Institut für Architektur von Anwendungssystemen
Universität Stuttgart
Universitätsstraße 38
70569 Stuttgart
Deutschland

Diplomarbeit Nr. 3591

Pattern-basierte Optimierung des Umwelteinflusses von Geschäftsprozessen

Lazar Davidkov

Studiengang:	Informatik
Prüfer:	Prof. Dr. Frank Leymann
Betreuer:	M.Sc. Wirt.-Inf. Alexander Nowak
begonnen am:	16.11.2013
beendet am:	16.05.2014
CR-Klassifikation:	H.3.3, H.4.2

Inhaltsverzeichnis

Abkürzungsverzeichnis

BP	– Business Process
BPEL	– Business Process Execution Language
DB	– Database
DSS	– Decision Support System
GBPP	– Green Business Process Pattern
GoF	– Gang of Four (Erich Gamma, Richard Helm, Ralph Johnson und John Vlissides)
KD-DSS	– Knowledge driven decision support system
MIS	– Management Information System (MIS)
NBM	– Naive Bayes Model
RS	– Recommender System
RDF	– Resource Description Framework
KEIDA	– KEI Dashboard
KEI	– Key Ecological Indicators

Abbildungsverzeichnis

1 Einleitung

Muster können in jeder Wissensdomäne gefunden werden. Sie haben ihren Vormarsch erfahren, nachdem Christopher Alexander sein Werk [Ac72] über Architektur-Muster vorgestellt hat. Alexander hat darin das Konzept der Mustersprache eingeführt und dieses beim Erstellen seiner Mustersprache auch in der Praxis umgesetzt. Die Idee, das Wissen in Form von Mustern darzustellen, wurde dann von vielen anderen Wissenschaftlern aufgegriffen. Somit finden sich Mustersprachen heute in nahezu jeder Wissensdomäne. Mustersprachen werden in Katalogen organisiert, die in den letzten Jahren mit dem Einführen der Webtechnologien in Wiki Systemen über Netzwerke (Intranet oder Internet) zugänglich gemacht worden sind. Diese Systeme erlauben es den Unternehmen beispielweise das Wissen, das dort generiert wird, zu verwalten und allen Mitarbeitern zur Verfügung zu stellen. Dadurch sind die Mitarbeiter des jeweiligen Unternehmens in der Lage, Lösungen zu schon bekannten Problemen zu finden. Das größte Problem dabei ist es, das passende Muster zu einer Aufgabe zu finden, weil die Information, die in den Wiki-Systemen gespeichert wird, unstrukturiert ist. Den Benutzern bleibt einzig die Möglichkeit durch aufwendige Recherchen an dem passenden Muster zu gelangen, da das Finden des richtigen Musters selbst für erfahrene Benutzer keine leichte Aufgabe ist.

Um das Finden einer Problemlösung in einer Mustersprache zu erleichtern, sind verschiedene Knowledge Based Decision Support Systems (KB-DSS) entstanden [HA06] [KZ07] [KP10] [SNK12]. Mit der Einführung des Web 2.0 und der raschen Entwicklung der Webtechnologien sind diese heutzutage als Webanwendungen aufgebaut [Mh98]. Sie implementieren Data Mining Methoden oder Inferenzalgorithmen und sind in der Lage, dem Benutzer beim Treffen von Entscheidungen zu helfen. In der Domäne der Mustersprachen können solche Systeme Benutzern beim Finden der richtigen Muster eine wesentliche Hilfe bieten. Die Konzepte solcher Systeme sind aber auf die Domäne der Mustersprache von Gang-of-Four (GoF) begrenzt. Die Muster von GoF sind eine Sammlung von Lösungen in der objektorientierten Programmierung. Alle anderen Wissensdomänen sind vernachlässigbar und die dort existierenden Mustersprachen sind nur mit großem Aufwand seitens der Benutzer anwendbar. Die entsprechenden Mustersprachen werden dann elektronisch in einem Wiki als Sammlung von Artikeln verwaltet und über einfache Links, die die Relationen darstellen, verbunden.

Diese Diplomarbeit zielt auf die Problematik, wie die Arbeit mit Mustersprachen vereinfacht und ein Einstieg in die Sprache gewährleistet werden kann. Es wird ein Konzept eines *Pattern Wiki* vorgeschlagen, das bei allen gängigen Mustersprachen anwendbar ist. Dieses Konzept sieht den Aufbau eines Decision Support Systems auf Basis der Struktur der Muster und ihrer Zusammenhänge innerhalb der Mustersprache vor. Neben der Verwaltung der Mustersprachen steht im Vordergrund, wie die Benutzer der Sprache, auf Basis bestimmter Merkmale ein zu ihrem Problem passendes Muster finden können. Zu diesem Zweck wird eine Komponente im Konzept vorgeschlagen, die die Suche ermöglicht. Schwerpunkte dabei sind die Kategorisierung der Muster innerhalb der Sprache, die Abgrenzung des Kontextes

der Problemstellung jedes Musters, sowie eine geeignete Beschreibung der Problemstellung. Da die Abgrenzung des Kontexts und die Beschreibung der Problemstellung ein sehr detailliertes Wissen über die Probleme und deren Lösungen erfordern, wird auch auf die Gemeinschaft, die hinter der Mustersprache steht, großer Wert gelegt. Die Gemeinschaft soll neben den Verfassern der Mustersprache in der Lage sein, zur Verbesserung des Systems und zur Erweiterung und Popularisierung der Mustersprache beizutragen. Einerseits passiert das direkt, indem der Gemeinschaft Werkzeuge zur Verfügung gestellt werden, die die Muster in der Sprache besser für die Zwecke des Entscheidungssystems beschreiben (durch Social Tagging) und andererseits geschieht dies indirekt, indem die Suchgeschichte jedes einzelnen Benutzers aufgefasst und ausgewertet wird und als Basis für Vorschläge bei zukünftigen Suchanfragen für andere Benutzer bereitgestellt wird. Ein weiterer wichtiger Aspekt im Bewahren des Wissens ist die Beschreibung der Lösungswege beim Lösen eines Problems mittels Muster in einer Mustersprache. Der Lösungsweg beschreibt also die Schritte, die eingesetzten Muster selbst und die Art und Weise, in der die Muster eingesetzt wurden.

Die vorliegende Diplomarbeit gliedert sich neben Einleitung und Zusammenfassung in sechs Hauptteile. Nach der Einleitung folgt in Kapitel 2 ein Überblick über die Begriffe, die eine große Rolle beim Aufbau des Pattern Wiki spielen. Dabei werden die Grundlagen der Muster und Mustersprachen, Entscheidungssysteme, Empfehlungssystem und das Social Tagging (kollektives Wissen) eingeführt und erläutert.

Kapitel 3 erklärt, wie die Analyse einer Mustersprache erfolgt. Der Kontext und das Problemfeld eines Musters werden identifiziert und mittels Metadaten beschrieben. Es wird das Konzept der Lösungswege und seine Bedeutung für die jeweilige Mustersprache vorgestellt.

Kapitel 4 führt das Konzept des Pattern Wiki ein. Die Bestandteile: Pattern Repository, Search Manager, Solution Manager und Recommender System werden erläutert. Die Grundgedanken und Überlegungen bei der Ermittlung des Einstiegspunktes werden beschrieben.

In Kapitel 5 wird eine Implementierung des Systems vorgestellt. Anhand von Ausschnitten aus dem Quellencode des Systems wird gezeigt, wie dieses aufgebaut ist und wie es funktioniert.

In Kapitel 6 wird dann anhand der Green Business Process Patterns [NL11] [NL13] [NL14] [NaL13] gezeigt, wie eine Mustersprache für das Einfügen im Pattern Wiki in der Praxis vorbereitet werden kann.

In Kapitel 7 wird mittels eines Anwendungsfalls gezeigt, wie das System zur Verbesserung des Umwelteinflusses eines Geschäftsprozesses mittels Green Business Process Patterns führen kann. Es wird veranschaulicht, wie der Einstiegspunkt ermittelt und wie ein Lösungsweg im Solution Repository gespeichert werden kann.

2 Grundlagen und verwandte Arbeiten

Ziel dieser Diplomarbeit ist die Verbesserung der Nutzbarkeit von Mustersprachen, d.h. es geht darum, die Relationen zwischen den Mustern abzubilden und auf Basis einer Mustersprache ein Entscheidungssystem (engl. Decision Support System (DSS)) aufzubauen. Das DSS wird dazu eingesetzt, das Ermitteln eines Einstiegspunktes in diese Sprache möglich zu machen. In diesem Kapitel werden die Grundlagen von Mustern und Mustersprachen, sowie die Grundlagen eines DSS erläutert. Kapitel 2.1 führt die Mustersprachen ein. Die Grundlagen der Entscheidungssysteme werden im Kapitel 2.2 eingeführt. Die Ergebnisse der Suche werden um zusätzliche Informationen erweitert, die aus der Suchgeschichte (eigene Suche und die der anderen Benutzer des Systems) hervorgehen und aufgrund früherer Entscheidungen ein (oder mehrere) zum Problem des Benutzers passende Muster vorgeschlagen. Es existieren bereits verschiedene Algorithmen und „Best Practices“, die sich mit dieser Problematik befassen. Diese werden in Kapitel 2.3 zusammengefasst. Social Tagging ist ein entscheidender Bestandteil des Suche-Konzeptes in einer Mustersprache und wird in Kapitel 2.4 eingeführt.

2.1 Muster und Mustersprachen

Das Konzept der Mustersprache (engl. Pattern Language) wurde zum ersten Mal von Christopher Alexander [Ac72] im Gebiet der Architektur eingeführt:

> “Die Elemente [einer Mustersprache] sind Muster. Es gibt eine Struktur über den Mustern, die beschreibt, wie jedes Muster seinerseits wieder ein Muster von anderen, kleineren Mustern ist. Außerdem gibt es Regeln in diesem Muster, die beschreiben, wie die Muster konstruiert und im Kontext mit anderen Mustern angeordnet werden können.
>
> In diesem Sinne sind Muster sowohl Elemente als auch Regeln [der Mustersprache], Regeln und Elemente sind also nicht voneinander zu unterscheiden. Die Muster sind Elemente. Und jedes Muster ist genauso auch eine Regel, die eine mögliche Anordnung der Elemente beschreibt - ihrer selbst oder anderer Muster.”

Die Definition besagt, dass zwischen den Mustern Regeln existieren und diese Regeln die Muster zusammen halten. Die Muster haben nur innerhalb der Sprache eine Bedeutung, in der sie definiert sind. Außerhalb der Sprache können diese Muster nicht existieren, weil sie immer eine Teillösung eines Problems sind. Sie hängen von anderen Mustern ab und nur zusammen sind sie in der Lage, ein Problem in der Domäne, für die die Sprache entwickelt wurde, zu lösen. Alexander spricht von Sprache, weil die Mustersprache, wie die Sprachen, die die Menschen sprechen, aus Wörtern und Regeln bestehen. Die Elemente der Mustersprachen sind aber komplexer und dienen nicht nur als Wörter, sondern auch als Regeln (siehe Definition). Manche Autoren [Bf98] sind der Meinung, dass man nicht über eine Sprache sondern über ein System (Mustersystem) sprechen sollte. Ihre Argumentation beruht auf der Unvollständigkeit der Mustersprachen. Eine solche Sprache soll in der Lage

sein, alle Probleme zu lösen, die in einer Domäne vorkommen. Sie deckt aber nur einen kleinen Teil dieser Aufgaben ab. In dieser Arbeit werden die Begriffe Mustersprache und Mustersystem gleich gesetzt. In den folgenden Kapiteln werden nur Mustersprachen beschrieben, wobei sie nicht den Anspruch haben, alle vorkommenden Probleme in der Domäne der Sprache lösen zu können.

