits Microsoft SharePoint ein [Mil11].

Die Mitarbeiterschaft stellt dabei selbst eine große dezentralisierte Wissensmenge dar. Den größten Mehrwert hätte man, wenn man das vorhandene Wissen z.B. in Form einer Knowledge Base gebündelt vorfinden könnte. Aufbau und Betrieb einer Knowledge Base bindet Expertenwissen und Ressourcen, die anderweitig dann nicht mehr verfügbar sind. Dabei stehen sich wohl definierte Geschäftsprozesse und das verteilte Mitarbeiterwissen konzeptuell gegenüber. Im Kern steht dabei der Geschäftsprozess, zu dessen Unterstützung ein ECM zwingend benötigt wird. Auf lange Sicht ist der Parallelbetrieb zweier so unterschiedlicher Systeme unwirtschaftlich. Wie also kann trotz Verzicht auf eine manuell gepflegte Knowledge Base das implizite Mitarbeiterwissen dennoch verfügbar gemacht werden?

Eine Antwort auf diese Frage lautet Semantisches Mapping: Bestehende Informationen werden zu maschinenlesbaren Aussagen transformiert, sodass es möglich ist, sie auf technischem Wege durchsuchbar zu machen und aufzubereiten.

## 1.1 Problemstellung

Gesucht ist ein semantisches Mapping für das gegebene System SharePoint, welches dessen Inhalte standardkonform als Linked Data exponiert. Als de facto-Standard zur Darstellung und Verarbeitung von Linked Data hat sich RDF etabliert.

Es muss also ein Verfahren entwickelt werden, welches eine große, komplexe Datenmenge des SharePoints durch automatische Verarbeitung als RDF-Datenquelle anbietet. Dazu muss zunächst geklärt werden, in welcher Form die Datenhaltung in SharePoint organisiert ist.

## 1.2 Motivation

Die Etablierung neuer technischer Standards ermöglicht Innovation. Deswegen bietet der Linked Data Ansatz eine neue unternehmensweite Platform zur Informationsintegration. Davon profitieren Mitarbeiter in ihrem Arbeitsumfeld unmittelbar. Das implizit vorhandene Unternehmenswissen wird zugänglich und einfacher findbar. Anders als ausschließlich dateissystembasierte Organisation von Dokumenten ist die Suche mithilfe von Indizierung und zentralen Datenbanken in ihrer Leistungsfähigkeit deutlich gesteigert. Bei einer klassischen Suche mit unscharfen Bedingungen leidet die Genauigkeit der Suchergebnisse immens. Linked Data kann auch ohne zusätzliche Strukturen das vorhandene Wissen erschließen.

Gewonnenes Wissen muss nicht aktiv verteilt werden, weil es bereits bei einer Suche im Informationsgraphen gefunden wird. Der automatisierte Ansatz spart Ressourcen: Keine Knowledge Base muss von spezialisierten Mitarbeitern aktuell gehalten werden. Außerdem kann Synergiepotential bei ähnlichen Problemstellungen identifiziert und ausgenutzt werden.

Eine Migration bestehender Geschäftsprozesse hin zu Linked-Data ist zu aufwendig. Wünschenswert ist der Erhalt bestehender Infrastruktur bei gleichzeitiger Einbindung semantischer Prinzipien.

Bisher gibt es keine Möglichkeit, SharePoints Inhalte als Linked Data zu betrachten. Die existierenden Erweiterungen und Schnittstellen sind für objektrelationale Anwendungen ausgelegt.

Da SharePoint einen SQL Server verwendet, liegt es nahe, SparqlMap zu untersuchen. Dieses Projekt bietet einen Software-Adapter an, der SQL-basierte Datenquellen als SPARQL Endpunkt darstellt.

## 1.3 Herausforderungen

Eine Herausforderung ist, eine Möglichkeit zu finden, auf SharePoints Metainformationen als Ausgangsbasis für das automatische Mapping zuzugreifen. Hierfür muss zunächst die Inhaltsverwaltung verstanden werden und die ihr zugrunde liegenden Konzepte, sodass man eine wohldefinierte Abbildungsvorschrift aufstellen kann.

Von Microsoft werden keine lokalen Demo-Installationen mehr zur Analyse der Metainformationen angeboten. Eine frische Installation ist aber leer, d.h. es liegen keine Beispieldaten vor, die man untersuchen könnte. Händisches Erzeugen von Beispieldatensätzen erfordert sehr viel Zeit, sodass eine Quelle für hinreichend komplexe Analyseobjekte gefunden werden muss.

Es soll der SparqlMap Ansatz verfolgt werden, um keine großen Datenmengen als RDF Triple exportieren zu müssen. Da jedoch SparqlMap eine SQL Quelle erfordert, ist es notwendig, diesen Ansatz fortzuführen und für SharePoint abzuwandeln.

## 1.4 Stand der Technik

Um die relational vorliegenden Daten zu exportieren, muss ein Mapping erzeugt werden. Es gibt kaum Editoren, die die vom W3C spezifizierte R2RML Sprache unterstützen. Das Erstellen solcher händischer Mappings ist ein zeitintensiver Prozess [Sen13]. Es ist noch ein Forschungsfeld, das Erzeugen dieses Mappings zu unterstützen oder semi-automatisiert zu lösen.

Die händisch erzeugten Mappings werden in der Regel in so genannte Triplestores exportiert. Ein Beispiel Virtuoso[1]. [Mic14]

Ontologien müssen händisch erstellt und gepflegt werden. Hierzu ist die Verwendung von speziellen Editoren zur Wissensverwaltung gebräuchlich. Ein Beispiel ist das open-source Projekt Protege[2].

Einige wenige Unternehmen bieten eine Integration von semantischen Web-Technologien. Diese sind aber alle proprietär und bedürfen immer noch eines hohen Maßes an Anwender-Interaktion. Dabei muss die Ontologie von Experten händisch gepflegt werden. In der Regel sind diese Erweiterungen von Sharepoint Webparts, die in die Seiten eingebunden werden. [Fil]

[1] http://virtuoso.openlinksw.com
[2] http://protege.stanford.edu

## 1.5 Zielsetzung

Ziel dieser Bachelorarbeit ist die Entwicklung eines Verfahrens zur semantischen Abbildung der Inhalte eines Microsoft SharePoint-Servers, so dass diese als RDF Triple durch einen SPARQL Endpunkt zur Verfügung gestellt werden können.

Im Rahmen dieser Arbeit werden verschiedene Ansätze zur Realisierung eines SPARQL Endpunktes für SharePoint-Server diskutiert. Dabei wird überprüft, inwiefern sie mit Microsofts Schnittstellendefinitionen verträglich sind. Um eine geeignete semantische Abbildungsvorschrift aufzustellen, muss zunächst die Inhaltsverwaltung des SharePoint-Servers analysiert werden. Anschließend werden die zur Verfügung stehenden Schnittstellen bewertet.

Hierzu werden im Kapitel 4 verschiedene Methoden diskutiert. Eine resultierende Lösungsarchitektur wird im Kapitel 5 vorgestellt. Das Kapitel 6 stellt die Implementierung eines Prototypen vor.

# 2 Grundlagen

Als Vorwissen werden die Inhalte des Bachelorstudiengangs Informatik der Universität Bonn vorausgesetzt. Insbesondere wird auf den Kenntnissen der Module Informationssysteme, Relationale Datenbanken und Web- und XML-Technologien aufgebaut.

Die darüber hinaus gehenden Prinzipien des Semantischen Web werden an dieser Stelle nur kurz eingeleitet. Es wird auf die jeweiligen Standards des World Wide Web Consortium[1] (W3C) verwiesen. Zum weiteren Verständnis werden in diesem Kapitel einige Grundlagen von SparqlMap und SharePoint erläutert.

## 2.1 Semantisches Web

Das World Wide Web Consortium (W3C) beschreibt Semantisches Web als plattformübergreifendes Prinzip zum Datenaustausch über Anwendungsgrenzen hinweg.

Im Sinne einer Erweiterung des klassischen Internets, welches zur Verbindung von Daten bzw. Dokumenten angedacht war, sollen die Prinzipien des semantischen Webs die Verbindung von Informationen ermöglichen. Oft wird es auch als Internet der Dinge oder Web 3.0 bezeichnet. [BL99]

Dabei sollen die Informationen für so genannte Agenten maschinenlesbar gemacht werden. Sie sollen dadurch ohne Künstliche Intelligenz in der Lage sein Informationen zusammenzutragen und aus ihnen Schlüsse ziehen zu können [BL01]. Der Begriff Linked Data bezeichnet dabei ein dezentrales Konzept, welches Informationen mittels Schnittstellen basierend auf dem Resource Description Framework anbietet [Hit06].

