Institut für Architektur von Anwendungssystemen
Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Diplomarbeit Nr. 3406

Verwendung von semantischen Wikis zur Lösungsdokumentation und Musteridentifikation

Daniel Andreas Kaupp

Studiengang:	Informatik
Prüfer:	Prof. Dr. Frank Leymann
Betreuer:	Dipl.-Inf. Christoph Fehling Johanna Barzen, M.A.
begonnen am:	19. Dezember 2012
beendet am:	5. Juli 2013
CR-Klassifikation:	D.2.10, D.2.11, D.2.13, H.2.3, I.2.3

Inhaltsverzeichnis

Abbildungsverzeichnis

Verzeichnis der Listings

Verzeichnis der Algorithmen

Abkürzungsverzeichnis

AIFB Institut für angewandte Informatik und formale Beschreibungsverfahren

Ajax Asynchronous JavaScript and XML

API Application Programming Interface

CSS Cascading Style Sheets

DAML DARPA Agent Markup Language

DARPA Defense Advanced Research Projects Agency

DIQA DIQA Projektmanagement GmbH

DOM Document Object Model

DV Datenverarbeitung

FIFO First In First Out

GoF Gang of Four

GPL General Public License

HTML Hypertext Markup Language

HTTP Hypertext Tansfer Protocol

IETF Internet Engineering Task Force

JSON JavaScript Object Notation

KIT Karlsruher Institut für Technologie

MW MediaWiki

MVC Model-View-Controller

OIL Ontology Inference Layer

OWL Web Ontology Language

PHP PHP: Hypertext Preprocessor

PLoP Pattern Language of Programs

RDF Resource Description Framework

RDFS Resource Description Framework Schema

RFC Request for Comments

SF SemanticForms

SFS SF Select

SFOS SF Ontology Select

SMW Semantic MediaWiki

SMW+ Semantic MediaWiki Plus

SPARQL SPARQL Protocol And RDF Query Language

SPARUL SPARQL Update Language

URI Uniform Resource Identifier

W3C World Wide Web Consortium

XML eXtended Markup Language

1 Einleitung

„Wissen ist Macht!“ Diese alte Weisheit gilt heute mehr denn je. Das Informationszeitalter ist geprägt durch Wissen und seine Verarbeitung. Viele Berufsbilder befassen sich ausschließlich mit der Verarbeitung von Daten. Die Flut von Informationen, die in der heutigen Welt generiert werden ist bei weitem zu groß, als dass ein einzelner Mensch sie verarbeiten könnte.
Das führt dazu, dass Menschen sich vor Informationen schützen müssen: In früheren Jahrhunderten breiteten sich Informationen mit der Geschwindigkeit des Menschen aus. Damals war es so, dass Informationen die jemanden erreichten auch zumindest in dessen mittelbarem Umfeld relevant waren. Mittlerweile fällt es schwer die Zeit zu messen, die eine Information braucht, um den Erdball zu umrunden. Die Welt ist zum mittelbaren Umfeld geworden und die Informationen der Welt strömen auf jeden Menschen ein. Doch diese Informationen sind bei weitem nicht alle relevant. Deutlich wird dies schon am eigenen E-Mail Posteingang: Wie viel Prozent unserer E-Mails sind relevant? Und das sind nur Nachrichten, deren mehr oder weniger intendierter Empfänger wir sind. Eine Filterung der Informationen ist also unerlässlich, um irrelevante Inhalte auszublenden.
Auf der anderen Seite gehen sicher auch viele Informationen an uns vorbei, ohne dass wir sie wahrnehmen. Die Chance, dass etwas dabei ist was für uns relevant gewesen wäre, ist recht groß. Ein großer Anreiz an sozialen Netzwerken ist sicherlich das einfache „Teilen“ von Inhalten, die die eigenen Freunde unbedingt sehen sollten.
Was wäre, wenn diese Filtersysteme tatsächlich beides könnten? Irrelevante Informationen ausblenden und alle relevanten Informationen weiterreichen? Tim Berners-Lee hat diese Vision im Artikel: „The Semantic Web“[BLHL01] in Form von „Agenten“ vorgestellt. Computerprogramme haben eine reele Chance mit der Fülle an Daten fertig zu werden. Doch das allein reicht nicht aus: Sie müssen auch verstehen was wichtig ist und was nicht. Man muss die Bedeutung der Informationen mittragen, damit daraus verwertbares Wissen wird. Während diese Idee in den letzten 12 Jahren bereits Zeit hatte, die IT und IT-nahen Fachgebiete zu durchdringen, nähern sich die Geisteswissenschaften erst jetzt nach und nach an.
Das Schlagwort dieser neuen Bewegung lautet „Digital Humanities“. Um den vollen Mehrwert aus einer Digitalisierung eines geisteswissenschaftlichen Fachgebiets zu gewinnen, ist es ratsam nicht nur Daten, sondern Wissen - also Information mitsamt ihrer Bedeutung - zu

erfassen.
Bei einer Vorsortierung des Wissens eines Fachgebiets treten oft Strukturen hervor, die bei der Gestaltung einer geeigneten Wissensrepräsentation helfen und darüber hinaus auch mit geringem Adaptionsaufwand auf Strukturen der Informatik umgemünzt werden können. Ein Beispiel für solche Strukturen sind Entwurfsmuster (engl.: Pattern). Daten, die nach einem Entwurfsmuster aufgebaut sind, werden in dieser Arbeit mit dem Begriff „Lösungen" bezeichnet.
Die Informatik hat für solche Strukturen Methoden und Werkzeuge, die eine Eingabe, Konservierung, Verarbeitung und Ausgabe des Wissens ermöglichen. Daraus ergibt sich für die Geisteswissenschaften mit geringem Adaptionsaufwand ein großer Mehrwert.

Ziel dieser Arbeit ist, das Fachwissen einer geisteswissenschaftlichen Disziplin in einem semantischen Wiki zu erfassen und Methoden zur Dokumentation von Lösungen zu finden. Desweiteren sollen die Möglichkeiten des Wikis dazu benutzt werden, Prozesse bei der Musteridentifikation zu unterstützen. Für die Evaluation dieser Ziele wurde eine Unterdisziplin der Theater-, Film- und Fernsehwissenschaft gewählt: Die Kostümwissenschaften.

Gliederung

Die Arbeit ist in folgender Weise gegliedert:

Kapitel 1 – Einleitung: Eine Einführung in die Arbeit und diese Gliederung.

Kapitel 2 – Grundlagen und Verwandte Arbeiten: Kapitel 2 bietet eine Einführung in semantische Informationen und einen Einblick in aktuelle Entwicklungen in der Kostümwissenschaft. Darüber hinaus werden einige Arbeiten zu Entwurfsmustern und deren Identifikationsprozess betrachtet.

Kapitel 3 – Konzept zur Erfassung einer Musterdomäne: Dieses Kapitel erklärt das Vorgehen bei der Erfassung einer Domäne und beschreibt konkrete Musteridentifikationsprozesse.

Kapitel 4 – Anforderungen durch die Kostümdomäne: Die Kostümdomäne hat spezielle Anforderungen, die in diesem Kapitel erörtert werden.

Kapitel 5 – Implementierung eines semantischen Wikis zur Erfassung der Kostümdomäne: Details zur Implementierung sowie der zugrundeliegenden Wikisoftware offenbart dieses Kapitel.

Kapitel 6 – Zusammenfassung und Ausblick: Dieses Kapitel fasst die Ergebnisse der Arbeit zusammen und bietet einen kurzen Ausblick.

2 Grundlagen und Verwandte Arbeiten

Diese Arbeit beschäftigt sich mit Methoden zur Lösungsdokumentation in einem semantischen Wiki und mit der Identifikation von Mustern. Die Daten, die als „Lösungen“ dokumentiert werden, sind einem Fachgebiet, einer Domäne zuzuordnen. Abschnitt 2.1 erklärt wichtige Begriffe rund um semantische Datenerfassung und Muster.
In Abschnitt 2.4 werden Musteridentifikationsprozesse in anderen Domänen betrachtet.

2.1 Grundlagen - Semantik

Seit Tim Berner-Lees wegweisendem Artikel „The semantic Web“ [BLHL01] aus dem Jahr 2001 hat dieser Begriff nicht nur die Welt der Informatik durchdrungen. Dabei sind die Konzepte hinter diesem Begriff weitgehend unbekannt.

2.1.1 Bedeutung von Daten

Semantik, aus dem Griechischen abgeleitet für „dem Zeichen zugehörig“, ist die Bedeutung eines oder mehrerer Zeichen. In natürlicher Sprache erschließt sich die Semantik dem Zuhörer meist sofort, da unser Gehirn mehr darauf trainiert ist, Bedeutung und Zusammenhang von Worten zu erfassen, als sich ihren exakten Laut zu merken. Dieses Verhalten ist bei computergestützten Systemen gerade gegensätzlich: Ein Computer ist in der Lage, den Wortlaut eines eingegebenen Satzes zu speichern und exakt wiederzugeben. Dabei hat er von der Bedeutung der Worte in natürlicher Sprache keine Ahnung. Ihm ist es nicht möglich, einfache Zusammenhänge oder Schlüsse aus Texten abzuleiten.

Eine der Grundaufgaben des Semantic Web ist, dem Computer ein Verstehen der Bedeutung unserer Daten zu ermöglichen. Dabei geht es natürlich nicht um ein Verstehen im menschlichen Sinn, sondern um die Fähigkeit, Daten in einem gegebenen Rahmen interpretieren zu können und daraus Schlüsse zu ziehen. Als Lösung für dieses Problem bietet es sich an, die Informationen mit Wissen zu versehen, das auch vom Computer interpretiert werden kann. Dafür gibt es einige Sprachen, die sehr einfach sind und in einem Satz nur eine

grundlegende Information verpacken können. Trotz dieser Beschränktheit lassen sich damit die meisten Sachverhalte (wenn auch über Umwege) modellieren. Das Sprachkonstrukt dieser Sprachen besteht aus einfachen Sätzen, die nur Subjekt, Prädikat und Objekt kennen. Web Ontology Language (OWL) und Resource Description Framework (RDF) / Resource Description Framework Schema (RDFS) sind die zwei bedeutendsten Vertreter und werden im Folgenden näher betrachtet.
Bevor diese Sprachkonstrukte vorgestellt werden, macht es Sinn die Daten zu betrachten, um deren Bedeutung es gehen soll. Wenngleich der Titel dieser Arbeit die Beschaffenheit der Daten nicht einschränkt, beziehen sich Ihre Forderungen auf Daten, die eine Domäne abbilden.

2.1.2 Strukturierung der Daten einer Domäne

Um eine Verarbeitung von Daten erst möglich zu machen, müssen diese in eine Struktur gebracht werden. Das komplette Fachgebiet, welches strukturiert wird, bezeichnet man dabei als Domäne. Diese Domäne kann ein beliebiges Wissensgebiet des „natürlichen“ Lebens repräsentieren. Die Struktur muss dabei flexibel genug sein, um das gesamte Fachwissen abzubilden und trotzdem so abstrakt, dass sie zur Weiterverarbeitung mit standardisierten Methoden eingesetzt werden kann.
Um diese Strukturierung etwas anschaulicher zu gestalten, werden die weiteren Erklärungen anhand der Domäne „Personendaten“ vorgenommen. Diese ist besonders geeignet, da jeder zumindest mit seinen eigenen persönlichen Daten umgehen muss und somit bereits ein Domänenexperte ist. Zudem ist diese Domäne recht überschaubar, was auch die Beispiele sehr leicht verständlich macht. In Kapitel 5 werden die hier vorgestellten Prinzipien auf die Domäne „Kostüme beim Film“ angewandt.

Überblick über vorhandene Daten Der Personalausweis enthält einige unserer wichtigsten Personendaten. Um eine Einteilung vorzunehmen genügen bereits die wenigen Informationen auf der Vorderseite. Exemplarisch wollen wir hier die Daten von Frau Mustermann betrachten. Abbildung 2.1 zeigt ihren Personalausweis.

Ein intuitiver Weg, diese Daten zu strukturieren ist, die beschreibendenden (blau hinterlegt) und die echten Daten (grün hinterlegt) zusammenzufassen. Die beschreibenden Daten werden dabei als **Klassen** oder Begriffe (engl.: concepts) bezeichnet. Diese Begriffe sind auf jedem Personalausweis gleich, sie bilden also auch kein Unterscheidungsmerkmal. Im Gegensatz dazu sind die grün hinterlegten Daten für jede Person eindeutig. Zumindest sollten sie in ihrer Summe zu einer eindeutigen Identifizierbarkeit der Person führen. Dabei spricht man von **Instanzen**.

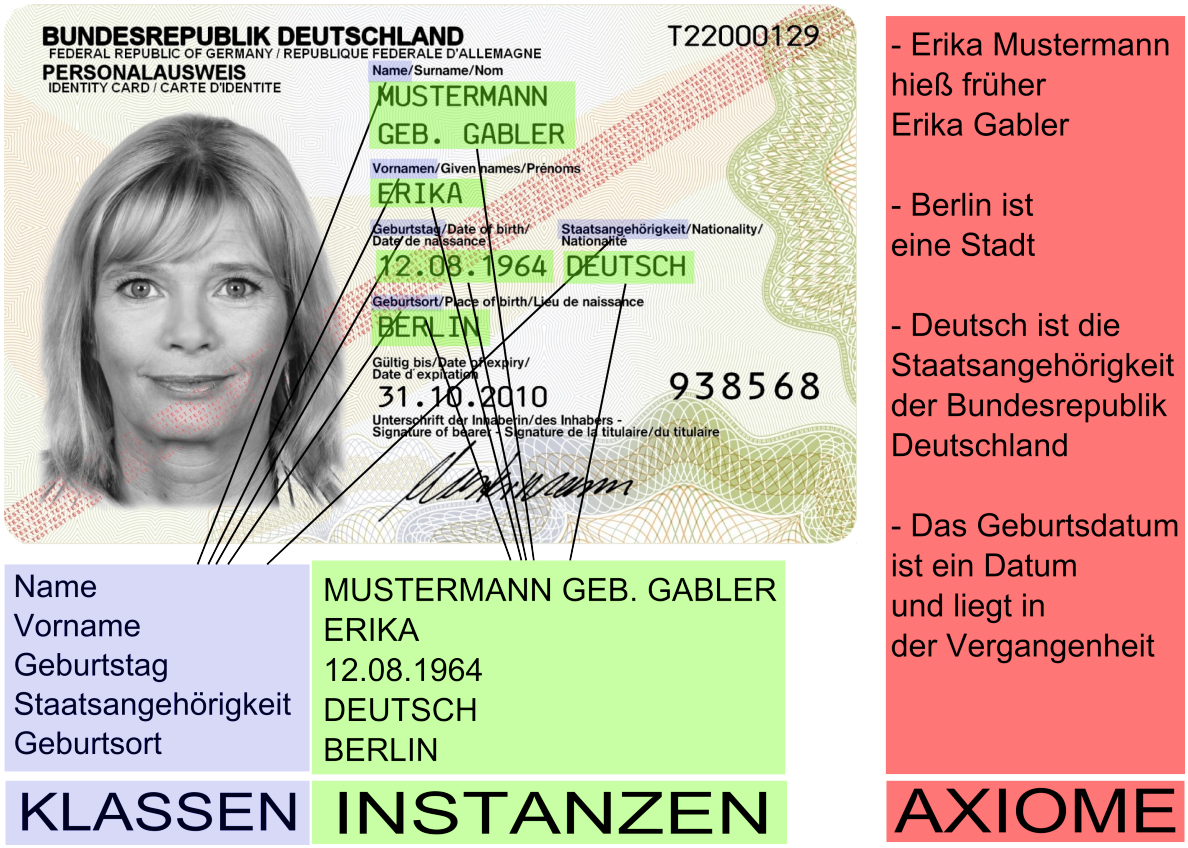

Abbildung 2.1: Daten eines Personalausweises. Quelle: BMI (Bundesministerium des Inneren)

Rot hinterlegt sind Zusammenhänge oder Tatsachen, die ein Mensch beim Lesen automatisch weiß und deswegen nicht extra wahrnimmt, die aber wichtige Informationen für die Bedeutung in sich tragen. Dies können Regeln, logische Schlüsse oder Hintergrundinformationen sein, die nicht unmittelbar mit der Domäne zu tun haben. Man bezeichnet diese Daten als **Axiome**.

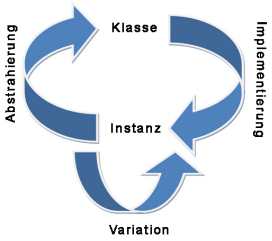

Abbildung 2.2: Zusammenhang zwischen Klassen und Instanzen

Klassen In den wenigsten Domänen fallen die Namen der Klassen, wie hier beim Personalausweis direkt vom Himmel. Möchte man eine Klassifizierung vornehmen, muss man zuerst die bereits vorhandenen Instanzen zuordnen. Dazu sortiert man sie nach ihren Eigenschaften. Diese Eigenschaften werden abstrahiert und einer Klasse zugeschrieben. Dann erhält die Klasse eine Bezeichnung, die möglichst exakt wiedergibt, welche Instanzen sie repräsentiert. Je weniger Eigenschaften eine Klasse besitzt, desto allgemeiner ist sie. Das bedeutet, dass es mehr Instanzen gibt, die dieser Klasse zuzuordnen sind. Eine sinnvolle Klasseneinteilung ist natürlich nur dann gegeben, wenn auch die Unterschiede zwischen den Eigenschaften zur Geltung kommen. Deswegen führt man Unterklassen ein, die die Eigenschaften der übergeordneten Klasse erben und diese durch weitere Eigenschaften weiter einschränken. Für den umgekehrten Weg von der Klasse zur Instanz sollen die Eigenschaften der Klasse als Bauplan für die Instanz dienen.

Vererbung und Hierarchie Für jede Instanz eine eigene Klasse zu erfinden, ist genauso wenig sinnvoll, wie alle Instanzen nur einer einzigen Klasse zuzuordnen. Letzteres wird in der Beschreibung der Instanzen sehr unscharf, da nur elementare Basiseigenschaften allen Instanzen gemein sind und sich die Klasse auf diese beschränken muss. In unserem Beispiel würde diese Klasse vermutlich „Personendaten“ heißen und nur die Eigenschaft „Enthält Daten, die für die Identifizierung oder Adressierung von Personen wichtig sind.“besitzen. Würde man hingegen den ersten Ansatz verfolgen, müsste man jede Eigenschaft, die zwei Instanzen gemein ist, auch in jeder dieser Klassen verankern. „Vorname“ und „Nachname“ unterscheiden sich in ihrem Aufbau nicht besonders, sind aber in ihrer Funktion verschieden. Hier würde es sich also anbieten, die gemeinsamen Eigenschaften in einer neuen Klasse „Name“ unterzubringen, und zu konstatieren, dass sowohl „Vorname“ als auch „Nachname“ Unterklassen von „Name“ sind. Hat jedes Element nur eine Oberklasse spricht man von Einfachvererbung, bei mehreren Oberklassen von Mehrfachvererbung.
Während eine Einfachvererbung in einer Baumstruktur dargestellt werden kann, ist es möglich, dass bei einer Mehrfachvererbung eine Raute entsteht, falls zwei Klassen, die die Oberklasse einer Dritten bilden, von derselben Klasse erben. Die Vorteile einer Baumstruktur liegen darin, dass der Pfad vom Blatt zur Wurzel eindeutig ist. Bei der Mehrfachvererbung kann man zwar komplexere Sachverhalte abbilden, die Datenstruktur wird dadurch aber unübersichtlicher. In der Objektorientierung gibt es das „Interface“ Konzept, um die Nachteile der Einfachvererbung auszugleichen. In Kapitel 5 wird eine vergleichbare Implementierung für das Semantic Media Wiki vorgestellt.

Eigenschaften Eine Klasse kann beliebig viele Eigenschaften besitzen. Diese Eigenschaften werden von Oberklasse zu Klasse vererbt; die Klasse besitzt somit automatisch alle Eigenschaften der Oberklasse(n). Eine Eigenschaft der Klasse „Geburtsdatum“ ist beispielsweise,

dass es nicht in der Zukunft liegen darf. Eine weitere Eigenschaften ist, dass es ein gültiges Datum sein muss. „Geburtsdatum“ würde weitere Eigenschaften von „Datum“ erben.

Bauplan Erstellt man unter Berücksichtigung aller Eigenschaften einer Klasse eine neue Instanz, kann diese Instanz der Klasse zugeordnet werden. In der Realität der Abbildung einer vorhandenen Domäne ist dieser triviale Zusammenhang nicht von Bedeutung. Mit der semantischen Erfassung der Domäne und den daraus resultierenden Möglichkeiten der Datenverarbeitung ist es durchaus denkbar, Instanzen generieren zu lassen und auf ihre Tauglichkeit zu testen. Dafür dient die Klasse als Bauplan.
Je genauer der Bauplan, desto kleiner ist die Variationsbreite der generierten Instanzen, desto größer ist aber auch die Chance, dass taugliche Lösungen generiert werden.

Instanzen Als Instanzen oder Fakten bezeichnet man die konkreten Ausprägungen einer Klasse: „Max“ und „Erika“ sind Instanzen von „Vorname“. Durch Instanzen gibt es erst eine Domäne. Wissen, das keine Ausprägung in Instanzen hat, braucht auch nicht abstrahiert werden; es gibt keine Eigenschaften und keine Regeln.
Während Klassen eher unbemerkt existieren, begegnen wir Instanzen ständig. Instanzen sind in der direkten Wahrnehmung: „Hallo, ich bin Max.“, während sich uns die Klassen erst dann eröffnen, wenn wir darüber nachdenken. Dass wir Menschen trotz allem intuitiv mit Klassen arbeiten, zeigt sich darin, wie wir uns Dinge merken: Wir bilden Klassen („Der heißt wie mein Freund Max.“) oder abstrahieren Eigenschaften aus Klassen („Max ist ein Männername.“). Lösungserfassung, ein Teil dieser Arbeit, bedeutet nichts anderes als dass man Instanzen findet und in die Klassifizierung einordnet.

Axiome Klassen hängen untereinander zusammen und können miteinander interagieren. Die Zusammenhänge in der Domäne werden durch Axiome ausgedrückt. Für den beschränkten Wissensausschnitt, den die Domäne repräsentiert, bilden Axiome zugleich die Abschlüsse nach außen hin. Ein Axiom stellt eine Tatsache fest die nicht begründet oder hinterfragt werden muss.
Diese Regeln zu finden und zu formulieren ist meist nicht leicht. Zum einen sind dem Domänenexperten viele Zusammenhänge so klar, dass er diese als gegeben sieht und beim Modellieren nicht beachtet, zum anderen ist es kaum möglich, alle Axiome zu finden. Eventuelle Lücken zeigen sich oft erst in der praktischen Anwendung der gesammelten Daten. Auch in der Personendaten-Domäne finden wir Axiome: Einige Personendaten wie der Nachname und die Anschrift sind veränderlich, andere - wie das Geburtsdatum - nicht. Eine weitere Regel, die im deutschen Sprachgebrauch zunehmend an Bedeutung verliert, ist dass sich die Anrede aus Geschlecht und Familienstand zusammensetzt: Die Folge davon ist, dass

das „Fräulein“ ausstirbt.
Fasst man die so strukturierte Domäne zusammen, spricht man von einer Ontologie.

Ontologie Dr. Sylvia Radeschütz beschreibt in ihrer Dissertation eine Ontologie folgendermaßen:
„Eine Ontologie stellt die eindeutige Wissensrepräsentation eines formal definierten Systems von Begriffen und ihren Beziehungen dar [SS04]. Eine Ontologie kann demnach durch ein Tupel $O = (C, H_C, P_C, I, R, A)$ definiert werden. Die Menge C aus der Ontologie sind prägnante Begriffe, die die Terminologie der Domäne beschreiben. Die Begriffe sind in einer Hierarchie H_C angeordnet, die durch Subkonzeptbeziehungen dargestellt wird. Begriffe sind durch Attribute P_C definiert, die ihre Beziehung mit anderen Begriffen oder Begriffen in einem angegebenen Wertebereich angeben. Instanzen I repräsentieren Objekte der Ontologie. Relationen in R beschreiben, welche Beziehungen zwischen den Begriffen bestehen. A beschreibt die Axiome. Das sind Regeln bestehend aus Begriffen, Attributen und Relationen, mit denen aus bereits definiertem Wissen neues Wissen abgeleitet werden kann.“[Rad11, S. 43 f.]