Die Mustersprachen sind Sammlungen (Katalog im Fall von GoF [GR11]) von Mustern und die Relationen zwischen den Mustern sind mit semantischer Information versehen. Wenn ein Muster auf ein Problem angewendet wird, ändert sich laut Alexander sofort der Kontext des Problems und dieser neue Kontext erlaubt die Anwendung eines weiteren Musters. Dadurch entstehen Sequenzen von Mustern, die den Lösungsweg beschreiben. Somit ist es möglich und das ist auch der Grundgedanke von Alexander, ein großes Problem in kleinere Probleme aufzuteilen und auf jedes Teilproblem ein Muster anzuwenden. Diese Vernetzung ermöglicht beliebige Kombinationen und dadurch entsteht die Sprache. Die Muster kann man als die Wörter in einer gesprochenen Sprache betrachten und die Relationen zwischen diesen Mustern, wie die Grammatik, die die Sätze aufbaut. Jeden Lösungsweg kann man als einen Satz betrachten. Das beliebige Kombinieren der Muster nach den Regeln, die sie enthalten, führt dazu, dass immer neue Sätze (Lösungswege) entstehen und zur Vielfältigkeit der Sprache beitragen [Ac72].

Die Mustersprachen haben sich im Laufe der Zeit als eine wichtige Hilfe bewiesen. Dafür sprechen die Anzahl der Sprachen, die heutzutage existieren, und auch die Vielfalt der Domänen, für die eine oder mehrere Mustersprachen entwickelt wurden. Die zwei Mustersprachen (die Mustersprache von Alexander und diese von GoF) sind für die Domäne der Architektur und der Softwareentwicklung und insbesondere für die objektorientierte Programmierung (OOP) konzipiert und entwickelt. Auf Basis der Mustersprache von GoF sind weitere Sprachen entstanden, die spezifische Bereiche des Software Engineering abdecken. So gibt es z.B. eine Mustersprache für den Entwurf von verteilten Systemen [Mf03], eine Mustersprache für die Planung und Ausführung von Geschäftsprozessen in serviceorientierten Architekturen [HZ09], eine Mustersprache für die AJAX Programmierung [MM06], eine Mustersprache, die sich mit der Human Computer Interaction befasst [Bj01], eine Mustersprache für die Erstellung des Front Ends von Web Applikationen [Gi03] usw. Es existieren aber auch Einsätze in weiteren Gebieten, wie z.B. der Filmindustrie. In dieser Kategorie fällt die Mustersprache zur Klassifizierung von Kostümen [BLD12]. Muster werden auch im Verhalten von Kindern erkannt und diese werden in der Pädagogik eingesetzt [Bp11].

Trotz der Anfangsschwierigkeiten beim Erkennen des relevanten Musters (Einstiegspunkt in der Sprache) stellen die Sprachen ein wichtiges Konzept beim Bewahren von Wissen dar. Die Herausgabe der Sprachen an das breite Publikum macht dieses Wissen allen zugänglich und erfüllt sie, laut Alexander, somit mit Leben. Das Wissen über die Lösungsmöglichkeiten in einer Domäne wird dann jedem, der im gegebenen Bereich tätig ist, zur Verfügung gestellt. Somit kann sich die Mustersprache als „Best Practice“ beweisen und weiter entwickelt werden. Das gilt insbesondere in der Sphäre des Software Engineering. Die dynamische

Entwicklung in diesem Bereich ist die „Best Practice“ von heute die unerwünschte Lösung von morgen. Deswegen sollen die Mustersprachen dem breiten Publikum vorgestellt werden. Die Anwender der Sprache können dann die Gültigkeit der Muster ständig prüfen und diese wenn nötig ändern und aktualisieren.

2.2 Entscheidungssysteme

Im Kapitel 2.1 wurde gezeigt, dass es wichtig ist, Wissen in Form einer Mustersprache zu bewahren, um es beim Entstehen eines schon bekannten Problems wieder nutzen zu können. Dabei wurde auch klar, dass das Finden geeigneter Muster in einer Mustersprache keine leichte Aufgabe ist. Für den unerfahrenen Benutzer kann diese Suche zu einem langwierigen Prozess werden und eine falsche Entscheidung kann dazu führen, ein nicht passendes Muster auszuwählen [BHS07]. Ein rechnergestütztes System könnte dem Benutzer in diesem Fall diese Aufgabe abnehmen und bei der Entscheidung helfen. Entscheidungssysteme existieren schon seit Anfang der 80-er Jahre des letzten Jahrhunderts und werden in vielen großen Unternehmen eingesetzt [MS03]. Früher waren sie unter dem Namen Management Information System (MIS) bekannt. Heutzutage spricht man von Decision Support System (DSS) [MS03]. Ein solches System hat die Aufgabe, dem Management eines Unternehmens beim Treffen von Entscheidungen zu helfen.

Das Treffen von Entscheidungen ist eine der kognitiven Fähigkeiten der Menschen. Sie ist ein bewusster Prozess, der die Wahl zwischen mehreren Alternativen (Möglichkeiten) darstellt und auf gedanklichen Überlegungen basiert [Hz14]. Die Entscheidung kann strukturiert, semi-strukturiert oder unstrukturiert sein [SS05]. Entscheidend ist der Typ der vorliegenden Information, auf deren Basis eine Entscheidung vorbereitet und getroffen werden soll. Ein DSS soll die semi-strukturierten und unstrukturierten Prozesse unterstützen [MM05]. Dabei wird die benötigte Information bearbeitet und dem Benutzer in einer geeigneten Form zur Verfügung gestellt. Obwohl das System einen Großteil der Arbeit erledigt, ist das Treffen einer Entscheidung allein dem Manager oder dem Benutzer des Systems überlassen.

Eine Definition eines DSS findet man in [Vc09]: “an interactive computer system helping decision makers to combine data and models to solve semi-structured and unstructured problems”. Ein solches System besteht aus drei Komponenten (Abb. 2.1):

- *Database management system (Data)* Die Datenkomponente besteht aus einer Datenbank (DB), die die relevanten Informationen enthält. Die Daten, die in diese DB einfließen, stammen aus externen oder internen Quellen des Unternehmens.
- *Model-base management system (Model)* Die Modellkomponente enthält die Regeln, die bei der Auswertung der Informationen in der Datenkomponente benötigt werden. Dabei werden verschiedene mathematische Modelle aus der Statistik und Wirtschaftslehre sowie verschiedene Methoden aus der Informationswissenschaft, Psychologie und Künstlichen Intelligenz (KI) eingesetzt [DF02].
- *Dialog generation and management system (Graphical user interface (GUI))*. (Abb. 1). Die Arbeit mit dem System erfolgt über ein *Graphical User Interface* (GUI). Das GUI stellt einen Dialogmanager dar, über den die Benutzer Anfragen an das System

stellen können.

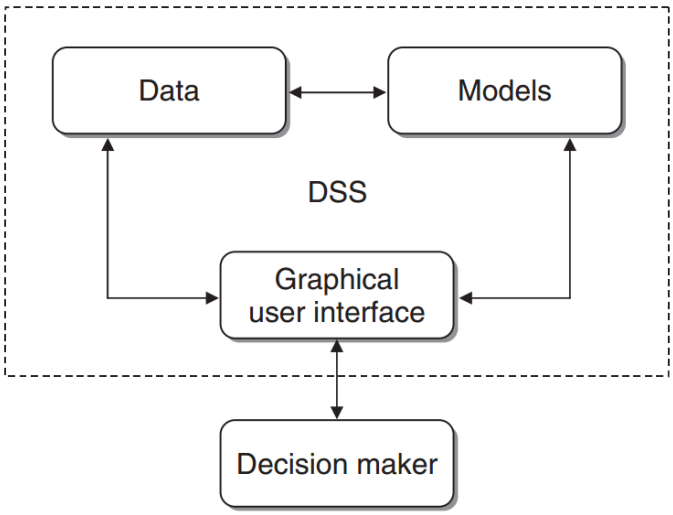

Abbildung 2.1 Komponenten eines DSS [Vc09]

Die Eigenschaften, die ein solches System aufweisen soll, sind wie folgt [LMM04]:

- ***einfach*** - leicht zu verstehen
- ***robust*** - das Modell soll immer die richtige Richtung aufweisen
- ***einfach zu kontrollieren*** - der Benutzer soll in der Lage sein, das Modell leicht an seine Wünsche anzupassen
- ***leicht anzupassen*** - das Modell soll in der Lage sein, sich beim Eintreffen neuer Informationen zu aktualisieren
- ***vollständig*** - mehrere Dimensionen des untersuchten Problems sollen aufgefasst werden
- ***einfach zu bedienen*** - die Eingaben sollen leicht vom Benutzer geändert werden können

Im Laufe der Zeit sind DSS für mehrere Typen entstanden, aber der Kern jedes solchen DSS bleiben die drei obengenannten Komponenten: data, model und dialog. Neben diesen drei Komponenten werden zusätzliche Komponenten gebraucht, die die spezifischen Anwendungsaufgaben widerspiegeln. Eine Klassifikation der DSS findet man in Abb. 2.2 [Pd02]. Das DSS, das für die Zwecke dieser Diplomarbeit erstellt wurde, kann der Gruppe der wissensbasierten DSS zugeordnet werden.

Die wissensbasierten DSS (KD-DSS) bauen auf dem Wissen von Experten in einer gegebenen Domäne auf. Experten sind Personen, die über Fachkenntnisse im gegebenen Bereich verfügen. Die Fachkenntnisse stellen fachspezifisches Wissen dar, das diese Personen in ihrer täglichen Arbeit und über Ausbildungen, Seminare usw. erworben haben [TMW02]. Mit diesem Wissen sind sie in der Lage, komplexe Probleme zu lösen, weil sie bessere und schnellere Entscheidungen als Nicht-Experten treffen können. Wird dieses Wissen in ein elektronisches Format überführt und interessierten Nicht-Experten bereitgestellt, spricht man von einem Experten System (ES). Dieses ES bildet den wichtigsten

Teil des KD-DSS. Ein solches System agiert wie ein Experte, indem es seinen Benutzern beim Entscheidungsprozess mit komplexen Schlussfolgerungen hilft [TMW02]. Beim Aufbau solcher Systeme werden hauptsächlich Web Technologien eingesetzt. Sie sind als Client/Server konzipiert. Ein solches System wurde in den USA im Department of Labor's Small Business Administration aufgebaut. Das System ist über einen Webbrowser zugänglich und hat zum Ziel, den kleinen und mittleren Unternehmen in den USA in komplexen Fragen der rechtlichen Bestimmungen, eine schnelle Hilfe zu bieten [Hd06]. Den Interessenten bleiben aufwendige Recherchen im Internet erspart. Stattdessen finden diese leicht und unkompliziert die relevanten Informationen mit Hilfe eines benutzerfreundlichen Webinterface. Alles, was sie machen müssen, ist es eine Art Gespräch mit dem System zu führen und Fragen zu beantworten. Das System und die Entscheidungsmechanismen, die dahinter stecken, leiten die Konversation in die eine oder andere Richtung. Dadurch wird auf Basis der eingegebenen Information eine spezifische Frage beantwortet. Das System basiert auf Exsys CORVID [COR] Knowledge Automation Expert Systems und liefert den Benutzern Wissen an Stelle von Information. Die gesetzlichen Bestimmungen lassen sich mit "If… Then…" Konstrukten leicht in eine Reihe von Fragen/Antworten überführen und ins System einbinden. CORVID verfügt zusätzlich über einen Inferenzmechanismus, der bestimmte Inferenzregeln verfolgt und aus dem Wissen Schlussfolgerungen ableitet, die zum Lösen des Problems beitragen sollen. Das so beschriebene System stellt dem breiten Publikum ein Werkzeug zur Verfügung, das immer die aktuellsten Informationen als Vorlage hat und den Benutzern Basiswissen im Bereich der gesetzlichen Bestimmungen vermitteln kann.