[1] http://www.w3.org

### 2.1.1 Resource Description Framework

Mit Hilfe des Resource Description Framework (RDF) können Informationen im Internet beschrieben werden.[1] Man bezeichnet diese als Ressourcen, zu dessen Eigenschaften (engl. property) RDF Aussagen macht. Eine solche Aussage wird Triple genannt und besteht dabei aus einem Subjekt (der Ressource), dem Prädikat (ihrer Eigenschaft) und ihrem Objekt (dem Wert). Ressourcen und Properties sind URIs. Ein Wert kann eine URI oder ein Literal sein. RDF wird üblicherweise im XML Format dargestellt. [Woo14]

Speichert man alle Triple in einem Informationssystem, dann spricht man von einem so genannten Triplestore. Ein Triplestore ist ein spezialisiertes DBMS. Das SPARQL Protokoll beschreibt die Art und Weise, wie RDF Triple abgefragt werden können [Har13].

### 2.1.2 SPARQL Protocol And RDF Query Language

Die SPARQL Protocol And RDF Query Language (SPARQL) ist eine graphbasierte Abfragesprache.[2] So genannte Triple Patterns spezifizieren die gewünschten Rückgabeergebnisse. Die Rückgabe ist eine tabellarische Struktur im XML oder JSON Format. Seit März 2013 ist mit der Version 1.1 auch ein ein Update-Befehl definiert worden. Mit SPARQL können Anfragen an sogenannte SPARQL Endpunkte gestellt werden. [Har13]

Ein SPARQL Endpunkt kann z.b. ein Triplestore sein. Liegen die RDF Triple direkt im Speicher vor, spricht man von materialisierten Daten. Im Gegensatz dazu kann ein SPARQL Endpunkt auch virtuell sein. Dann werden die Triple erst bei einer Abfrage durch eine Abbildungsvorschrift (engl. Mapping) erzeugt. [Unb13]

### 2.1.3 RDB to RDF Mapping Language

Die RDB to RDF Mapping Language (R2RML) ist eine standardisierte Sprache zur Definition von Abbildungen (engl. mapping) von relationalen Daten nach RDF.[3] Mit ihr ist es möglich, Inhalte einer bestehenden relationen Datenbank in einen RDF-Graph zu überführen. Das Mapping wird mittels Terse RDF Triple Language (Turtle) beschrieben. Dabei ist das Mapping selbst ebenfalls ein RDF-Graph. [Das12]

Um ein Triple zu erzeugen wird eine sogenannte TripleMap als Vorlage definiert. Als Eingabe für die TripleMap muss eine logische Tabelle angeben werden. Diese kann eine Tabelle, Sicht oder SQL-Abfrage sein. Jeder Datensatz des Rückgabeergebnisses wird entsprechend zu einem Triple umgeformt. [Das12]

[1] http://www.w3.org/standards/techs/rdf
[2] http://www.w3.org/standards/techs/sparql
[3] http://www.w3.org/standards/techs/rdb2rdf

## 2.2 SparqlMap

SparqlMap ist ein SPARQL-zu-SQL-Übersetzer, welcher 2012 als open-source Projekt von der Forschungsgruppe Agile Knowledge Engineering and Semantic Web[1] (AKSW) entwickelt wurde. Er ermöglicht, SPARQL Abfragen an bestehende relationale Datenbanken zu stellen. Die Übersetzung der Abfragen basiert auf R2RML. Dabei wird die Abfrage in eine einzige SQL Abfrage übersetzt, sodass die Kommunikation zwischen SparqlMap und Datenbank minimiert und zugleich von der Abfrageoptimierung des RDBMS profitiert werden kann. Das Ergebnis der übersetzten SQL Abfrage wird, abhängig vom Abfragetyp, in eine SPARQL Rückgabe umgewandelt. [Unb13]

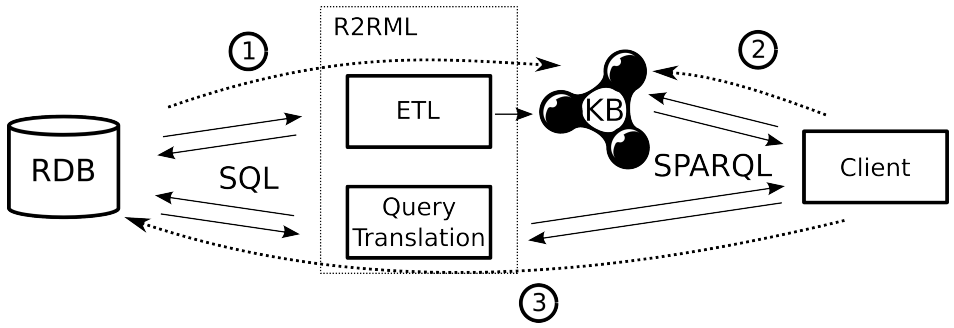


**Abbildung 2.1:** Modelle zur Abbildung von RDB nach RDF,
Quelle: [Unb13]

Die Abbildung 2.1 zeigt beide Ansätze zur Abbildung von relationalen Datenbanken nach RDF. Dabei können die Daten entweder mittels Extract, Transform, Load-Prozess (1) geladen und materialisiert werden, sodass diese danach (2) direkt als RDF abgefragt werden können, oder mittels Abfragenübersetzung (3) direkt aus der RDB abgerufen werden.

SparqlMap nutzt letzteren Ansatz. Oberstes Ziel bei der Entwicklung war es, eine möglichst leicht in bestehende Systeme zu integrierende Lösung zu finden. Somit wurde SparqlMap als standalone Anwendung konzipiert, die keine Materialisierung der RDF Daten benötigt. [Unb13]

Benchmarks haben gezeigt, dass SparqlMap nur einen Mehraufwand von 10% produziert. SparqlMap ist in Java entwickelt und nutzt ARQ aus Apaches Jena API[2] um SPARQL Anfragen zu parsen. Der SPARQL Endpunkt wird mittels Java Servlets realisiert. [Unb13]

[1] http://aksw.org/Projects/SparqlMap.html
[2] http://jena.apache.org

## 2.3 SharePoint

Seit 2001 vertreibt die Firma Microsoft die Produktlinie SharePoint. Zielgruppen sind dabei mittelständische und größere Unternehmen. Die Kernfunktionalität ist das Dokumentenmanagementsystem mit Weboberfläche. Microsoft bewirbt SharePoint als Kollaborationssoftware und die nahtlose Integration ihrer Office Anwendungen, welche die betriebliche Zusammenarbeit verbessern soll.

| Jahr | Produktname | Schwerpunkt |
|---|---|---|
| 2001 | SharePoint Portal Server | |
| 2003 | Office SharePoint Portal Server 2003 | Portal Integration |
| 2006 | Office SharePoint Server 2007 | Soziale Medien |
| 2010 | SharePoint 2010 | Mobile Endgeräte |
| 2013 | SharePoint 2013 | Cloud-Dienste |

**Tabelle 2.1:** Übersicht der SharePoint Produktlinie mit dem jeweiligen Schwerpunkt der Weiterentwicklung, Quelle: Microsoft

Im folgenden wird in dieser Arbeit der Begriff SharePoint, sofern die Version nicht ausgeschrieben ist, stellvertretend für die Technologie SharePoint verwendet. Sowohl der Begriff Microsoft als auch SharePoint sind eingetragene Marken der Microsoft Corporation.

Eine umfangreiche Dokumentation stellt Microsoft für Entwickler in der Microsoft Software Developer Network-Bibliothek[1] zur Verfügung. Die für das Verständnis wichtigen Konzepte werden in den folgenden Abschnitten Top-Down vorgestellt. Zunächst wird die Architektur erläutert, daraufhin folgt die Inhaltsverwaltung mit ihrer Representation der Information.

### 2.3.1 Architektur

SharePoint ist ein verteiltes dreischichtiges System. Mehrere Anwendungs-, Datenbank- und Webserver werden zu einer Serverfarm zusammengeschlossen. Dies soll eine angemessene Performance und Ausfallsicherheit garantieren. Hauptkomponenten des SharePoint sind der Internet Informations Server (kurz IIS) und der SQL Server.

Die Anwendungsebene abstrahiert von der physikalischen Topologie. Ihr oberstes logisches Element bildet die Webseite, welche beliebig viele verschachtelte Unterseiten besitzen kann. Die Inhalte des SharePoint werden jeweils pro Seite abgelegt.

[1] http://msdn.microsoft.com/library/

### 2.3.2 Inhaltsverwaltung

Von den vielen verschiedenen SharePoint Datenbanken wird in Abschnitt 4.3 die Inhaltsdatenbank näher betrachtet. In dieser werden sämtliche Inhalte gespeichert. Es kann mehrere Inhaltsdatenbanken geben, welche Datenbank zu welche Website gehört ist in der Konfigurationsdatenbank definiert. [Micd]

Im SharePoint wird jeder Inhalt einem Inhaltstyp (engl. content type) zugeordnet. Der Inhaltstyp beschreibt die Struktur des Inhaltes, indem er Feldbezeichnungen mit Datentypen und Einschränkungen festlegt. Durch Vererbung werden die übergeordneten Definitionen weitergegeben. Jeder Inhaltstyp besitzt genau einen übergeordneten Inhaltstyp.