2.1.3 RDF/RDFS - Resource Description Framework und RDF Schema

RDF ist ein Standardmodell für Datenaustausch im Internet. Es setzt dabei auf die selben Paradigmen, die schon das World Wide Web zu dem gemacht haben, was es heute ist: Alles ist dezentral und jeder kann es beliebig erweitern. Dean Allemang drückt es treffend aus:„In the Semantic Web we refer to the things in the world as resources; a resource can be anything that someone might want to talk about.“[AH11] Sinngemäß: „Alles, worüber jemand reden wollen könnte ist eine Ressource.“ Betrachtet man unsere Beispiel-Personendaten, würden diese klassisch zentral vermutlich in einem Datenbankschema dargestellt, wie es in Abbildung 2.3 zu sehen ist.

Id	Name	Vorname	Geburtstag	Staatsangehörigkeit	Geburtsort
1	Mustermann geb. Gabler	Erika	12.08.1964	Deutsch	Berlin
2	Mustermann	Max	01.02.1976	Deutsch	Musterstadt

Abbildung 2.3: Personendaten als Datenbankeinträge

Stellt man sich nun vor, diese Daten müssten auf mehreren unterschiedlichen Systemen verteilt sein die nur lose miteinander verbunden sind, ergeben sich mehrere Probleme:

1. **Verteilung der Daten:** Eine Möglichkeit, die Daten aus der Datenbank in Abbildung 2.3 zu verteilen, ist die Daten zeilenweise auf den unterschiedlichen Rechnern zu

speichern. Somit würde jeder Rechner einen oder mehrere vollständige Datensätze verwalten, wie in Abbildung 2.4 zu sehen ist. Dieses Vorgehen fördert zwar die Flexibilität, bringt aber auch einen hohen Koordinationsaufwand mit sich. Wodurch wird sichergestellt, dass die Reihenfolge der Spalten auf jedem Rechner gleich bleibt?

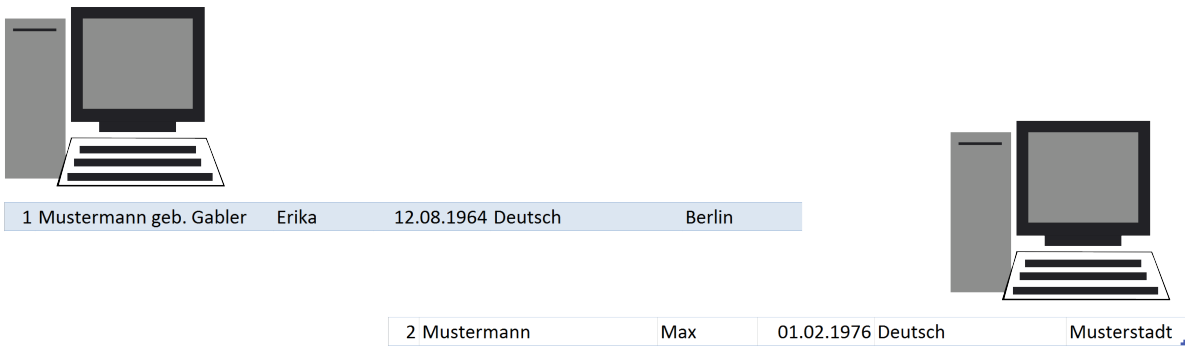

Abbildung 2.4: Mögliche Verteilung von Daten, Zeile um Zeile

Abbildung 2.5 zeigt ein alternatives Verteilungsschema: Hier werden die Informationen der einzelnen Spalten auf unterschiedliche Rechner verteilt. Diese Lösung ähnelt der Vorigen und hat wie sie auch das Problem, sicherzustellen, dass die „Datenreihe 2 auf Server 1“ auch der „Datenreihe 2 auf Server 2“ entspricht. Für das Hinzufügen eines Datensatzes ist ein großer Aufwand nötig.

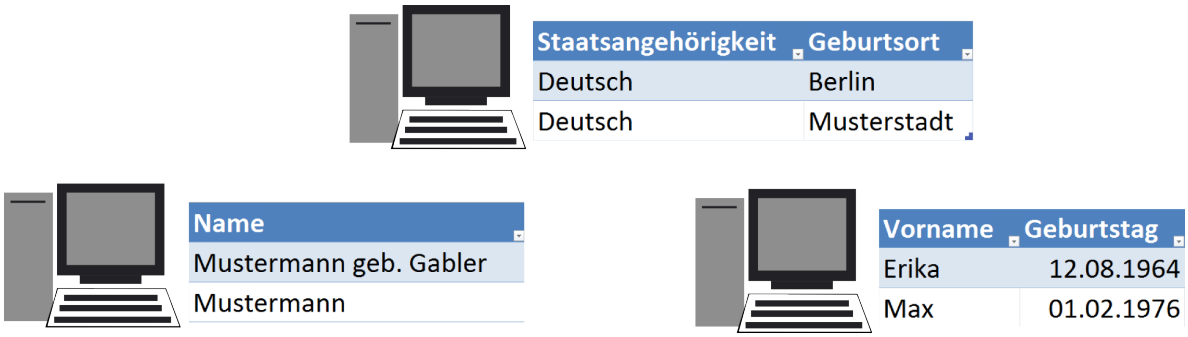

Abbildung 2.5: Mögliche Verteilung von Daten, Spalte um Spalte

Eine Verteilung, die der Anforderung „jeder soll alles über jedes Thema sagen dürfen“ gerecht werden will, muss die Vorteile dieser beiden Ansätze vereinigen. Der in Abbildung 2.6 vorgestellte Ansatz tut dies. Allerdings kauft man sich dabei auch die Nachteile der beiden anderen Ansätze ein: Für jede Zelle muss man Informationen zu Zeile und Spalte transportieren.
Für diese drei Informationen bietet sich die Darstellung als Tripel an: Zeile, Spalte, Wert. Diese Tripel werden auch, folgend dem englischen Satzaufbau

mit Subjekt, Prädikat und Objekt betitelt. Beispielsweise wäre ein solches Tripel: `[Zeile] 2 [hat] Geburtstag [am] 01.02.1976.`

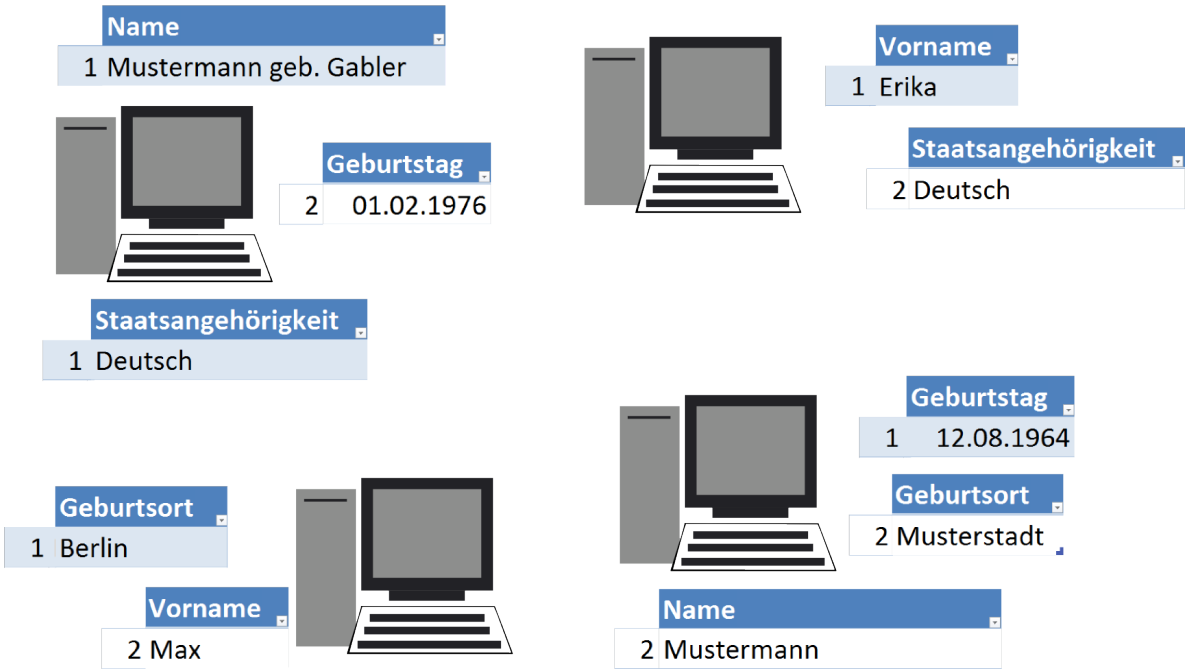

Abbildung 2.6: Mögliche Verteilung von Daten, Zellenweise

2. **Zusammenfügen der verteilten Daten:** Wenn man nun Tripel mit gleichem Subjekt oder Objekt findet, lassen sich diese zusammenfassen. Daraus lässt sich ein gerichteter Graph wie in Abbildung 2.7 erstellen.

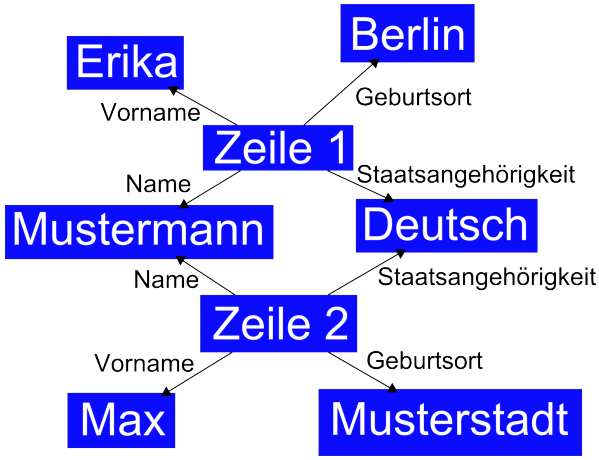

Abbildung 2.7: Graph aus zusammengefügten Daten

So lassen sich nun nicht nur die eigentlich zusammengehörigen Daten zusammenfügen, sondern auch beliebige andere. Berlin als Entität bietet beispielsweise viele Anknüpfpunkte. Hierin liegt die größte Stärke von RDF, denn nun ist jeder in der Lage diesen Graphen zu erweitern. Das kann einen Wachstumseffekt wie den des Internet bewirken.

3. **Identifikation von Daten:** Ein letztes Problem bleibt bestehen: Wo findet man in einem hochgradig verteilten und dezentralen Netz immer die richtige Ressource? Die Antwort darauf liefert das Internet gleich mit: Der Uniform Resource Identifier (URI) weißt jeder Ressource im Internet eine eindeutige Id zu. Das allerdings liegt in der Verantwortung der Betreiber der jeweiligen „Authority", also den Betreibern der Website, auf der die Entitäten hinterlegt sind. Die Definition des URI ist als Request for Comments (RFC) unter der Nummer RFC3986 bei der Internet Engineering Task Force (IETF) hinterlegt.

Sprachkonstrukte von RDF: RDF liefert bei weitem keine vollständige Beschreibungssprache für Ressourcen. Tatsächlich definiert es nur einige wenige Begriffe:

- `rdf:type`
- `rdf:Property`
- `rdf:Statement`
- `rdf:subject`
- `rdf:predicate`
- `rdf:object`
- `rdf:Bag`
- `rdf:Seq`
- `rdf:Alt`
- `rdf:value`
- `rdf:List`

Ein Ausdruck in RDF wird mit seiner vollen URI in spitzen Klammern dargestellt: Die Information „Berlin liegt in Deutschland" könnte also wie folgt aussehen:

```
<http://www.example.com/Resources/Geo/Cities#Berlin>
<http://www.example.com/Resources/Location#Lies_within>
<http://www.example.com/Resources/Geo/Countries#Germany>
```

Da diese Schreibweise sehr lang und wenig übersichtlich ist, bietet RDF die Möglichkeit, ähnlich wie in eXtended Markup Language (XML) Namensräume zu definieren. Außerdem gibt es noch eine Kompaktschreibweise namens *Turtle*. Die gängigen Anwendungen, die mit RDF arbeiten, akzeptieren auch Eingaben in Turtle.

RDFS - RDF Schema: Bei RDFS handelt es sich um eine Erweiterung von RDF, die einige weitere grundlegende Konzepte umsetzt. Mit den Begriffen

- `rdfs:subClassOf` (Unterklasse von)
- `rdfs:subPropertyOf` (Unterattribut von)
- `rdfs:domain` (Definitionsbereich)
- `rdfs:range` (Wertebereich)

ermöglicht RDFS eine Ableitung vieler neuer Fakten. So kann man in RDFS beispielsweise ein Attribut „hatMädchenName“ definieren, und diesem Attribut einen Definitionsbereich „VerheirateteFrau“ zuweisen. „VerheirateteFrau“ ist natürlich eine Unterklasse von „Frau“. Dadurch ist bei der Angabe eines Tripels „:Erika :hatMädchenName "Gabler"“ ein Schluss auf „:Erika :type :Frau“ möglich. Durch den Klassenbegriff werden auch einfache Mengenoperationen auf Tripeln möglich.

2.1.4 OWL - Web Ontology Language

Auch OWL basiert auf der Syntax von RDF. Bei OWL handelt es sich um eine Erweiterung der RDF Sprachkonstrukte hin zu einer Sprache, mit der auch komplexe prädikatenlogische Aussagen getroffen werden können. Als Vorgänger von OWL, das seit 2001 vom World Wide Web Consortium (W3C) als Empfehlung ausgegeben wurde, gelten die beiden Standards DARPA Agent Markup Language (DAML) und Ontology Inference Layer (OIL). Beide basieren ebenfalls auf RDF.
OWL liegt mittlerweile in der zweiten Version vor und ist der de facto-Standard für Ontologie-Sprachen. Es erweitert RDFS um viele Funktionen. Hier sollen nur einige wichtige Sprachkonstrukte erwähnt werden:

- `owl:Restriction` - Beschreibt eine Klasse indem es die erlaubten Werte eines bestimmten Attributs definiert.
- `owl:hasValue` - Definiert einen bestimmten Wert, den ein Attribut haben muss.
- `owl:someValuesFrom` - Definiert eine Menge von Werten, die ein Attribut annehmen kann.
- `owl:allValuesFrom` - Definiert alle Werte, die ein Attribut annehmen kann.
- `owl:onProperty` - Stellt eine Verknüpfung zwischen einer Restriktion und dem betroffenen Attribut her.

Damit lassen sich in Kombination mit den RDFS Sprachkonstrukten weitgehende Schlüsse ziehen und auch komplexe Zusammenhänge modellieren. Beispielsweise können automatisch Wahlunterlagen an eine Adresse versandt werden, die einer Person zugeordnet wird, die die deutsche Staatsangehörigkeit hat und zum Zeitpunkt der Wahl volljährig ist.

2.1.5 Abfragesprache SPARQL

Das Semantic Web brächte keinen Nutzen, wenn man die hinterlegten Tripel nicht auch abfragen und auswerten könnte. Dafür wurde die Abfragesprache SPARQL Protocol And RDF Query Language (SPARQL) entwickelt. Sie basiert auf einem sehr einfachen Prinzip: Was ich weiß gebe ich an, was ich suche versehe ich mit einem Fragezeichen.

Angenommen, der RDF Datengraph besteht aus folgenden Tripeln:[1]

```
:Erika :geborenIn :Berlin .
:Max :geborenIn :Musterstadt .
:Erika :geborenAm "1964-08-12"^^xsd:date .
:Max :geborenAm "1976-02-01"^^xsd:date .
:Berlin :liegtIn :Deutschland .
:Musterstadt :liegtIn :Deutschland .
:Deutschland :liegtIn :Europa .
:Berlin rdf:type :Stadt .
:Musterstadt rdf:type :Stadt .
```

Eine Abfrage kann nach Subjekt, Prädikat oder Objekt fragen. Dazu gibt man lediglich eine Variable anstelle des gesuchten Teils in die Suchabfrage ein.

Die folgende Abfrage 2.1 zeigt alle (im Datengraph bekannten) Städte in Deutschland an:

```
SELECT ?Stadt
  WHERE
  { ?Stadt :liegtIn :Deutschland . }
```

Listing 2.1: Abfrage nach Städten in Deutschland

?Stadt
Berlin
Musterstadt

Genauer gesagt zeigt diese Abfrage alles an, was ein Attribut „liegtIn“ mit dem Wert „Deutschland“ hat. Möchte man alle Ressourcen die mittels dieses Attributs verbunden sind ermitteln, muss man wie in Listing 2.2 auch das Objekt durch eine Variable ersetzen.

```
SELECT ?Was ?Wo
  WHERE
  { ?Was :liegtIn ?Wo . }
```

[1] Hier wird die verkürzte Turtle Schreibweise (siehe Abschnitt 2.1.3) verwendet

Listing 2.2: Abfrage nach Attribut „liegtIn“

?Was	?Wo
Berlin	Deutschland
Musterstadt	Deutschland
Deutschland	Europa

Diese Abfrage lässt sich nun durch die Bedingung, dass die Subjekte eine Stadt sein müssen, weiter eingrenzen.

```
SELECT ?Was ?Wo
  WHERE
  { ?Was :liegtIn ?Wo .
    ?Was rdf:type :Stadt .}
```

Listing 2.3: Abfrage nach Attribut „liegtIn“ mit Eingrenzung Stadt

?Was	?Wo
Berlin	Deutschland
Musterstadt	Deutschland

Das Grundprinzip von SPARQL ist, dass ein Suchgraph aufgebaut wird, der mit dem Datengraph abgeglichen wird. Auf diesem Prinzip lassen sich auch komplexe Abfragen durchführen. Dazu liefert SPARQL noch einige weitere Mechanismen, wie Transitivität, das Filtern der Ergebnismenge und viele aus anderen Query Languages bekannte Aggregatsfunktionen. Im letzten Beispiel (Listing 2.4) dieser kurzen Einführung wird die Transitivität auf das Attribut „liegtIn“ angewandt (mit * gekennzeichnet), um alle Personen zu finden, die einen Geburtstag in Europa haben, dessen Datum nach 1970 liegt. Erwartungsgemäß gibt die vorhandene Datenmenge auf diese Abfrage genau einen Datensatz zurück:

```
SELECT ?Name ?Geburtsdatum
  WHERE
  { ?Ort :liegtIn* :Europa .
    ?Ort rdf:type :Stadt .
    ?Name :geborenAm ?Geburtsdatum
    ?Name :geborenIn ?Ort
    FILTER (?Geburtsdatum > 1970-01-01^^xsd:date)
  }
```

Listing 2.4: Abfrage nach Geburtsdaten in Europa nach 1970

?Name	?Geburtsdatum
Max	1976-02-01

2.1.6 Semantische Wikis

Der Masseneffekt, den Wikipedia bewirkt hat, ist unvergleichlich: Millionen von Menschen arbeiten freiwillig und unentgeltlich an einer gemeinsamen Wissensplattform. Die große Befürchtung, dass die Qualität der Artikel schlecht oder schlecht recherchiert ist, hat sich nicht bewahrheitet. Durch einfache Regeln, Nachvollziehbarkeit jeder Änderung und einige freiwillige Administratoren, hält sich die Anzahl schlechter Artikel sehr in Grenzen. Dies ist nicht zuletzt auch so, weil ein großes öffentliches Interesse darin besteht, dass die Informationen in Wikipedia vertrauenswürdig sind.
Diese Wikis (Wikipedia ist nur das populärste Beispiel) sind für Menschen optimiert. Sie tragen als ein wichtiger Bestandteil zum „herkömmlichen“ Internet bei. Für das Semantic Web sind diese Wikis jedoch nicht gemacht. Wie alle herkömmlichen Webseiten enthalten sie Informationen in textueller Form, deren Bedeutung sich einer Maschine entzieht.[2] Dabei sind gerade Wikis für das Semantic Web sehr spannend, denn das sind die Orte an denen Wissen abgelegt wird. Folglich ist also klar, dass auch das Semantic Web seine Wikis braucht: Eben semantische Wikis.

Wichtige Anforderungen an ein semantisches Wiki für Funktionalität und Akzeptanz sind:

- Intuitives Mapping zwischen Tripeln und Wikidarstellung
- Ansprechende Darstellung der Inhalte für den Menschen
- Unkomplizierte Eingabe von semantischen Daten
- Im- und Export von und nach RDF / OWL
- Unterstützung von Suchfunktionen
- Umsetzung der OWL-Sprachkonstrukte
- Leichte Erweiterbarkeit durch Plugins

Weitere Anforderungen sind eine möglichst weite Verbreitung und eine große Benutzergemeinde, sowie guter Support und gute Dokumentation.

[2] Eine Ausnahme dazu bildet dbPedia.org. Hier werden die Inhalte der Faktenboxen auf Wikipedia als semantische Daten zur Verfügung gestellt.

2.2 Grundlagen - Muster

2.2.1 Muster nach Christopher Alexander

Christopher Alexander, ein Architekt, hat mit seinem Standardwerk „A Pattern Language - Towns, Buildings, Construction“[AIS77] nicht nur einen Meilenstein in der Architektur gelegt, sondern auch das Denken in der Informatik grundlegend verändert. Ihm ist es zu verdanken, dass man von Software-Architektur spricht. Nicht genug, dass sich Software und Architektur in ihren Entwurfsprozessen sehr ähneln - es lassen sich auch die von Alexander beschriebenen Patterns auf die Software übertragen.
Alexander beschreibt in seinem Buch städtebauliche Herausforderungen und zerlegt diese geschickt in Teilprobleme, die an verschiedensten Stellen wieder auftreten können. Für diese Herausforderungen erarbeitet er elegante Lösungen, erfasst Voraussetzungen und listet die Probleme auf, für die dieser Ansatz eine Lösung ist. Die Patterns die Alexander beschreibt gehorchen alle der folgenden fünfgliedrigen Form[Lea94]:

1. **Name:** Ein aussagekräftiger Name.
2. **Example:** Ein Beispiel für das Vorkommen, mit Bildern oder Zeichnungen verdeutlicht.
3. **Context:** Zeigt Einsatzort, Hintergründe und Diskussion dieses Patterns auf.
4. **Problem:** Hier werden die Kräfte und Einschränkungen beschrieben und wie sie interagieren. Auch Design und Konstruktion können Teile des Problems darstellen.
5. **Solution:** Beschreibung, wie eine Lösung zu konstruieren ist; oft unter Zuhilfenahme von Subpatterns und in mehreren Variationen.

Weiterhin zeichnet sich ein Pattern nach Alexander laut Lea [Lea94] durch folgende Eigenschaften aus:

- **Encapsulation:** Das Problem ist nach außen hin abgegrenzt. Die Lösung stellt sicher, dass das Problem im Ganzen auch in Details bearbeitet wird.
- **Generativity:** Das Pattern enthält eine Prozessbeschreibung, mit deren Hilfe sich Instanzen erzeugen lassen.
- **Equilibrium:** Das Pattern muss in der Problembeschreibung eine Invariante erhalten, die den Konflikt zwischen Kräften und Einschränkungen minimalisiert. In jedem Designschritt muss die Designentscheidung durch diese Invariante begründbar sein. Dies ist ein Gratwanderung und wird oft nicht erreicht.
- **Abstraction:** Das Pattern abstrahiert konkrete Vorkommnisse.

- **Openness:** Ein Pattern kann beliebig detailliert sein; es gibt keine Untergrenze für die Feingranularität des Patterns. Das meint Lea mit Offenheit.
- **Composibility:** Die hierarchische Struktur von Patterns muss sich dadurch ausdrücken, dass Patterns zu neuen Superpatterns komponiert oder in Subpatterns zerlegt werden können.