Dominant DSS Component	**User Groups: Internal, External**	**Purpose: General, Specific**	**Enabling Technology**
Communications **Communications-Driven DSS**	Internal teams, now expanding	Conduct a meeting Bulletin board Help users collaborate	Web or Client/Server
Database **Data-driven DSS**	managers, staff, now suppliers	Query a Data Warehouse	Main Frame, Client /Server Web
Document base **Document-driven DSS**	Specialists and user group is expanding	Search Web pages find documents	Web
Knowledge base **Knowledge-driven DSS**	Internal users, now customers	Management advice Choose Products	Client/Server, Web
Models **Model-driven DSS**	Managers and staff, now customers	Crew scheduling Decision analysis	Stand-alone PC

Abbildung 2.2 Erweitertes DSS Framework [Pd02]

Das Beispiel zeigt, wie wichtig es ist, die unstrukturierten Informationen in einer geeigneten

Form bereitzustellen, damit aus diesen Informationen maschinell logische Schlussfolgerungen gezogen werden. In diesem Fall spricht man nicht mehr nur über Informationen, sondern auch über den Sinn dieser Informationen und über das Wissen, das in der Information steckt.

Die Natur der Mustersprachen macht sie zu einem interessanten Gebiet, auf dem KB-DSS eingesetzt werden können. Dafür spricht einerseits die große Anzahl der Muster, die zum Teil komplizierte und in einigen Fällen unendliche Kombinationsmöglichkeiten aufweisen, was das Suchen nach dem Einstiegspunkt erheblich erschwert. Andererseits sind die Muster für sich selbst ein komplexes Wissen, das sich nicht einfach extrahieren lässt. Es existieren schon mehrere Konzepte, wie ein solches System aufgebaut werden kann. Dabei ist es wichtig anzumerken, dass alle diese Konzepte sich nur auf Software Design Patterns beschränken. Einige setzen direkt auf einer Ontologie [HA06], [KZ07] und versuchen, die Muster formal mit RDF[1] zu beschreiben, um eine automatisierte Schlussfolgerung zu ermöglichen. Ein anderer Ansatz [KP10] klassifiziert die Muster nach ihrer Anwendung. Dabei werden wichtige Qualitätsmerkmale hervorgehoben und mit dem Anwendungstyp in Relation gesetzt. Dadurch wird die Verbindung Muster-Anwendungstyp sichtbar und ein Muster kann einem Problemfeld zugeordnet werden. Eine dritte Variante [SNK12] setzt direkt auf einer indexierten Suche im Text der Muster sowie auf einem typischen DSS-Dialog-System mit verschieden Fragen auf, die zum Herausfinden des Problemkontexts gestellt werden. Dadurch wird für jedes Muster ein Rang berechnet wobei das Muster mit dem höchsten Rang das geeignetste sein soll. Als Repository (Database) in dieser Variante dienen XML-Dateien, die mit Hilfe von *Apache Lucene Search Engine* [LA] indexiert werden. Die Suche besteht aus drei Schritten. Im ersten Schritt gibt der Benutzer einen freien Text ein und auf dessen Basis findet eine Vorauswahl der Muster statt. Im zweiten und dritten Schritt werden die Fragen nach dem Kontext und nach dem Problemfeld des Musters gestellt. Jeder dieser drei Schritte hat dann ein bestimmtes Gewicht im berechneten Rang. Dieser Ansatz wird auch in dieser Diplomarbeit verfolgt, aber in einigen Punkten verbessert, erweitert und benutzerfreundlicher gemacht. Kapitel 4 gibt einen ausführlichen Überblick über das vorgeschlagene Konzept und Kapitel 5 - über eine mögliche Implementierung. Das neue System setzt zusätzlich auf Social Tagging, um den Kontext der Problemstellung des Musters zu beschreiben, sowie ein Recommender System, um die Suche zu erweitern. Das Ziel ist es an erster Stelle ein System zu konzipieren, das unabhängig von einer Mustersprache ist und zum Zweiten, die Suche nach dem passenden Muster (Einstiegspunkt in die Sprache) genauer zu machen.

DSS finden Einsatz in vielen Bereichen, in denen Expertenwissen gefragt ist. Beim Aufbau eines solchen Systems spielen nicht nur die Kosten für die Experten, deren Dienstleistungen sehr teuer sind, eine Rolle, sondern auch das Wissen, das im Unternehmen generiert und bewahrt werden muss. Beispiele [Pd06], [Hd06] von DSS zeigen, dass diese Systeme ein wichtiger Teil jedes Unternehmens geworden sind und ihre Aufgaben in vielen Bereichen erfolgreich erledigen.

[1] Resource Description Framework

2.3 Empfehlungssysteme (Recommender System)

In den letzten Jahren ist mit dem Boom der neuen Technologien, insbesondere im Bereich des Mobile Computing und mit dem Vernetzen aller möglichen Geräte in jeder Sphäre des Lebens, die Zeit vom sogenannten Big Data gekommen. Daten in großen Mengen werden nicht nur von Menschen, sondern auch z.B. von Sensoren auf Fließbändern in großen Fabriken generiert. Die Herausforderungen, vor denen Unternehmen im Moment stehen, sind die Auswertung dieser Daten, um dazu beizutragen, sich Wettbewerbsvorteile (Optimierung von Prozessen, Erstellung von Nutzerprofilen für das Einblenden gezielter Werbung usw.) gegenüber der Konkurrenz zu verschaffen. BITKOM[2] definiert Big Data als [UW12]:

> "Big Data stellt Konzepte, Technologien und Methoden zur Verfügung, um die geradezu exponentiell steigenden Volumina vielfältiger Informationen noch besser als fundierte und zeitnahe Entscheidungsgrundlage verwenden zu können und so Innovations- und Wettbewerbsfähigkeit von Unternehmen weiter zu steigern."

Big Data kann als eine Grundlage benutzt werden, Entscheidungen zu treffen. Die Daten, die hier gemeint sind, bestehen bis zu 95% aus unstrukturierten Daten [MMD12]. Sie stellen die Basis des Datenbestands jedes DSS [SB11] dar. Von großem Interesse für die Unternehmen sind die Daten, die Menschen im Internet generieren [BLR14]. Die Webseiten der größten Onlinehändler sind so ausgelegt, dass dort jeder Klick erfasst und gespeichert wird. Jede Meinung zu einem Produkt oder Warenkorbinhalt und der nachfolgende Kauf werden gespeichert und ausgewertet. Dadurch entstehen sehr genaue Profile der Benutzer und somit kann das Kaufverhalten beim nächsten Besuch genau vorhergesagt werden. Bei der Auswertung werden verschiedene Konzepte aus der Statistik oder künstlichen Intelligenz genutzt [Fw12]. Dadurch entstehen Recommender Systems (RS). Sie sind in der Lage dem Benutzer mit den Inferenzmechanismen, über die sie auf Basis des vergangenen Verhaltens des Benutzers verfügen, Vorschläge in einer gegebenen Domäne zu unterbreiten. Solche Systeme finden insbesondere im Online Marketing eine breite Anwendung [FWW06]. Onlinehändler wie Amazon[3] und Zalando[4] oder die Suchmaschine (und auch alle anderen Services) von Google machen einen extensiven Gebrauch solcher Systeme. Webseiten wie Yahoo! versuchen, gezielt die Artikel mit der größten Klickrate[5] besser zu positionieren. Eine wichtige Informationsquelle der RS sind die Ratings, die von den Benutzern vergeben werden. Das geschieht beim Einkauf eines Produkts bei Amazon oder beim Anschauen eines Videos auf YouTube[6]. Diese Ratings werden als Referenz von den RS benutzt und bei den Berechnungen berücksichtigt.

Empfehlungssysteme sind ein Teil des maschinellen Lernens. Bei maschinellem Lernen wird

[2] Bundesverband Informationswirtschaft, Telekommunikation und neue Medien e.V.
[3] http://www.amazon.com
[4] http://www.zalando.de
[5] "Die Klickrate ist eine der wichtigsten Messgrößen, wenn eine Werbekampagne hinsichtlich ihres Erfolgs analysiert werden soll." : http://goo.gl/s0VfyN
[6] http://www.youtube.com

eine “offline” und eine “online” Modellierung der vorliegenden Daten unterschieden [BKM97]. Offline Modellierung beschäftigt sich mit der Auswertung historischer Daten. “Online” Modellierung dagegen mit “the science of designing explore/exploit schemes that serve items to users in optimal fashion” [AC11]. In den RS werden drei Typen von Objekten unterschieden [Rf11]: Gegenstände, Benutzer und Transaktionen. Gegenstände sind die Objekte, die vom System vorgeschlagen werden. Objekte werden anhand ihres Nutzens für den Benutzer bewertet. Falls der Nutzen groß ist und das Objekt als richtig eingestuft wird, wird ein positiver Wert zugewiesen, ansonsten ein negativer. Objekte in einem Online-Shop sind z.B. die Gegenstände, die dort verkauft werden. In einer Online Zeitung oder einem Portal wie Yahoo sind das die einzelnen Artikel. Die Benutzer der RS sind die Personen, die nach etwas suchen. Damit die RS auf den Benutzer ausgerichtete Gegenstände empfehlen können, werden verschiedene Informationen gebraucht. Wie oben beschrieben kommt ein Teil davon von Big Data und der damit verbundenen Such- oder Einkaufsgeschichte. Ein weiterer Teil der Information wird vom Benutzer selbst eingegeben, z.B. beim Registrieren im System. So entstehen die Benutzerprofile, die in den einzelnen Empfehlungsmethoden eingesetzt werden. Der dritte Typ Objekte, die im Kontext der RS unterschieden werden, sind die Transaktionen. Transaktionen stellen das Verhalten der Benutzer dar. Sie dienen dazu, die Benutzer in Relation mit einem Gegenstand zu bringen. Transaktionen sind die Schritte, die ein Benutzer in einem Kontext vornimmt, um zu einem Gegenstand zu kommen. Sie werden sehr detailliert gespeichert und sind ein wichtiger Teil jedes RS. Sobald die Information über die drei Objekttypen im RS zur Verfügung steht, kann ein bestimmter Algorithmus angewendet werden, um die Vorschläge zu berechnen.

Die RS werden auf Basis der adressierten Domäne, des benutzten Wissens und des verwendeten Algorithmus in sechs Kategorien aufgeteilt [Rf11]:

- **Content-based**. Diese Systeme setzen auf Informationen, die der Benutzer in der Vergangenheit angegeben hat. Sie werden bei zukünftigen Besuchen als Referenz benutzt, um spezifische Gegenstände vorzuschlagen.
- **Collaborative filtering**. Die Benutzer werden nach ihren Vorzügen aufgeteilt. Das System nutzt die Suchgeschichte der anderen Benutzer in der Gruppe, um dem aktuellen Benutzer Vorschläge zu machen.
- **Demographic.** Die Benutzer werden, wie bei *Collaborative filtering,* in Gruppen aufgeteilt, wobei die Gruppierung nach demographischen Merkmalen, wie Alter, Geschlecht, Staatsangehörigkeit usw. erfolgt.
- **Knowledge-based**. *Knowledge-based* Systeme sind speziell auf eine Wissensdomäne zugeschnitten. Die Vorschläge, die diese Systeme machen, sind eine mögliche Lösung des von dem Benutzer beschriebenen Problems.
- **Community-based**. Dieses System bewertet die Präferenzen des Freundeskreises eines Benutzers innerhalb des Systems (andere Benutzer, die als Freunde des gegebenen Benutzers markiert sind). Das können Bekannte, Kollegen oder Freunde sein.
- **Hybrid recommender systems.** Sie sind Kombinationen der vorgestellten Typen von RS.

Damit die RS zuverlässige Vorschläge machen können, werden viele Daten gebraucht. In der Regel gilt, dass je länger das System benutzt wird und je mehr Daten erfasst worden sind, desto besser und genauer entsprechen die gemachten Vorschläge den Präferenzen der Benutzer. Bei den Berechnungen werden viele Methoden aus dem Data Mining eingesetzt [Rf11]. Data Mining erfolgt in drei Schritten (siehe Abb. 2). Im ersten Schritt findet eine Vorbereitung der Daten (Data Preprocessing) statt. Im zweiten Schritt werden verschiedene Klassifizierungsalgorithmen eingesetzt und die Daten in verschiedene Kategorien aufgeteilt (Klassifikation). Im dritten Schritt werden Algorithmen eingesetzt, die die klassifizierten Daten analysieren (Cluster Analysis) [Rf11].