SharePoint ist listenbasiert. Sämtliche Inhalte werden in Listen abgelegt. Dabei werden zwei besondere Arten von Listen unterschieden. Listen und Bibliotheken. Jeder Listeneintrag (engl. list item) kann beliebige Dokumente als Anhang besitzen.

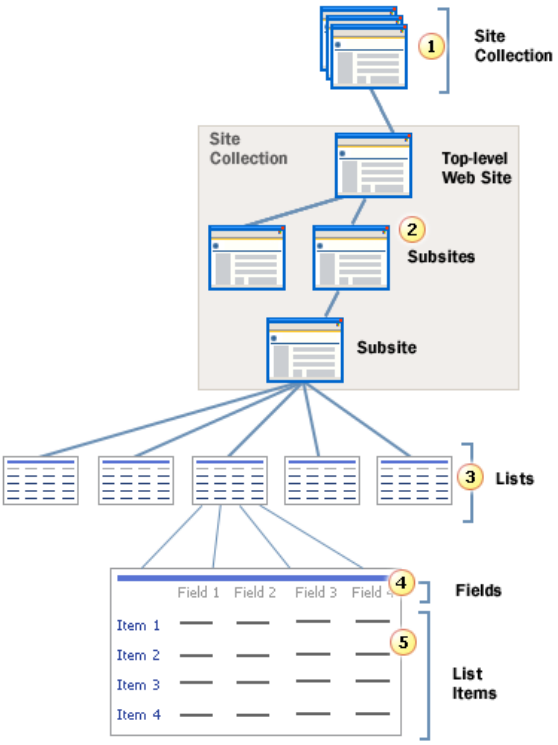


**Abbildung 2.2:** SharePoint Grundelemente,
Quelle: MSDN

# 3 Anforderungsanalyse

Die Gliederung der Anforderungsanalyse erfolgt nach der gebräuchlichen Unterscheidung zwischen funktionalen und nicht-funktionalen Anforderungen [Rob06]. Dazu werden zunächst die Anwendungsfälle und ihre Probleme identifiziert, woraus sich die Anforderungen ableiten lassen.

## 3.1 Anwendungsfälle

Die zu betrachtenden Anwendungsfälle beschränken sich in diesem Fall durch die hohe Zielebene und weiten Rahmen auf einen einzigen Systemanwendungsfall. Dieser lässt sich nach Cockburns Schablone [Coc] wie folgt beschreiben:

| **Veröffentlichung eines SPARQL Endpunktes für SharePoint** | |
|---|---|
| Akteure | SharePoint Server; SPARQL Client |
| Rahmen | System (white-box) |
| Ebene | sehr hohe Abstraktion |
| Auslöser | Etablierung semantischer Standards im Unternehmensumfeld |
| Beschreibung | Die Inhalte eines SharePoint sollen als SPARQL Endpunkt veröffentlicht werden, so dass diese mit SPARQL abgerufen werden können. |
| Invarianten | Microsofts Richtlinien sind einzuhalten und die Stabilität des Systems darf nicht durch die Erweiterung gefährdet sein. |

**Tabelle 3.1:** UseCase 01

## 3.2 Probleme

### 3.2.1 Richtlinien und Standards

Da das System und somit seine Randbedingungen festgelegt sind, muss die Lösungsarchitektur innerhalb dieser Grenzen verbleiben. Das bedeutet, dass weder Microsofts Richtlinien noch die W3C Standards missachtet werden dürfen.

### 3.2.2 Datenmenge

Die teilweise großen Mengen an Dokumenten bzw. Inhalten beanspruchen viele Speicher-Ressourcen, so dass eine Materialisierung der RDF Triple unwirtschaftlich wird. Es ist also notwendig das Mapping erst nach Anforderung anzuwenden.

### 3.2.3 Umfang des Mapping

Der große Umfang des SharePoint macht es Endanwendern kaum zumutbar, ein Mapping von Hand zu erstellen. Deswegen muss eine Lösung gefunden werden, welche das Eingreifen des Endanwenders reduziert bzw. vollständig überflüssig macht.

Die Erstellung eines Mappings zerfällt in zwei weitere Teilprobleme: Zunächst den Zugriff auf die Inhaltstypen, welche die Struktur der abzubildenden Inhalte definieren und den Zugriff auf die Inhalte selbst.

### 3.2.4 Zugriff auf Inhaltstypen

Der Zugriff auf die Inhaltstypen ist notwendig, weil diese die Strukturinformationen beschreiben, welche die Grundlege für das automatisierte Mapping darstellen. Auf die Inhaltstypen des SharePoint kann nur mit dessen Schnittstellen zugegriffen werden.

### 3.2.5 Zugriff auf Daten

Der Zugriff auf die Daten ist für den SPARQL Endpunkt notwendig. Hierbei ist es unerheblich, ob dies über die selbe Schnittstelle wie bei dem Zugriff auf die Inhaltstypen geschieht.

## 3.3 Anforderungen

### 3.3.1 Funktionale Anforderungen

1. Automatisiertes Generieren des Mapping

   **Beschreibung** Das Mapping soll automatisch generiert werden.

   **Begründung** Eine manuelle Erstellung ist wegen des Umfanges nicht vertretbar.

   **Quelle** UseCase 01; Problem 3; Problem 4

2. Keine Materialisierung der RDF Triple

   **Beschreibung** Die Daten werden erst auf Aufforderung gemappet.

   **Begründung** Vermeidung redundanter Speicher-Ressourcen und Inkonsistenz.

   **Quelle** Problem 2; Problem 5

### 3.3.2 Nicht-funktionale Anforderungen

Die nach [Rob06] folgenden Anforderungen sind mit absteigender Wichtigkeit sortiert und nur sofern relevant aufgeführt.

| | | |
|---|---|---|
| **Korrektheit** | Die Ergebnisse der SPARQL Abfragen sollen korrekt abgebildet werden. | |
| **Performance** | **Speicher** | Das Mapping soll keinen hohen Ressourcenbedarf haben. |
| | **Zeit** | Die Antwortzeiten der SPARQL Abfragen sollen ein flüssiges Arbeiten ermöglichen. |
| **Skalierbarkeit** | Das System ist in der Lage, mit vertretbaren Aufwand größere Datenmengen und kompliziertere Abfragen zu behandeln. | |
| **Portierbarkeit** | Das System soll zukunftssicher sein und somit Versionsänderungen tolerieren. | |
| **Wartbarkeit** | Anpassungen des Systems lassen sich ohne großen Aufwand vornehmen. | |

# 4 Methodologie

In diesem Kapitel werden die verschiedenen Vorgehensweisen und Überlegungen auf dem Weg zur Lösungsfindung vorgestellt. Ebenso werden aufgestellte Arbeitsansätze besprochen, welche nicht zum gewünschten Ziel führen. Einen großen Teil des Arbeitsaufwandes nahm die Erstellung geeigneter Versuchsdaten in Anspruch.

## 4.1 Konfiguration des Entwicklungssystems

Um alle Schnittstellen von SharePoint untersuchen zu können, muss gemäß Microsofts Vorgaben zur SharePoint Entwicklung die Entwicklungsumgebung auf der selben Maschine wie die Installation des SharePoint installiert sein [Micg]. Dies ist Grundvoraussetzung für die Nutzung des im Abschnitt 4.5.1 beschriebenen Sever-Object-Model [Mic11].

Die Entwicklung wurde auf einem Apple MacBook Pro Mid 2010 mit Intel Core 2 Duo Prozessor und 4 GB Arbeitsspeicher durchgeführt. Die Auswahl des Windows Betriebsystems, ist durch Apples Bootcamp Treiber auf die Version Windows 7 beschränkt [App].

Die aktuelle Version von SharePoint lässt sich ausschließlich auf Windows Server Betriebsystemen installieren [Micf]. Durch die Einschränkung auf Windows 7 muss auf SharePoint 2010 zurückgegriffen werden. Die unterstützte Version des Datenbankservers kann nach Microsofts 'SharePoint $n-1$ on SQL Server $n+1$' Regel bestimmt werden: Die eingesetzten Versionen dürfen sich um maximal ein Release unterscheiden [Mice].

Die Festlegung auf SharePoint 2010 ist rein technisch und stellt keinen Einfluss auf die Überlegungen zum Mapping dar. Die im Abschnitt 4.5 untersuchten Schnittstellen stehen auch in SharePoint 2013 zur Verfügung [Micc].