2.2.2 Muster in der Informatik

Lösungen zu alltäglichen Softwareproblemen haben Gamma, Helm, Johnson und Vlissides in ihrem Buch „Entwurfsmuster: Elemente wiederverwendbarer objektorientierter Software“[GHJV96] zum Standard erhoben. Sie werden oft auch als die „Gang of Four (GoF)“ bezeichnet und ihr Buch nicht selten mit GoF-Book referenziert. Sie legen eine detailreichere Struktur zur Notation von Patterns zugrunde und füllen damit die weiteren Forderungen von Alexander (vgl.: Abschnitt 2.2.1) ebenfalls mit Leben. Auf nähere Einzelheiten zu dieser Struktur wird diese Arbeit nicht eingehen.
Nahezu alle neueren Entwicklungen im Bereich Programmier-Patterns können in einem der Konferenzbände der Konferenz Pattern Language of Programs (PLoP) nachverfolgt werden. PLoP stellt unter anderem auch Empfehlungen zum Entwickeln von Entwurfsmustern (Patternwriting) bereit. Dieses Vorgehen soll neben anderen Pattern-Identifikationsprozessen im nächsten Abschnitt näher betrachtet werden. Diese Betrachtungen werden auch dieses Grundlagenkapitel abschließen und nahtlos in das konzeptionelle Vorgehen beim Pattern-Identifikationsprozess in der Kostümdomäne überleiten.

2.3 Kostüme im Film

Obgleich es befremdlich scheint, einen Abschnitt über Kostüme beim Film in einer Arbeit über Musteridentifikation und semantische Wikis zu schreiben, sind doch die Kostüme ein zentraler Punkt dieser Arbeit. Im Zuge einer zunehmenden Durchdringung der Welt durch digitale Informationssysteme rücken auch immer mehr Teile dieser bis dato fremden Welt in den Fokus der Informatik. Das Schlagwort „Digital Humanities“ spielt hier eine immer größere Rolle. Rollen im Film unterliegen Mustern. Diese drücken sich vor allem durch die gewählte Kostümierung der Darsteller aus. Dieses Thema ist ein gemeinsames Forschungsinteresse an der Universität zu Köln und der Universität Stuttgart. Die Autoren von „A Pattern Language for Costumes in Films“[SBLE12] und „Taxonomien kostümrelevanter Parameter: Annäherung an eine Ontologisierung der Domäne des Filmkostüms“[Ba13] schlagen eine Mustersprache für Kostüme vor. Darin beschreiben sie, wie mit klassischen Methoden der Informatik eine Erfassung und Auswertung von Kostümmustern geschehen kann. Sie führen

aus, dass es bisher für Kostüme noch keine einheitliche Beschreibungssprache gibt.[SBLE12, S. 7] Dadurch mangelt es auch an DV-Unterstützung. Kostümbildner und Kostümforscher behelfen sich derzeit mit klassischen Methoden (Sichten von Filmmaterial und Kostümfundus), bzw. benutzen Kostüm-Verwaltungssoftware, die keinen hohen Detaillierungsgrad unterstützt, um einen Film auszustatten oder Forschung zu betreiben. Viele Fragestellungen zu Kostümen könnten beantwortet werden, wenn eine entsprechende Datenbasis zur Verfügung stünde.
Die Domäne Kleidung ist jedoch sehr umfangreich und auch die Anzahl an Filmen ist immens. Die Autoren planen daher, auf einem begrenzten Filmkorpus Untersuchungen durchzuführen. Dafür wird ein Informationssystem benötigt. Abbildung 2.8 zeigt die Anforderungen an ein solches System.

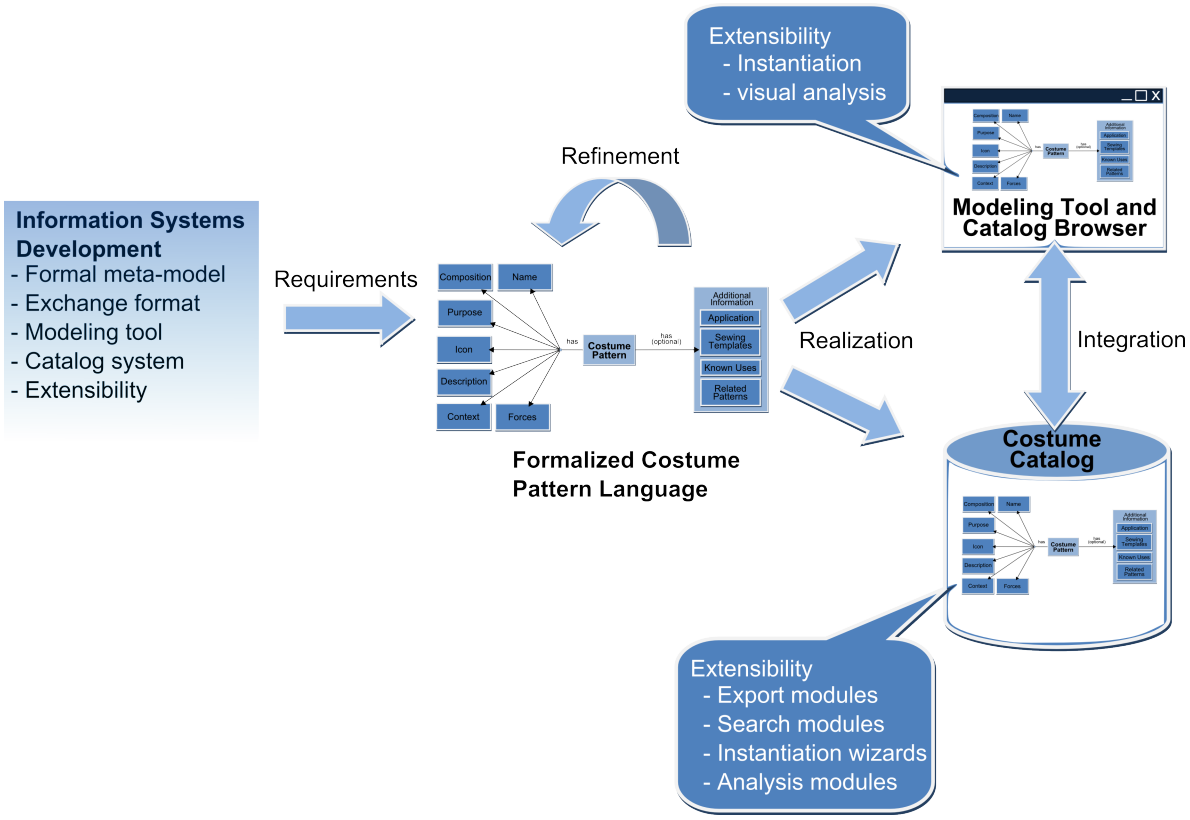

Abbildung 2.8: Anforderungen an ein Informationssystem, angelehnt an [SBLE12]

Die Anforderungen sind ein Katalogsystem zu schaffen, in dem Kostümprimitive und Kostümmuster in einer formalisierten Sprache abgelegt werden können. Desweiteren soll ein Modellierungswerkzeug und eine Anwendung zum Durchsuchen des Katalogsystems entstehen. Die Daten sollen durchsuchbar sein und exportiert werden können.
In RDF gespeicherte Daten erfüllen die Kriterien an Durchsuchbarkeit und Exportmöglich-

keiten. Für das Modellieren der Domäne bietet sich eine Ontologiesprache an. Diese Gründe sprechen für eine Lösung im Semantic Web. Ein Katalogsystem spricht für den Einsatz eines Wikis.

Vestimentäre Kommunikation „Kleider machen Leute!“ Dieses alte Sprichwort stimmt heute genauso wie vor 100 Jahren. Heute weiß man sogar, dass Kleider Botschaften transportieren können. Für den Film ist diese vestimentäre (vestimentär: die Kleidung betreffend) Kommunikation sogar noch bedeutender. Kleider schaffen es, auf den ersten Blick einen Eindruck einer Rolle zu vermitteln. Das ist insbesondere dann wichtig, wenn das Drehbuch keine Zeit vorsieht, den Charakter langsam zu entwickeln. Jede Rolle in einem Film spielt im wahrsten Sinn des Wortes eine Rolle und erfüllt somit eine Funktion. Je weniger die Rolle zu sagen hat, umso mehr muss ihre Kleidung die Funktion der Rolle transportieren. Wie sich Kostümierung und Charaktereigenschaften zueinander verhalten ist eine der spannenden Fragen in der Kostümforschung.

2.4 Muster-Identifikationsprozesse

Ein zentrales Thema dieser Arbeit ist das Identifizieren bzw. Entwickeln von Entwurfsmustern. Die hier betrachteten Identifikationsprozesse stammen aus unterschiedlichen Domänen und agieren zum Teil auch auf unterschiedlichen Ebenen. Sie wurden ausgewählt, weil sie einen interessanten Querschnitt durch die Abstraktionsebenen bilden.

2.4.1 Identifikationsprozesse bei Cloud-Computing Patterns

Dieser Abschnitt zeigt das Vorgehen am praktischen Beispiel der Cloud-Computing Domäne wie es in „Capturing Cloud Computing Knowledge and Experience in Patterns“[FEL+12] beschrieben wird:

1. **Customization of the Pattern Format:** In „Capturing Cloud Computing Knowledge and Experience in Patterns“[FEL+12] wird das Patternformat von Alexander zugrunde gelegt und um einige Strukturpunkte erweitert:
   - Icon: Ein Icon, das den Wiedererkennungswert des Patterns erhöht und nach Möglichkeit bereits dessen Inhalt andeutet.
   - „Example“ wird zu „Driving Question“: Bietet einen schnellen Überblick über die wichtigsten Beweggründe für das Pattern.
   - Scetch: Eine Zeichnung, die die Lösung darstellt.

- Variations: Hier werden unterschiedliche Varianten des Patterns vorgestellt.
- References: Dieser Abschnitt bildet die Beziehung zu anderen Patterns ab: Komposition (Zusammensetzung mehrere Pattern), Verfeinerung (Detailierung eines abstrakten Pattern) und Generalisierung (Abstraktion eines konkreteren Pattern - beispielsweise einer speziellen Implementierung).
- Known Uses: Zeigt praktische Anwendungen des Musters.
- Annotations: Hier können weiterführende Hilfen für den Leser untergebracht werden.

2. **Collection of Information Sources:** Stellt eine Sammlung von Quellen für Patterns in Form von Links zu Provider-Anleitungen, Büchern und Journalartikeln dar.
3. **Classification of Information Sources:** Es erfolgt die Einteilung der Informationen bezüglich ihrer Architekturdomäne in Klassen.
4. **Abstraction of Architectural Concepts:** In diesem Schritt werden die in den Klassen enthaltenen Daten abstrahiert, wobei providerspezifische Details entfernt werden.
5. **Creation and Classification of Patterns:** Die Patterns werden aus den abstrakten Modellen komponiert. Dabei ist die schwierigste Aufgabe, das richtige Maß an Abstraktion walten zu lassen, um einerseits möglichst dem speziellen Problem gerecht zu werden, andererseits aber auch die unterschiedlichen Implementierungen für dieses Problem zu vereinen.
6. **Iterative Improvement:** In diesem Prozessschritt wird das Pattern immer wieder in der praktischen Anwendung auf Tauglichkeit und Verständlichkeit getestet und bei Bedarf angepasst.

Die praktische Erfahrung hat gezeigt, dass sich dieses Vorgehen für die Erfassung von Cloud-Computing Pattern gut eignet.

2.4.2 Der Pattern Evolution Process

René Reiners beschreibt in „A Pattern Evolution Process - From Ideas to Patterns“[Rei12] ein Katalogsystem zur Verwaltung und Verbesserung von Patterns. Der Ansatz dabei ist, so früh wie möglich die Community mit in den Pattern-Verbesserungsprozess hinein zu nehmen. Durch dieses Vorgehen möchte Reiners den Prozess, der „top down“ stattfindet, also von einer kleinen Gruppe entwickelt wurde und anschließend der breiten Masse zugänglich gemacht wird, umkehren.
In einem ersten Schritt wird automatisch geprüft, ob alle geforderten Felder ausgefüllt

sind. Ist dies der Fall wird das Pattern mit einem zusätzlichen Statusattribut versehen, das ausdrückt, wie die Entwicklung über die Zeit stattgefunden hat:

- **just created:** Der initiale Status, der aussagt, dass das Pattern noch nicht überprüft wurde.
- **under consideration:** Das Pattern ist vielversprechend, bedarf aber weiterer Überprüfung.
- **pattern candidate:** Dieses Pattern ist kurz davor, anerkannt zu werden.
- **approved:** Ein Pattern mit diesem Status ist als Design Pattern anerkannt.

Stellt sich heraus, dass ein Patternkandidat nicht zum Pattern taugt, kann es vermutlich als „Antipattern“ noch gute Dienste tun. Dieser Ansatz setzt ebenfalls auf die Effekte, die von Wikipedia bekannt sind, wenngleich sicherlich die Zahl derer, die an einem Softwarepattern mitarbeiten um einige Größenordnungen kleiner ist. Reiners erhofft sich von diesem Prozess mehr Patterns, deren Potential zu einem früheren Zeitpunkt abschätzbar ist.

2.4.3 Muster für Muster - Die Sprache des Hirten

In diesem Journalbeitrag mit dem ungewöhnlichen Titel „The Language of Shepherding“ citeshepherd beschreibt Neil Harrison, wie das Anleiten eines Patternentwicklers (Schaf) durch einen erfahrenen Patternentwickler (Hirte) den Pattern-Entwicklungsprozess unterstützen kann. Den Vorgang dieses Anleitens bezeichnet er als „Shepherding“. Harrison betont, wie wichtig es ist, dass der Anleitende nicht nur beratend zur Seite steht, sondern auch einige Schritte in die Wege leitet, die den Prozess für den Patternschreiber vereinfachen sollen. Diese Schritte beziehen sich auf *Patterns for Patternwriting*, also Muster, mit deren Hilfe man selbst wieder Muster schreiben kann. Diese Muster unterteilt er dabei in zwei Gruppen: Muster, die dazu dienen, dem Hirten die Arbeit leichter zu machen und den Prozess des Shepherding zu vereinfachen und Muster, die direkt Einfluss auf das Pattern haben. An dieser Stelle werden nur letztere kurz vorgestellt:

- **Big Picture:** Ziel dieses Patterns ist, dass sowohl Autor als auch Hirte das zentrale Problem des Patterns verstehen und dabei an das Gleiche denken. Die Idee des Patterns soll dabei beim einmaligen Lesen des Problems und der Lösung klar werden.
- **Matching Problem and Solution:** Oft sind Problem und Lösung nicht deckungsgleich. Das sollen sie allerdings sein. Das Problem sollte komplett durch die Lösung erfasst sein und die Lösung sollte nicht mehr als das Problem lösen. Hier ist eine Balance zwischen Lösung und Problem gesucht, wobei in einem lösungsorientierten Ansatz die Lösung der entscheidende Teil ist und eher das Problem angepasst werden sollte.

- **Convincing Solution:** Die Lösung des Problems muss überzeugend sein. Der Hirte soll darauf achten, dass die Lösung so auch existiert. An mancher Stelle muss der Autor die Lösung etwas schmälern, da er sonst Gefahr läuft, dass nicht alle Fälle seiner breiten Lösung funktionieren.
- **Forces Define Problem:** Die Problem-Sektion des Pattern ist oft nicht klar genug formuliert. Die Kräfte-Sektion des Pattern kann helfen, das Problem besser zu verstehen, indem der Leser versteht, welche Überlegung welcher Entscheidung zugrunde liegt. Dafür muss der Autor in mehreren Iterationen zwischen Problem und Kräften vermitteln.
- **Balanced Context:** In der Kontext-Sektion sollte sowohl der Ausgangszustand stehen, auf den das Pattern anwendbar ist, als auch das Ergebnis nach der Anwendung des Pattern. Um hier einen vernünftigen Ausgleich zu erhalten ist es sinnvoll „vorher“, die Lösung und „nachher“ zu betrachten. Es hilft auch, wenn man „vorher“ mit dem Problem und „nachher“ mit der Lösung vergleicht.
- **War Stories:** Unter den Kriegsgeschichten versteht Harrison ein Beispiel von einem echten Einsatz des Patterns. Oft leidet die Lesefreude an der mangelnden Fähigkeit des Autors, zu begeistern. Bei Kriegsgeschichten hören die meisten jedoch aufmerksam zu. Diesen Effekt soll sich der Autor zu eigen machen.
- **Form Follows Function:** Es gibt viele unterschiedliche Patternformate. Meist bleibt der Autor bei dem Pattern, das er bereits kennt. Dabei soll sich die Form dem Pattern anpassen und nicht das Pattern der Form. Der Hirte kann darauf hinarbeiten, indem er nicht plump ein neues Patternformat vorschlägt, sondern den Autor neue Abschnitte herausarbeiten lässt.
- **Small Patterns:** Ziel des letzten Patterns ist es, ein Pattern klein und übersichtlich zu halten. Das ist allerdings während des Shepherding-Prozesses nicht möglich, da von außen viele Anregungen an den Autor kommen und das Pattern sich so erst einmal aufbläht. Nun kommt zum Ende des Prozesses die Anforderung, alles was nicht essentiell ist, aus dem Pattern zu entfernen.

3 Konzept zur Erfassung einer Musterdomäne

Dieses Kapitel beschreibt den Identifikationsprozess von Patterns und Lösungen und deren Darstellungsform in der Kostümdomäne. Desweiteren werden die Zuständigkeiten rund um die Erfassung in einem Rollenmodell erläutert. Im letzten Abschnitt werden die einzelnen Vorgänge bei der Erfassung und Musteridentifikation geschildert. Außerdem wird gezeigt, wie sich ein Teil dieses Prozesses automatisieren lässt. Wie sich diese Konzepte in Semantic Media Wiki umsetzen lassen, wird in Kapitel 5 beschrieben.

3.1 Identifikationsprozesse in der Kostümdomäne

Der hier dargestellte Prozess basiert in großen Teilen auf [Ba13].

3.1.1 Zugrundeliegendes Rollenmodell

Für die Kostümdomäne gibt es folgende Rollen:

- Kostümexperte
- Wikiexperte
- Lösungserfasser
- Kunden: Kostümverleih, Kostüm-Forscher, Kostümbildner
- Bot

Abbildung 3.1 verdeutlicht die Zusammenhänge.

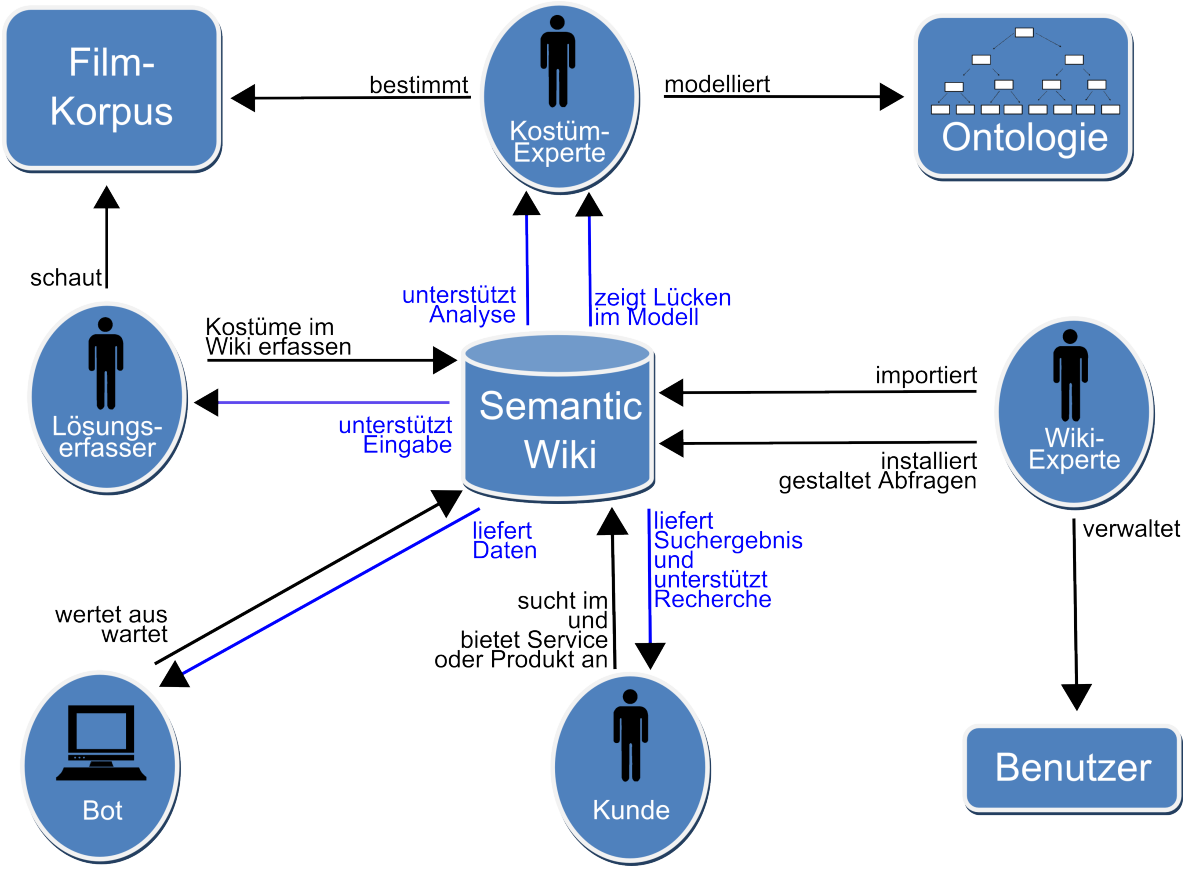

Abbildung 3.1: Das Rollenmodell in der Kostümdomäne

Kostümexperte Der Kostümexperte verfügt über vollständiges Domänenwissen und ist in der Lage, dieses Wissen in Taxonomien zu erfassen. Die Aufgabe des Kostümexperten ist es nicht nur, die Ontologie seiner Domäne zu modellieren, sondern auch, einen Filmkorpus für die Lösungserfasser vorzugeben und deren Arbeit zu prüfen, bzw. darauf zu achten, dass die Ontologie, sollte sie Lücken aufweisen, ergänzt wird. Hierbei wird er vom Wiki unterstützt. Der Kostümexperte pflegt auch bereits bekannte Muster nach dem in Abschnitt 3.1.2 beschriebenen Verfahren ein.

Wikiexperte Der Wikiexperte steht in enger Zusammenarbeit mit dem Kostümexperten. Er ist dafür verantwortlich, dass ein geeignetes Wiki installiert wird und dass die Domänendaten aus der Ontologie ins Wiki importiert werden können. Er muss sicherstellen, dass die Eingabe von Kostümen geführt erfolgt. Außerdem muss er die Restriktionen und zusätzlichen Zusammenhänge in der Domäne modellieren, die nicht durch einen Taxonomie-Import gelöst werden können. Er gestaltet nach den Vorgaben des Kostümexperten Abfragen im Stil

von Reportingseiten, die wichtige Fragestellungen aufgrund der aktuell gesammelten Daten beantworten. Er ist auch dafür verantwortlich, dass die Lösungserfasser mit entsprechenden Zugriffsrechten ausgestattet werden. Dabei muss er beachten, dass die Ontologiedaten vor unbefugtem Zugriff geschützt bleiben. Nur der Kostümexperte darf an der Ontologie modellieren.