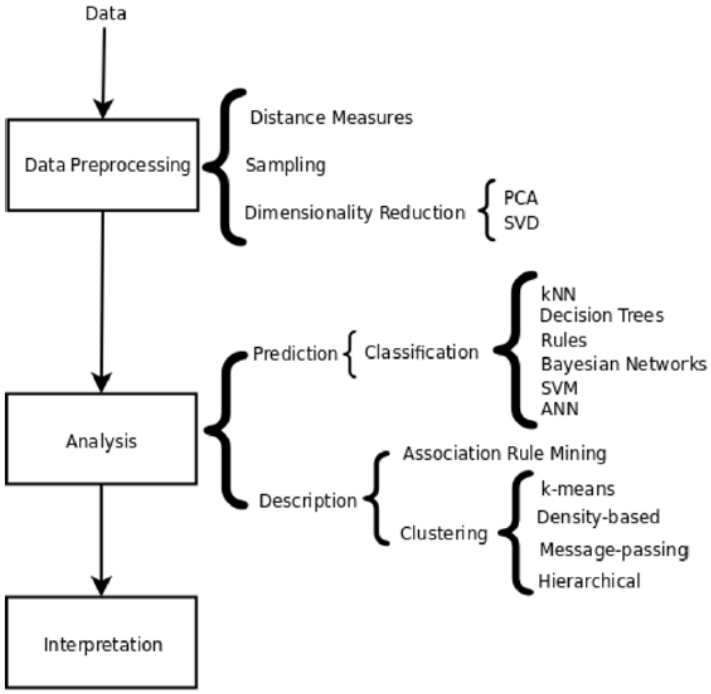

Abbildung 2. Schritte und Methoden beim Lösen eines Data Mining Problems [Rf11]

Für die Zwecke dieser Diplomarbeit wurde die Bayesian Klassifizierung eingesetzt. Diese wird ausführlich beschrieben. Alle anderen Algorithmen, die im jeweiligen Schritt des Data Mining Prozesses gebraucht werden, können im „Recommender System Handbook“ von Amatrian et al. gefunden werden.

Im System, das für diese Diplomarbeit konzipiert und entwickelt wurde, ist der Data Mining Prozess auf die Analyse der Daten beschränkt. Der Schritt “Data Preprocessing” war auf Grund der Natur der aufgefassten Daten nicht notwendig. Die Daten sind sehr strukturiert und benötigen keine Vorarbeit.

Das Naive Bayes Modell (NBM) findet eine breite Verwendung in der Textklassifikation. Dadurch wird versucht, Text-Dokumente einer oder mehrerer inhaltlich gleicher Kategorien,

auch Klassen genannt, zuzuordnen. Die Kategorien werden strikt voneinander getrennt gehalten oder hierarchisch geordnet. Damit die Kategorisierung einen Sinn ergibt, werden mindestens zwei Kategorien gebraucht. Sind nur zwei Kategorien vorhanden, entsteht die sogenannte binäre Klassifizierung. Die Anzahl der Klassen ist nach oben nicht begrenzt und kann unendlich hoch sein. Typische Beispiele für eine Textklassifikation sind die Spam-Filter [SDH98]. Sie erfolgen in der Regel mit einer Binär-Klassifizierung. Die E-Mail Nachrichten werden als Spam oder als Nicht-Spam gekennzeichnet. Verschiedene Tests zeigen, dass trotz seiner Einfachheit, das NBM ein zuverlässiger Text-Klassifikator sein kann. Mit einem solchen Filter ist eine Genauigkeit der Treffer, mit bis zu 99.691% zu erzielen [Zh10]. Spam-Filter wie SpamAssassin [7] oder solche, die im Mozilla Email Client Thunderbird oder Microsoft Outlook [CS] verwendet werden, implementieren ebenfalls NBM.

NBM setzt auf einem probabilistischen Modell aus den Kategorien und Wörtern (Terme) auf, die darin vorkommen. Beim Eintreffen eines unbekannten Dokuments *d* (z.B. eine E-Mail Nachricht) wird auf Basis der Terme (t_1….t_n), die *d* enthalten, die Wahrscheinlichkeit berechnet, mit der es einer Kategorie (c) angehört. Die bedingte Wahrscheinlichkeit lässt sich wie folgt berechnen [MRS08]:

$$P(c|d) = P(c)\Pi P(t_k|c),$$

wobei $P(t_k|c)$ die Wahrscheinlichkeit ist, dass Term t_k in c enthalten ist. $P(t_k|c)$ beschreibt die Gewichte der einzelnen Terme t_k und ihre Zugehörigkeit zu der Kategorie c. Da nur eine Kategorie meist sinnlos ist und immer mindestens zwei oder mehr erwartet werden, wird das Maximum von $P(c_j|d)$ mit $j > 1$, oder die *maximum a posteriori* (*MAP*) Klasse c_{map} gesucht. Die Klassifikationsformel lässt sich folgendermaßen umschreiben:

$$c_{map} = \arg\max P(c|d) = \arg\max P(c)\Pi P(t_k|c)$$

Das Dokument kann mehreren Kategorien zugeordnet werden, für das Endergebnis ist jedoch die wahrscheinlichste von Bedeutung. Dieses Modell wird als “naiv” bezeichnet, da die Annahme getroffen wird, dass alle Attribute des zu klassifizierenden Dokuments nur abhängig von der Kategorie und unabhängig unter einander sind. Die Klassifizierung einer Frucht wie z.B. Apfel könnte mit folgenden Attributen beschreiben werden: rund, rote Farbe, wächst auf einem Baum. Obwohl alle drei Attribute an einander hängen, um einen Apfel als solchen zu klassifizieren, werden diese im NBM als komplett getrennte Eigenschaften betrachtet. Dadurch wird das Model sehr einfach und wie oben gezeigt sehr effektiv.

Die Vorteile, die es so verbreitet in der Praxis machen, sind [Wp09]:

- ***Effektivität***: bis zu 99.691% Richtigkeit bei der Klassifikation
- ***Einfachheit***: einfach zu verstehen, zu implementieren und zu trainieren

[7] http://spamassassin.apache.org/tests.html

- ***Genauigkeit***: arbeitet mit einer sehr großen Genauigkeit, falls die Trainingsdaten entsprechend umfassend sind
- ***Schnelligkeit***: die Zeit, die gebraucht wird, um den Algorithmus zu trainieren, ist linear zu den Daten und der Speicherplatz ist linear zur Anzahl der Attribute im Modell. Dadurch wird das Benutzen von NBM sehr zeit- und platzeffektiv.

Obwohl NBM viele positive Eigenschaften hat, muss man auch die negativen erwähnen. Dazu gehören die Trainingsdaten. Es dauert eine gewisse Zeit, bis in der Datenbank genug Informationen vorhanden sind, um richtige Entscheidungen zu produzieren. Zu den negativen Seiten gehört auch die falsche Zuordnung von Dokumenten. Solche Fälle treten selten auf, sind aber durchaus möglich. Dies hätte negative Folgen im Fall eines Spam-Filters.

Trotz der negativen Seiten ist NBM ein Modell, das sich in der Praxis bewiesen hat. Neben der breiten Anwendung als Textklassifikator und in den Spam-Filtern wird NBM in den Hybriden sowie Collaborative Filtering Recommender Systemen benutzt. Diese Systeme versuchen, auf Basis der entdeckten Abhängigkeiten in den vorhandenen Informationen einem Benutzer relevante Vorschläge zu geben. Diese Strategie wird in der Implementierung des Systems verfolgt, das als Grundlage für diese Diplomarbeit dient. Die genaue Implementierung des Bayes Modells im Recommender Systems der beilegenden Software wird in Kapitel 5.3 ausführlich beschrieben.

2.4 Social Tagging

Der letzte Aspekt der theoretischen Grundlagen, die bei dem Konzept von der zu erstellenden Software im Rahmen dieser Diplomarbeit gebraucht wird, ist das Social Tagging (ST). Social Tagging hat sich in den letzten Jahren mit dem Boom der sozialen Netzwerke schnell verbreitet und sich als eine einfache Möglichkeit zur Beschreibung von unstrukturierten Daten bewiesen. Webseiten wie YouTube, Flickr und delicious.com verlassen sich hauptsächlich auf dem Benutzer, um Videodateien, Bilder oder im Fall von delicious.de Links mit Tags zu beschreiben. Videodateien lassen sich im Gegensatz zu einer Webseite nicht automatisch mit Metadaten annotieren und von den Suchmaschinen indexieren. Die Information über den Inhalt des Dokuments (Videodatei, Bild) soll auf einem anderen Weg erfolgen. Im Fall einer HTML-Seite ist diese Aufgabe dagegen einfacher. Der Text der Seite wird mit Hilfe von Data Mining Methoden, z.B. mit NBM, einer Kategorie automatisiert zugeordnet. Für Videodateien oder Bilder werden dagegen andere Algorithmen aus der Bildbearbeitung gebraucht. Es existieren schon solche Algorithmen, die z.B. Objekte in einer Videodatei erkennen können, diese sind aber sehr rechenaufwendig [OMS14]. Es ist auch zu erwähnen, dass jede Minute 100 Stunden Videomaterial auf YouTube hochgeladen werden [YT], was den Einsatz solcher Algorithmen nicht profitabel macht. Deswegen können sie sich in der Praxis im Moment nicht durchsetzen. In diesen Fällen wird auf den Faktor Mensch gesetzt. Die Benutzer der großen Web-Portale sind in der Lage, durch Schlagwörter (Tags) die Dateien sehr genau zu kennzeichnen. Die Menschen sind im Unterschied zu den Maschinen in der Lage, besser abzuschätzen, in welcher Kategorie ein Bild oder ein Video

zugeordnet werden soll. Dadurch werden die Dateien indexiert und die Suchmaschinen sind in der Lage, diese Dateien leichter zu finden. Die Idee, die hinter dem Social Tagging steht, besteht darin, das Wissen der Gemeinschaft zu benutzen, um ein Dokument (Artikel, Video Datei, Bild usw.) präzise zu beschreiben.

Mit dem Einführen des Web 2.0 und den neuen Möglichkeiten, die den Webentwicklern eröffnet wurden, wurde auch das Social Tagging populär. Beim Social Tagging wird Wert auf das Wissen der einzelnen Benutzer gelegt. Dabei ist es wichtig, wie die Benutzer die vorliegenden Ressourcen interpretieren. Dadurch entsteht eine „Folksonomy". Der Begriff wurde 2004 von Vander Wal [Wt07] eingeführt und ist ein zusammengesetztes Wort aus Folk und Taxonomie. Der Unterschied zur Taxonomie ist, dass die Beschreibung der Ressourcen jedes Benutzers geöffnet ist. Die Taxonomien sind geschlossene Wörterbücher, die von Experten in einem Gebiet über eine Menge von Ressourcen erstellt und streng kontrolliert werden. Die Taxonomien sind konsistent und werden „in-house", also im Unternehmen, auf ein spezielles Gebiet zugeschnitten. Die Folksonomy dagegen ist jedem Benutzer einer Webplattform frei zugänglich und jeder kann zu ihr beitragen [Wt07]. Sie entsteht beim Konsumieren der Information seitens des Benutzers der Webplattform. Die Benutzer stecken ihr persönliches Wissen und ihre Interpretation der Ressourcen in die Schlagwörter. Man kann sagen, dass die Folksonomy "tagging that works" ist [Wt07]. Die drei Aspekte einer Folksonomy sind die 1) Tags (Metadata), 2) Objekte (Ressourcen) und 3) Identity (Benutzer) (Abb. 2.3).

Die Benutzer haben Interesse an bestimmten Objekten (Blog Artikeln, Videoclips oder Bilder). Sie sind dann in der Lage, durch Einfügen von zusätzlichen Informationen in Form von Metadaten (Tags), diese Objekte als solche zu definieren. Die Tags sind Begriffe, die mit Komma oder Leerzeichen getrennt sind, und die Interpretation der Information darstellen. Das Tagging ordnet dem Benutzer des Systems Ressourcen zu. Auf Basis der Tags ist es dann möglich, Benutzer mit gleichen Interessen zu identifizieren. Eine weitere Möglichkeit besteht in der Erkennung von gleichen oder ähnlichen Ressourcen.