Somit setzt sich die Entwicklungsumgebung aus folgender Konfiguration zusammen:

| | |
|---|---|
| **Betriebssystem** | Windows 7 Professional |
| **Anwendungsserver** | SharePoint 2010 |
| **Datenbankserver** | SQL Server 2008 R2 Developer |
| **Entwicklungsumgebung** | VisualStudio 2010 und<br>SQL Management Studio 2008 R2 |
| **Framework** | Dot.Net 3.5 mit Service Pack 1 |

Die oben aufgeführte Software ist, sonfern nicht frei erhältlich, über das Institut für Informatik[1] mit dem Microsoft DreamSpark Premium-Abonnement beziehbar. Dieses gestattet die Nutzung für nichtkommerzielle Forschungszwecke [Micl].

## 4.2 Erstellung von Versuchsdaten

Nach einer neuen Installation von SharePoint liegt nur ein leeres System vor. Deshalb müssen zur genaueren Untersuchung der Inhaltsverwaltung und Datenhaltung geeignete Versuchsdaten erstellt werden. Das für Demonstrationszwecke angebotene Festplattenabbild von SharePoint 2010 ist nicht mehr erhältlich. Es bestand aus drei virtuellen Maschinen, auf denen jeweils ein vorkonfigurierter Server mit umfangreichen Beispielinhalten ein fiktives Unternehmen Namens Contoso darstellt [Plo].

Microsoft hat seine Produktlinie, beeinflusst durch den Trend von Social Media und der Umstellung auf Office Webanwendungen, umgestaltet. Seit der Veröffentlichung von SharePoint 2013 können Entwickler nun, durch Abschluss eines Online-Abonnements, ihre Lösungen auf von Microsoft gehosteten Systemen testen. Möchte man umfangreiche Tests mit SharePoint durchführen, so ist man auf die Erstellung eines eigenen Demo-Systems mit eigenen Versuchsdaten angewiesen.

In Microsofts Developer Center gibt es zwei Projekte, welche Dokumente im SharePoint für Testzwecke anlegen. Das Projekt bulkLoader erzeugt mithilfe des Office SDK und einem Wikipedia XML-dump Word, Excel, PowerPoint und Hypertext Dokumente [Mica]. Das zweite Projekt loadBulk nimmt die erzeugten Dokumente und lädt diese über die Schnittstellen des SharePoint in eine bestimmte Dokumentenbibliothek [Micb].

Versuche mit bulkLoader und loadBulk ergaben, dass die erzeugten Dokumente nicht genügend Komplexität bzw. Strukturtiefe besitzen, und somit keine ausreichend allgemeine Aussage für alle Inhaltstypen hergeleitet werden kann. Bei diesem Vorgehen besteht das Problem darin, dass ausschließlich Dokumente des selben Inhaltstypes 'Dokument' zu einer Dokumentenbibothek hinzugefügt werden. Dies genügt, um sich einen groben Eindruck

[1] http://www.informatik.uni-bonn.de/de/start/it-services/dienste/microsoft-dreamspark-premium/

über die Datenspeicherung zu verschaffen, doch mangelt es an Beziehungen zwischen den einzelnen Dokumenten.

Die Entwicklung eines eigenen Projektes zur Erzeugung von Versuchsdaten / Dokumenten ist erforderlich gewesen. Angelehnt am Prinzip von loadBulk wurde das Werkzeug SampleContentLoader entwickelt.

Als Datenquelle dient das Projekt DBpedia[1], welches Informationen aus Wikipedia extrahiert und und diese als Linked Data mit einer konsistenten Ontologie anbietet [Aue07].

Die DBpedia DataSets werden gelesen und in einer SQL Datenbank zwischengespeichert. Die SQL Datenbank vereinfacht den Umgang mit dem Problem der Verarbeitungsreihenfolge. Der erste Versuch, den tabellenbasierten Export der DBpedia zu parsen und mittels SQL Bulk Insert einzufügen, scheiterte an den nicht SQL-verträglichen Daten. Dabei traten z.B. folgende Probleme auf:

- Überschreitung der maximal zulässigen Anzahl an Spalten,
- Überschreitung der maximal zulässigen Zeichenanzahl als Spaltenbezeichner,
- Vergabe doppelter Spaltenbezeichner.

Beim zweiten Versuch konnten die DBpedia DataSets im turtle Format erfolgreich verarbeitet werden. Das Erzeugen der Versuchsdaten geschieht in zwei Phasen mit jeweils zwei Schritten:

1. Erstellung der Inhaltstypen
   a) Ontologie-Datei parsen und in eine SQL-Tabelle zwischenspeichern.
   b) Aus der SQL-Tabelle rekursiv die Inhaltstypen erzeugen.
2. Laden der Daten
   a) Dokumente / Listeneinträge erzeugen und einem Inhaltstyp zuweisen.
   b) Dokumente / Listeneinträge in den SharePoint hochladen.

Bei diesem Versuch wurde bemerkt, dass die dbpedia.owl Datei nicht alle in den DataSets referenzierten Klassen enthält. Jedoch konnte auf ein vollständiges Importieren verzichtet werden, da nur einige Versuchsdaten zur Rekonstruktion benötigt werden.

[1] http://dbpedia.org

## 4.3 Analyse der Datenhaltung

SharePoint besitzt eine Vielzahl verschiedener Datenbanken. Die Aufteilung und Bedeutung ist von Microsoft in [Micd] genau beschrieben. Für diese Arbeit ist hauptsächlich die Inhaltsdatenbank von Interesse. Microsofts Dokumentation bezüglich des inneren Aufbaus der Datenbank ist nicht besonders aufschlussreich, weil Microsoft den direkten Zugriff auf die Datenbank nicht unterstützt. Deswegen wurde versucht, das Datenbankschema zu rekonstruieren. Das Problem des direkten Zugriffs auf die Datenbank wird im Abschnitt 4.6 näher behandelt.

Der erste Versuch, mit den Werkzeugen des Microsoft SQL Management Studio ein Schemadiagramm erzeugen zu lassen, lieferte kein überschaubares Ergebnis. Wegen des Fehlens von Fremdschlüsseln und Einschränkungen werden im Diagramm sämtliche Tabellen nur aufgelistet.

Als Nächstes wurde versucht, sich durch genauere Betrachtung mit dem Microsoft SQL Management Studio einen Überblick über die Datenbank zu verschaffen. Die Datenbank beinhaltet überwiegend Tabellen. Die 117 fremdschlüsselfreien Tabellen werden durch 12 Sichten verbunden. Erschwerend kommt hinzu, dass die vergebenen Spaltenbezeichnungen nicht konsistent vergeben sind. Zusätzlich sind 899 Prozeduren und 33 Funktionen angelegt.

Nach näherer Betrachtung der Prozeduren konnte händisch ein unvollständiges Schema der Inhaltsdatenbank rekonstruiert werden.

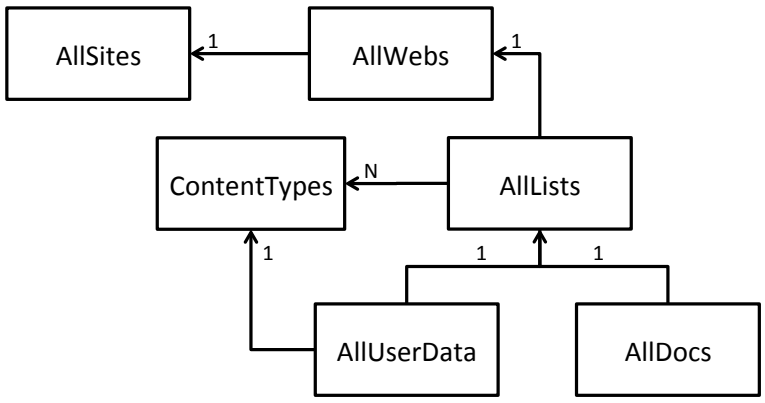


**Abbildung 4.1:** Zusammenhänge innerhalb der Inhaltsdatenbank (nicht vollständig)

Sofort fällt auf, dass der gesamte Inhalt in den Tabellen AllUserData und AllDocs abgelegt wird. Wie die Werte der Tabelle AllUserData zu interpretieren sind, ist in der Spalte Fields in der Tabelle ContentTypes mittels Collaborative Application Markup Language (CAML) in XML definiert.

Weil eine Indexierung so viele Werte-Spalten nicht umfassen kann, werden durchsuchbare Werte in eine Tabelle als Schlüsselpaare aufgenommen. Mit der Version 2013 werden durch die Einführung von wide tables die typisierten Werte-Spalten zu einer wide column im XML zusammengefasst. Dies erhöht die maximale Speichergröße eines Listeneintrages.