Lösungserfasser Der Lösungerfasser sollte ein grundlegendes Verständnis von Kostümen und Mode mit sich bringen, aber auch die Fähigkeit, Daten in einem Wiki zu erfassen. Seine Aufgabe besteht in der Durchsicht der Filme des Filmkorpus und der Erfassung der darin vorkommenden Kostüme. Die Hauptschwierigkeit besteht dabei in der Identifikation der verschiedenen Kleidungsstücke, die für eine spätere Auswertung essentiell ist. Ein weiterer wichtiger Punkt, der dieser Rolle zukommt, ist die Bestimmung der Relevanz des einzelnen Kleidungsstücks für ein Kostüm. Dies kann Faktoren wie die Art der Darstellung, die Zeitspanne in der das Element zu sehen ist oder auch Relevanz in der Funktionalität oder für den Ablauf der Handlung beinhalten.
Der Lösungserfasser weist jedem Kostümteil ein oder mehrere Eigenschaften aus der Ontologie zu. Dabei kann es vorkommen, dass er auf eine Lücke in der Ontologie stößt. Das fehlende Ontologiewissen muss vom Kostümexperten nachmodelliert werden. Dies kann direkt im Wiki geschehen, so dass ein weiterer Import nicht nötig ist. Manche Wikis bieten auch die Möglichkeit, dass der Lösungserfasser das fehlende Element einfach eingibt. Der Kostümexperte bekommt dann gemeldet, dass dieses Element verwendet wurde, obwohl es nicht modelliert ist.

Kunden Der Kreis der Kunden ist noch nicht näher spezifiziert. Denkbare Szenarien sind jedoch, dass Kostümverleihe direkt ihre Teile mit dem Wiki verknüpfen könnten, um so einem Kostümbildner einen schnelleren und direkteren Zugriff zu ermöglichen.
Kostümbildner und Kostüm-Forscher können aufgrund der gesammelten Daten schnell eine Recherche durchführen, um typische Kostüme oder spezielle Instanzen nachzuschlagen. Hierfür eignet sich besonders eine Volltextsuche, wie sie in Semantic Media Wiki implementiert ist. Auch komplexe Abfrageergebnisse lassen sich aufgrund der semantischen Annotation extrahieren. Dazu können auf die vom Wikiexperten modellierten Abfragen zurückgegriffen werden oder eigene Reportingseiten erstellt werden.
Für Kleidungshersteller eröffnen sich neue Möglichkeiten, da mit hinterlegten Herstellungsanleitungen schnell und nach Bedarf produziert werden kann.
Auch wenn bislang die Recherche im Vordergrund steht, soll der Mehrwert, der durch

diese semantische Datenerfassung entsteht, allen Kostümschaffenden zugänglich gemacht werden.

Bot Obwohl es auf den ersten Blick verwunderlich scheint, dass ein Programm zusammen mit Menschen Teil eines Rollenmodells ist, macht diese Einteilung bei näherer Betrachtung doch Sinn.
Obwohl ein Wiki schon weit über den Stand eines „Datenspeichers“ hinausgewachsen ist, gibt es dennoch Funktionen, die eines weiteren, externen Programms bedürfen. Dies sind meistens Wartungsfunktionen. In dieser Domäne übernimmt der Bot eine Analysefunktion. Für solche Bot-Programme bietet Mediawiki eine Schnittstelle. Der Bot muss sich allerdings wie jeder andere Benutzer mit Benutzername und Passwort identifizieren, bevor er Änderungen vornehmen oder Seiten auslesen kann.

3.1.2 Der Muster-Identifikationsprozess

Am Anfang der Identifikation von Patterns steht die Frage, was ein Pattern eigentlich ist und wie es sich darstellen lässt. Die allgemeine Antwort darauf liefert Abschnitt 2.2.1. Zur Beantwortung des konkreten Falles eines Kostümpatterns soll der nächste Abschnitt dienen.

Darstellungsform eines Kostümpattern

Ein Kostümpattern wird im Wiki als eigener Artikel dargestellt. Dieser implementiert die in „A Pattern Language for Costumes in Films“[SBLE12] vorgestellte Form (Abb.: 3.2), die sich an den Architekturmustern nach [AIS77] und [HW03] orientiert und sich wie folgt darstellt:

- **Name:** Ein eindeutiger Name mit hohem Wiedererkennungswert. Der Name des Wiki-Artikels.
- **Icon:** Eine grafische Darstellung des Kostüms.
- **Zweck:** Kurze Beschreibung der vestimentären Botschaft, die dieses Kostümmuster transportiert.
- **Zusammensetzung:** Hier werden die einzelnen Kostümteile oder Unterpatterns beschrieben, aus denen das Pattern zusammengesetzt ist. Es werden auch die Beziehungen zwischen den einzelnen Kostümteilen abgebildet.

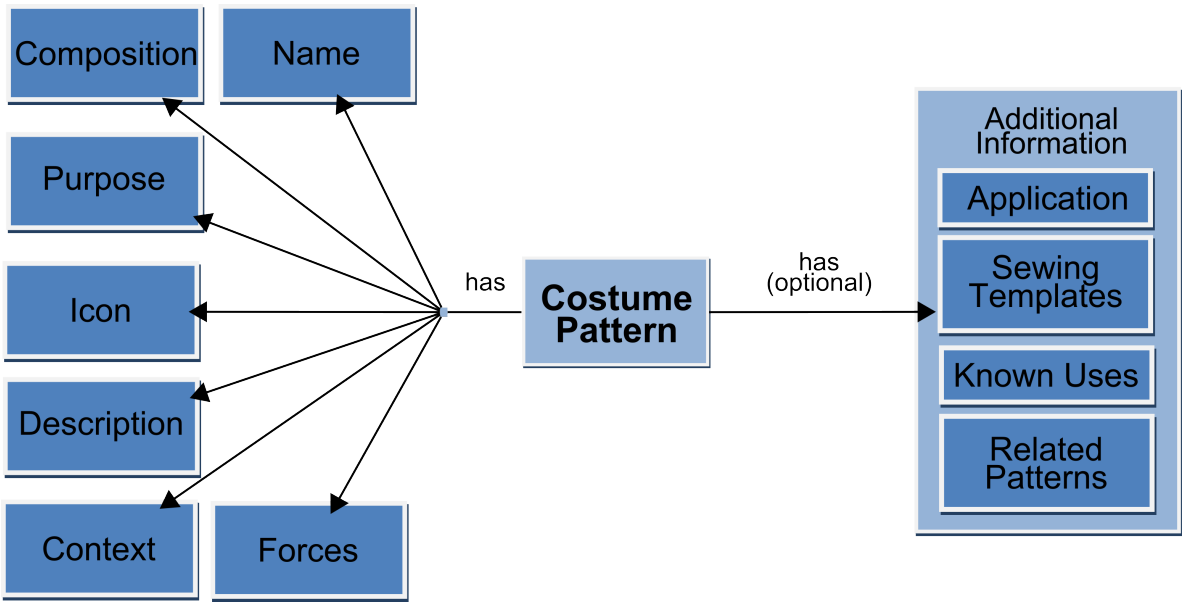

Abbildung 3.2: Eigenschaften und Format eines Patterns angelehnt an [SBLE12, Abschnitt 5, S. 12]

- **Kräfte:** Dieser Abschnitt beschäftigt sich mit den innerfilmischen Wechselwirkungen die beim Einsatz dieses Musters zum Tragen kommen.
- **Kontext:** Genre und weiterer Kontext zum Pattern
- **Beschreibung:** Eine textuelle Beschreibung des Patterns
- **Optionale Eigenschaften:**

  Verwandte Kostümmuster: Verlinkungen zu weiteren Mustern, die im Zusammenhang stehen.

  Bekannte Vorkommen: Eine Sammlung aller Instanzen, die dieses Pattern implementieren.

  Anwendung: Hier können weitere Informationen, zum Beispiel eine Ankleidereihenfolge hinterlegt werden.

  Nähanweisung: Falls vorhanden können hier Informationen zum Herstellungsprozess eingetragen sein.

Unter Zusammensetzung finden sich Links auf die zugehörigen Kostümteile, die wiederum eigene Artikel sind.
Diese besitzen ebenfalls eine Zuordnung zu den Taxonomien. Die vom Kostümexperten modellierten Taxonomien sind hierarchisch angeordnet und werden vom abstrakten „Schuh“ zur

„Riemchensandalette“ immer konkreter. Alle Eigenschaften (wie in Abschnitt 2 beschrieben), die für ein Kostümteil gültig sind, können auch dem abstrakten Kostümteil eines Patterns annotiert werden. In den meisten Fällen sollte ein Musterkostüm eher abstrakten, in der Hierarchie höher stehenden Elementen zugeordnet sein. Dies ist jedoch nicht zwingend.

Identifizierung von Mustern

Wie in vielen anderen Domänen, ist auch bei den Kostümen die Identifikation eines Patterns Handarbeit. Der Kostüm-Experte kennt aus seinem professionellen Umgang mit Kostümen die gängigen Rollen und deren Merkmale.
Diese Muster sind immer auch soziale Rollen und unterliegen dem gesellschaftlichen Wandel. Dies bringt mit sich, dass das Muster auf dem eine Filmrolle basiert, je nach Kontext einem Beruf, Stereotyp oder auch einer Charaktereingenschaft zuzuordnen ist.
Im Gegensatz zu dem in „A Pattern Language for Costumes in Films“[SBLE12] vorgestellten Pattern-Identifikationsprozess entfällt die Wahl des Genres und Filmkorpus als expliziter Schritt. Implizit sind beide Schritte in der Auswahl von Instanzen vertreten. Der hier gewählte Ansatz ermöglicht auch die Genregrenzen zu verlassen und übergreifende Muster zu bestimmen. Ein weiterer Unterschied besteht darin, dass hier zu Beginn des Identifikationsprozesses ein Patternkandidat gewählt wird und nicht alle gängigen Rollen eines Genres betrachtet werden.
Die einzelnen Schritte könnten also wie folgt beschrieben werden:

1. Wahl eines Patternkandidaten
2. Auswahl von Instanzen
3. Dekomposition der Instanzen
4. Ähnlichkeitsanalyse der Komponenten
5. Filterung
6. Abstraktion zum gemeinsamen Oberbegriff
7. Komposition zum Pattern

Die folgenden Schritte werden manuell durchgeführt. Wo die Unterstützung durch das Wiki wie möglich ist, steht explizit dabei. Abschnitt 3.2.7 verfolgt einen anderen, weitestgehend automatisierten Ansatz.

Abbildung 3.3: Der Pattern-Identifikationsprozess in der Kostümdomäne, angeleht an [Ba13]

Wahl eines Patternkandidaten Der Kostümexperte wählt aus Stereotypen, Beruf oder Charaktereigenschaften einer Rolle einen Patternkandidaten aus. Die vorhandenen Attributwerte kann er dabei aus dem Wiki ablesen, sofern bereits die entsprechenden Lösungen im Wiki hinterlegt wurden.

Auswahl von Instanzen Der Kostümexperte wählt nun diejenigen Instanzen, die den gewählten Patternkandidaten implementieren. Das Wiki kann sehr leicht eine Liste mit allen Instanzen erzeugen, die den entsprechenden Attributwert besitzen.

Dekomposition der Instanzen In diesem Schritt müssen die Kostüme in ihre einzelnen Teile zerlegt werden. Bis zur Komposition werden alle weiteren Schritte auf den Kostümteilen durchgeführt. Ist eine Lösung im Wiki eingetragen, sind auch ihre einzelnen Kostümteile vorhanden und entsprechend zugeordnet. Benutzt man eine entsprechende Abfrage, kann man diesen und den vorhergehenden Schritt zusammenfassen.

Ähnlichkeitsanalyse der Komponenten Dies ist der Kern des hier vorgestellten Identifikationsprozesses. Im Gegensatz zu „A Pattern Language for Costumes in Films“[SBLE12],

wo nur von „Describe Components“ die Rede ist, wird hier eine Analyse durchgeführt, die zwei miteinander verglichenen Kostümteilen einen numerischen Wert zuordnet, der die Ähnlichkeit ausdrückt. Dies geschieht über die Hierarchieebenen der einzelnen Taxonomien. Dabei wird jede Ebene mit einem Wert versehen, der ihrem Abstand zur Wurzel entspricht. Nun sucht man zu den zwei Elementen (in jeder Eigenschaft) den ersten gemeinsamen Oberbegriff in der Taxonomie und summiert die Werte von dort bis zur Wurzel auf. Die Summe dieser Werte bringt die Ähnlichkeit zweier Kostümteile zum Ausdruck, da zwei Elemente, die im Taxonomiebaum näher beieinander stehen automatisch einen höheren Wert erzielen. Je ähnlicher sich mehrere Kostümteile sind, desto größer ist die Wahrscheinlichkeit, dass diese Teile für das Pattern relevant sind.
Diese Analyse kann in Semantic Media Wiki von einer SPARQL-Abfrage übernommen werden.

Filterung Der Nachteil dieser Methode wird im folgenden Schritt, „Abstraktion zum gemeinsamen Oberbegriff“ klar: Vermutlich wird man viele Instanzen finden, die sich sehr ähneln, in wenigen Details jedoch signifikant unterscheiden. Möchte man nun für die Eigenschaft mit der großen Differenz einen gemeinsamen Oberbegriff finden, landet man sehr nahe am Wurzelelement. Wiederholt sich dieser Vorgang mit anderen Instanzen bei anderen Eigenschaften, hat man zuletzt ein sehr unscharfes Pattern, mit dem man nichts anfangen kann.
Deswegen ist es wichtig, diese „Ausreißer“ herauszufiltern und nicht in den Pattern-Kompositionsprozess einfließen zu lassen. Dies ist eine Aufgabe, die der Kostümexperte im referenzierten Modell[SBLE12] intuitiv übernimmt.

Abstraktion zum gemeinsamen Oberbegriff Nachdem die Daten gefiltert wurden, werden für die korrespondierenden Kostümteile die gemeinsamen Oberbegriffe gesucht.

Komposition zum Pattern Diese werden nun im Abschnitt „Zusammensetzung“ verlinkt. Die weiteren Abschnitte auf der Patternseite werden ausgefüllt.
Im Anschluss daran sollte das Pattern gegen weitere Instanzen geprüft und gegebenenfalls angepasst werden.

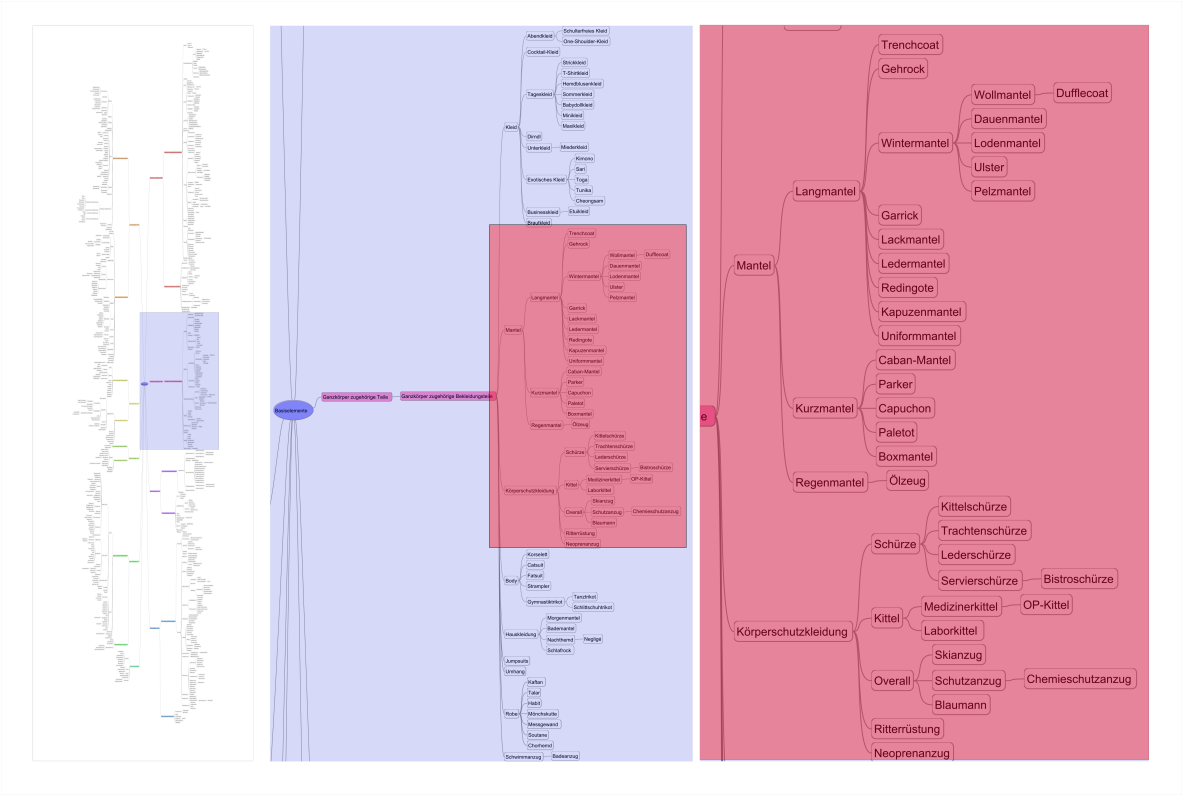

Abbildung 3.4: Ausschnitt aus der Basiselemente-Taxonomie aus [Ba13]

3.2 Modellierung der Kostümdomäne

3.2.1 Domänenmodellierung

Das Herz der Kostümdomäne sind die einzelnen Kostüme, wie sie in Filmen vorkommen. Um ein Kostüm zu erfassen, gilt es, die einzelnen Kostümteile exakt aufzunehmen und auch weitere Eigenschaften wie Zustand und Trageweise sowie Assoziationen mit dem Kostüm zu erfassen. Abbildung 3.5 bietet einen Überblick über die Eigenschaften, die bei einem Kostüm modelliert werden und zeigt die Zusammenhänge zwischen Film, Rolle, Kostüm, Kostümteil und Teilbereich.
Kostüme werden aus Kostümteilen komponiert. Um ein gemeinsames Vokabular vorzugeben und Referenzen auf diese Vokabeln anzugeben, wurden Farben, Basiselemente, Teilelemente, Formen und Materialien in Taxonomien vom Domänenexperten modelliert und standardisiert. Diese hierarchisch aufgebauten Fakten ermöglichen eine Ähnlichkeitsanalyse wie in Abschnitt 3.1.2 beschrieben. Tabelle 3.1 zeigt, für welche Eigenschaften ein Vokabular in Form einer Taxonomie vorgegeben ist, während Abbildung 3.4 einen Ausschnitt aus der Basiselemente-Taxonomie zeigt.

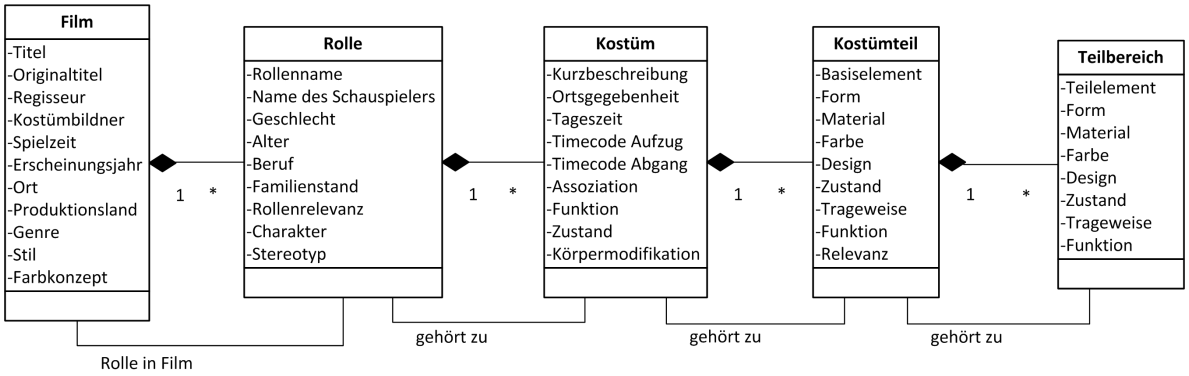

Abbildung 3.5: Kostüme als UML-Klassendiagramm

Dateiname	Zeilenanzahl	Hierarchietiefe	Bsp. tiefstes Element
Alter	13	2	in den 20ern
Basiselement	889	7	Trainingshorts
Design	29	4	gestreift
Familienstand	4	1	ledig
Farbe	53	3	dunkelviolett
Farb-Eigenschaft	6	1	glänzend
Farbkonzept	7	1	überzeichnet
Form	30	3	knöchellang
Funktion	219	5	Chirurgenkleidung
Geschlecht	4	1	weiblich
Körpermodifikation	44	4	Hochsteckfrisur
Material	99	5	Spandex
Material-Eigenschaft	7	1	fließend
Operator	15	2	angesteckt
Ortsgegebenheit	4	1	draußen
Rollenrelevanz	4	1	Nebenrolle
Tageszeit	7	2	nachmittags
Teilelement	203	5	Stiletto
Trageweise	10	2	hochgekrempelt
Zustand	28	3	ausgewaschen

Tabelle 3.1: Eigenschaften, die mittels Taxonomie in [Ba13] modelliert wurden

Um eine saubere Katalogisierung der Kostüme vorzunehmen, werden diese einer Rolle zugeordnet, die wiederum einem Film zugehört. Somit lassen sich Kostüme auch getrennt nach Genres oder Produktionsländern und Produktionsjahren betrachten.
Für das Kostüm werden zusätzliche Informationen wie der Zeitpunkt des Auf- und Abtritts erfasst. Der Charakter und Stereotyp der verkörperten Rolle werden ebenfalls notiert.

Ersteres ermöglicht eine Betrachtung über die Zeit hinweg und kann aufschlussreiche Informationen liefern, wie lange ein Kleidungsstück oder eine Farbe zu sehen ist, letzteres soll Korrelationen zwischen Charaktereigenschaften oder Stereotypen und Kleidung, Farben, Materialien, etc. aufzeigen. Je kürzer eine Kostüm zu sehen ist, desto wichtiger ist die in Kapitel 2.3 beschriebene vestimentäre Kommunikation. Da der Rolle kaum Zeit bleibt, ihre Funktion und Charaktereigenschaften auszuspielen, muss die Kostümierung versuchen, dies dem Zuschauer klar zu machen.
Der Kompositionsabschnitt eines Patterns ist die abstrahierte Form einer Kostümlösung: Aus der konkreten Instanz des Sheriffs in „Rio Bravo“ wird beispielsweise das Pattern des Wild-West Sheriffs. Während die Instanz alle Details enthält, beinhaltet das Pattern nur die charakteristischen Kleidungsstücke, bzw. deren charakteristische Eigenschaften.
Eine Kostümlösung beschreibt ein Kostüm einer Rolle in einem Film, das Pattern versucht, alle Kostüme aller Rollen, die vom gleichen Typ (Beruf, Charakter, Stereotyp) sind, abzubilden. Der größte gemeinsame Nenner aller (oder der meisten (vgl. Abschnitt 3.1.2)) Instanzen gibt somit die Abstraktionsebene des Patterns vor. Dabei ist es schwierig zu entscheiden, ob eine Instanz nun typisch für ein Pattern ist, oder nicht. Lösungen für diese Entscheidung kommen erst mit einer großen Anzahl erfasster Kostüme oder durch fachliche Bewertung des Domänenexperten.

3.2.2 Erfassung des Domänenwissens

Das Domänenwissen besteht aus Fakten (vgl. Abschnitt 2.1.2) und ihren Zusammenhängen. Dieses Wissen kann in Form von Ontologien abgespeichert werden. Für unterschiedliche Domänen ist das Ontologiewissen natürlich nicht einheitlich. Daher gibt es keine Musterlösung mit der alle Domänen gleichermaßen im Wiki dargestellt werden können. Einzelne Bereiche einer Ontologie können aber durchaus verallgemeinert werden: so sind viele Fakten beispielsweise hierarchisch aufgebaut. Diese Hierarchie lässt sich durch eine „ist eine“ Beziehung abbilden. Diese hierarchische Struktur der Fakten ist besonders wichtig, um später Auswertungen auf den gesammelten Daten durchzuführen. Die Zusammenhänge zwischen den einzelnen Fakten sind sehr domänenspezifisch, jedoch lässt sich in den einzelnen Domänen meist eine Unterscheidung in unterschiedliche Ebenen vornehmen. Die Fakten, die der Domänenexperte modelliert hat, sollen nicht verändert werden. Eine wichtige Aufgabe des Systems ist es, dem Erfasser der Daten die richtigen Fakten zur Auswahl vorzugeben. Mit der Lösung dieses Problems befasst sich Abschnitt 3.2.5. Nur so kann gewährleistet werden, dass auf erfassten Daten auch sinnvoll gesucht werden kann.