Wal sieht zwei Typen von Folksonomien: breite (eng. broad) und schmale (eng. narrow). In einer breiten Folksonomy kann eine Ressource mehrmals mit einem Tag getaggt werden. Ein Beispiel dafür ist delicious.com. Die schmale Folksonomy dagegen erlaubt das Tagging einer Ressource mit einem Tag nur einmal. Beispiele dafür sind flickr.com und YouTube.

Die Vorteile, die eine Folksonomy mit sich bringt, sind [Pi09]:

- **authentische Sprache**. Jeder Benutzer hat eine eigene Ausdrucksweise und ein eigenes Wörterbuch.
- **Aktualität**. Die Tags sind immer aktuell, da die Benutzer diese zu jeder Zeit ändern können.
- **mehrere Interpretationen**. In einer Taxonomie mit strengen Regeln existiert nur eine Interpretation. In einer Folksonomy dagegen kann jeder sein persönliches Wissen und seine Interpretation der Information einfügen.
- **günstige Indexierung.** Die Ressourcen werden vom Benutzer beschrieben

und vom System leichter indexiert. Diese Lösung ist deswegen günstig, weil die Indexierung auf den einzelnen Benutzern verteilt ist. Das ermöglicht eine größere Skalierbarkeit.

- **Identifizierung von Communities**. Auf Basis der Tags werden Verbindungen zwischen den Nutzern des Systems identifiziert. Dadurch werden diese in Gruppen aufgeteilt und somit entstehen Communities.
- **bessere Empfehlungssysteme.** Durch die Identifizierung der Communities und die Verbindungen Benutzer-Ressourcen werden jedem einzelnen Benutzer einerseits Benutzer mit gleichen Interessen und andererseits Ressourcen, die ihn interessieren können, vorgeschlagen.
- **schneller als Taxonomie**. Taxonomie wird in einem kleinen Team von Experten in einem Wissensgebiet erstellt, was eine zeitaufwendige Aufgabe ist. Die Communities dagegen bestehen in der Regel aus tausenden Benutzern, die sehr schnell auf das Auftauchen neuer Ressourcen reagieren und bereit sind, schnell ihr Wissen über diese Ressourcen zu teilen.

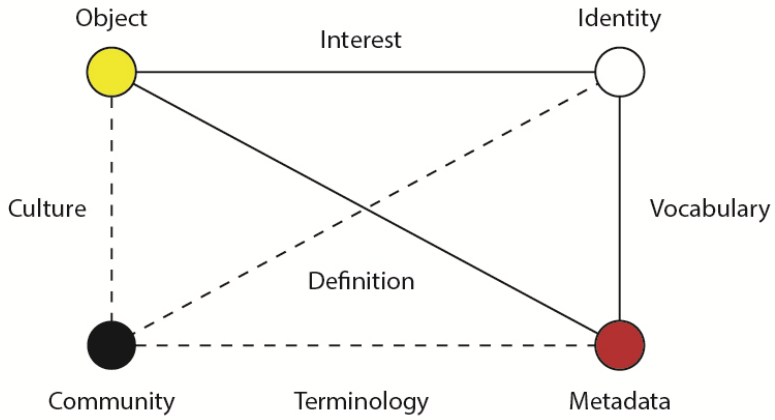

Abbildung 2.3 Dual Folksonomy Triad [Wt06]

Eine Folksonomy hat mehrere Vorteile, wobei man ihre Nachteile [Pi09] nicht unterschätzen darf. An erster Stelle muss angemerkt werden, dass das Fehlen eines einheitlichen Wörterbuchs Probleme mit sich ziehen kann. Die Synonyme und Antonyme können sehr schnell zu Missverständnissen führen. Dazu kommt auch das Benutzen verschiedener Sprachen. Ein Apfel auf einem Bild kann als [Apfel, pomme, apple, manzana] gekennzeichnet werden. Dadurch werden Relationen Benutzer-Ressource versteckt. Ein weiteres Problem sind Spam-Tags, die absichtlich eine Ressource falsch kennzeichnen. Zum Spam-Problem kommt auch das ungenaue Setzen von Tags durch unerfahrene Benutzer.

Der Nutzen einer Folksonomy oder der Beschreibung von Ressourcen durch den Benutzer ist sehr groß. Sie wird immer dort eingesetzt, wo die Volltextrecherche keinen Erfolg haben kann. Das Tagging ermöglicht auch Suche-Funktionalitäten, wie z.B *Filter by Tag* oder *Tagclouds*. Bei der Filterung nach Tags können Relationen wie *UND* und *ODER* definiert werden. Dadurch ist die Suche nach Äpfel & (Mann | Frau) möglich (Abb. 2.4.). Die Tags kann man bei Visualisierungen dazu verwenden, versteckte Muster zu entdecken.

Folksonomien können auch als eine Erweiterung der Volltextrecherche eingesetzt werden. Dieser Ansatz wird in dieser Diplomarbeit verfolgt. Die Schlagwörter, die zu jedem Muster angegeben werden, dienen als Beschreibung des Kontexts des Musters und haben ein hohes Gewicht in der Suche nach dem Einstiegspunkt in der Mustersprache.

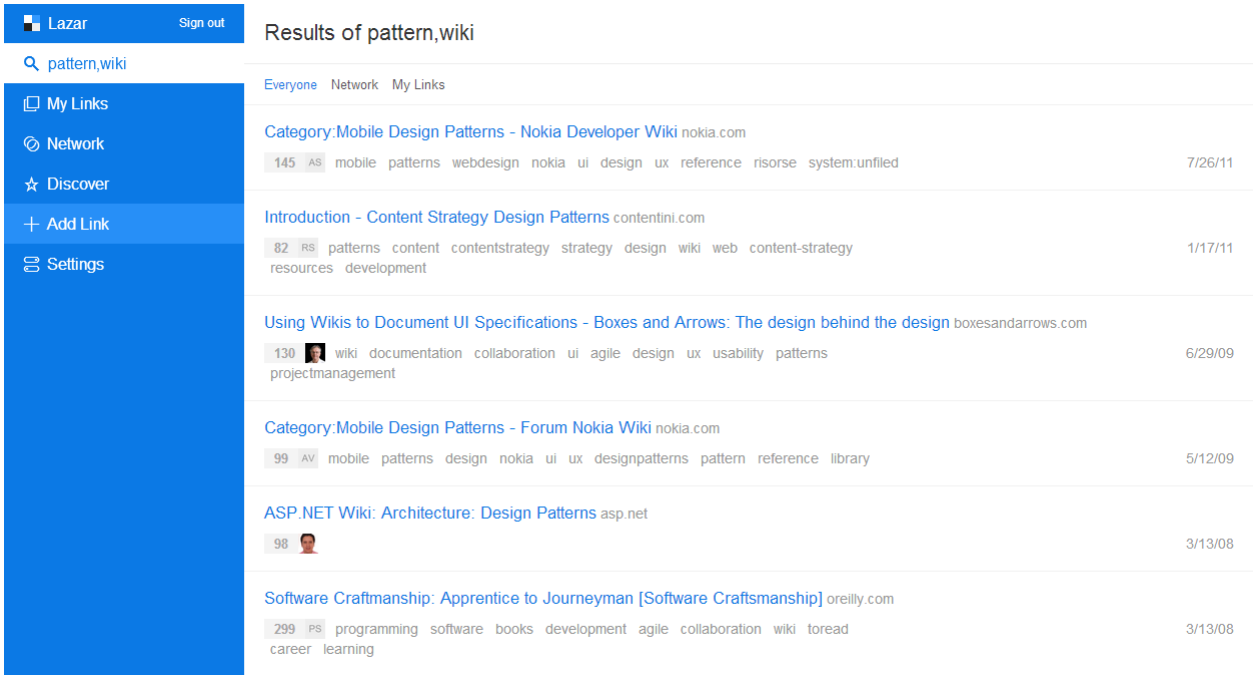

Abbildung 2.4 Suche nach Schlagwörtern “pattern, wiki” in delicious.com

Die Beispiele von flickr.com, Amazon und delicious.com zeigen, dass die Beschreibung und die daraus ermöglichte maschinelle Indexierung von unstrukturierten Daten dem Nutzer des jeweiligen Systems überlassen werden kann. Dieser Ansatz ist enorm skalierbar mit dem Nachteil, dass nicht alle Ressourcen gleiche Aufmerksamkeit bekommen. Trotzdem bleiben Menschen bei ihren Einschätzungen über die Information, die sie von den Ressourcen bekommen, viel genauer als Maschinen. Menschen sind in der Lage, schneller und genauer Bilder, Video-Dateien oder einfach Artikel zu interpretieren und mit Schlagwörtern zu beschreiben. Wie gerade gesehen machen die vielen Vorteile, die diese Technik mit sich bringt, sie sehr nützlich und weit verbreitet.

In diesem Kapitel wurden alle wichtigen theoretischen Grundlagen erläutert, die beim Ausarbeiten des Konzepts der Softwarelösung zu dieser Diplomarbeit angewendet wurden. Die Mustersprachen sind ein wichtiges Konzept und stellen eine Möglichkeit dar, das Wissen in einer Domäne zu speichern. Damit dieses Wissen jedoch zugänglich gemacht werden kann, ist das reine Vorliegen der Sprache nicht ausreichend. Die Mustersprache von Alexander enthält mehr als 250 Muster. Für unerfahrene Benutzer der Sprache ist es eine fast unmöglich erscheinende Aufgabe, sich in der Sprache zurechtzufinden. Hier werden andere Ansätze aus der Informationstechnologie gebraucht wie z.B. Decision Support Systeme, die wiederum für eine solche Aufgabe bestens geeignet sind.

3 Analyse einer Mustersprache

Mustersprachen sind in vielen Wissensdomänen weit verbreitet und bieten Lösungsstrategien für immer wieder vorkommende Probleme. Laut Alexander entsteht eine Mustersprache genau dann, wenn die Muster miteinander verbunden sind. Ein Problem lässt sich in Unterprobleme aufteilen und das für das übergeordnete Problem existierende Muster lässt sich seinerseits in kleinere Muster aufteilen, die die Teilprobleme lösen. Dadurch wird ein Problem in einer gegebenen Domäne gelöst. GoF dagegen bezeichnen die Design Patterns, die sie in ihrer täglichen Arbeit identifiziert haben, nicht als Sprache, sondern als Katalog. Als Kataloge von Mustern kann man auch viele Ansätze in der Domäne der Geschäftsprozesse bezeichnen. So sind die Workflow Patterns [WP], die als Grundlage der *Workflow Language* oder der *Business Process Modelling Language* dienen, als Katalog konzipiert. Ihre Struktur, setzt keine Verbindung zwischen den Mustern voraus, obwohl in vielen Fällen eine solche zu beobachten ist.

In diesem Kapitel wird eine Analyse der Mustersprachen durchgeführt, um wichtige Eigenschaften der Sprachen hervorzuheben. Ziel dabei ist es, sich auf Basis der Muster in der Sprache, die Relationen und die Klassifikation, sowie das versteckte Wissen in den Mustern über das Problem, die Grundlagen zum Aufbau eines DSS zu verschaffen. In Kapitel 3.1 wird die Struktur der Muster eingeführt, die als Grundlage einer einheitlichen Musterrepräsentation innerhalb der Mustersprachen dient. In Kapitel 3.2 wird die Klassifizierung der Muster erläutert und anhand von Beispielen aus der Praxis gezeigt. Kapitel 3.3 beschäftigt sich mit den Beziehungen, die die Muster miteinander verbinden. Danach werden in Kapiteln 3.4 und 3.5 die Beschreibung des Kontexts jedes Musters sowie die Beschreibung des Problemfelds erläutert. Abschließend werden in Kapitel 3.6 die Lösungswege, die das Einsetzen einer Menge von Mustern auf einem bestimmten Problem beschreiben, eingeführt.