## 4.4 Mapping

Das W3C unterscheidet zwischen dem so genannten direkten und benutzerdefinierten Mapping. Direktes Mapping erzeugt generisch aus einem beliebigen Datenbankschema ein Mapping. Dabei wird jede Zeile einer Tabelle oder Sicht zu einem Triple übersetzt. Jede Spalte wird als Property des Triple übersetzt. Wenn es sich bei der Spalte um einen Fremdschlüssel handelt, so wird diese Referenz ebenfalls in RDF als Property, die auf eine andere Ressource zeigt, umgeformt. [Are12]

Die bei der Analyse (in 4.3) gewonnenen Erkenntnisse über das nicht streng normalisierte Datenbankschema lassen einen Ansatz mittels direktem Mapping nicht in Frage kommen. Die Interpretation der Werte innerhalb einer Spalte der Tabelle AllUserData ist nicht einheitlich. Somit kann eine Spalte nicht konsistent in eine Property übersetzt werden. Die Strukturinformationen zu den Werten sind über ihren Inhaltstyp in der Tabelle ContentType in der Spalte Field als XML codiert.

Gemäß dem Problem 3.2.2, dass aufgrund der Datenmenge kein benutzerdefiniertes Mapping händisch erfolgen kann, muss ein benutzerdefiniertes Mapping generiert werden. Aus der im Abschnitt 2.3.2 vorgestellten Inhaltsverwaltung des SharePoint Server lässt sich angelehnt an [Fil] ein semantischer Mapping Ansatz wie folgt ableiten:

- Inhaltstypen werden zu ontologischen Klassen transformiert

```
<owl:Class rdf:about="http://sp/ontology/ContentType1">
 <rdfs:label xml:lang="de">Inhaltstyp1</rdfs:label>
 <rdfs:comment xml:lang="de">ContentType.Description</rdfs:comment>
</owl:Class>
```

- Die Vererbung der Inhaltstypen wird ebenfalls abgebildet

```
<owl:Class rdf:about="http://sp/ontology/ContentType2">
 <rdfs:label xml:lang="de">Inhaltstyp2</rdfs:label>
 <rdfs:subClassOf rdf:resource="http://sp/ontology/ContentType1"/>
</owl:Class>
```

- Die Felder eines Inhaltstyps werden zur Property transformiert.

```
<owl:DatatypeProperty rdf:about="http://sp/ontology/Field1">
 <rdfs:label xml:lang="de">Feldbezeichnung1</rdfs:label>
 <rdfs:domain rdf:resource="http://sp/ontology/ContentType2"/>
 <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#s..."/>
</owl:DatatypeProperty>
```

- Felder mit Referenz auf Inhaltstyp werden als ObjectProperty abgebildet.

```
<owl:ObjectProperty rdf:about="http://sp/ontology/Field2">
 <rdfs:label xml:lang="de">Beziehung1</rdfs:label>
 <rdfs:domain rdf:resource="http://sp/ontology/ContentType2"/>
 <rdfs:range rdf:resource="http://sp/ontology/ContentType3"/>
</owl:ObjectProperty>
```

## 4.5 Analyse der Schnittstellen

In diesem Abschnitt werden die Schnittstellen von SharePoint vorgestellt.

### 4.5.1 Sever-Object-Model

Das Server-Object-Model ist eine sehr umfangreiche API für das Dot.Net Framework. Sie besitzt die weitestreichend Eingriffsmöglichkeiten in das SharePoint System und benötigt daher sehr hohe Sicherheitsanforderungen. Die Verwendung der Microsoft.SharePoint.dll Assembly ist ausschließlich auf dieselbe lokale Maschine begrenzt, was zur Folge hat, dass die Software auf allen physikalischen Servern verteilt werden muss. [Micc]

### 4.5.2 Client-Server-Object-Model

Das Client-Server-Object-Model (CSOM) bietet ebenfalls eine große, aber dennoch im Vergleich zum Sever-Object-Model eingeschränktere, API. Dagegen ist es jedoch nicht nur auf das Dot.Net Framework beschränkt, sondern auch aus Silverlight Anwendungen oder JavaScript heraus verwendbar. Das CSOM basiert auf einem Web Service, der das Server-Object-Model kapselt, und kann daher auch auf anderen Maschinen genutzt werden. [Mic11]

Es ist asynchron konzipiert, sodass alle benötigten Objekte explizit geladen werden müssen. Bis zum Erhalten der Serverantwort werden die Objekte durch Proxies repräsentiert. Ein Zugriff auf dessen ungeladene Werte führt unweigerlich zu einem Ausnahmefehler. Werden jedoch alle asynchronen Entwurfsmuster wie z.B. das Request Batching berücksichtigt, erhält man eine sehr leistungsstarke Schnittstelle. [Mici]

### 4.5.3 Web Service

Die von SharePoint 2010 angebotenen Web Services bieten einen größeren Funktionsumfang als das CSOM. Doch die Performance leitet unter der wesentlich aufwändigeren Verwaltung des Web Service. Es ist ist dem CSOM gegenüber in der Stapelverarbeitung und dem Ausnahmeverhalten im Nachteil. [Mic11]

Microsoft hat die Web Service Schnittstelle abgekündigt und empfiehlt, sofern möglich, auf die Nutzung von Web Services zu verzichten und stattessen für neue Entwicklungen das CSOM zu vorzuziehen [Mich].

### 4.5.4 REST API

Die Representational State Transfer (REST) Schnittstelle bietet als Web Service den resscourenorientierten Zugriff auf SharePoint Inhalte mittels der HTTP Methoden GET, POST, PUT und DELETE. Die API kann als Antwort verschiedne Formate wie z.B. XML, JSON, Atom oder AtomPub zurückliefen. Ähnlich der des CSOM ist die Schnittstelle für Zusammenarbeit von verschiednen plattformübergreifenden Anwendungen konzipiert. [Mic11]

Indem alle SharePoint Inhalte als HTTP Ressourcen verfügbar gemacht werden entfallen aufwendige Zwischenschichten. Der Einsatz von Batching, HTTP ETags und paging gewährleisten auch das effiziente Abrufen umfangreicher Dokumentenbibilotheken. [Mic11]

## 4.6 Zugriff auf die Datenbank

Im folgenden soll geklärt werden, wie der im Abschnitt 4.4 aufgestellte Mapping-Ansatz ohne Missachtung der folgenden seitens Microsoft nicht unterstützten Datenbankoperationen gelöst werden kann.

- Datenbank-Trigger hinzufügen
- Vorhandene gespeicherte Prozeduren direkt aufrufen
- Hinzufügen der neuer gespeicherter Prozeduren
- Hinzufügen, Ändern oder Löschen von Daten in einer beliebigen Tabelle
- Hinzufügen, Ändern oder Löschen von Spalten in einer Tabelle

Für die Definition eines benutzerdefinierten Mappings muss eine TripleMap auf eine logische Tabelle verweisen, welche laut Spezifikation eine valide SQL Anfrage sein muss [Das12]. Damit das Mapping die Anforderung an die Versionssicherheit erfüllt, sollten die im vorherigen Abschnitt 4.5 vorgestellten Schnittstellen berücksichtigt werden. Gesucht wird eine Lösung, die SQL Anfragen ins Object-Model oder REST übersetzt.

Hierfür kann auf die SQL Server CLR-Integration von Microsoft zurückgegriffen werden. Sie ermöglicht es, innerhalb des SQL Servers beliebige stored procedures, triggers, user-defined functions, user-defined types, und user-defined aggregates als Dot.Net code zu definieren und auszuführen [Micj].

## 4.7 Ereignisbasierte Steuerung der Automatisierung

Damit das Mapping aktuell bleibt, müssen einige Gedanken zur Pflege gemacht werden. Eine Möglichkeit ist es, das Mapping in regelmäßigen Abständen vollständig neu zu generieren. Das ist korrekt, hat aber bei sehr großen SharePoint Lösungen einen großen Rechenaufwand. Dabei muss man zwischen den Aktualisierungen ein inkonsistentes Mapping in Kauf nehmen. Wenn man jedoch nicht das gesamte Mapping aktualisieren will, dann muss man die Veränderungen separat behandeln können. In dem Paper von [DR13] werden Ereignisse für Ontologie eingefügt. Sie beschreiben Verhaltensmuster, bei deren auftreten sich die Ontologie verändert.

Im SharePoint sind bestimmte Ereignisse definiert, welche sich über das Object-Model abrufen lassen [Mick]. Mit ihnen kann die Aktualisierung des Mappings ausgelöst werden. Sie lassen sich eindeutig auf die von [DR13] beschriebenen Verhalten übertragen. So kann ein vollständiges Neuerstellen der Mappings vermieden werden.