3.2.3 Struktur des Wikis

Dieser Abschnitt beschreibt kurz die Struktur eines Wikis und erläutert die für die Kostümdomäne relevanten Elemente.

Artikel Sie sind der zentrale Datentyp eines Wikis. Fast alles im Wiki wird von einem Artikel repräsentiert. Jeder Artikel hat einen eindeutigen Namen (innerhalb seines Namensraums) und liegt in einem Namensraum, der dem Namen als Präfix vorangestellt ist. Ein Artikel hält seine Änderungshistorie in Revisionen vor und stellt seine Inhalte in einer optisch ansprechenden Form dar.

Kategorien Dieses Konzept ist ein Spezialfall eines Artikels und auch der Grund für semantische Attribute: Eine Kategorie ist ein Artikel, der mit einem speziellen Tag von einem anderen Artikel aus referenziert werden kann und diesen als zugehörig anzeigt. Ein Artikel kann beliebig vielen Kategorien zugehören. Im Prinzip ist das nichts anderes als das erste semantische Attribut „zugehörig zu einer Kategorie“.

Semantische Attribute Die Ausdrucksschwäche des Kategorie-Konzepts führte zur Entwicklung semantischer Attribute: Sie erweitern das Vokabular eines Wikis um eine ganze Wortklasse, die Verben. Mit diesen Attributen lassen sich Zusammenhänge erst richtig darstellen. Die Regeln für grundlegenden Satzbau, Subjekt - Prädikat - Objekt, gelten nach „The Semantic Web“[BLHL01] auch, wo Zusammenhänge semantisch beschrieben werden sollen. Subjekt und Objekt sind dabei Artikel des Wikis. Objekte können auch von anderen Datentypen sein, die das Wiki erlaubt. Prädikate sind durch die Attribute selbst dargestellt.

Namensräume Artikel werden in Namensräumen organisiert. Dies hilft Artikel nach ihrer Funktion zu trennen und verlagert das Problem, zwei Artikel nicht gleich benennen zu können. Für Namensräume können unterschiedliche Zugriffsrechte vergeben werden. Dadurch lassen sich Inhalte vor ungewollter oder böswilliger Veränderung schützen.

Vorlagen Wiederkehrende Inhalte wie bestimmte Abfragen, Zugehörigkeit zu einer Kategorie oder die Verfügbarmachung bestimmter semantischer Attribute in einem Artikel können durch die Benutzung von Vorlagen stark vereinfacht werden. Dabei wird die Vorlage im Artikel nur referenziert. Die Vorlage ist ein Artikel in einem speziellen Namensraum. Vorlagen eignen sich auch hervorragend, um zentrale Änderungen vorzunehmen.

Formulare Vorlagen können von Formularen interpretiert werden. Semantic Media Wiki mit der Erweiterung SemanticForms bietet die Möglichkeit, einen neuen Artikel mit Hilfe eines Formulars zu erstellen, dem eine Vorlage zugrunde liegt. Das Formular füllt die Vorlage anhand der Benutzereingaben aus. Formulare sind ein bequemer Weg, Artikel im Wiki anzulegen.

Abfragen Der Nutzen abgelegter Daten, die nicht durchsucht oder verwendet werden können, ist fraglich. Deswegen bringt das Wiki einige Befehle, die Abfragen von einfach bis sehr komplex ermöglichen. Dabei kann entweder auf die Wiki-Datenbank oder auf ein externes Programm zurückgegriffen werden. Auch die Abfragebefehle unterscheiden sich in Syntax und Komplexität.

3.2.4 Taxonomie-Import ins Wiki

Für den Import von Ontologien bietet Semantic MediaWiki (SMW) eine eigene Funktion. Dabei werden zwar alle modellierten Daten richtig in das Wiki übertragen, die Benennung der Artikel erfolgt jedoch hierarchisch, so dass der komplette Pfadname des Ontologiebaumes als Artikelname angegeben wird. Dadurch wird das Wiki unübersichtlich. Eine Verwendung von Namensräumen sieht der Import ebenfalls nicht vor. Aus diesen Gründen ist von der Benutzung dieser Import-Funktion abzuraten.
Alternativ kann der Import über die Programmierschnittstelle von MediaWiki (MW) erfolgen. Dieses Interface existiert für viele Programmiersprachen.[med13c] So ist es möglich, aus einem gewählten Eingabe-Format einen Import ins Wiki zu erhalten, der ebenfalls sehr flexibel angepasst werden kann. Beispielsweise kann die Zuordnung in Namensräume und die Ausgestaltung der Artikel mit vorgefertigten Abfragen erfolgen.
Im Referenzprojekt haben wir uns für die Implementierung der Taxonomien in Freeplane bzw. Freemind entschieden. Dieses unter der GNU-General Public License erschienene Mindmap-Programm ermöglicht die einfache hierarchische Gliederung von Fakten und verfügt über eine Reihe von Exportformaten.
Die nötigen Zusammenhänge zwischen den Fakten der Taxonomien sowie die Restriktionen, ob ein Attribut beispielsweise mehrere Werte haben darf und Vorgaben für die Eingabe müssen in Vorlagen und Formularen verankert werden. Dies ist die Hauptaufgabe des Wiki-Experten. Hier gilt es, in enger Zusammenarbeit mit dem Domänenexperten dessen Vorstellungen mit den Möglichkeiten, die Media Wiki und seine Extensions bieten in Einklang zu bringen. An manchen Stellen muss hier die Domäne mit einer gewissen Ungenauigkeit modelliert werden, an anderer Stelle muss eine Erweiterung für das MediaWiki selbst geschrieben oder um eine gewisse Funktionalität erweitert werden.

3.2.5 Lösungsdokumentation und Eingabemodelle

Unterstützte Eingabe von Instanzen

Dieser Abschnitt beschreibt das Vorgehen des Lösungserfassers bei der Dokumentation einer Lösung.
Alle Schritte sind durchgängig durch Formulare und Übersichtsseiten mit Eingabemasken unterstützt. Die nötigen semantischen Attribute, die den Zusammenhang zwischen den Seiten sichern, werden dabei von den Formularen selbstständig erstellt. Diese Zusammenhänge sind auch in Abb. 3.5 in den Verbindungen die mit „gehört zu“ annotiert sind, abgebildet.

1. **Film erfassen:** In einem ersten Schritt ist es nötig, den Filmtitel, Daten zu Genre, Entstehungsland und -zeit und kostümverantwortlichen Personen zu erfassen. Dieser Schritt muss für jeden Film natürlich nur einmal vorgenommen werden. Möchte man an einem bestehenden Film weiterarbeiten, genügt ein Klick auf den entsprechenden Link auf der Startseite.

2. **Rolle erfassen:** Bevor Kostüme erfasst werden können, muss festgelegt werden, welcher gespielten Rolle ein Kostüm zugeordnet werden soll. Weitere wichtige Informationen sind der Stereotyp den die Rolle verkörpert, sowie Charaktereigenschaften der Rolle oder ob es sich um eine Haupt- oder Nebenrolle handelt. Der Name der Rolle setzt sich aus Film und eingegebenem Rollenname zusammen und wird im Namensraum „Rolle:“ abgelegt. Auch hier gilt wie beim Film, dass eine einmal angelegte Rolle durchaus mehrere Kostüme tragen kann, ohne dass die Rolle deswegen neu eingegeben werden müsste.

3. **Kostüm erfassen:** Der Kostümname wird fortlaufend und automatisch vergeben. Für jedes Kostüm muss eine Kurzbeschreibung angegeben werden. Dann wählt man aus Taxonomien die Ortsgegebenheit und Tageszeit. In die Felder für Timecode-Aufzug und -Abgang gibt man die Zahl der Sekunden ab Filmstart ein. Mit diesen Werten kann das Wiki auch rechnen. Assoziation bietet ein Feld zur Freitexteingabe der subjektiven Wahrnehmung des Lösungserfassers. Im Gegensatz zum Stereotyp der für die ganze Rolle gilt, ist die Assoziation nur auf das Kostüm bezogen. Funktion, Zustand und Körpermodifikation werden aus den entsprechenden Taxonomien ausgewählt.

4. **Kostümteil erfassen:** Auch hier wird der Name des Artikels vom Wiki vergeben und fortlaufend durchnummeriert. Der Lösungserfasser wählt ein Basiselement aus der Taxonomie und spezifiziert, ebenfalls mit Hilfe der Taxonomien Form, Material, Farbe, Design und Trageweise. Die gewählten Werte für Funktion und Zustand aus der Kostümeingabe werden automatisch übernommen, können aber hier noch bearbeitet werden. Der Wert für die Relevanz gibt auf einer Skala von 1-10 an, wie wichtig oder

typisch das Kleidungsstück für das Kostüm und die Rolle ist. Dies unterliegt natürlich ebenfalls der subjektiven Wahrnehmung des Erfassers.

5. **Teilbereich erfassen:** Optional kann man zu jedem Kostümteil noch beliebig viele Teilbereiche näher spezifizieren. Solche sind beispielsweise ein Kragen, Ärmel, Reißverschluss-Partien. Dies führt dazu, dass ein Kostümteil mit einem sehr hohen Detailgrad erfasst werden kann. Für jeden Teilbereich muss man ein Teilelement wählen und dann, wie beim Kostümteil Eigenschaften angeben, die dieses näher bestimmen. Die möglichen Eigenschaften sind die gleichen, wie im Kostümteil und werden von dort auch übernommen, können dann jedoch angepasst werden.

Durch diese geführte Eingabe wird sichergestellt, dass bei jedem Kostümteil für jede Eigenschaft aus der richtigen und gleichen Taxonomie Werte zum Einsatz kommen. Durch Vorgabe der Fakten wird auch die Gefahr von Tippfehlern minimiert, da nur wenige Freitexte vorhanden sind. Diese strukturierte Darstellung der Kostümlösungen ist die Grundlage für den Pattern-Identifikationsprozess. Nur wenn alle Werte eines semantischen Attributs auch in der selben Taxonomie zu finden sind, lassen sie sich miteinander vergleichen.

Referenzierung von Fakten durch konfektionierte Auswahl

Aufgabe des Wikiexperten ist es, die Formulare für die Lösungserfasser möglichst einfach und intuitiv zu gestalten. Dabei muss er die Möglichkeit bieten, schnell auf die Fakten der richtigen Taxonomie zu verweisen. In Formularen kann dies durch Texteingabe mit Autovervollständigung erreicht werden.
Da im Fall der Kostümdomäne zum Teil sehr große Taxonomien vorliegen und man allein bei den Basiselementen aus 890 Fakten wählen kann, bietet die Einarbeitung in diese Taxonomie eine große Einstiegshürde. Aus diesem Grund wurde ein Formular entwickelt, das sowohl die klassische Eingabe unterstützt, als auch den Taxonomie-Lernprozess durch die grafische Darstellung des Taxonomiebaumes vereinfacht. Der Lösungserfasser kann sich so entweder durch den Taxonomiebaum hangeln oder die Direktverbindung zu einem Fakt wählen. Diese beiden Verfahren lassen sich auch an beliebiger Stelle durchmischen.
Der Domänenexperte spezifiziert auch Attribute, die mehrere Werte haben können. (Beispielsweise Farben). Um auch hier die Eingabe so intuitiv wie möglich zu gestalten, gibt es bei den Formularfeldern, die mehrere Werte erlauben, einen Knopf der den aktuellen Wert sichert, und die Baumansicht wieder auf das Standardelement zurücksetzt, so dass man einen neuen Wert auswählen kann.

3.2.6 Informationen aus dem Wiki auswerten

Ziel der semantischen Erfassung von Informationen ist, Maschinen zu ermöglichen, die Bedeutung eines Sachverhaltes zu erkennen. Darin sind Maschinen, verglichen mit Menschen, nicht besonders gut. Auch semantische Daten „verstehen“ Maschinen nicht in dem Sinne, dass sie ihre wahre Bedeutung erfassen könnten. Das schreibt auch Tim Berners-Lee: „The computer doesn't truly „understand“ any of this information, but it can now manipulate the terms much more effectively in ways that are useful and meaningful to the human user.“[BLHL01]
Vielmehr ermöglicht Semantik, die Daten in einem Format darzustellen mit dem die Maschine arbeiten kann, auf dem sie Schlüsse ziehen und Vorkommnisse suchen kann. Das sind Vorgänge, die Maschinen besser beherrschen, als Menschen, denn es sind nun einfach Berechnungen.

Triplestore Um diesem Sinn gerecht zu werden, sind auch die semantischen Informationen, die in einem Wiki abgespeichert werden dergestalt, dass sie von einem Computer verarbeitet werden können. Um die Mächtigkeit ihrer Darstellungsform zu demonstrieren ermöglichen viele Wikis die Verarbeitung des Wissens mit Bordmitteln. Andere greifen auf spezielle Abfragemaschinen zurück. SMW tut beides: Die Sprache, auf der die Computer mit den Daten operieren können, ist eine Tripelsprache: Dabei werden Zusammenhänge in der Form Subjekt - Prädikat - Objekt dargestellt. Das Wiki verwandelt die ihm bekannten Informationen in solche Tripel. Das ist recht einfach, da das Subjekt immer der Artikel ist, in dem ein Attribut steht, das Prädikat das Attribut ist und das Objekt durch den Wert des Attributs bestimmt wird. Man kann sich leicht vorstellen, dass so eine große Anzahl an Tripeln entsteht. Diese schickt das Wiki nun an ein Programm, das diese Tripel verwalten und auf ihnen arbeiten kann, den Triplestore (Tripel-Speicher).
Der Triplestore ist von seinem Aufbau her dreigeteilt. Eine detaillierte Beschreibung des Aufbaus anhand des TripleStore Basic von DIQA ist in Abschnitt 5.3.11 zu finden.
Eine Logik-Einheit sorgt dafür, dass Attribute transitiv (Attribut „gehört zu“ in Abb. 3.5), oder symmetrisch verwendet werden können. Dort können auch Regeln hinterlegt sein, die beispielsweise aus den Informationen: `X hat_Geschlecht weiblich` und `Y ist_Vater_von X` ein weiteres Tripel `X ist_Tochter_von Y` ableiten.
Ein weiteres Element des Triplestore ist die Query Engine, die Abfragen auf den Tripeln ausführt. Eine Abfrage liefert im Grunde immer eine Menge von Tripeln zurück, bei denen Subjekt, Prädikat oder Objekt mit den in der Abfrage spezifizierten Werten übereinstimmen. Alle komplexeren Abfragen sind Mengenoperationen auf den unterschiedlichen Ergebnismengen.
Im Vordergrund läuft eine Benutzerschnittstelle, die Anfragen in einer Querysprache ent-

gegennimmt und die gefundenen Ergebnisse formatiert und zurückschickt. Abbildung 3.6 verdeutlicht den Aufbau des Triplestore. Der Triplestore implementiert somit den von Dean Allemang beschriebenen RDF-Store ([AH11, Seite 57]).
Aus Abbildung 3.6 geht auch hervor, dass der Triplestore sowohl aus dem Wiki als auch von außerhalb abgefragt werden kann.

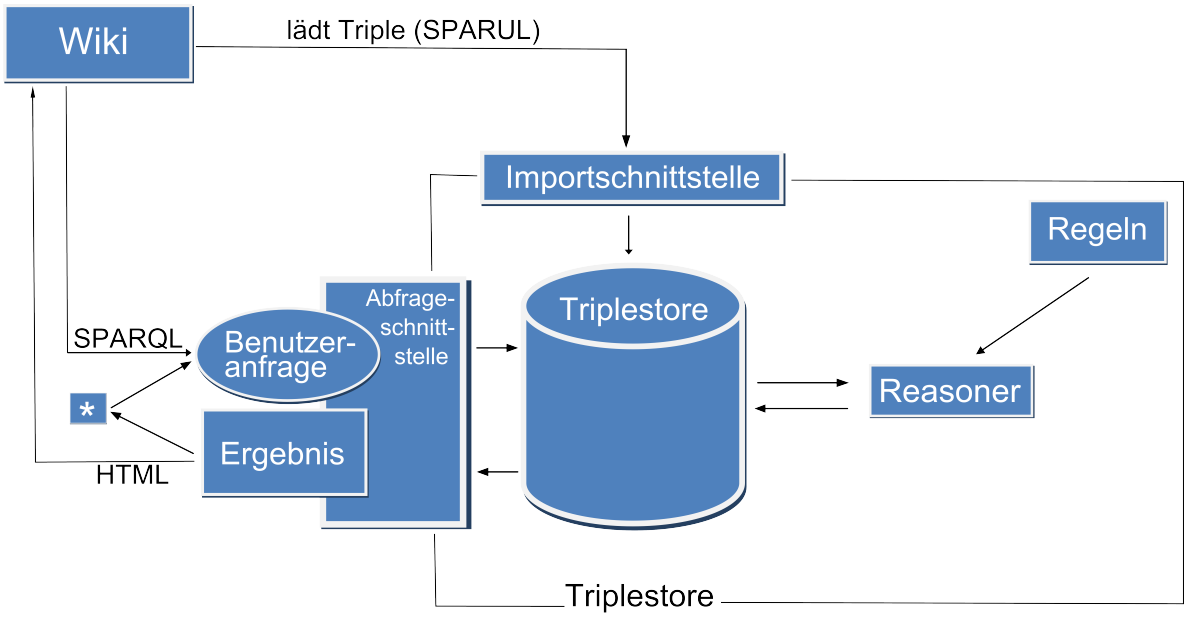

Abbildung 3.6: Schematische Übersicht Triplestore

Abfragen Anfragen an den Triplestore erfolgen in SPARQL (siehe Abschnitt 2.1.5). Semantic Media Wiki bietet zwar auch die Möglichkeit, Abfragen in der wikieigenen Sprache „Wikitext“ zu formulieren, da aber letztlich alle Abfragen in SPARQL umgewandelt werden, verzichtet diese Arbeit auf deren Betrachtung.
Der Wikiexperte ist angehalten für den Domänenexperten wichtige Daten in der Art von Reports zur Verfügung zu stellen. Da mit dem Laden einer Seite auch die Abfrage neu ausgewertet wird, spiegeln die Ergebnisse immer den aktuellen Wissensstand des Wiki wider. In der Kostümdomäne könnten folgende Kenngrößen aufschlussreich sein: Anzahl der eingegebenen Kostüme, durchschnittliche Anzahl an Rollen pro Film, durchschnittliche Anzahl angelegter Kostümteile pro Kostüm.
Für die Kostümforschung wichtige Fragen können ebenfalls mit SPARQL-Queries beantwortet werden. Ein Beispiel hierfür ist, welche Farbe in einem Film dominant ist. Das Wiki kann diese Frage natürlich nur bezogen auf die Kostüme beantworten und auf Kameraführung oder relative Größe der Kleidungsstücke zueinander nimmt es auch keine Rücksicht. Doch wenn man, wie in diesem Fall die Anzahl der gleichfarbigen Kleidungsstücke mit der

Auftrittsdauer des Kostüms multipliziert, erhält man einen Wert, der die Dominanz einer Farbe ausdrückt. Über die Höhe dieses Werts lässt sich auch die Signifikanz dieser Aussage relativieren.
Für den Pattern-Identifikationsprozess lassen sich die Abfragen auch nutzen, um eine Ähnlichkeitsanalyse durchzuführen. Dieses Vorgehen wird im folgenden Abschnitt genauer beschrieben.

3.2.7 Teilautomatisierter Patternidentifikationsprozess

Der Identifikationsprozess von Patterns wurde in Abschnitt 3.1.2 bereits vorgestellt. Dieser Abschnitt beschreibt, wie dieser Prozess mit Hilfe der erfassten Daten zumindest teilweise automatisiert werden kann.
Der nachfolgend beschriebene Prozess geht von einer grundsätzlich anderen Annahme aus: Er sucht kein Pattern basierend auf einem Patternkandidaten, sondern eine Menge von Instanzen, die sich untereinander ähneln. Diese bietet er dem Domänenexperten zur Überprüfung an. Dahinter steht die Annahme, dass Kostüme, die sich stark ähneln auch ein gemeinsames Muster implementieren. Das nachfolgend beschriebene Verfahren zur Ähnlichkeitsanalyse basiert auf einer Abfrage, die eine Ähnlichkeit zwischen 2 Kostümen ausdrückt. Die Güte dieses Ansatzes steht und fällt mit der Qualität dieser Abfrage. Für eine erste Implementierung des Verfahrens wurde eine sehr einfache und vermutlich auch schlechte Heuristik zur Ähnlichkeitsbestimmung angenommen. Eine wirklich gute Heuristik für die Ähnlichkeit von 2 Kostümteilen anzugeben, liegt eher im Forschungsbereich der Kostümwissenschaften und würde den Rahmen dieser Arbeit sprengen.

Ähnlichkeitsanalyse Um die Ähnlichkeit zwischen zwei beliebigen Kostümen messen zu können, benötigt man Vergleichskriterien. Diese sind zum einen in der Zahl der angegebenen Eigenschaften gegeben, zum anderen in der tatsächlichen Ähnlichkeit der einzelnen Teile. Die Relevanz eines Teiles muss genauso berücksichtigt werden wie die Nähe zweier Teile zueinander in der Taxonomie. Für eine erste Testimplementierung wurde eine Heuristik herangenommen, die nur die Basiselemente und deren Abstand in der Taxonomie betrachtet. Dabei werden nicht nur die korrespondierenden Kleidungsstücke (also Kopfbedeckung von Kostüm1 mit Kopfbedeckung von Kostüm2) verglichen, sondern es findet ein Kreuzvergleich statt. Dies scheint auf den ersten Blick unnötig kompliziert, vereinfacht aber die Abfrage und verfälscht das Ergebnis nicht. Der numerische Wert einer Ähnlichkeit berechnet sich durch das semantische Attribut „Ebene“, das bereits beim Import der Taxonomie jedem Fakt zugewiesen wurde. Der Wert dieses Attributs ist wie in Abschnitt 3.1.2 beschrieben, der Abstand des entsprechenden Knotens zur Wurzel.
Da ein Vergleich aller Instanzen eine große und rechenintensive Aufgabe ist, wird für

diese Aufgabe ein Wiki-Bot, also ein automatisiertes Wartungsprogramm, eingesetzt. Dieses überprüft in regelmäßigen Abständen, ob ein neues Kostüm angelegt wurde und überprüft es auf Ähnlichkeit mit allen anderen vorhandenen Kostümen. Dazu verwendet es eine vorgefertigte Abfrage, die in Listing 3.1 zu sehen ist und ersetzt vor Ausführung das „XXX“ in der Anfrage durch den Namen des betreffenden Kostüms. Die Ergebnisse seiner Analyse legt der Bot in den betroffenen Artikeln ab. So gibt es eine Ähnlichkeitsbeziehung in Form eines semantischen Attributs von Kostüm A nach Kostüm B. Diese Informationen werden allerdings so angebracht, dass sie beim normalen Betrachten des Wikis nicht auffallen.