3.1 Struktur der Muster

Die Muster brauchen als Teil der Mustersprachen eine einheitliche Struktur. Sie wird benötigt, um die Konsistenz bei der Beschreibung der einzelnen Muster zu gewährleisten, sowie den Vergleich zweier Muster zu ermöglichen. Im Laufe der Zeit sind immer mehr neue Mustersprachen entstanden und jede davon hat eine eigene Musterstruktur. Die meisten dieser Sprachen greifen auf die Struktur zurück, die Alexander eingeführt hat. Sie wird hier als klassische Struktur bezeichnet. Ein Muster nach diesem Schema besteht aus:

- ***Name***
- ***Bild***, das als Ikone verwendet wird,
- ***Zweck des Musters***: Kontext des Musters und kurze Beschreibung des zu lösenden Problems,
- ***Problemkontext***: stellt eine Zusammenfassung des Problems dar
- ***Problembeschreibung***: ausführliche Beschreibung des Problems

- ***Lösung***: stellt eine mögliche Lösung des Problems dar
- ***Lösung Skizze***: graphische Darstellung der Lösung
- ***Relationen*** zu den Mustern, die als nächstes Muster angewendet werden sollen

Diese Struktur kann auch als Muster betrachtet werden, das beim Aufbau neuer Mustersprachen immer wieder verwendet werden kann. In diesem Fall ist die Rede von einer Mustersprache für den Aufbau von Mustersprachen [MD96].

Eine weitere Mustersprache ist die Zusammengefasste von GoF im Buch „Design Patterns: Elements of Reusable Object-Oriented Software“. Sie ist für die Domäne des Entwurfs von objektorientierter Software entwickelt worden. Diese Tatsache ist in der Struktur, die jedes Muster aufweist, gespiegelt. Die Hauptteile, die man bei Alexander findet, sind auch hier zu erkennen. Es gibt Unterschiede und sie sind auf die Natur der Domäne, für die diese Sprache entwickelt wurde, zurück zu führen. Die Entwurfsmuster haben wie das klassische Muster einen Namen und einen Zweck. Eine Besonderheit ist der Abschnitt “Auch bekannt als”. Diese Entwurfsmuster sind kein neues Wissen, sondern stellen bekannte Praktiken beim Aufbau von Software dar. Erfahrene Softwareentwickler setzen auf diese Praktiken in der täglichen Arbeit. Daraus sind mehrere Namen für die gleichen Muster entstanden. Als nächster Teil in der Struktur, folgt die Problembeschreibung, die in Motivation und Anwendbarkeit aufgeteilt ist. Danach folgt eine Lösungsskizze in Form von einem UML Diagramm. Sie ist gefolgt von der eigentlichen Lösung, aufgeteilt in Teilnehmer, Interaktionen, Konsequenzen, Implementierung, Beispielquellencode und bekannte Verwendungen. An letzter Stelle kommen die Relationen zu den anderen Mustern unter dem Namen “Verwandte Muster”. Es ist leicht zu sehen, dass auch hier auf die von Alexander entworfene Sprache zurückgegriffen wird. Jede neue Sprache kommt mit ihren Besonderheiten, aber das Prinzip des Musters bleibt bestehen.

Nimmt man Beispiele von Mustern anderer Mustersprachen, ergibt sich das gleiche Bild. Die Muster folgen einem Schema, das der Klassischen sehr ähnelt. Der Grund ist, dass die interne Repräsentation der Muster innerhalb der Sprache konsistent bleiben soll, damit sind die Benutzer der Sprache in der Lage, sich leicht mit den Mustern der Sprache vertraut zu machen.

3.2 Klassifizierung

Eine Mustersprache stellt Problemlösungen in einer Domäne dar. Sie stehen den Benutzern der Sprache bereit, um beim Lösen von wieder vorkommenden Problemen angewendet zu werden. Eine Schwierigkeit dabei ist die Auswahl des passenden Musters. Die Fragen, die dabei entstehen, sind: wie soll das richtige Muster, das genau zu der Situation passt, gefunden werden und muss man die Sprache genauer kennen oder gibt es andere Auswahlmöglichkeiten, sich in der Sprache zu orientieren und leichter einen Einstiegspunkt zu finden? In diesem Fall bietet die Klassifikation der Muster innerhalb der Sprache Abhilfe. Das ist eine wichtige Eigenschaft der Sprachen, weil somit eine erste Vorauswahl seitens der Benutzer der Sprache getroffen werden kann.

Die Muster in einer Mustersprache werden in verschiedene Kategorien aufgeteilt, die bestimmte Gruppen von Problemen erfassen. Die Muster in der Sprache von Alexander folgen z.B. eine strikte Hierarchie. Als Beispiel nehmen wir die Gruppe der Muster, die eine Stadt und Gemeinschaft definieren. Diese werden gefolgt von den Mustern, die den Aufbau von Wohnflächen und dazugehörigen zusätzlichen Wohnbereichen beschreiben. Diese ihrerseits werden von der Gruppe der Muster gefolgt, die sich mit dem eigentlichen Aufbau der Wohnflächen und Wohnbereiche befassen.

Die Klassifizierung bei GoF dagegen unterliegt keiner strengen Hierarchie und orientiert sich einmal an der Aufgabe und einmal am Gültigkeitsbereich der Muster [GR11] (siehe Abb. 3.1).

		Aufgabe		
		Erzeugungsmuster	**Strukturmuster**	**Verhaltensmuster**
Gültigkeitsbereich	**klassenbasiert**	Fabrikmethode	Adapter	Interpreter Schablonenmethode
	objektbasiert	Abstrakte Fabrik Erbauer Prototyp Singleton	Adapter Brücke Dekorierer Fassade Fliegengewicht Kompositum Proxy	Befehl Beobachter Besucher Iterator Memento Strategie Vermittler Zustand Zuständigkeitskette

Abbildung 3.1 Dimensionen der Ausprägung von Entwurfsmustern

Eine solche Klassifizierung hilft dem Benutzer, sich in der Sprache zu orientieren und das richtige Muster zu identifizieren. Das kann aber nur dann zu einem Erfolg führen, wenn es im Voraus eine klare Vorstellung über das Problem gibt. Wegen der strengen Hierarchie in der Mustersprache von Alexander sind die Architekten eines Hauses in der Lage, sehr schnell an das passende Muster zu kommen. Sie können alle Muster von der obersten Gruppe, die sich mit dem Bau und Konzept von Städten befasst, ausschließen und sich direkt an der darunter liegenden Gruppe orientieren. Bei den Entwurfsmustern von GoF ist die Aufgabe dagegen nicht so einfach. Die Schwierigkeit besteht darin, die einzelnen Objekte im Voraus im aufzubauenden Softwaresystem ausfindig zu machen. Viele Objekte, die dabei identifiziert werden, existieren in der realen Welt nicht und müssen auf einer anderen Abstraktionsebene betrachtet werden. Viele Entwickler verlassen sich dann auf ihre eigene Erfahrung und die Klassifizierung hilft ihnen bis zu einem gewissen Grad, die Muster zu sortieren, um den Einstiegspunkt herauszufinden. Für den ungeübten Entwickler dagegen ist es keine leichte Aufgabe, das richtige Muster zu erkennen.

3.3 Beziehungstypen

Eine weitere Eigenschaft der Muster sind die Relationen zu anderen Mustern. In allen Mustersprachen sind diese in der einen oder anderen Form ausgeprägt und in einen speziell für diese Zwecke ausgelegten Teil gegliedert. Bei Alexander sind diese unter „Relationen“ und bei GoF unter „Verwandte Muster“ zu finden. Die Relationen zwischen den Mustern sind eines der wichtigsten Merkmale einer Mustersprache und einer der Gründe, wie in Kapitel 2 gezeigt wurde, für die Bezeichnung eines Katalogs mit Mustern als Mustersprache.

Die ausgeprägte Hierarchie in der Mustersprache von Alexander ermöglicht eine Zuordnung der Probleme, die die Muster lösen. Ein Problem kann in kleinere Probleme aufgeteilt werden, die von Mustern gelöst werden, die sich ihrerseits weiter unten in der Hierarchie befinden. Die Relationen zwischen den Mustern sind einseitig und gehen von einem Hauptmuster zu den untergeordneten Mustern aus. Die Semantik dieser Verbindungen ist sehr einfach und begrenzt sich auf “gefolgt von” und “Alternative zu”. Nehmen wir die GoF Mustersprache, ist schnell zu erkennen, dass Relationen eine klare Richtung aufweisen. Im Unterschied zur Mustersprache von Alexander existieren bei der GoF Mustersprache viele unterschiedlichen Relationen [Nj98]:

- ***Uses***: ein Muster benutzt ein anderes
- ***Refines***: ein Muster spezifiziert ein allgemeines
- ***Conflicts***: ein Muster steht im Konflikt mit einem anderen
- ***Used by***: ein Muster wird von einem zusammengesetzten benutzt
- ***Refined by***: ein Muster stellt eine Erweiterung eines anderen dar
- ***Variant***: ein Muster ist eine Variante von einem anderen
- ***Variant Uses, Similar, Combines, Requires*** usw…

Diese Relationen stellen keine einfachen Verbindungen zwischen den Mustern dar. Sie haben eine tiefere semantische Bedeutung. Diese Bedeutung kann in jeder Mustersprache eine andere Definition haben. Die Definition der Semantik folgt dann den Gegebenheiten der jeweiligen Mustersprache.

3.4 Kontext eines Musters

Die Muster einer Mustersprache sind nur innerhalb dieser Sprache gültig. Sie bieten Lösungen von Problemen, die in einer gegebenen Domäne entstehen. Laut Alexander sind die Muster Teillösungen zu einem gegebenen Problem. Wird ein Muster an diesem Problem angewendet, ändert sich sofort sein Kontext. Damit aber ein Muster zu einem Problem als passend eingestuft werden kann, muss der Anwender die Rahmenbedingungen des jeweiligen Problems kennen. Nur dann kann das Muster zur Lösung dieses Problems beitragen. Diese Abgrenzung des Kontexts der Problemstellung jedes Musters kann neben der Klassifikation vom Verfasser der Sprache dazu eingesetzt werden, dem Benutzer der Sprache den Einstiegspunkt in die Sprache zu erleichtern.

Aus dem Muster „Abstrakte Fabrik“ [GR11] von GoF Mustersprache lässt sich z.B. folgende Abgrenzung ableiten:

- Unabhängigkeit des Systems von der Erzeugung, Zusammensetzung und Repräsentation der Objekte
- Konfiguration eines Systems mit mehreren Objektfamilien
- Sicherstellung der Konsistenzbedingung einer Objektfamilie

Dadurch ist der Benutzer in der Lage, sich zu entscheiden, ob dieses Muster in den Rahmen seines Problems passt oder nicht. Diese Rahmen stellen sicher, dass eine Lösung des Problems mit dem gegebenen Muster möglich ist.

Diese Abgrenzung soll im ersten Schritt von den Autoren der Mustersprache übernommen werden. Grund dafür ist, dass sie sich mit den Problemen und den dazugehörigen Lösungen auseinander gesetzt haben und die Thematik und Problematik sehr gut kennen. In einem zweiten Schritt sollen alle Benutzer der Sprache die Möglichkeit haben, den Kontext zu vervollständigen. Für sie bleibt einzig die Möglichkeit, sich mit dem Text der Muster zu befassen. Bei praktischen Erfahrungen mit dem gegebenen Muster anhand der vorliegenden Informationen über den Kontext kann man ihn erweitern und genauer definieren. Sowohl für die Autoren der Sprache, als auch für die Benutzer, sind die Fragen, die beim Definieren des Kontexts gestellt werden sollen: was sind die Rahmenbedingungen im gegebenen Fall (z.B. innerhalb des Unternehmens oder innerhalb des Geschäftsprozesses) und (besonders wichtig für die Benutzer der Mustersprache) welche Teile der Struktur des Musters können diese Information liefern?