Da unser Mapping nur auf Inhaltstypen basiert, und wir somit nicht die anderen SharePoint Ereignisse betrachten müssen, erhalten wir diesen Ausschnitt. Im SharePoint kann außerdem die Beziehung der ContentTypes nicht verändert werden, somit muss das addR und delR nicht behandelt werden.

| **SharePoint Events** | **Ontology Change Operations** |
|---|---|
| ItemAdded | addC(c) |
| ItemDeleted | delC(c) |
| FieldAdded | addA(a, c) |
| FieldDeleted | delA(a, c) |
| FieldUpdated | addR(a, c) oder delR(a, c) |
| FieldUpdated | chgA(c, a, v) |
| FieldUpdated | chgA(c, a, v) |

**Tabelle 4.1:** SharePoint Ereignisse und entsprechende Ontologie Operationen

# 5 Lösungsansatz

Aus den im vorherigen Kapitel 4 erlangten Erkenntnissen wird nun ein Lösungsansatz abgeleitet. Der Ansatz wurde maßgeblich durch die Einschränkungen im Zugriff auf die Datenbank in Abschnitt 4.6 beeinflusst. Hierzu wurde zentral ein Schnittstellenadapter in Form einer CLR-Assembly entwickelt, welcher die SQL-Anfragen von SparqlMap in SharePoints REST API übersetzt. Die Komponente SharePoint2RDF erzeugt die R2RML Mapping-Beschreibung und legt für jeden abgebildeten Inhaltstyp eine User Defined Function (UDF) an. Die UDF der Inhaltsypen ruft innerhalb der Adapterdatenbank die CLR-Assembly aus dem Projekt SharePoint2SQL auf. Neben der Mapping-Beschreibung kann dem Client eine Ontologie-Datei zur Verfügung gestellt werden.

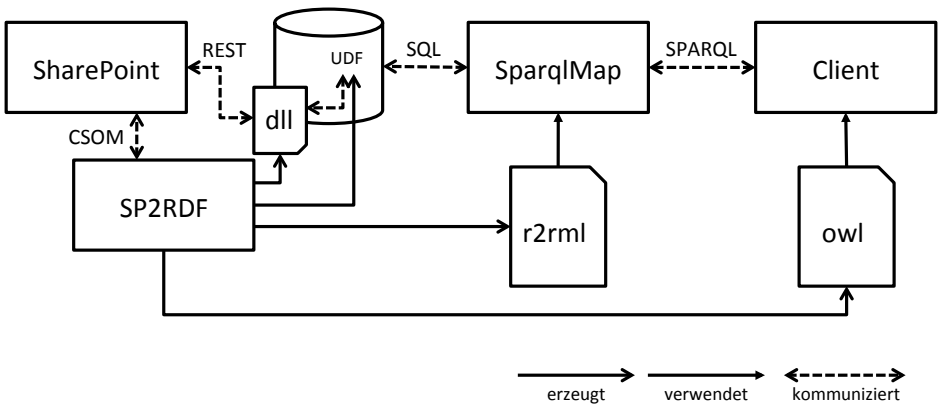


**Abbildung 5.1:** Übersicht der Architektur

# 6 Implementierung

Die Implementierung basiert vollständig auf dem Dot.Net Framework von Microsoft. Als Programmiersprache wurde C# gewählt. Sie ist von der Syntax und dem Verhalten der im wissenschaftlichen Umfeld weit verbreiteten Programmiersprache Java sehr ähnlich. Die Lernkurve von C# ist gering und gewährleistet mögliche Weiterentwicklungen. Besonderheit hierbei ist die Common Language Runtime CLR, sodass mühelos die verschiedenen von Dot.Net unterstützten Sprachen gemischt werden können. Dazu zählen C#,VB und F#. Die von Microsoft veröffentlichten Programmierschnittstellen Object-Model für SharePoint und SQL-Server beziehen sich auf das hauseigene Dot.Net Framework, sodass eine Verwendung nicht proprietärer Frameworks nicht möglich ist, sofern nicht REST oder WebService genutzt wird. Um keine Änderungen am SparqlMap zu verursachen wurde eine zusätzliche SQL-Datenbank erstellt. Sie wird wie ein Interface verwendet um SparqlMap einfache Select Abfragen absetzen lassen zu können. Intern werden die Daten über einen WebService geholt.

Die Implementierung (von VisualStudio .sln Solution genannt) gliedert sich in mehrere Teil-Projekte. Im folgenden werden die einzelnen Teil-Projekte vorgestellt.

## 6.1 SharePoint2RML

Dieses Projekt erzeugt aus einer Menge von Sharepoint ContentTyps eine R2RML Mapping Defintionsdatei. Es wird zu einer ausführbaren Binärdatei kompiliert. Bei der Ausführung erhält man die Mappingdatei. Die Generierung der Mappingdatei geschieht in zwei Phasen. In der ersten Phase werden die Mappingkandidaten ermittelt. Hierfür wird

über die CSOM Schnittstelle eine Anfrage an den SharePointserver geschickt. Dieser antwortet mit einer Auflistung der für den übergebenen Site-Bereich zur Verfügung stehenden ContentTypes. In der zweiten Phase wird ein StringWriter erzeugt. Dieser schreibt die Mappingdatei. Der Programmaufbau ist analog zur Grammatik des R2RML Standards aufgebaut. Durch den Streaming Ansatz ist gewährleistet, dass auch Dateien, die größer als der Hauptspeicher sind, korrekt geschrieben werden können. Transformierungen / Codierungen geschehen in den Methoden, die die Terminalsymbole in der Grammatik darstellen.

## 6.2 SharePoint2OWL

Dieses Projekt erzeugt aus einer Menge von SharePoint ContentTypes eine Ontologie-Datei im Format rdf/xml. Es wird zu einer ausführbaren Binärdatei kompiliert. Die Ontologie enthält die im SharePoint vorhanden Inhaltstypen und ihre Beziehungen zueinander.

## 6.3 SharePoint2SQL

Dieses Projekt beinhaltet User Defined Functions und Prozeduren die im SQL Server registriert werden. Es bildet den Schnittstellenadapter zwischen SparqlMap und der REST Schnittstelle des SharePoints. Die kompilierte DLL wird in den SQL-Server geladen und dort von User Definiend Functions als managed code aufgerufen.

## 6.4 Integration

Die Dot.Net Projekte lassen sich nahtlos in Microsofts Betriebsystem integrieren. Eine Möglichkeit ist, einen Windows Service zu schreiben, der die Events von SharePoint abfängt. Der Service verwaltet den Aufruf der Teilprojekte und startet gegebenenfalls die Generierung. Als Mehrwert für die Systemadministration entsteht dadurch, dass bestehende Monitoring Lösungen mitverwendet werden können.

SparqlMap benötigt als Java Anwendung eine VM, diese muss aber nicht zwingend auf der selben Maschine wie SharePoint2RDF ausgeführt werden. Es genügt in regelmäßigen Abständen die Mapping-Deffinitionsdatei auszutauschen. So kann SparqlMap eigenständig betrieben werden und muss nicht mit bestehenden Windows Installationen gemischt werden.

# 7 Evaluierung

Nun folgend sollen die im Kapitel 3 aufgestellten Überlegungen zu den Anforderungen erneut betrachtet und ihre Umsetzung bewertet werden.

## 7.1 Funktionale Anforderungen

Alle in Abschnitt 3.3.1 aufgestellten funktionalen Anforderungen können mit dem im Kapitel 5 vorgestellten Lösungsansatz erfüllt werden.

### 7.1.1 Automatisiertes Generieren des Mapping

Die im Abschnitt 4.4 gezeigte Verwendung von Inhaltstypen als Ausgangspunkt für das Mappings liefert ohne zusätzliche Benutzerinteraktion bereits ein umfangreiches Mapping. Dadurch, dass durch die Nutzung des SharePoint ohnehin für jeden Listeneintrag bzw. Dokument die Festlegung eines Inhaltstyps obligatorisch ist, wird die Arbeitsweise des Endanwenders nicht verändert. Die Geschäftsprozesse, die Inhalte im SharePoint erzeugen, stehen somit implizit semantisch zur Verfügung. Dieser Ansatz unterscheidet sich von Fillies und Weichhardt [Fil], indem er vollständig alle definierten Inhaltstypen ohne Einschränkung direkt abbildet und somit eine gute Standardvorlage ergibt.

Es wäre besser, wenn man für erfahrene Anwender ebenso die Möglichkeit schafft, das Mapping zu beeinflussen. Vorteil der Lösung von [Fil] ist, dass die Ontologie mittels Werkzeuge wie Protégé bearbeitbar ist und mehr Flexibilität bietet, jedoch bedarf es dafür einen zusätzlichen Mapping-Schritt außerhalb des SharePoint. Der in dieser Arbeit vorgestellt Ansatz lässt sich hinsichtlich der fehlenden Eingriffsmöglichkeit in das

Mapping leicht erweitern. Dafür können dem obersten Inhaltstyp noch zusätzliche Felder, die das Mapping beschreiben hinzugefügt werden, sodass diese an alle weiteren vererbt und den Nutzern im SharePoint transparent angezeigt werden. Zusätzlich kann die Implementierung bezüglich der Auswertung des '_Hidden' Feldes so erweitert werden, dass unsichtbare oder abgeschaltete Inhaltstypen nicht abgebildet werden.

Die im Abschnitt 4.7 vorgestellte Liste von SharePoint Ereignissen macht die Automatisierung effizienter. So ist es möglich nur Teile des Mapping zu aktualisieren und eine rechenintensive Neuerstellung des Ganzen zu vermeiden.