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX tsctype: <http://www.ontoprise.de/smwplus/tsc/unittype#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT ?kostüm2 (SUM(?wert) AS ?Summe)
 WHERE
 {
  {
  ?kostüm1 a cat:Kostüm .
  ?kostümteil1 prop:Gehört_zu ?kostüm1 .
  ?kostümteil1 a cat:Kostümteil .
  ?kostüm2 a cat:Kostüm .
  ?kostümteil2 prop:Gehört_zu ?kostüm2 .
  ?kostümteil2 a cat:Kostümteil .
  FILTER
  (
    (?kostüm1 != ?kostüm2) && (?kostüm1 = XXX)
  )
  ?kostümteil1 prop:Basiselement ?compElement1 .
  ?kostümteil2 prop:Basiselement ?compElement2 .
  ?compElement1 prop:Unterelement_von* ?hierarchie1 .
  ?compElement2 prop:Unterelement_von* ?hierarchie2 .
  FILTER (?hierarchie1 = ?hierarchie2)
  ?hierarchie1 prop:Ebene ?wert .
  FILTER (?wert != 0)
  }
 }
GROUP BY ?kostüm2
ORDER BY DESC(?Summe)
```

Listing 3.1: Abfrage um Ähnlichkeit zwischen 2 Kostümen zu erhalten

Einteilung in Ähnlichkeitsklassen Die Art dieses semantischen Attributs stellt ein Problem dar, das an mehreren Stellen im Umgang mit dem Wiki existiert: Ein Attribut kann keine

Gewichtung haben. Das liegt an der Triplestruktur, die neben Subjekt - Prädikat - Objekt keinen Platz bietet, ein Prädikat mit einem quantitativen Argument zu versehen.
Würde man das Attribut als Kante eines Graphen zwischen den zwei Knoten Subjekt und Objekt betrachten, wäre die Kante zwar gerichtet, aber nicht gewichtet. Eine naheliegende Lösung dafür wäre es, zu jedem Grad an Ähnlichkeit, ein semantisches Attribut zu bilden. Daraus folgt aber im schlechtesten Fall (keine zwei Kostümpaare sind sich gleich ähnlich), dass plötzlich bei n-Kostümen $\frac{n\times(n-1)}{2}$ neue semantische Attribute dem Wiki hinzugefügt werden würden.
Dies ist keinesfalls wünschenswert. Durch eine Normalisierung der Werte auf einen Prozentsatz zwischen 0 und 100, würde die Anzahl der neuen Attribute konstant 100 betragen, allerdings verliert man auch ein wenig Genauigkeit, da bei der Berechnung des Prozentsatzes Nachkommastellen ignoriert werden müssen, um die Beziehung zwischen den beiden Elementen einer Ähnlichkeitsklasse zuzuordnen. Als Referenzwert zur Normalisierung kann man die Identität eines vollständig spezifizierten Kostüms mit 100% ansetzen. Daraus lässt sich nun ein Prozentwert wie folgt berechnen:

$$\lfloor \frac{\text{absoluter Ähnlichkeitswert} \times 100}{\text{absoluter Identitätswert}} + 0,5 \rfloor = \text{Ähnlichkeitswert in Prozent} \tag{3.1}$$

Dieser Wert wird nun bei beiden Kostümen wie folgt eingetragen:
Für Kostüm A: `[[ähnelt zu XX %::Kostüm B]]`,
für Kostüm B: `[[ähnelt zu XX %::Kostüm A]]`,
wobei XX jeweils der Ähnlichkeitswert in % ist. Natürlich muss bei diesen Ähnlichkeitswerten berücksichtigt werden, dass ein höherer Wert automatisch auch die Bedingungen eines niedrigeren Wertes erfüllt. Hierzu sollte man bei den semantischen Attributen von einer Unterattribut-Eigenschaft (subProperty) Gebrauch machen und jedem Attribut mitteilen, dass es ein Unterattribut des Attributes mit einem Prozentpunkt weniger ist. Dadurch baut sich eine Attributkette auf und bei Suchabfragen werden die Unterattribute automatisch mitberücksichtigt.
Hat man die Ähnlichkeiten zwischen je 2 Kostümen ermittelt, ist der nächste Schritt, die Ähnlichsten zu gruppieren. Davon handelt der folgende Abschnitt.

Gruppierung der ähnlichsten Kostüme Für die Lösung dieses Problems kann man die Kostüme als Knoten eines Graphen und die Ähnlichkeits-Attribute als Kanten betrachten. Da alle Ähnlichkeits-Attribute bidirektional sind, kann man von einem ungerichteten Graphen ausgehen. Da von jedem Kostüm zu jedem anderen eine Kante besteht, ist der Graph auch vollständig.

Streicht man jetzt alle Kanten heraus, die kleiner als ein gewünschter Ähnlichkeits-Schwellenwert sind, erhält man einen ungerichteten Graphen, der partitioniert sein kann. Die ähnlichsten Kostüme sind nicht nur zu einem dritten Kostüm ähnlich, sondern auch zueinander. Somit suchen wir jetzt einen Teilgraph, der ebenfalls vollständig ist. Dieses Problem ist in der Informatik als CLIQUE bekannt. Hier suchen wir nun den vollständigen Teilgraph maximaler Größe, also die maximale CLIQUE. Die NP-Vollständigkeit dieses Problems wurde 1972 von Richard Karp bewiesen, indem er es auf das 3-SAT-Problem reduzierte. [Kar72]
Nachfolgender primitiver Algorithmus löst dieses Problem und liefert die Ergebnismengen $E[]$ zurück, deren Größte auch die maximale Clique darstellt.

Algorithmus 3.1 Algorithmus zum Auffinden der maximalen CLIQUE

Require: $U \leftarrow$ alle Knoten in einem zusammenhängenden ungerichteten Graphen
  $B, S, E[\mathcal{P}(U)] \leftarrow \{\}$ // Hilfs- und Ergebnismengen definieren
  $index \leftarrow 0$
  while $B \neq U$ **do** // Wenn alle Knoten bearbeitet sind, breche ab
    $S \leftarrow U$ // Kopiere alle Elemente aus U in eine Menge S
    for all $s \in S \wedge s \notin B$ **do** // Wähle ein beliebiges s nicht Element von B aus S
      if $S = U$ **then** // erster Durchlauf
        $B \leftarrow B \cup \{s\}$ // füge s der Bearbeitet-Menge hinzu
      end if
      $S \leftarrow S - \{x | x \in S \wedge x$ hat keine Kante zu $s\}$ // Entferne alle Knoten aus S,
      // zu denen s keine Kante hat
      $E[index] \leftarrow E[index] \cup \{s\}$ // Füge s der Ergebnismenge hinzu
      $S \leftarrow S - \{s\}$ // Entferne s aus S
    end for
    $index \leftarrow index + 1$ // Erhöhe den Index für eine neue Ergebnismenge
  end while

Es gibt einige E(i) die identisch sind, diese kann man noch vereinigen. Das größte E(i) ist die maximale CLIQUE.

Für die Implementierung wurde der Algorithmus nach Bron und Kerbosch [BK73] gewählt, da dieser als Teil der Java Graph Bibliothek JGraphT[jgr13] bereits implementiert ist. Dieser Algorithmus wird von einem weiteren Bot im Hintergrund ausgeführt, der die Instanzen in der CLIQUE mit einer einzigartigen ID versieht und als
`[[CliqueID::<Ähnlichkeitswert>-<uniqueID>]]` kennzeichnet.

Darstellung und Auswertung Das Aufzeigen aller Kostüme, die eine CliqueID mit einem bestimmten Ähnlichkeitswert enthalten ist nun eine einfache Reporting-Aufgabe die von einer SPARQL-Abfrage übernommen wird. Lediglich der Schwellwert für diese Suche muss noch gesetzt werden.
Der Domänenexperte analysiert im letzten Schritt die vorgeschlagenen Gruppen von ähnlichen Kostümen. Vielleicht entsteht aus einer dieser Cliquen ein neues Pattern.

Abschnitt 5.4.3 zeigt das Ergebnis dieses Algorithmus an einem kleinen Beispielgraphen.

4 Anforderungen durch die Kostümdomäne

Das Fachgebiet der Kostümwissenschaften ist sehr umfangreich. Das zentrale Element, das Kostüm ist hoch dynamisch. Es kann aus einer Vielzahl von Fakten in vielen unterschiedlichen Anordnungen komponiert werden. Äußere Restriktionen durch Film und Rolle schränken diese Möglichkeiten wieder ein. Die Anforderungen, die bei der Modellierung dieser Domäne erfüllt werden mussten sind in Abschnitt 4.1 aufgeführt.
Auch bei der Integration ins Wiki mussten spezielle Anforderungen beachtet werden. Die Umsetzung dieser Anforderungen sind in Abschnitt 5.4 beschrieben, die Anforderungen werden in Abschnitt 4.2 beschrieben.

4.1 Domänenmodellierung

Ziele der Modellierung der Kostümdomäne sind:

1. Ein Nachschlagewerk zu schaffen, das sowohl vom Menschen als auch der Maschine ausgewertet werden kann: Durch die Erfassung von Kostümen an einem zentralen Ort soll diese Plattform auch einen Mehrwert für andere Kostümschaffende bilden: Es sollen Zusatzinformationen zu den Kostümen abgelegt werden wie Nähanleitungen, Verleih- und Bezugsmöglichkeiten, Checklisten zur Kostümvorbereitung. Filmschaffenden, die neue Kostüme kreieren soll es zur Recherche und als Ablageort dienen. Forschern soll es eine rechnergestützte Recherche auf den Daten ermöglichen.
2. Eine formale Kostüm-Mustersprache zu entwerfen und zu verfeinern: Diese Sprache muss so beschaffen sein, dass sie die Realität von Kostümmustern abbilden kann. Durch Hinzufügen neuer Kostümmuster muss sich zeigen, wie gut die Sprache die Bedürfnisse der Domäne abdeckt.

Kostüme sind sehr detailreich. Ein Katalogsystem muss diese Details geeignet wiedergeben können:

- Kostüme sind zusammengesetzt aus Kostümteilen:
  - Jedes Kostümteil ist ein Kleidungsstück oder Accessoire: Die Anzahl dieser Basiselemente ist groß aber durch eine Taxonomie hinreichend erfassbar.

  - Jedes Kostümteil hat mindestens eine Farbe.
  - Jedes Kostümteil hat mindestens eine Form-Eigenschaft.
  - Jedes Kostümteil ist aus mindestens einem Material gefertigt.
  - Jedes Kostümteil hat mindestens ein Design.
  - Jedes Kostümteil kann in einem oder mehreren Zuständen (z.B.: nass) sein.
  - Jedes Kostümteil kann eine oder mehrere Funktionen erfüllen. (z.B.: Sportbekleidung)
  - Jedem Kostümteil kann eine oder mehrere Trageweisen zugeordnet sein.
  - Kostümteile sind unterschiedlich wichtig für eine Rolle. Diesem Umstand muss Rechnung getragen werden.
  - Jedes Kostümteil kann beliebig detailliert beschrieben werden. Es muss eine Möglichkeit geben, Teilbereiche zu beschreiben.
- Jedes Kostüm kann aus beliebig vielen Kostümteilen bestehen.
- Jedes Kostüm ist einer oder mehreren Szenen zugeordnet.
- Jedes Kostüm ist einer Rolle zugeordnet.

Durch eine semantische Erfassung für eine maschinelle Verarbeitung des Wissens ergeben sich folgende Forderungen:

- Alle Eigenschaften dürfen nur Werte annehmen, die Teil der Ontologie sind.
- Film, Rolle, Kostüm, Kostümteil und Teilbereich müssen durch semantische Attribute miteinander verknüpft sein. Eine textuelle Zuordnung genügt nicht.
- Die Benennung von Artikeln muss eindeutig sein.

Alle Kostümteile und Kostüme müssen die Möglichkeit zur Erweiterung durch Zusatzinformationen bieten.

4.2 Integration ins Wiki

Aus diesen Anforderungen ergeben sich Designentscheidungen die bei der Integration ins Wiki berücksichtigt werden müssen:

- Bei der Eingabe der Kostüme muss sichergestellt sein, dass nur bereits angelegte Fakten gewählt werden können und diese zur Auswahl ansprechend dargestellt werden. Die Ontologie darf nur vom Domänenexperten bearbeitet werden.
- Die grundlegende Struktur des Wiki sind einzelne Artikel: Jede Entität (Film, Rolle, Kostüm, Kostümteil, Teilbereich) wird durch einen eigenständigen Artikel repräsentiert und mit semantischen Links verknüpft.
- Es existieren nur Taxonomien der Eigenschaften, keine Ontologie. Die Axiome müssen mit der Einrichtung des Wikis in Wiki-Strukturen abgelegt werden, um die Ontologie zu vervollständigen. Um spätere Änderungen an der Ontologie in den erfassten Daten nachführen zu können, ist eine Programmierschnittstelle nötig.
- Der Taxonomie-Import muss automatisiert aus den modellierten Daten erfolgen.

5 Implementierung eines semantischen Wikis zur Erfassung der Kostümdomäne

In diesem Kapitel werden die Technologien erläutert, die für DataWiki von besonderer Bedeutung sind. Dabei werden hauptsächlich die Konzepte der Technologien beleuchtet, die für den Einsatz in oder das Zusammenspiel mit DataWiki entscheidend sind.
Im ersten Abschnitt dieses Kapitels werden die Beweggründe erörtert, die für die Entscheidung für DataWiki maßgeblich waren. Nach einem Überblick über DataWiki werden im letzten Abschnitt dieses Kapitels die Anstrengungen dokumentiert, die Unternommen wurden, um eine völlige Abdeckung der Kostümdomäne mit DataWiki zu erreichen.

5.1 Auswahl des Wikis

Es gibt einige semantische Wiki Implementierungen, die zum Teil eigenständige Projekte, zum Teil auch Erweiterungen herkömmlicher Wikis sind. Viele davon sind spezialisiert auf einzelne semantische Fähigkeiten wie Ontologiemodellierung oder Konsistenzüberprüfung der Ontologie durch prädikatenlogische Schlüsse. Das umfangreichste semantische Wiki ist SMW. Es erweitert die MW Plattform um semantische Daten und wird seinerseits wieder erweitert, so dass für SMW bereits viele Methoden und Werkzeuge aus unterschiedlichsten Anwendungsbereichen existieren. Viele davon sind sehr speziell, beispielsweise zur Verknüpfung von Geodaten mit Kartenmaterial - andere sehr allgemein gehalten.
SMW entstammt einem Projekt des Institut für angewandte Informatik und formale Beschreibungsverfahren (AIFB) des Karlsruher Institut für Technologie (KIT), wo 2005 die erste Version entwickelt wurde. Es liegt mittlerweile in Version 1.8.0.5 vor und wird ständig weiterentwickelt. Für SMW gibt es eine sehr lebendige Unterstützergemeinde und eine umfassende Dokumentation.
Eine Weiterentwicklung von SMW mit stärkerem Fokus auf Durchsuchbarkeit und Verwendbarkeit als wissenschaftliches Expertensystem ist Semantic MediaWiki Plus (SMW+). SMW+ wurde von Ontoprise gepflegt und weiterentwickelt. Dabei entstand eine Zusammenarbeit

mit der Investment Firma Vulcan Inc[1]. Vulcan brachte dabei Erkenntnisse aus dem Projekt Halo in SMW+ mit ein.
Im Mai 2012 stellte Ontoprise einen Insolvenzantrag. Daraufhin gingen die Rechte an SMW+ an die DIQA Projektmanagement GmbH (DIQA), die SMW+ unter dem Namen DataWiki weiterentwickelt und vertreibt. DataWiki als Erweiterung von SMW+ unterliegt der General Public License (GPL), die besagt, dass Weiterentwicklungen ebenfalls unter der GPL stehen und quelloffen sein müssen. DIQA vertreibt DataWiki kostenlos und bietet kostenpflichtigen Support an. Mit der Insolvenz von Ontoprise ist auch das Forum zu SMW+ und die Unterstützung durch die Benutzergemeinde weggebrochen.
Die Vorzüge von DataWiki, die letztlich zu einer Entscheidung für dieses Wiki geführt haben, ist die komfortable Installation als Komplettpaket (zumindest unter Windows) und die mögliche Anbindung an einen Triplestore, der auch von DIQA kommt. Da Semantic MediaWiki ohne die Halo-Extension über keinen geeigneten aktuellen Reasoner verfügt, schied dies als Alternative aus.
Durch die Entscheidung für DataWiki mussten auch einige Nachteile in Kauf genommen werden:

- **Wenig Dokumentation und kaum Support:** DataWiki liegt lediglich eine Installationsanweisung bei, in der einige (aber leider nicht alle) Konfigurationsmöglichkeiten erläutert werden. Supportanfragen per Email haben die Mitarbeiter von DIQA zwar freundlich beantwortet, aber das Fehlen eines Forums, eines Handbuchs und einer Benutzergemeinde machen sich in der täglichen Arbeit negativ bemerkbar.
- **Inkompatibilität mit neueren Versionen von Extensions:** DataWiki basiert auf MediaWikis Version 1.17.0. Die aktuelle Version ist 1.21.1. Dadurch bleiben viele Verbesserungen, nicht nur von MW sondern auch von seinen Erweiterungen, die oftmals zur Version 1.17.0 inkompatibel sind, unbenutzbar.
- **Kein Support für alte Versionen von SMW-Community:** Die Entwickler einer Erweiterung in einem OpenSource Projekt sind natürlich darauf bedacht, dass ihre Erweiterung besser wird, und die Kompatibilität zur aktuellen Version der Hauptsoftware gegeben ist. Fragt man nach 2 Jahre alten Fehlern, bekommt man meistens zu hören, dass diese in der neuesten Version nicht mehr auftreten. Das hilft leider nichts, wenn man aus Kompatibilitätsgründen an eine alte Version gebunden ist.

[1]Microsoft Mitbegründer Paul Allen ist Gründer und Alleinaktionär von Vulcan Incorporated

5.2 Verwendete Technologien

Dieser Abschnitt liefert einen kurzen Überblick über die Technologien, die bei der Implementierung des Wikis eine Rolle spielen. Die ersten 3 Abschnitte sind dabei dem XAMP-Stack gewidmet, der in der Web-Programmierung eine große Rolle spielt. Das X steht austauschbar für das Betriebssystem, A, M und P für Apache HTTP Server, MySQL und PHP.

5.2.1 Apache HTTP Server

Der Apache HTTP Server, der oft nur mit „Apache“ referenziert wird, ist der am weitesten verbreitete Webserver. Er bietet ein einfaches Deployment und kann mit dem entsprechenden Plugin jede der gängigen serverseitigen Sprachen interpretieren. Zusammen mit MySQL und PHP ist er in vielen Webprojekten erfolgreich im Einsatz. Durch den Einsatz von Caches kann der Apache auch bei großen Datenmengen und vielen Anfragen schnelle Ergebnisse liefern. Das ist gerade für Wiki-Anwendungen ein entscheidender Vorteil.

5.2.2 MySQL

Wie in vielen Webanwendungen kommt auch bei MediaWiki MySQL als Datenbank zum Einsatz. MySQL ist nach eigenen Angaben[ORA13] die populärste Open-Source-Datenbank der Welt. Diese relationale Datenbank arbeitet auf verschiedenen Datenbank-Schemata. Die bekanntesten sind MyISAM und InnoDB. MySQL wurde seit 1994 von Michael Widenius und David Axmark entwickelt, 2008 von Sun Microsystems aufgekauft, und gehört seit 2010 zu Oracle.
Der MySQL Erfinder Michael Widenius hat zum bestehenden MySQL einen Ableger entwickelt, der auch in Zukunft quelloffen bleiben muss. Dieser Ableger heißt MariaDB. Aufgrund der totalen Kompatibilität zu MySQL ist der Umstieg besonders leicht. Die Wikipedia ist bereits auf MariaDB umgestiegen.[Wik13]
MediaWiki legt alle Daten in der Datenbank ab. Dieses Vorgehen unterscheidet es beispielsweise von TWiki, das alle Artikel als Textdateien ablegt. Die Benutzung von MySQL ermöglicht MW-Erweiterungen wie SMW oder Halo, die Datenbank für Ihre Zwecke mitzubenutzen.

5.2.3 PHP

PHP ist eine Programmiersprache, die im Web-Umfeld sehr verbreitet ist. Der Name PHP steht für „PHP: Hypertext Preprocessor“. Das P in der Abkürzung ist rekursiv zu verstehen.

PHP-Quellcode wird von allen gängigen Webservern unterstützt. Der Webserver interpretiert den Code und generiert dabei HTML-Markup, das an den Client gesendet wird. PHP ist leicht zu erlernen, da seine Syntax stark an die von Java und c++ angelehnt ist. Die besonders einfache Anbindung an Datenbanksysteme und die Erweiterbarkeit durch zahllose Bibliotheken machen es zu einem festen Bestandteil eines Großteils aller Webseiten.
Mit PHP lassen sich auch große Webprojekte realisieren. Bibliotheken können leicht eingebunden werden und Quellcode lässt sich gut in kleinere Module zerlegen. Ein großer Vorteil ist, dass benutzerdefinierte Funktionen und Methoden ausgeführt werden können. Dabei wird die Funktion oder Methode als String, und ihre Argumente als weiterer String an die Funktion `call_user_func($func, $param)` oder `call_user_method($func, $param)` übergeben. Wollte man so etwas beispielsweise in Java realisieren würde man nicht um den Gebrauch der Java Reflect API herumkommen. Hierbei müssten die unterschiedlichen Datentypen in Einklang gebracht werden was umfangreiche Typkonvertierungen zur Folge hätte. Durch PHPs Typlosigkeit ist es hier sehr viel flexibler.
Dieses einfache Einbinden von Funktionen ermöglicht ein Vorgehen, bei dem sich eine Erweiterung bei einer globalen Instanz registriert. An jeder beliebigen Stelle im Quellcode (Hook) kann nun überprüft werden, ob eine registrierte Erweiterung an dieser Stelle mit dem Programm interagieren möchte. Aus der Sicht der Erweiterung ist diese Stelle ein Haken, an dem der eigene Code eingehängt werden kann. Deswegen nennt man solche Stellen FunctionHooks. MediaWiki bedient sich dieses Konzepts durchgängig. Über 430 dieser Hooks sind in MediaWiki dokumentiert.[med13b]
MediaWiki verwendet auch ParserFunctions und MagicWords, deren Konzepte ebenfalls auf die Verwendung von Hooks zurückgehen, jedoch dergestalt spezialisiert sind, dass dabei Schlüsselwörter in Benutzereingaben ausgewertet werden. So können sich hinter einem MagicWord eine Variable, eine Funktion oder eine vom Benutzer angelegte Vorlage verstecken.
PHP ermöglicht ein gutes Design im Sinne des Model-View-Controller (MVC)-Entwurfsmusters. Das MVC-Design sieht eine Trennung zwischen der Anwendungslogik (Model), der Benutzeroberfläche (View) und der Benutzerinteraktion (Controller) vor. Bei Webanwendungen ist hier immer noch zu betrachten, dass PHP auf dem Server interpretiert wird, der Benutzer jedoch an einem Webbrowser sitzt. Deswegen wird PHP oft zusammen mit CSS und JavaScript verwendet.
CSS übernimmt dabei einen Teil der Gestaltung, und gibt Antwort auf die Frage „Wie sieht ein einzelnes Element aus? “ JavaScript und PHP sind für die Positionierung der Elemente im HTML-Markup verantwortlich, JavaScript unterstützt bei der Eingabe und Navigation auf der Seite und kann Benutzereingaben validieren. PHP interpretiert die HTTP-Requests und reagiert darauf.

5.2.4 JavaScript

JavaScript ist eine Scriptsprache auf Clientseite. Seine Syntax basiert auf Java, im Gegensatz zu Java kennt es aber keine Typen. JavaScript wird häufig dazu verwendet, auf Ereignisse im Browser zu reagieren (`onClick()`, `onLoad()`), Eingaben zu validieren oder Hilfestellungen anzubieten. Außerdem bietet es viele Möglichkeiten, Webseiten dynamisch wirken zu lassen, indem visuelle Effekte auf das HTML-Markup angewendet werden. Für alle lokalen Belange, für die keine Interaktion mit dem Server nötig ist, sollte eine clientseitige Scriptsprache zum Einsatz kommen.
Im Gegensatz zur Architektur des Servers, ist die Software, die auf dem Client eingesetzt wird, nicht im Voraus bekannt. So unterstützen verschiedene Browser nicht alle JavaScript Befehle, bzw. interpretieren diese unterschiedlich. Glücklicherweise gibt es hier mit jQuery eine Bibliothek, die diese Unterschiede zwischen den Browsern größtenteils transparent behandelt.

jQuery

jQuery ist eine Bibliothek für JavaScript. Sie implementiert viele Funktionen zum Durchsuchen und zur Manipulation des Document Object Model (DOM)-Baumes. Weiterhin bietet sie Funktionen für komplexe Events und visuelle Effekte an.
Der Einsatz von jQuery ist sehr simpel: In jeder JavaScript-Umgebung kann man mit einem `$()` oder der Funktion `jQuery()` eine Filterfunktion auf den DOM-Baum anwenden, die eine Menge von jQuery-Objekten zurückgibt, auf denen eine Vielzahl von Funktionen durchgeführt werden kann.