Eine universelle Antwort dieser Frage kann es nicht geben. Jede Sprache hat ihre Besonderheiten und die Bestandteile der Struktur der Muster, wie sie in Kap. 3.1 definiert wurden, sind nicht verpflichtend. Das bedeutet, dass in einer Sprache Teil *x* die Information über den Kontext liefern kann, was aber nicht für Sprache *y* zutreffen muss. Für jede Sprache ist eine andere Vorgehensweise nötig, wobei sich diese Vorgehensweisen je nach Sprache nicht nur marginal sondern auch komplett unterscheiden können. Deswegen wird hier auch keine konkrete Lösung gegeben. Diese Aufgabe wird den Verfassern der Sprache und der Gemeinschaft hinter der Sprache überlassen.

3.5 Beschreibung des Problemfelds

Alexander hat die Muster als Lösungen von immer wieder vorkommenden Problemen definiert. Interessierte, die die Sprache einsetzen wollen, müssen sich mit dem zu lösenden Problem gut auskennen. Erst wenn der Benutzer einer Mustersprache den Kontext des Problems gut abgrenzen und das Problem beschreiben kann, ist dieser in der Lage, ein passendes Muster in der Sprache zu finden. Das sind die zwei Grundvoraussetzungen, um den Einstiegspunkt in der Sprache leichter ausfindig zu machen. Deswegen spielt neben der Abgrenzung des Kontexts, die Beschreibung des Problemfelds eine wichtige Rolle.

Wie aber kann das Problem beschrieben werden? Wie soll die Information aus dem Muster entnommen werden und welche Form soll diese haben? Da beim Lösen von Problemen immer wieder Fragen gestellt werden müssen, wird diese Form der Beschreibung im Rahmen

dieser Diplomarbeit vorgeschlagen. Die Fragen sind in diesem Sinn die passendste Form einer Problembeschreibung. Dadurch kann mit dem Anwender des Musters ein Dialog geführt werden, der ihm die Richtung zeigt. Die Aufgabe besteht dabei in der Formulierung der Fragen. Sie müssen unmissverständlich auf das Problem des Musters zielen. Die Granularität der gestellten Fragen ist ebenfalls von Bedeutung und je feiner diese ist, desto besser wird ein Muster beschrieben.

Für das „Abstrakte Fabrik"-Muster passende Fragen wären:

- Soll eine Klassenbibliothek von Objekten angeboten werden, von denen nur die Schnittstellen, nicht aber ihre Implementierungen offengelegt werden?
- Soll eine Schnittstelle von Familien verwandter oder voneinander abhängiger Objekte, ohne ihre konkreten Klassen zu benennen, angeboten werden?

Die zwei Fragen sind nur ein Beispiel, an Hand derer sich der Benutzer schnell und sehr präzise entscheiden kann, ob das Muster passend zu seinem Problem ist. Diese Information könnte in Kombination mit dem abgegrenzten Kontext, für das weitere Betrachten des Musters seitens des Benutzers, ausschlaggebend sein.

3.6 Beschreibung eines Lösungswegs

In jeder Mustersprache werden Kombinationen von Mustern zum Lösen von Problemen genutzt. In der GoF-Sprache gibt es zusammengesetzte Muster, wie MVC (Model, View, Controller), die immer in der gleichen Reihenfolge und der gleichen Konstellation angewendet werden müssen. Sprachen mit einer strikten Hierarchie teilen die Probleme in kleinere Probleme, die dann weiter in noch kleinere Probleme aufgeteilt werden usw. Dadurch entstehen Lösungswege, die aber zum Lösen des gleichen Problems immer wieder anders sein können. Dabei spielen die Relationen zwischen den Mustern eine wichtige Rolle, ebenso wichtig ist, welche Freiheit diese Relationen dem Benutzer beim Treffen von Entscheidungen geben. Das bedeutet allerdings nicht, dass dieses Konzept bei der GoF-Sprache nicht angewendet werden kann. Die Lösungswege betrachten das Problem auf einer höheren Abstraktionsebene und könnten mehrere Probleme mit einzubeziehen. Die Lösungswege sind von der Mustersprache von Alexander abgeleitet, allerdings mit einem wesentlichen Unterschied zu seiner Idee. Bei einem Lösungsweg muss nicht zwingend ein komplexes Problem in kleinere Probleme aufgeteilt werden, sondern die Probleme können komplett voneinander getrennte Teile der Wissensdomäne umfassen und die Lösung von einem viel größeren und umfassenderen Problem sein.

Die Lösungswege werden in diesem Sinne mehr als Dokumentation der Wege zum Lösen eines Problems betrachtet. Sie sollen wie alle Muster eine einheitliche Struktur haben. Eine vereinfachte Version dieser Struktur könnte aus folgenden Teilen bestehen:

- ***Name des Lösungswegs***
- ***Beschreibung des Lösungswegs***
- ***Eingesetzte Muster***
- ***Zusätzliche Informationen***

Der Name muss eindeutig sein, denn die Namen der Muster innerhalb einer Sprache werden für ihre eindeutige Identifikation gebraucht. Damit man die Lösungswege auch untereinander unterscheiden kann, ist auch hier die Indexierung der Lösungswege mit einer eindeutigen Bezeichnung nötig. Die Beschreibung kann alle Schritte enthalten, die beim Lösen des Problems unternommen wurden. Die Reihenfolge der eingesetzten Muster könnte auch noch zusätzlich gespeichert werden. Die Muster, die dabei eigesetzt wurden, müssen ebenfalls erfasst werden. Hier ist es denkbar, dass innerhalb einer Wissensdomäne die Relationen zu den Mustern mit semantischer Information versehen werden. Der letzte Teil der vorgeschlagenen Struktur sind die zusätzlichen Informationen. Dabei sind verschiedene Dateien möglich, wie z.B. Quellencode-Dateien, Word-Dateien mit den Anforderungen, BPEL-Dateien falls es sich um Geschäftsprozesse handelt usw.

In diesem Kapitel wurden die Mustersprachen untersucht und wichtige Konzepte hervorgehoben. Diese sollen als Grundlage des Konzepts einer Softwarelösung zum Verwalten von Mustersprachen dienen, sowie den Benutzern der Mustersprache einen leichten Einstieg ermöglichen. Welche Anforderungen sich aus dieser Analyse ableiten lassen, wird im nächsten Kapitel gezeigt.

4 Konzeption eines Pattern Wiki

Das Konzept eines Pattern Wiki baut auf die Anforderungen auf, die aus der Analyse der Mustersprachen hervorgehen. Dabei werden die zusätzlichen Informationen zu jedem Muster in der Sprache (Kontextbeschreibung und Beschreibung des Problemfelds) aktiv ausgenutzt. Das Konzept ist sprachenunabhängig und kann für jede beliebige Mustersprache angewendet werden. In Kapitel 4.1 werden die Anforderungen an ein solches System eingeführt. In Kapitel 4.2 werden dann das Konzept des Systems sowie alle wesentlichen Teile vorgestellt. In den nachfolgenden Kapiteln werden diese Teile im Detail beschrieben.

4.1 Anforderungen eines Pattern Wiki

Die Analyse der Mustersprachen in Kap. 3 bestimmt die Ankerpunkte, die man beim Erstellen des Konzepts eines Pattern Wiki beachten muss. Diese sind wie folgt:

Verwaltung der Mustersprachen:
Die Muster, die in der Sprache zu finden sind, sind, wie schon gezeigt wurde, nur innerhalb dieser Sprache gültig. Deswegen wird ein zentraler Speicherplatz gefordert, der als Repository der Mustersprache dienen soll. Als ein wesentlicher Teil der Verwaltung ist eine Möglichkeit gefordert, die es erlaubt, die Muster nach einer bestimmten Struktur dort abzulegen. Unter Verwaltung ist sowohl die Bearbeitung der bestehenden Muster, als auch das Löschen von Mustern zu verstehen.

Verwaltung von Lösungswegen:
Das Konzept der Lösungswege kann als ein wesentlicher Teil jeder Mustersprache betrachtet werden. Lösungswege geben den Benutzern die Möglichkeit, sich bereits erfolgreich durchgeführte Problemlösungen anzuschauen. Dafür wird, wie es auch bei den Mustern der Fall ist, ein zentraler Speicherplatz, sowie die Möglichkeit, neue Lösungswege zu erstellen, bestehende zu ändern oder zu löschen, benötigt.

Suche nach einem Einstiegspunkt in der Sprache:
Für unerfahrene Benutzer einer Mustersprache ist das Finden eines Einstiegspunkts eine der größten Hürden. Das ist selbst für Benutzer, die sich mit der Sprache auskennen, keine leichte Aufgabe. Deswegen wird nach einer Möglichkeit gesucht, auf Basis des in der Mustersprache versteckten Wissens, einen Einstiegspunkt in der Sprache zu finden.

Feedback auf Basis früherer Entscheidungen:
Eine Sprache wird in der Regel von mehreren Benutzern (einer Gemeinschaft) eingesetzt. Beim Suchen des passenden Musters wird dann eine Menge (siehe Kap. 3.5) an Informationen vom Benutzer gebraucht. Dieser vergleicht die mitgebrachte Information mit der Information, die die Sprache zusätzlich zu jedem Muster bereitstellt, und kann auf dieser Basis eine Entscheidung treffen, ob das Muster passend zu dem Problem ist. Wird diese Information dann bei der nächsten Suche als Feedback an den nächsten Interessierten

weitergegeben, weil ihm ähnliche Kriterien bei der Suche vorliegen, kann sich dieser leichter und genauer für eine passende Lösung entscheiden. Aus diesem Grund wird ein Empfehlungssystem erwartet, das die Suchgeschichte speichern und auswerten kann.

4.2 Konzept eines Pattern Wiki

Aus den gerade eingeführten Anforderungen lassen sich folgende Komponenten eines Pattern Wiki ableiten: *Pattern Repository*, *Pattern Manager*, *Solution Manager*, *Recommender System* sowie *Search Manager* (Abb. 4.1).

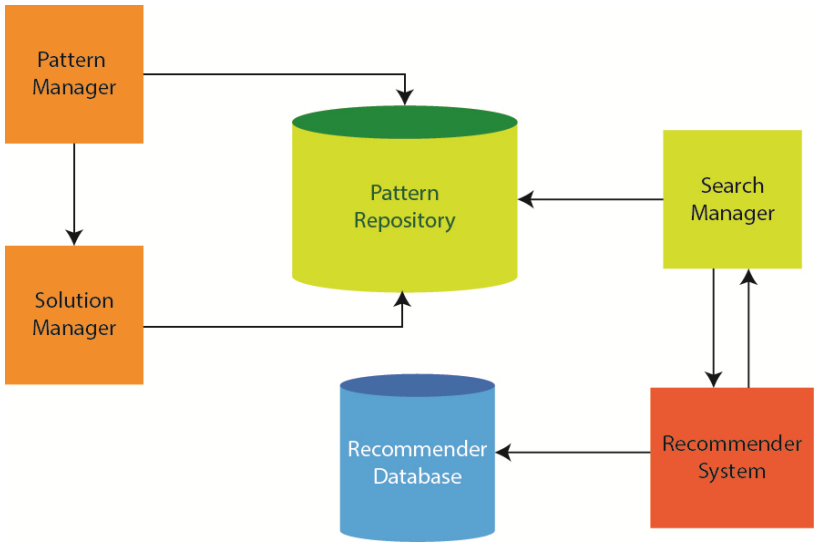

Abbildung 4.1 Konzept des Systems

Im *Repository* (Abb. 4.1) werden die Muster abgelegt. Darauf haben die drei Komponenten *Pattern Manager*, *Solution Manager* und *Search Manager* Zugriff. Die Aufgabe des *Pattern Managers* besteht darin, eine leichte und unkomplizierte Verwaltung der Muster in der Sprache zu gewährleisten. Einfügen und Bearbeitung von Mustern in der Sprache sind die Aufgaben, die diese Komponente übernimmt. Die Muster sind jedem Benutzer des Systems frei zugänglich und durch die Relationen zwischen ihnen ist der Benutzer in der Lage, sich innerhalb der Sprache frei zu bewegen. Der *Search Manager* begleitet den Benutzer bei der Suche nach einem Einstieg in die Mustersprache. Die *Recommender System*-Komponente ist für die Berechnungen zuständig, die ausgeführt werden, und die auf den Suchanfragen der Benutzer und den Entscheidungen, die sie am Ende der Suche treffen, basieren. Dadurch lernt das System vom Benutzer und es gilt, dass je mehr Benutzer es verwenden, desto passender werden die Empfehlungen, die das Empfehlungssystem macht. Die letzte Komponente ist das *Solution Repository*. Sie ist in einer engen Verbindung mit dem Pattern Repository. Dadurch ist es für interessierte Benutzer möglich, sich bei der Suche nach dem passenden Muster direkt auch Lösungen anzusehen, bei denen ein spezifisches Muster gebraucht wurde und wie das Problem insgesamt gelöst wurde. Diese Komponente übernimmt die Aufgaben zur Verwaltung der Lösungswege.