### 7.1.2 Keine Materialisierung der RDF-Triple

Die Lösungsarchitektur, welche auf dem im Abschnitt 2.2 beschriebenen SparqlMap Ansatz basiert, vermeidet redundante Speicherung und Inkonsistentz. Ferner ermöglicht sie vor allem eine einfache Integration in bestehende SharePoint Lösungen.

## 7.2 Nicht-Funktionale Anforderungen

### 7.2.1 Korrektheit

Im allgemeinen lässt sich die Korrektheit einer Implementierung nicht beweisen [Dij72]. Dennoch wurde versucht, dies bei der Entwicklung des Prototypes sicherzustellen. Jede Teilkomponente wurde hinsichtlich der aufgestellten Invarianten getestet, sodass jede Schnittstelle an sich wie erwartet bzw. Standardkonform verhält. Aus der Annahme der Korrektheit der einzelnen Komponenten lässt sich eine Korrektheit des gesamten Systems folgern.

### 7.2.2 Performance

Die Performance steht im Widerspruch zu Anforderung der nicht materialisierten Triple. Einerseits möchte man auf Redundanz verzichten, anderseits erwartet man schnelle Antwortzeiten. Bei der vorgestellten Lösungsarchitektur erhält man somit, bedingt durch den Transport der Daten zwischen SharePoint, dem Schnittstellenadapter und SparqlMap, eine verlängerte Wartezeit.

Desweiteren kann sich bei Abfragen, welche sich auf einen Inhaltstyp einer weit übergeordneten Ebene beziehen, die dadurch große Rückgabemenge negativ auf die Wartezeit auswirken. Ebenso wird durch die sehr allgemein formulierte CAML Abfrage die Optimierungen der Rest API nicht begünstigt. Besser wäre es, das Projekt SharePoint2SQL, welches den Schnittstellenadapter implementiert, hinsichtlich der Performance zu optimieren.

### 7.2.3 Skalierbarkeit

Die Skalierbarkeit ist durch den SparqlMap Ansatz und dem der REST API zugrundeliegende Hypertext Transfer Protokoll gewährleistet. Die Menge an Inhaltstypen im SharePoint ist beliebig wählbar. Einzig der zur Verfügung stehende Speicher bildet eine begrenzende Schranke. Eine Verallgemeinerung der Anfrage skaliert mit der Rückgabemenge und der Antwortzeit.

### 7.2.4 Portierbarkeit

Durch die Beschreibung des Mapping mittels des R2RML und Erstellung eines Schnittstellenadapters kann auf beliebige W3C-konforme Mapping-Anwendungen portiert werden. Bei der Entwicklung wurde darauf geachtet, dass die Lösungsarchitektur auch für zukünftige Versionen von SharePoint verwendet werden kann.

Einzig der Schnittstellenadapter ist wegen CLR Integration Plattformgebunden und benötigt zwingend das Dot.Net Framework. Das SharePoint2RDF Projekt lässt sich unter Verwendung der im Abschnitt 4.5.4 vorgestellten REST API auch in Java implementieren. Es empfiehlt sich jedoch zur Vereinheitlichung und leichteren Windows-Integration, das Dot.Net Framework zu nutzen.

### 7.2.5 Wartbarkeit

Die Implementierung wurde zur besseren Wartbarkeit in einzeln konfigurierbaren Teilprojekten Modular aufgebaut. So können Anpassungen ohne großen Aufwand vorgenommen werden.

# 8 Zusammenfassung und Ausblick

In diesem Kapitel werden die in dieser Arbeit gewonnenen Erkenntnisse nochmal kurz zusammengefasst. Im folgenden Abschnitt wird die Lösung mit verwandten Arbeiten und dem Stand der Technik verglichen bzw. eingeordnet. Der Abschnitt 8.2 soll die Einschränkungen dieser Lösung zusammenfassen. Zum Schluss folgen das Fazit und ein Ausblick auf mögliche weitere Entwicklung.

Im Rahmen dieser Arbeit wurde die Inhaltsverwaltung und Datenhaltung des SharePoint näher untersucht. Dabei hat die Betrachtung der Inhaltsverwaltung ergeben, dass sich die den Inhalten eindeutig zugeordneten Inhaltstypen gut für eine semantische Abbildung eignen. Sie werden auf Klassen und ihre Felder auf entsprechende Eigenschaften abgebildet.

Aus der Analyse der Datenhaltung ging hervor, dass wegen des Datenbankschemas der direkte Zugriff auf die Datenbank nur sehr umständlich möglich ist. Um die Werte der verschiedenen Inhalte zu erhalten, müssten diese zuerst aus verschiedenen Tabellen und Spalten zusammengefügt werden. Zusätzlich würde bei einem solchen Vorgehen gegen Microsofts Richtlinien zur Nutzung der Inhaltsdatenbank verstoßen.

Um einen stabilen Zugriff auf die Inhalte zu gewährleisten, wurden eine Abwägung der zur Verfügung stehenden Schnittstellen des SharePoint durchgeführt. Nachdem das Server-Object-Model wegen zu hohen Sicherheitsanforderungen und Web Services wegen Microsoft Abkündigung als Schnittstellen ausgeschieden sind, fiel die Wahl auf das Client-Server-Object-Model sowie die REST API.

Damit SparqlMap die Inhalte mittels SQL abfragen kann, wurde ein Schnittstellenadapter entwickelt, welcher die SQL Anfragen in einen REST Aufruf übersetzt. Dieser basiert auf einer CLR Assembly, welche im SQL Server registriert wird. Sie führt innerhalb der Datenbank den REST Aufruf durch und liefert die Antwort als SQL Tabelle zurück.

## 8.1 Verwandte Arbeiten

Über den Stand der Technik hinaus, wurde inspiriert durch den SparqlMap Ansatz eine leicht in bestehende Systeme integrierbare Lösung gefunden, welche keine Benutzerinteraktion und Materialisierung der RDF Triple erfordert.

Es konnten keine mit diesem Ansatz unmittelbar verwandte Arbeiten identifiziert werden.

Microsoft hat auf der SharePoint Konferenz 2014 ein neues Produkt namens Delve (vormals Codename Oslo) vorgestellt. Laut Microsofts Pressemitteilung[1] vom 09.09.2014 soll Delve ab dem 15.09.2014 ausgerollt werden. Leider wird aus Microsofts Produktdarstellung nicht die verwendete Technologie und dessen Ansatz deutlich. Es soll zumindest den im Abschnitt 1.2 motivierten Funktionsumfang bieten, Mitarbeitern Wissen einfacher zugänglich zu machen.

## 8.2 Einschränkungen

Sicherheitsaspekte wurden bei der Bearbeitung dieses Arbeit nicht behandelt. Vertrauliche Dokumente würden ohne weitere Maßnahmen im gesamten Unternehmen veröffentlicht. Dies stellt für einen realen Betrieb eines semantischen SharePoint die größte Einschränkung dar. Hierzu ist z.B. eine Integration von Microsofts Active Directory-Rechteverwaltung erforderlich.

Die mit der Einführung von SPARQL 1.1 neue Fähigkeit, Triple mittels SPARQL einzufügen, wird von diesem Lösungsansatz nicht unterstützt. Dies liegt daran, dass SparqlMap in der bisherigen Implementierung nur DESCRIBE Abfragen behandeln kann. Diese Einschränkung gilt jedoch nicht für die REST Schnittstelle. Somit ist es möglich, später diese Funktionalität in dem Projekt SharePoint2SQL hinzuzufügen.

Die Ontologie wird vollständig aus den Inhaltstypen erstellt. Daher fehlt die Möglichkeit, diese mit benutzerdefinierten Einstellungen zu beeinflussen.

## 8.3 Schlussfolgerung

In dieser Arbeit wurde gezeigt, dass ein generisches Mapping möglich ist. Unter den gegeben Randbedingungen hat sich der SparqlMap Ansatz als machbar erwiesen. So ist man in der Lage, beliebige SPAQRL Abfragen auf den SharePoint Inhalten auszuführen.

Standardisierung der Schnittstellen ist erkennbarer Entwicklungstrend umständlicher Zugriff wegen SQL - REST REST ist jedoch gut geeignet weil versionssicherer

[1] http://www.microsoft.com/de-de/news/pressemitteilung.aspx?id=535160

## 8.4 Ausblick

Da im Rahmen dieser Arbeit explorativ gearbeitet wurde, lag kein besonderer Schwerpunkt auf optimiertem Datendurchsatz. Es bietet sich an, Bottlenecks im Rahmen eines systematischen Lasttests zu finden und ggf. zu beheben.