```
$('.target').hide(2000);
```

blendet beispielsweise alle Elemente aus, die mit `.target` selektiert wurden. Der vorstehende Punkt ist der Selektor für das Hypertext Markup Language (HTML)-Attribut „class“. Es werden in diesem Beispiel alle DOM-Elemente, die `"class=target"` besitzen in einer 2 Sekunden dauernden Animation ausgeblendet. Ebenso einfach kann man einem Element ein Event zuweisen:

```
$('.target').click(function(){alert($(this) + " wurde geklickt"});
```

Eine besonders nützliche Eigenschaft von jQuery ist die Funktion `delegate()` oder `live()`, mit der man einem Element einen Eventhandler zuweisen kann, das zu diesem Zeitpunkt noch garnicht existiert. Wird das Element erzeugt, überprüft jQuery, ob es mit dem Filter

der `delegate()` Funktion übereinstimmt, und führt im Erfolgsfall die Anweisung darauf aus. Mit dieser Funktion kann man das Verhalten fremder JavaScript-Dateien beeinflussen, auch wenn man auf die Datei selbst keinen Zugriff hat.

Dynatree Dynatree ist eine Erweiterung von jQuery die Operationen auf Baumstrukturen zur Verfügung stellt. Knoten lassen sich nach Belieben einfügen, sortieren, auf- und zuklappen und entfernen. Die Baumknoten werden im HTML-Markup als `<ul>`-Liste dargestellt. Der Baum kann mit unterschiedlichen CSS-Dateien gestaltet werden. Im Standard liefert Dynatree ein Custom-Thema und eines im Stil von Windows Vista.

5.2.5 Asynchronous JavaScript and XML (Ajax)

[Win09] Ajax ist ein Konzept, um Webseiten dynamisch zu gestalten. Vor Ajax mussten Webseiten immer am Stück geladen werden. Formulareingaben wurden als Parameter zwischen Client und Server hin und hergeschickt, oder in Session-Variablen oder Cookies gespeichert, um eine Illusion von Dynamik zu erzeugen. Ajax hingegen ist in der Lage, einzelne Bereiche einer Website dynamisch nachzuladen, indem es asynchron, also nicht zum Zeitpunkt der Generierung der Seite, Nachrichten zum Server schickt. Dieser kann nun die Anfrage bearbeiten (beispielsweise mit einer Datenbankabfrage) und die Antwort als XML oder JavaScript Object Notation (JSON) Objekt zum Client zurücksenden. Dort kann JavaScript einfach den Inhalt in die bestehende Webseite einbinden, ohne Daten in anderen Bereichen anzufassen. Das erspart viel Parametrisierungsaufwand und macht echtes dynamisches Verhalten erst möglich.
Wie bereits erwähnt kann das Ergebnis als XML-Markup oder JSON Objekt verschickt werden. Geht es schlicht darum, direkt HTML Inhalte einzublenden, sind JSON Objekte überflüssig und nicht empfehlenswert. Muss man die Daten zuerst noch weiterverarbeiten und braucht dazu Javascript - Objekte, empfiehlt es sich, die Nachrichten als JSON Objekt zu versenden.

5.2.6 Cascading Style Sheets (CSS)

CSS enthalten wichtige visuelle Informationen, die auf die unterschiedlichen HTML-Elemente angewendet werden. Die Referenzierung geschieht dabei über das DOM, das jeder Website zugrunde liegt.
Diese Hierarchie sorgt dafür, dass untergeordnete HTML-Elemente die Style-Informationen ihrer übergeordneten Elemente erben. Auf jeder Ebene können Style-Informationen neu angepasst werden. DataWiki enthält mit „ontoskin“ ebenfalls ein eigenes CSS-Design.

5.2.7 Java

Java spielt als Programmiersprache durch ihre klare Struktur, den großen Funktionsumfang und die weite Verbreitung in fast allen Bereichen eine große Rolle. Unter Verwendung von Servlet-Containern wie Apache Tomcat, GlassFish oder Apache Jetty lassen sich mit Java umfangreiche Webservices bauen. Java ist im Zusammenhang mit DataWiki nur im Triplestore und SOLR[Fou13b] vorhanden. Die Wiki-Bots sind ebenfalls in Java geschrieben.

5.3 DataWiki

DataWiki ist hauptsächlich eine Kombination aus Erweiterungen zu MediaWiki. In diesem Kapitel sollen die Eigenschaften von MediaWiki und den in DataWiki enthaltenen Erweiterungen beschrieben werden, um so einen Gesamteindruck von DataWiki zu gewinnen.

5.3.1 MediaWiki (MW)

MediaWiki ist das Wiki, das für die Wikipedia entwickelt wurde. Es bietet das Grundgerüst für alle semantischen Erweiterungen. Es bietet eine Benutzer- und Versionsverwaltung, die Markup-Sprache „WikiText“ und das Kategorienkonzept. Das wichtigste Feature von MW ist aber seine Erweiterbarkeit. Die große Anzahl an verfügbaren Extensions ist ein Zeugnis dafür, wie gut das Erweiterungskonzept durchdacht ist.

5.3.2 Erweiterung: Semantic MediaWiki (SMW)

SMW ist die umfangreichste Erweiterung von MediaWiki. Sie führt semantische Attribute ein. Sie ermöglicht Abfragen auf diesen strukturierten Daten und stellt einige Formate zur Verfügung, in denen Abfrageergebnisse präsentiert werden können. SMW hat sich dem Gedanken des Semantic Web stark verpflichtet, indem es eine einfache Funktion geschaffen hat, um das Wiki zu veröffentlichen, das heißt, die semantischen Daten für Dritte zugänglich zu machen.

5.3.3 Erweiterung: Halo

Halo ist eine der wichtigsten Erweiterungen für DataWiki. Halo verbessert die Eingabemöglichkeit durch die „Data Toolbar“. In dieser Toolbar lassen sich semantische Annotationen schnell und unkompliziert durchführen. Bei allen Feldern in dieser Toolbar wird die Auswahl

der Attribute zusätzlich durch Autovervollständigung vereinfacht. Ein weiterer wichtiger Punkt, den Halo erfüllt ist die Anbindung an den DIQA Triplestore über den Triplestore-Connector. Halo sorgt dafür, dass die Daten aus dem Wiki in einen Graphen geladen werden und an den Triplestore geschickt werden. Auch das Query-Interface und der Data-Explorer sind durch Halo in DataWiki vorhanden.

5.3.4 Erweiterung: Semantic Result Formats

Abfrageergebnisse in Semantic MediaWiki lassen sich in den unterschiedlichsten Formaten anzeigen. Diese Erweiterung liefert von der einfachen Tabelle bis zur TagCloud alles. Eine Übersicht über die unterschiedlichen Darstellungsformate findet sich auf den Seiten der Erweiterung.[med13a]

5.3.5 Erweiterung: WYSIWYG

Diese Extension liefert einen Editor für Wikiseiten, der anstelle von Wiki-Markup bereits formatierten Text (What you see is what you get) und andere Elemente(Bilder, Abfragen) als Grafiken anzeigt. Daraus wird dann Wikitext generiert. Die Eingabe im WikiText-Editor ist etwas performanter und wenn man direkt mit den semantischen Attributen arbeiten möchte ist der WYSIWYG-Editor eher hinderlich. Für den User ist dieser Editor jedoch sehr nützlich und komfortabel, da keine Kenntnisse über das Wiki-Markup nötig sind.

5.3.6 Erweiterung: Semantic Drilldown

Semantic Drilldown ist eine Erweiterung, die es erlaubt über die Attributwerte eine Filterung vorzunehmen und somit Artikel anhand ihrer annotierten Attribute zu filtern. Das entspricht dem Semantischen Browsen, das durch SMW implementiert wurde, nur mit dem Unterschied, dass nicht auf allen Artikeln gesucht wird, sondern nur auf einer bereits im vorhergehenden Suchschritt eingeschränkten Ergebnismenge.

5.3.7 Erweiterung: Validator

Validator ist Teil des Semantic Bundle und eine Pflichterweiterung, wenn man SMW benutzen möchte. Es validiert vom Benutzer eingegebene Parameter und generiert besser verständliche Fehlermeldungen.

5.3.8 Erweiterung: Enhanced Retrieval

SOLR ist eine Suchmaschine aus dem Projekt Apache Lucene.[Fou13b] Es bietet eine klassische Volltextsuche, Facettensuche, Text-Highlighting, schnelle Indexierung, Datenbankunterstützung und ist durch viele Programmierinterfaces anbindbar. SOLR läuft als eigenständige Software mit einem Jetty-Server und führt die Suchanfragen von DataWiki aus. Dafür ist die SMW Extension „Enhanced Retrieval“ zuständig, die mit DataWiki ausgeliefert wird.

5.3.9 Erweiterung: SemanticForms (SF)

Diese Erweiterung generiert auf Vorlagen basierend HTML-Formulare. Um ein Formular zu benutzen, muss es auf eine Vorlage abgestimmt sein. Ein Formular kann auch mehr als eine Vorlage befüllen. Für das Anlegen der Formulare stellt SF eine komfortable Benutzeroberfläche zur Verfügung. Formulare können die Restriktionen, die in einer Vorlage hinterlegt sind darstellen. Akzeptiert ein Attribut beispielsweise nur eine begrenzte Anzahl an Werten und eine Mehrfacheingabe ist erlaubt, werden die Werte in Form von Checkboxen dargestellt. Die Formularfelder arbeiten mit Autovervollständigung, wenn dies konfiguriert ist.
Um einen neuen Artikel im Wiki mit Hilfe eines Formulars zu erzeugen, wählt man eine spezielle Seite von SF `Special:FormStart` und gibt dort an, welches Formular verwendet werden, und wie der neue Artikel heißen soll. SF liefert auch eine Methode, mit der man diese Eingabefelder direkt in eine vorhandene Seite oder Vorlage einbinden kann. Es bietet die Möglichkeit, den Artikel- und Formularnamen vorzugeben, so dass sich der Lösungserfasser nicht um die richtige Abfolge von Formularen und Seitennamen kümmern muss. Generierte Artikelnamen können eine fortlaufende oder zufällige Nummer enthalten. Da alle diese Informationen in der URL codiert werden, lassen sich auch beliebige Attribute und ihre Werte über den Querystring des HTTP-GET Protokolls mitsenden. SF wertet diese Attribute aus. Dieses Verhalten kann dazu verwendet werden, semantische Verknüpfungen zwischen Artikeln herzustellen. Zum Beispiel gehört ein Kostümteil zu einem Kostüm. Klickt man die Schaltfläche „Kostümteil anlegen“ auf der Seite eines Kostüms erhält automatisch das Attribut „gehört zu“ des neuen Kostümteils den Seitennamen des Kostüms als Wert. Schickt man das Formular ab, werden die eingegebenen Werte den verknüpften Attributen in den Vorlagen zugewiesen und anschließend im neu angelegten Wiki-Artikel angezeigt. Abbildung 5.1 zeigt den Ausschnitt eines SF-Formulars, und die zugehörigen Wikitext Einträge.

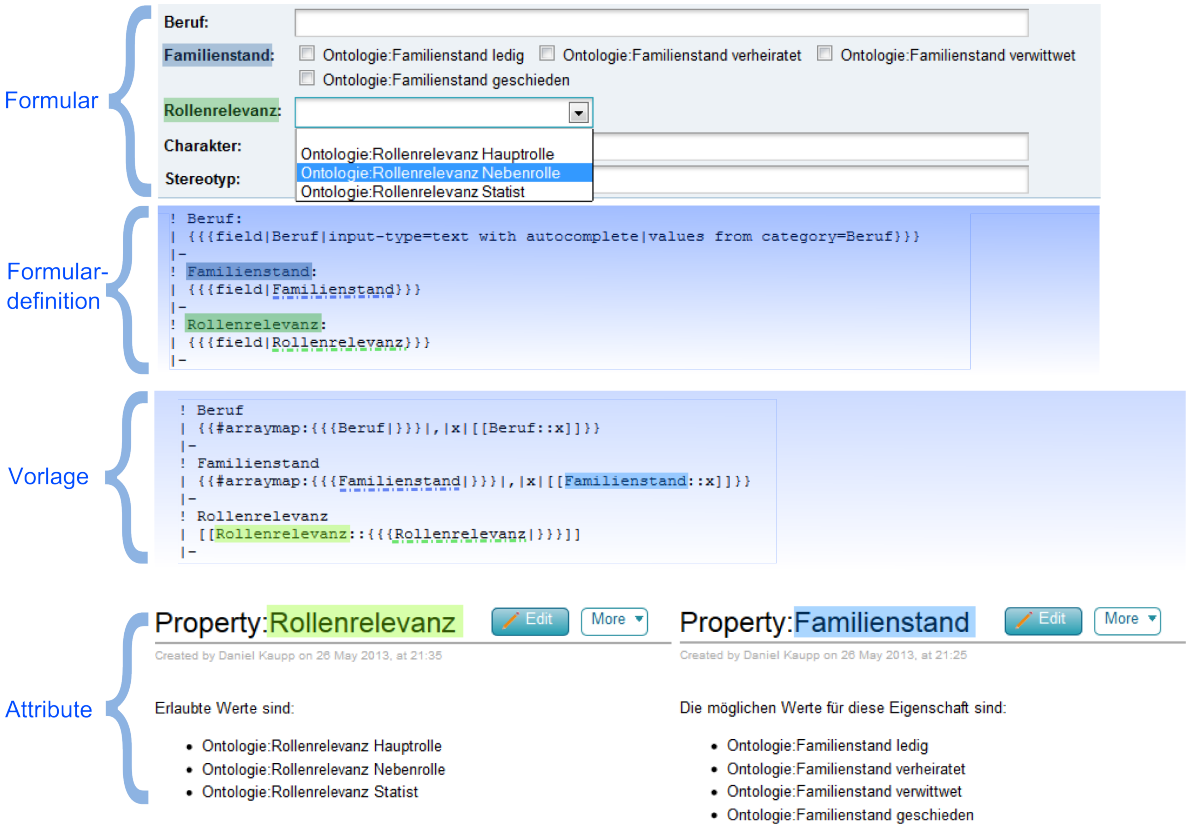

Abbildung 5.1: Formular, Formulardefinition, Vorlage und semantische Attribute einer Eingabe von SemanticForms

5.3.10 Erweiterung: SF Select (SFS)

SFS erweitert SF dergestalt, dass es eine vordefinierte Query auswertet, und die Ergebnisse dieser Abfrage mit Ajax ins Formular transportiert und als zulässige Wertemenge für das Eingabefeld zur Verfügung stellt. Dabei kann die Query eine Variable enthalten, die den Wert eines anderen Feldes annimmt. Sobald eines der von SFS überwachten Felder geändert wird, werden alle Queries neu abgeschickt und die Werte, die in Dropdown-Feldern angeboten werden bei Bedarf aktualisiert.
Dieses Vorgehen ermöglicht es, durch hierarchische Daten zu manövrieren. Da eine Hierarchie jedoch üblicherweise in einer Baumstruktur dargestellt wird, wurde die Vorgehensweise von SFS zum Vorbild für SFOS.

5.3.11 DIQA Triplestore Basic

Der DIQA Triplestore ist im Gegensatz zu DataWiki keine quelloffene Software. Er wird trotzdem kostenlos in Form einer Evaluationslizenz zur Verfügung gestellt. Der Triplestore benutzt Teile von Apaches Jena Framework[Fou13a]. Dieses steht unter der Apache 2.0 Lizenz, welche besagt, dass Programme, die Code verwenden, der unter der Apache 2.0 Lizenz veröffentlicht wurde auf diesen Umstand hinweisen müssen, aber ihre Lizenz frei wählen dürfen. Triplestore Basic lässt sich in 3 Ebenen[2] aufteilen:

- **RDF-Ebene:** Auf dieser Ebene kommuniziert der Triplestore über SPARQL und SPARQL Update Language (SPARUL). Über einen Webserver empfängt der Triplestore auf Port `:8080` SPARQL Abfragen und auf Port `:8081` SPARUL Updates. DataWiki benutzt diese Schnittstelle, um die Tripel aus dem Wiki in Form eines Graphen in den Tripelstore zu laden. Der Triplestore ist auch in der Lage, MediaWiki Ask-Queries auszuführen. Auf dieser Ebene werden die Sprachkonstrukte geparst und zur internen Datenverarbeitung in das Graph Format umgewandelt.
- **Inferenz-Ebene:** Der Triplestore kann mit verschiedenen Reasonern[3] zusammen benutzt werden. In der Standardinstallation wird ein „GenericRuleReasoner“ verwendet, der Transitivität, Symmetrie und Inversion von Attributen unterstützt. Diese Regeln sind über Textfiles auch erweiterbar. An dieser Stelle wird vermutlich die Inference-API von Jena verwendet, die auch eine Auswahl von externen Reasonern zulässt.
- **Speicher-Ebene:** Zur Speicherung der Tripel sieht Triplestore Basic mehrere Möglichkeiten vor, die sich an der Store API von Jena orientieren: Eine Speicherung im Hauptspeicher ist die Standardauswahl, doch auch die Jena TDB und SDB Modelle können gewählt werden.
  Jena SDB basiert auf einer SQL-Datenbank, in der die Tripel persistent abgelegt werden. Bei Jena TDB werden die Tripel in Tabellen auf der Festplatte abgespeichert. Als Datenstruktur dienen hier Tabellen und B^+-Bäume. Auch die Verwendung weiterer semantischer Datenbanken ist möglich.
  Der Austausch zwischen der Speicher- und der Inferenz-Ebene erfolgt über einen Graph-Datentyp. Möchte man also eine andere Datenbank anbinden, muss lediglich das Interface für diesen Datentyp angepasst werden.

[2] Angelehnt an die Grafik unter http://jena.apache.org/about_jena/architecture.html
[3] Programm, mit dem Schlussfolgerungen gezogen werden (engl.: to reason)

5.4 Anpassung und Anwendung

Obwohl Semantic Media Wiki mit den Erweiterungen SemanticForms und SemanticForms-Select schon sehr gute Möglichkeiten bietet, um die Benutzereingabe zu vereinfachen und vereinheitlichen, gibt es an vielen Stellen noch Bedarf für weitere Funktionalitäten. Zwar wird SemanticForms kontinuierlich weiterentwickelt, allerdings ist es zusammen mit DataWiki nur in der Version bis 2.4 kompatibel.
Um diese Problematik zu umgehen, entstand die Erweiterung: SFOS, die stark auf SFS basiert, diese aber um einige optische Effekte, eine zweite Eingabemethode mit Baumstruktur und eine vereinfachte Integration in das Formular erweitert. SFOS kann überall dort zum Einsatz kommen, wo aus einer hierarchischen Struktur Fakten gewählt werden sollen. Es unterstützt auch die Eingabe von mehreren Werten und wird in Abschnitt 5.4.2 ausführlich beschrieben.
Zum Import der Taxonomien und zur Analyse von Ähnlichkeiten entstanden drei Wiki-Bots, die in Abschnitt 5.4.3 beschrieben sind. Diese basieren auf der MediaWiki Java Programmierschnittstelle und führen ihre Funktionen über das Hypertext Tansfer Protocol aus.
Für die Repräsentation der Patterns wurde die Erweiterung „PatternRepository“ verwendet. Diese entstand im Zuge der Diplomarbeit von Norbert Fürst.[Fü13] PatternRepostitory wird kurz in Abschnitt 5.4.2 vorgestellt.

5.4.1 Einrichtung

Namensräume

Um eine Abgrenzung zwischen Ontologiewissen und Instanzen zu schaffen, und letztere sauber zu gliedern, wurden folgende zusätzliche Namensräume in den Semantic MediaWiki Einstellungen definiert:

- `$wgExtraNamespaces[120]="Ontologie"`
- `$wgExtraNamespaces[121]="Ontologie_Talk"`
- `$wgExtraNamespaces[122]="Rolle"`
- `$wgExtraNamespaces[123]="Rolle_Talk"`
- `$wgExtraNamespaces[124]="Film"`
- `$wgExtraNamespaces[125]="Film_Talk"`
- `$wgExtraNamespaces[126]="Kostuem"`
- `$wgExtraNamespaces[127]="Kostuem_Talk"`

- `$wgExtraNamespaces[128]="Pattern"`
- `$wgExtraNamespaces[129]="Pattern_Talk"`
- `$wgExtraNamespaces[130]="Kostuemteil"`
- `$wgExtraNamespaces[131]="Kostuemteil_Talk"`
- `$wgExtraNamespaces[132]="Abfrage"`
- `$wgExtraNamespaces[133]="Abfrage_Talk"`

In den Namensräumen musste aus Kompatibilitätsgründen mit dem Triplestore auf Umlaute verzichtet werden, da diese immer wieder zu Fehlern führten. Der Ontologie Namensraum wird vor Bearbeitung durch einen nicht Administrator geschützt.

5.4.2 Erweiterungen

Erweiterung: SF Ontology Select (SFOS)

Diese Erweiterung entstand im Rahmen dieser Arbeit. Sie erweitert die Eingabemöglichkeiten von SFS um eine Baumstruktur. Diese Baumliste kann traversiert werden und lädt dabei dynamisch die Unterelemente der nächsten Ebene nach, wenn diese aufgeklappt wird. Da parallel auch noch zwei weitere Eingabemöglichkeiten Verwendung finden sollen, ist es wichtig, Synchronizität zwischen den Eingabeelementen zu erzeugen. Dies wird durch Abbildung 5.2 illustriert. Die aufgeklappte Baumebene enthält die selben Elemente wie das Dropdown. Für das Basiselement ist keine Mehrfachauswahl gestattet.

Die anderen Eingabemöglichkeiten sind die im vorausgehenden Abschnitt beschriebenen Dropdown-Felder und die SF-Standardtexteingabe mit Autovervollständigung. Abbildung 5.3 zeigt diese drei Eingabemöglichkeiten.

Bei der Programmierung dieser Erweiterung wurden auch Schritte unternommen, um den Wikitext, der für die Erzeugung dieser Felder notwendig ist, zu optimieren. So musste in SFS für jedes Feld ein Eintrag im Formular vorhanden sein. Für SFOS genügt ein Eintrag mit einem Argument, bis zu welcher Hierarchietiefe Felder erzeugt werden sollen. Ausserdem bietet SFOS eine ansprechende Form, einem Attribut mehrere Werte zuzuweisen. Ob dies erlaubt ist, wird ebenfalls direkt dem in der Vorlage hinterlegten Wert entnommen. Wird ein weiterer Wert gewünscht, wandert der bereits ausgewählte Wert in eine Zeile über der Formularzeile. Durch Klick kann er bei Bedarf wieder gelöscht werden.

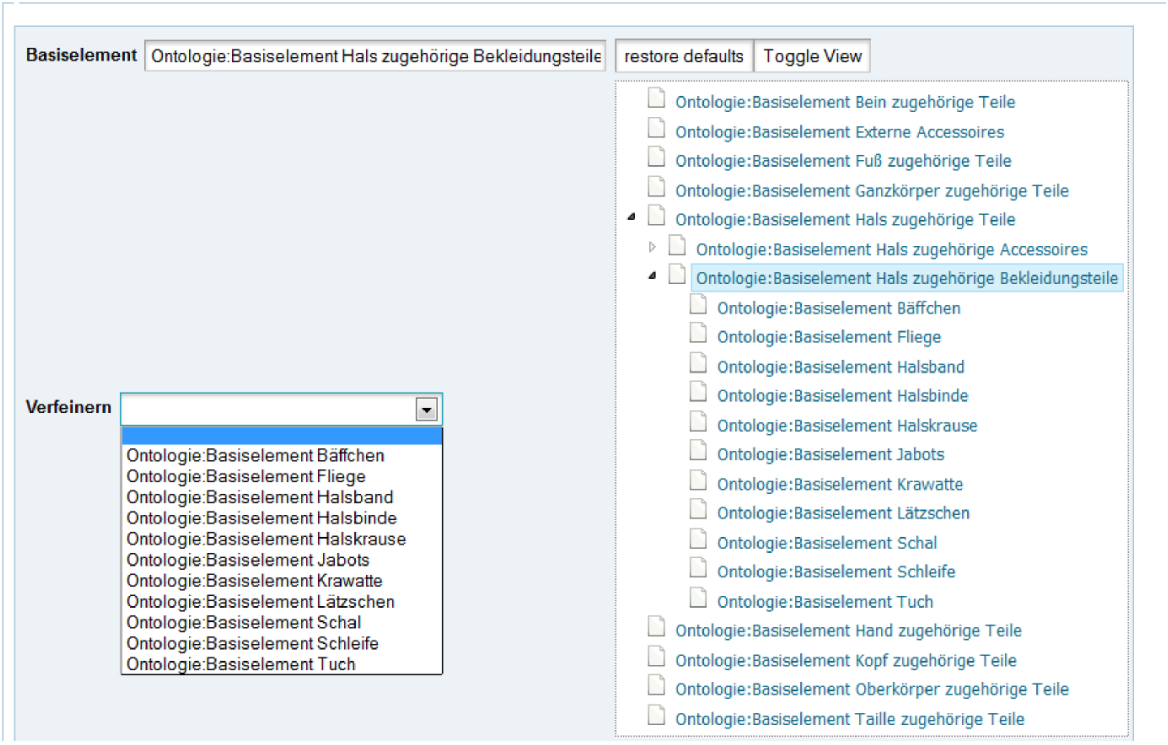

Abbildung 5.2: SFOS Darstellung für die Basiselement-Auswahl

Erweiterung: PatternRepository

PatternRepository stellt das in Abschnitt 3.1.2 vorgestellte Format zum Ablegen von Patterns zur Verfügung. Es bietet zudem komfortable Funktionen zum semantischen Annotieren von Patterns, zum Anlegen eines neuen Pattern und zur Verwaltung. Ausserdem werden verwandte Pattern als Leseempfehlung angegeben. PatternRepository wird in der Diplomarbeit von Norbert Fürst ausführlich beschrieben.[Fü13]

5.4.3 Wiki-Bot Implementierungen

Wiki Import-Bot

MediaWiki verfügt über Programmierschnittstellen für die gängigen Programmiersprachen. Hier wurde die API für Java verwendet. Um die Daten der Ontologie in das Wiki zu importieren, genügen zwei Funktionen der Java Wiki API: Das Programm muss sich im Wiki anmelden (`login()`) und Artikel editieren (`edit(articleName, newText, comment)`). Die Hauptaufgabe besteht im Auslesen des Taxonomieformats. Dabei müssen die Unterelement-Beziehungen zwischen den einzelnen Fakten berücksichtigt werden. Für die Darstellung

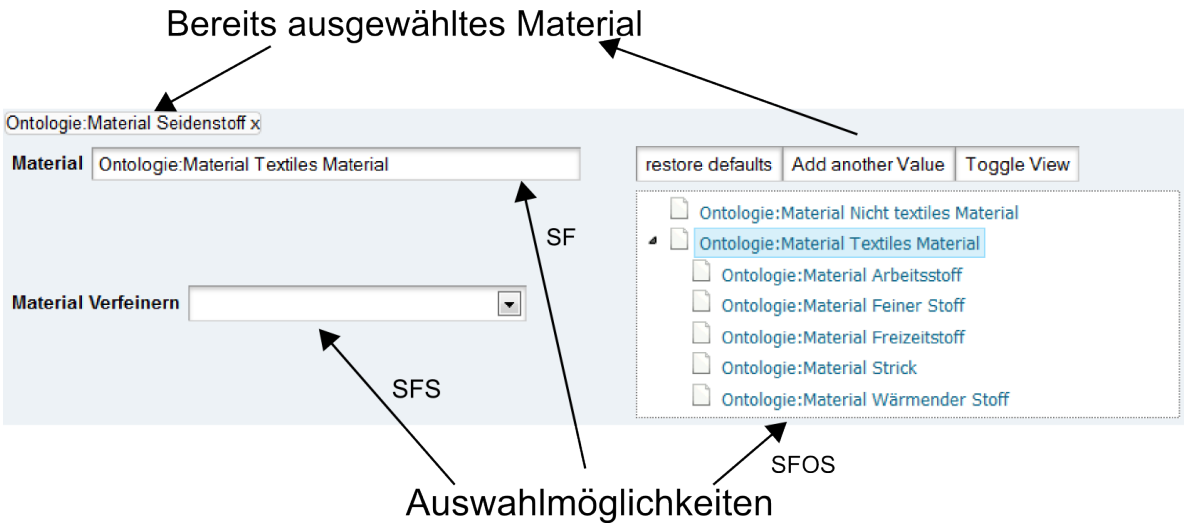

Abbildung 5.3: SFOS Darstellung für die Material-Eigenschaft

im Wiki muss eine geeignete Form gefunden werden. Das Schema für die Artikelbenennung ist dabei dergestalt, dass alle importierten Artikel das Präfix „Ontologie:“ erhalten, und somit dem Namensraum „Ontologie“ zugeordnet werden. Danach folgt der Name der Taxonomie und der Name des Fakts. Der Taxonomiename bildet eine Kategorie „Ontologie Taxonomiename“, und jeder in der Taxonomie vorkommende Fakt wird dieser Kategorie zugeordnet. Die Kategorie bietet einen guten Überblick über die bereits importierten Elemente. Da eine Taxonomie hierarchisch angeordnet ist, kommt hier keine Mehrfachvererbung zum Einsatz. Für manche Fakten der Kostümdomäne wäre es wünschenswert, nicht nur einer Oberklasse zuzugehören, sondern auch einer weiteren. In der objektorientierten Programmierung kennt man hierfür das Konzept der Interfaces. Für SMW kann man in diesem Fall die Kategorie als vergleichbares Konzept sehen. Der Vergleich hinkt zwar etwas, da eine Kategorie ja nur eine Eigenschaft transportiert, und ein Interface eine ganze Klasse an Eigenschaften, doch auch die hierarchische Verknüpfung durch ein semantisches Attribut vererbt nur die Hierarchie-Beziehung selbst. Folgendes Beispiel aus der Kostümdomäne macht dies deutlich:
Die Basiselement-Hierarchie ist nach den Körperbereichen sortiert, an denen ein Kleidungsstück getragen wird. Möchte man jetzt eine Abfrage nach allen Schmuckstücken vornehmen, kann man in der Hierarchie kein gemeinsames Oberelement (außer dem Wurzelelement) finden. Sucht man im Gegensatz dazu nach allen Kleidungsstücken die am Oberkörper getragen werden gibt es einen solchen Artikel. Diesem Problem kann man dadurch begegnen, dass man alle Schmuckstücke in eine Kategorie:Schmuck einordnet. Das ist allerdings Teil des Domänenwissens und muss vom Domänenexperten modelliert werden. Über eine Suche nach Teilbegriffen eines Artikels (z.Bsp. Reif, Ring, Kette für Schmuck) kann in einem fortgeschrittenen Importer auch solches Wissen extrahiert und ins Wiki importiert werden. Diese

Modellierung von „nicht-hierarchischen“ Eigenschaften kann auch als Wartungsfunktion von einem Programm ausgeführt werden.

Wiki Similarity-Bot

Im Gegensatz zum Import-Bot, der auf einmaligen Gebrauch zum Import der Taxonomien ausgelegt ist, sollte der Similarity-Bot ständig im Hintergrund arbeiten. Dieser Bot beobachtet die Kostüm-Seiten, die sich ändern oder neu angelegt wurden und führt zu jeder dieser Seiten eine Ähnlichkeitsanalyse wie in Abschnitt 3.2.7 beschrieben aus. Die Ergebnisse dieser Analyse schreibt er auf die Wikiseite der entsprechenden Kostüme. Obwohl diese Einträge nicht angezeigt werden, erzeugen sie doch eine Leerzeile, wodurch der Artikel pro Kostüm eine Zeile tiefer rutscht. Bei einer Überarbeitung des Bots muss überprüft werden, ob es Vorteile bringt, wenn die Ähnlichkeitswerte direkt über SPARQL CONSTRUCT in den Datengraphen modelliert werden.
Da die Abfragen zur Ähnlichkeit eventuell etwas länger dauern können, wenn die Datenmenge steigt, werden die zu untersuchenden Elemente in einer First In First Out (FIFO) Warteschlange verwaltet. Die anschließenden Änderungen werden ebenfalls in einem separaten Thread in einer Queue abgearbeitet. Für den Zugriff auf eine Wikiseite gibt es einen Schwellwert, der in jedem Fall abgewartet wird, bis die Aktion als beendet gilt, so dass die Zeit, die zum Editieren der Seiten gebraucht wird die Analysedauer des Abfrageergebnisses bei weitem übersteigt.

Wiki CLIQUE-Bot

Der CLIQUE-Bot arbeitet ebenfalls im Hintergrund und bildet aus den Paaren ähnlicher Kostüme für jeden diskreten Ähnlichkeitswert über eine Abfrage einen Graphen. Dazu wurde die freie Java Bibliothek JGraphT[jgr13] verwendet. Diese Bibliothek liefert auch eine fertige Implementierung des CLIQUE-Algorithmus nach Bron und Kerbosch.

Die Zugehörigkeit zu einer gefundenen Clique wird im Artikel des betroffenen Kostüms vermerkt. Das Ergebnis für den Wert 7 zeigen die Abbildungen 5.4 und 5.5

5.4.4 Vorlagen und Formulare

Die im Namensraum „Ontologie“ enthaltenen Begriffe bilden nur einen Teil der Ontologie ab. Bestandteil der Ontologie sind auch die Zusammenhänge zwischen Kostüm und Kostümteil oder die Kardinalitäten eines Attributs. Dieses Wissen muss nachträglich in das Wiki eingebracht werden. Dies geschieht hauptsächlich über Vorlagen: Eine Vorlage kann an

	CLIQUE-id
Kostüm-0000002	CLIQUE-7-1921cf88-5
Kostüm-0000004	CLIQUE-7-1921cf88-5
Kostüm-0000005	CLIQUE-7-1921cf88-5
Kostüm-0000006	CLIQUE-7-1921cf88-5
Kostüm-0000001	CLIQUE-7-e8817a9c-1
Kostüm-0000002	CLIQUE-7-e8817a9c-1
Kostüm-0000006	CLIQUE-7-e8817a9c-1
Kostüm-0000002	CLIQUE-7-effd4adc-e
Kostüm-0000003	CLIQUE-7-effd4adc-e
Kostüm-0000004	CLIQUE-7-effd4adc-e
Kostüm-0000005	CLIQUE-7-effd4adc-e

Abbildung 5.4: Abfrageergebnis zur CLIQUE mit Wert 7

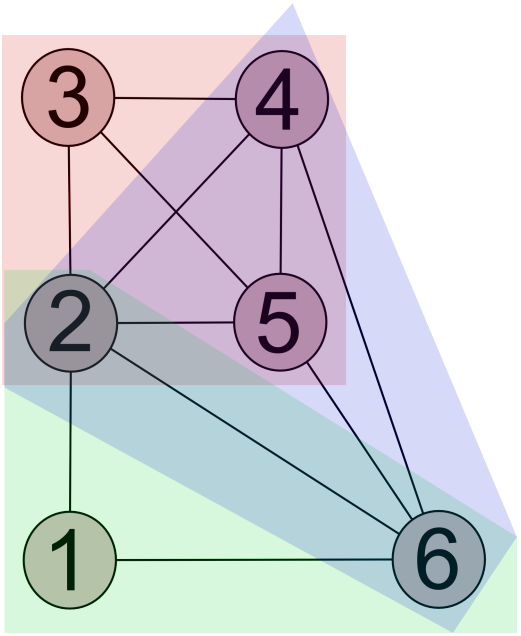

Abbildung 5.5: Graph für Wert 7 mit eingefärbten CLIQUEN

beliebiger Stelle in eine Seite eingebunden werden. Der Artikel wird dann so angezeigt, als Stünde der Text der Vorlage direkt im Artikel. SemanticForms macht von dieser Eigenschaft Gebrauch und erstellt eine neue Seite mit den dem Formular zugrundeliegenden Vorlagen. In der Vorlage kann auch weiterer Text stehen, den SemanticForms nicht auswertet. Dadurch können leicht Abfragen spezifiziert werden, die alle dem Kostüm zugeordneten Kostümteile auflisten. Desweiteren kann in der Vorlage auch eine Schaltfläche definiert werden, die ein neues Formular basierend auf einer weiterführenden Vorlage, öffnet. Listing 5.1 zeigt einen Ausschnitt aus der Vorlage, die für das Erstellen eines neuen Kostüms verantwortlich ist.

```
<includeonly>{|
|-
! Kurzbeschreibung
| [[Kurzbeschreibung::{{{Kurzbeschreibung|}}}]]
|-
! Ortsgegebenheit
| [[Ortsgegebenheit::{{{Ortsgegebenheit|}}}]]
|-
! Tageszeit
| {{#arraymap:{{{Tageszeit|}}}|,|x|[[Tageszeit::x]]}}
|-
! Timecode Anfang
| [[Timecode von::{{{Timecode Anfang|}}}]]
|-
! Timecode Ende
```

```
| [[Timecode bis::{{{Timecode Ende|}}}]]
|-
! Charaktereigenschaften / Assoziation
| {{#arraymap:{{{Assoziation|}}}|,|x|[[Assoziation::x]]}}
|-
! gehört zu
| [[gehört zu::{{{gehört zu|}}}]]
|}

== Neues Kostümteil anlegen ==
{{#formlink:
|form=Kostümteil anlegen
|link text=neues Kostümteil anlegen
|link type=button
|query string=Kostümteil anlegen[Gehört zu]={{FULLPAGENAME}}
|Kostümteil anlegen[Funktion]={{#ask: [[{{FULLPAGENAME}}]]
            | ?Funktion =
            | source=wiki
            | link=none|mainlabel=- }}
| Kostümteil anlegen[Zustand]={{#ask: [[{{FULLPAGENAME}}]]
            | ?Zustand=
            | source=wiki
            | link=none
            | mainlabel=- }}
}}

== Kostümteil bearbeiten ==
{{#forminput:form=Kostümteil anlegen|query string=Kostümteil anlegen[Gehört
    zu]={{FULLPAGENAME}} }}

{{#ask:
[[gehört zu::{{FULLPAGENAME}}]]
| intro=zugehörige Kostümteile
| ?Basiselement
| ?Material
| ?Farbe
| ?Design
| ?Relevanz
| format=fancytable
| style=table_zebra_grid
| source=wiki
| merge=false
|}}
</includeonly>
```

Listing 5.1: Vorlage: Neues Kostüm erstellen

Die mit `#arraymap:` gekennzeichneten Attribute können mehrere Werte annehmen, die anderen nur einen. Unter `#formlink:` wird ein Button erzeugt, bei dessen Klick eine neue Seite mit der `form:Kostümteil anlegen` erzeugt wird. Dieser neuen Seite werden bereits drei Parameter mitgegeben: Zum einen der Seitenname der eigenen Seite als Referenz vom Kostümteil zum Kostüm, zum anderen die Ergebnisse zweier Abfragen: Damit werden die Werte für Funktion und Zustand aus dem Kostüm an das Kostümteil weiterpropagiert. Nach diesem Schema müssen alle Verbindungen zwischen den Taxonomien vorgenommen werden, sofern die Ontologie das vorsieht. Diese Arbeit ist sehr fehleranfällig, und sollte nach Möglichkeit im Wiki komfortabel modelliert werden können.

6 Zusammenfassung und Ausblick

Dieses Kapitel bietet einen kurzen Durchgang durch die Arbeit und zeigt auf, wo zukünftig noch Verbesserungen am System vorgenommen werden können.

6.1 Zusammenfassung

Im Mittelpunkt dieser Arbeit standen Betrachtungen zur Modellierung einer geisteswissenschaftlichen Domäne. Ziel dieser Modellierung ist es, die Vorteile semantischer Annotation geschickt auszunutzen, um neue Erkenntnisse über die Domäne zu gewinnen.
In der Kostümdomäne wurden Strukturen entdeckt, die mit Patterns in der Informatik vergleichbar sind. Daraus entstand die Forderung nach einem Informations- und Katalogsystem für Kostüme und Kostümpatterns. Dieses wurde im Rahmen dieser Arbeit implementiert. Es wurden Lösungen erarbeitet, um das Domänenwissen in das Katalogsystem zu transferieren. Die nötigen Verbindungen zwischen den Elementen der Ontologie, die nicht importiert werden konnten, wurden mit Vorlagen und Formularen nachgebildet. Im Zuge einer möglichst einfachen Eingabe von Lösungen entstand eine Erweiterung für Semantic MediaWiki, die eine komfortable Auswahlmöglichkeit für hierarchisch strukturierte Daten anbietet.
Der Pattern-Identifikationsprozess wurde eingehend beschrieben und es wurde ein alternativer Prozess vorgestellt, der auf einer IT-gestützten Ähnlichkeitsanalyse basiert. Das generelle Vorgehen in diesem Prozess wurde positiv getestet, die Ergebnisse sind allerdings Aufgrund einer schlechten Heuristik und einer geringen Datenmenge nicht aussagekräftig. Reportingseiten, die auf Abfragen des Datenbestandes beruhen, wurden implementiert.

6.2 Ausblick

Die Kostümdomäne hat ein riesiges Datenpotential. In jedem Film tragen viele Schauspieler viele Kostüme. Und Filme gibt es auch nicht gerade wenige. Wenn die Anzahl der erfassten Kostüme steigt, werden gute Algorithmen zur Mustererkennung nötig. Die von mir vorgestellte Ähnlichkeitsanalyse kann mit einer guten Heuristik sicher einiges dazu beitragen.
Es wäre wünschenswert, dass DataWiki den Anschluss an die MediaWiki-Community nicht

ganz verliert. Im letzten Jahr hat sich Semantic MediaWiki stark weiterentwickelt. Mit einem Update von DataWiki auf die aktuelle MediaWiki Version würden sich sowohl für DataWiki-, als auch für MediaWiki-User tolle Synergieeffekte ergeben.

Literaturverzeichnis

[AH11] D. Allemang, J. Hendler. *Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2nd Auflage, 2011. (Zitiert auf den Seiten 16 und 47)

[AIS77] C. Alexander, S. Ishikawa, M. Silverstein. *A pattern language: towns, buildings, construction*. Oxford Univ. Press, New York, 1977. (Zitiert auf den Seiten 24 und 34)

[BK73] C. Bron, J. Kerbosch. Algorithm 457: finding all cliques of an undirected graph. *Commun. ACM*, 16(9):575–577, 1973. doi:10.1145/362342.362367. URL http://doi.acm.org/10.1145/362342.362367. (Zitiert auf Seite 52)

[Ba13] J. Barzen. Taxonomien kostümrelevanter Parameter: Annäherung an eine Ontologisierung der Domäne des Filmkostüms. Technischer Bericht 2013/04, Universität Stuttgart. (Zitiert auf den Seiten 5, 25, 31, 37, 39 und 40)

[BLHL01] T. Berners-Lee, J. Hendler, O. Lassila. The Semantic Web. *Scientific American*, 284(5):34–43, 2001. URL http://www.sciam.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21. (Zitiert auf den Seiten 9, 11, 42 und 46)

[Fü13] N. Fürst. Semantic Wiki for Design Pattern Capturing, 2013. (Zitiert auf den Seiten 68 und 70)

[FEL+12] C. Fehling, T. Ewald, F. Leymann, M. Pauly, J. Rütschlin, D. Schumm. Capturing Cloud Computing Knowledge and Experience in Patterns. In R. Chang, Herausgeber, *IEEE CLOUD*, S. 726–733. IEEE, 2012. URL http://dblp.uni-trier.de/db/conf/IEEEcloud/IEEEcloud2012.html#FehlingELPRS12. (Zitiert auf Seite 27)

[Fou13a] A. S. Foundation[HRG]. Apache Jena, 2013. URL http://jena.apache.org/. (Zitiert auf Seite 67)

[Fou13b] A. S. Foundation[HRG]. Lucene and SOLR, 2013. URL http://lucene.apache.org/solr/. (Zitiert auf den Seiten 63 und 65)

[GHJV96] E. Gamma, R. Helm, R. Johnson, J. Vlissides. *Entwurfsmuster: Elemente wiederverwendbarer objektorientierter Software*. Addison-Wesley, Bonn, 1. Aufl. Auflage, 1996. Design Patterns, 1995, Deutsche Übersetzung von Dirk Riehle. (Zitiert auf Seite 25)

[HW03] G. Hohpe, B. Woolf. *Enterprise integration patterns: designing, building, and deploying messaging solutions*. Addison-Wesley, Boston, Mass. [u.a.], 2003. (Zitiert auf Seite 34)

[jgr13] jgrapht.org[HRG]. jGraphT: A Free Java Graph Library, 2013. URL http://jgrapht.org/. (Zitiert auf den Seiten 52 und 72)

[Kar72] R. M. Karp. Reducibility Among Combinatorial Problems. In R. E. Miller, J. W. Thatcher, Herausgeber, *Complexity of Computer Computations*, S. 85–103. Plenum Press, 1972. (Zitiert auf Seite 51)

[Lea94] D. Lea. Christopher Alexander, An Introduction for Object-Oriented Designers. *Software Engineering Notes*, 1994. URL http://gee.cs.oswego.edu/dl/ca/. (Zitiert auf Seite 24)

[med13a] semantic mediawiki.org[HRG]. Extension: Semantic Result Formats, 2013. URL http://semantic-mediawiki.org/wiki/Semantic_Result_Formats. (Zitiert auf Seite 64)

[med13b] mediawiki.org[HRG]. Hooks in MediaWiki, 2013. URL http://www.mediawiki.org/wiki/Category:MediaWiki_hooks. (Zitiert auf Seite 60)

[med13c] mediawiki.org[HRG]. MediaWiki API Client Code, 2013. URL http://www.mediawiki.org/wiki/API:Client_code. (Zitiert auf Seite 43)

[ORA13] ORACLE[HRG]. About MySQL, 2013. URL http://www.mysql.de/about/. (Zitiert auf Seite 59)

[Rad11] S. N. Radeschütz. *Business Impact Analysis - Konzept und Realisierung einer ganzheitlichen Geschäftsanalyse*. Dissertation, Universität Stuttgart, Holzgartenstr. 16, 70174 Stuttgart, 2011. URL http://elib.uni-stuttgart.de/opus/volltexte/2012/7262. (Zitiert auf Seite 16)

[Rei12] R. Reiners. A Pattern Evolution Process - From Ideas to Patterns. In *Informatiktage*, Band S-11 von *LNI*, S. 115–118. GI, 2012. URL http://dblp.uni-trier.de/db/conf/informatiktage/informatiktage2012.html#Reiners12. (Zitiert auf Seite 28)

[SBLE12] D. Schumm, J. Barzen, F. Leymann, L. Ellrich. A Pattern Language for Costumes in Films. In *Proceedings of the 17th European Conference on Pattern Languages of Programs (EuroPLoP 2012)*. 2012. (Zitiert auf den Seiten 5, 25, 26, 34, 35, 36, 37 und 38)

[SS04] S. Staab, R. Studer, Herausgeber. *Handbook on Ontologies*. International Handbooks on Information Systems. Springer, 2004. (Zitiert auf Seite 16)

[Wik13] WikiMedia[HRG]. Wikipedia adopts MariaDB, 2013. URL http://blog.wikimedia.org/2013/04/22/wikipedia-adopts-mariadb/. (Zitiert auf Seite 59)

[Win09] J. Winkler. *JavaScript und Ajax: das Praxisbuch für Webentwicklung ; [JavaScript-Grundlagen beherrschen und anwenden; Ajax-Anwendungen verstehen und selbst programmieren; inklusive umfassender Objektreferenz zum Nachschlagen]*. Know-how ist blau. Franzis, 2009. URL http://books.google.de/books?id=IR3RPwAACAAJ. (Zitiert auf Seite 62)

Alle URLs wurden zuletzt am 04.07.2013 geprüft.

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommenen Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

(Daniel Andreas Kaupp)