Hinsichtlich der Erweiterung des SPARQL-Protokolls wäre es möglich, die Systemarchitektur von einigen Indirektionen zu befreien und analog zu SparqlMap eine Query Engine zu entwickeln, die als SPARQL Endpunkt die REST-Schnittstelle von SharePoint direkt verwendet. In Dot.Net entwickelt, könnte eine solche Komponente eine native und leistungsstarke Anbindung liefern. Außerdem könnte dann die Rechteverwaltung an den Daten über ADS direkt verwaltet werden: Die Credentials des Endnutzers können bis zum SharePoint durchgereicht werden.

Schließlich erscheint es interessant, Microsofts Lösung Delve mit den Ergebnissen dieser Arbeit zu vergleichen.

# Literaturverzeichnis


[App] Apple: Boot Camp: System requirements for Microsoft Windows operating systems, URL http://support.apple.com/kb/HT5634 (Zitiert auf Seite 13)

[Are12] Arenas, Marcelo; Bertails, Alexandre; Prud'hommeaux, Eric und Sequeda, Juan: A Direct Mapping of Relational Data to RDF, W3C recommendation, W3C (2012), http://www.w3.org/TR/rdb-direct-mapping/ (Zitiert auf Seite 17)

[Aue07] Auer, Sören; Bizer, Christian; Kobilarov, Georgi; Lehmann, Jens und Ives, Zachary: DBpedia: A Nucleus for a Web of Open Data, in: *In 6th Int'l Semantic Web Conference, Busan, Korea*, Springer, S. 11–15 (Zitiert auf Seite 15)

[BL99] Berners-Lee, Tim und Fischetti, Mark Collaborateur.: *Weaving the Web : the original design and ultimate destiny of the World Wide Web by its inventor*, Harper San Francisco 2000, San Francisco (1999), URL http://opac.inria.fr/record=b1096780 (Zitiert auf Seite 5)

[BL01] Berners-Lee, Tim; Hendler, James und Lassila, Ora: The Semantic Web. *Scientific American* (2001), Bd. 284(5):S. 34–43, URL http://www.sciam.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21 (Zitiert auf Seite 5)

[Coc] Cockburn, Alistair: *Use Cases effektiv erstellen.*, Mitp-Verlag, 1 Aufl. (Zitiert auf Seite 10)

[Das12] Das, Souripriya; Cyganiak, Richard und Sundara, Seema: R2RML: RDB to RDF Mapping Language, W3C recommendation, W3C (2012), http://www.w3.org/TR/2012/REC-r2rml-20120927/ (Zitiert auf Seiten 6 and 19)

[Dij72] Dijkstra, Edsger W.: The Humble Programmer. *Commun. ACM* (1972), Bd. 15(10):S. 859–866, URL http://doi.acm.org/10.1145/355604.361591 (Zitiert auf Seite 25)

[DR13] Dos Reis, Julio Cesar; Dinh, Duy; Pruski, Cédric; Da Silveira, Marcos und Reynaud-Delaître, Chantal: Mapping Adaptation Actions for the Automatic Reconciliation of Dynamic Ontologies, in: *Proceedings of the 22Nd ACM International Conference on Conference on Information & Knowledge Management*, CIKM '13, ACM, New York, NY, USA, S. 599–608, URL http://doi.acm.org/10.1145/2505515.2505564 (Zitiert auf Seite 20)

[Fil] Fillies, Christian und Weichhardt, Frauke: Linked Data Interface, Semantics and a T-Box Triple Store for Microsoft SharePoint, white paper (Zitiert auf Seiten 3, 17, and 24)

[Har13] Harris, Steven und Seaborne, Andy: SPARQL 1.1 Query Language, W3C recommendation, W3C (2013), http://www.w3.org/TR/2013/REC-sparql11-query-20130321/ (Zitiert auf Seite 6)

[Hit06] Hitzler, Pascal und Sure, York: *Semantic Web: Grundlagen*, eXamen.press, Springer, Heidelberg (2006) (Zitiert auf Seite 5)

[Mica] Micosoft: Load Bulk Content to SharePoint 2010, URL http://code.msdn.microsoft.com/Bulk-Loader-Create-Unique-eeb2d084 (Zitiert auf Seite 14)

[Micb] Microsoft: Bulk Loader - Create Unique Documents based on Wikipedia Dump File, URL http://code.msdn.microsoft.com/office/Load-Bulk-Content-to-3f379974 (Zitiert auf Seite 14)

[Micc] Microsoft: MSDN article - Choose the right API set in SharePoint 2013, URL http://msdn.microsoft.com/library/office/jj164060(v=office.15).aspx (Zitiert auf Seiten 13 and 18)

[Micd] Microsoft: MSDN article - Database types and descriptions (Office SharePoint Server), URL http://technet.microsoft.com/en-us/library/cc678868(v=office.12).aspx (Zitiert auf Seiten 9 and 16)

[Mice] Microsoft: MSDN article - Hardware and software requirements (SharePoint Foundation 2010), URL http://technet.microsoft.com/library/cc288751.aspx (Zitiert auf Seite 13)

[Micf] Microsoft: MSDN article - Set up the development environment for SharePoint 2013, URL http://msdn.microsoft.com/library/office/ee554869(v=office.15).aspx (Zitiert auf Seite 13)

[Micg] Microsoft: MSDN article - Setting Up the Development Environment for SharePoint 2010 on Windows Vista, Windows 7, and Windows Server 2008, URL http://msdn.microsoft.com/library/ee554869(v=office.14).aspx (Zitiert auf Seite 13)

[Mich] Microsoft: MSDN article - SharePoint 2013 Deprecated API sets, URL http://msdn.microsoft.com/en-us/library/office/jj164060(v=office.15).aspx#DeprecatedAPIs (Zitiert auf Seite 19)

[Mici] Microsoft: MSDN article - Using the Client Object Model, URL http://msdn.microsoft.com/en-us/library/ff798388.aspx (Zitiert auf Seite 18)

[Micj] Microsoft: MSDN article -Introduction to SQL Server CLR Integration, URL http://msdn.microsoft.com/library/office/ff464297(v=office.14).aspx (Zitiert auf Seite 19)

[Mick] Microsoft: MSDN article -Table of SharePoint Events, Event Receivers, and Event Hosts, URL http://msdn.microsoft.com/en-us/library/office/ff408183(v=office.14).aspx (Zitiert auf Seite 20)

[Micl] Microsoft Corporation: Microsoft DreamSpark Premium-Abonnement, https://www.dreamspark.com/Institution/DSP-EULA.aspx (Zitiert auf Seite 14)

[Mic11] Microsoft: MSDN article - Deciding Which SharePoint 2010 API to Use (2011), URL http://msdn.microsoft.com/library/hh313619.aspx (Zitiert auf Seiten 13, 18, and 19)

[Mic14] Michel, Franck; Montagnat, Johan und Faron-Zucker, Faron@polytech.Unice.Fr, Catherine: A survey of RDB to RDF translation approaches and tools, Techn. Ber. (2014), URL http://hal.archives-ouvertes.fr/hal-00903568, iSRN I3S/RR 2013-04-FR 24 pages (Zitiert auf Seite 3)

[Mil11] Miles, Doug: State of the ECM Industry 2011, Techn. Ber., Association for Information and Image Management, 1100 Wayne Avenue, Suite 1100, Silver Spring, MD 20910 (2011) (Zitiert auf Seite 1)

[Plo] Plotnikov, Dmitri und Ozcifci, Gokan: Sharepoint 2010: Information Worker Demonstration and Evaluation Virtual Machine (SP1), TechNet Articles, URL http://social.technet.microsoft.com/wiki/contents/articles/8910.sharepoint-2010-information-worker-demonstration-and-evaluation-virtual-machine-sp1.aspx (Zitiert auf Seite 14)

[Rob06] Robertson, Suzanne und Robertson, James: *Mastering the Requirements Process (2Nd Edition)*, Addison-Wesley Professional (2006) (Zitiert auf Seiten 10 and 12)

[Sen13] Sengupta, Kunal; Haase, Peter; Schmidt, Michael und Hitzler, Pascal: Editing R2RML Mappings Made Easy., in: Eva Blomqvist und Tudor Groza (Herausgeber) *International Semantic Web Conference (Posters & Demos)*, Bd. 1035 von *CEUR Workshop Proceedings*, CEUR-WS.org, S. 101–104 (Zitiert auf Seite 3)

[Unb13] Unbehauen, Jörg; Stadler, Claus und Auer, Sören: Accessing Relational Data on the Web with SparqlMap, in: Hideaki Takeda; Yuzhong Qu; Riichiro Mizoguchi und Yoshinobu Kitamura (Herausgeber) *Semantic Technology*, Bd. 7774 von *Lecture Notes in Computer Science*, Springer Berlin Heidelberg (2013), S. 65–80, URL http://dx.doi.org/10.1007/978-3-642-37996-3_5 (Zitiert auf Seiten 6 and 7)

[Woo14] Wood, David; Lanthaler, Markus und Cyganiak, Richard: RDF 1.1 Concepts and Abstract Syntax, W3C recommendation, W3C (2014), http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/ (Zitiert auf Seite 6)