4.2.1 Pattern Repository und Pattern Manager

Bei der Erstellung von Wissensmanagementsystemen hat sich in den letzten Jahren das Konzept des Wiki Systems etabliert. Das erste Wiki-System "Portland Pattern Repository"[8] wurde als Webplattform entwickelt, auf dem Programmierer ihre Entwicklungsmuster anderen Programmierern zur Verfügung stellen könnten. Im Unterschied zu den üblichen Webseiten, die nur das Betrachten der Inhalte erlauben, war das Wiki-System so konzipiert, dass jeder seine eigenen Muster dort publizieren könnte. Jeder Benutzer kann aktiv dazu beitragen, dass sich das Repository erweitert, indem immer neues und aktuelles Wissen dort gespeichert und verfügbar gemacht wird. Im Jahr 2001 wurde auf Basis des Portlands Pattern Repository das erfolgreichste Wiki-System Wikipedia vorgestellt. Seine Popularität verdankt es an der Möglichkeit, eigene Beiträge in einer unkomplizierten und benutzerfreundlichen Weise zu erstellen, Beiträge anderer Benutzer zu kommentieren oder zu korrigieren [FM13]. Im Laufe der Zeit sind viele Wissensmanagementsysteme auf Basis des Wiki-Konzepts entstanden. Diese werden in vielen Unternehmen dazu eingesetzt, das interne Wissen zu bewahren. Die Mitarbeiter sind in der Lage, Dokumentationen zur verschiedenen Themen zu erstellen. Da die Wiki-Systeme in der Regel web-basiert sind, können dann alle Mitarbeiter über einen Web-Browser auf diese zentrale Wissensdatenbank zugreifen.

Ein Wiki-System bietet die richtigen Funktionalitäten, die man bei der Verwaltung einer Mustersprache braucht. Es existieren viele Erweiterungen für das System, die es sehr vielfältig und anpassbar machen. Eine davon ist das Semantik Wiki Bundle[9]. Sie ermöglicht es, die Links zwischen den Mustern mit semantischer Information zu versehen. Diese sind dann nicht mehr nur einfache Links, sondern haben eine spezifische vordefinierte semantische Bedeutung. Diese Bedeutung der Links ist ein wichtiger Aspekt in einer Mustersprache. Es erlaubt dem Benutzer beim Suchen in der Sprache, leichter die Relationen zwischen den Mustern zu erkennen und zwischen ihnen zu navigieren. Die Links sind nicht mehr nur eine Verbindung zu den anderen Mustern, sondern geben zusätzliche Information und vereinfachen den Entscheidungsprozess.

Auf Grund dieser Eigenschaften eines Wiki Systems wird für das Pattern Repository ein Wiki System als Grundlage vorgeschlagen. Daraus lässt sich der Name des Konzepts ableiten: Pattern Wiki.

Nicht nur die Links sind von großer Bedeutung im Repository des Pattern Wiki. Wie in Kapitel 3.4 erläutert wurde, wird zu jedem Muster im System eine Beschreibung des Kontexts gebraucht. Diese könnte im *Pattern Repository* in Form von "Tags" erfolgen. In Kapitel 2.4 wurde deutlich, dass Social Tagging eine wichtige Rolle spielt, wenn man sich auf die Community verlässt, um Objekte zu klassifizieren. Hier wird genau dieser Ansatz angestrebt. Die Community soll ihren Beitrag leisten, damit der Kontext jedes Musters in Form von Tags immer besser beschreiben wird. Je besser und treffender diese Beschreibung ist, desto genauer sind die vom Search Manager gelieferten Ergebnisse. Diese semantischen

[8] http://c2.com/cgi/wiki?WikiHistory
[9] http://semantic-mediawiki.org/

Informationen sollen als Metadaten jedem Muster zugewiesen werden und zu jeder Zeit änderbar sein. Dadurch kann jeder Nutzer dazu beitragen, dass die Muster den Problemstellungen immer genau angepasst sind. Diese Aufgaben übernimmt der *Pattern Manager*. Er hat einen direkten Zugriff auf die Datenbestände des Repository und beschäftigt sich mit der Verwaltung der Muster.

Das *Repository* übernimmt eine der wichtigsten Rollen. Die Information, die sie anbietet, wird aktiv von den anderen Komponenten genutzt und es ist als Kern des Systems zu bezeichnen.

4.2.2 Suche nach einem Einstiegspunkt

Der *Search Manager* (Abb. 4.1) dient als Schnittstelle zwischen den Benutzern und dem Repository. Der Benutzer des Pattern Wiki ist, wie gerade erläutert, in der Lage, sich frei durch die Sprache zu bewegen und die Muster, die zu seinem Problem passen, auszusuchen. Diese Suche ist insbesondere für unerfahrene Benutzer der Sprache eine aufwendige Aufgabe. Diese sollen in der Lage sein, ohne Kenntnisse über die Muster, das Richtige schnell zu finden.

Der *Search Manager* ist wie die Dialog-Komponente (Abb. 2.1) in einem DSS aufgebaut und besteht aus Schritten, die den Benutzer des Systems zur Auswahl eines passenden Musters führen. Voraussetzung dafür ist Vorwissen über das zu lösende Problem. Das Vorwissen, welches der Benutzer mitbringen sollte, umfasst an erster Stelle ein erkanntes Problem (Absicht), weiterhin ist es wichtig, dass der Benutzer den Kontext des Problems kennt (Kontextbeschreibung) sowie die genauen Änderungsmöglichkeiten in der Domäne, in der das Problem entstanden ist (Problembeschreibung). Diese drei Hauptteile dienen als Grundlage beim Treffen der Entscheidung, welches Muster ausgewählt werden soll. Jedes dieser drei Teile hat ein Gewicht im Endkoeffizient, welches das System anhand der Benutzereingabe berechnet. Je höher der Koeffizient ist, desto größer ist die Wahrscheinlichkeit, dass dieses Muster eine passende Lösung des beschriebenen Problems darstellt.

Die Gewichtsverteilung dieser drei Komponenten könnte wie folgt definiert werden: Absicht 0.2, Kontextbeschreibung 0.3 und Problembeschreibung 0.5. Dadurch wird für jedes Muster, das passend zum Problem des Benutzers sein kann, ein Koeffizient von maximal *1* berechnet. Dieser entspricht dann dem Rang des Musters in der Ergebnisliste.

Die Suche nach dem Muster lässt sich in drei Schritte aufteilen:

Schritt 1 (Vorauswahl). Im ersten Schritt stehen dem Benutzer alle Kategorien von Mustern zur Auswahl. Entscheidet sich dieser für eine oder mehrere Kategorien, werden nur die Muster in diese(n) Kategorie(n) weiter betrachtet. Ziel dabei ist es, eine Vorauswahl zu treffen und die Suche in den Mustern einzuschränken. Dieser Schritt ist notwendig, weil die Benutzer die Suche erstens mit einem Vorwissen über das Problem starten und zweitens eine Beschränkung der Auswahl in einem so frühen Stadium der Suche, diese viel genauer

gestalten kann. Dadurch sind die Optionen, die dem Benutzer in den nächsten Schritten zur Wahl stehen, nicht so detailliert und umfassend. Der Benutzer kann seine Wahl leichter treffen. In Schritt 1 wird kein Teilkoeffizient berechnet und diese Auswahl hat keinen Einfluss auf das Endergebnis.

Schritt 2 (Absicht und Kontextbeschreibung). Dieser Schritt kombiniert die Beschreibung der Absicht und des Kontexts. Die Absicht hat ein kleines Gewicht, da die eingegebene Information nur von dem Benutzer abhängig ist und auch sehr ungenau sein kann. Die Absicht wird in einem Textfeld als freier Text eingegeben. Dieser Text wird dann in einzelne Terme (Wörter) aufgeteilt. Im Text der Muster wird nach den Termen gesucht. Im Pattern Wiki wird anhand der Anzahl der Treffer der Teilkoeffizient berechnet. Dieser kann Maximum 0.2 sein.

Nachdem der Benutzer die Absicht beschreiben hat, folgt die Abgrenzung des Kontexts. Um den Kontext beschreiben zu können, müssen die Benutzer sich gut mit dem Problem und der Domäne, in der dieses entstanden ist, auskennen. Sogar erfahrene Benutzer, die sich mit der vorliegenden Mustersprache gut auskennen, sind nicht immer in der Lage, diesen Kontext zu beschreiben [KK07]. Die Information sollte von dem Verfasser der Sprache im Voraus in Form von Tags (siehe *Kontextabgrenzung* in Kapitel 3.4) eingegeben und durch die Community erweitert und vervollständigt werden. Die Schwierigkeiten, die die Nutzer der Sprache beim Bestimmen des Kontexts haben, sind im Gewicht dieses Teils im Endkoeffizient widergespiegelt. Dieses Gewicht beträgt Maximum *0.3* und ist durch die Anzahl der Tags, die vom Benutzer ausgewählt wurden, bestimmt. Je mehr Tags in der Auswahl in einem Muster zu finden sind, desto näher an 0.3 wird das Teilgewicht sein.

Schritt 3 (Problembeschreibung). Im letzten Schritt geht es darum, Fragen über das zu lösende Problem richtig zu beantworten. Sie sind im Voraus vom Verfasser der Sprache oder von der Community festgelegt. Wie im Kapitel 3.5 einführend erklärt wurde, dienen sie dazu, das Problem möglichst konkret zu beschreiben. Sie sind auf das Problem gerichtet und eine positive Antwort hat einen viel größeren Wert und hohes Gewicht im berechneten Koeffizienten. Die Trefferquote kann dadurch erhöht werden, dass zu jedem Problem mehrere Fragen gestellt werden. Sie stellen sicher, dass die Absicht des Benutzers vom jeweiligen Muster abgedeckt ist. Aus diesem Grund wurde für diesen Schritt das größte Gewicht gewählt.

Diese drei Schritte haben zum Ziel, eine möglichst breite und umfassende Information zu jedem Muster zu sammeln. Dadurch entsteht ein vollständiges Bild des Problems und seines Kontexts. Auf Basis der zusätzlichen Information, die vom Verfasser der Mustersprache und von der Gemeinschaft im Voraus eingegeben wird, wird die Suche nach dem richtigen Muster erst ermöglicht.

4.2.3 Solution Manager

Ziel des Solution Repository ist es neue Lösungswege (siehe Kap. 3.6), die beim Lösen von realen Problemen entstanden sind, abzuspeichern. Dadurch sind Interessierte am Lösungsweg

in der Lage, leicht nachzuvollziehen, wie dieser entstanden ist und können ihn wiederholen. Die Lösungswege werden im Pattern Repository abgelegt und stehen dadurch den Benutzern des Systems immer zur Verfügung. Somit wird ermöglicht, dass die Interessierten beim Durchsuchen des Muster-Repository nachschauen können, wie die einzelnen Muster dazu beigetragen haben, diese spezifischen Probleme zu lösen. Die allgemeinen Lösungen, die die Muster in jeder Sprache beschreiben, werden mit konkreten Umsetzungen erläutert und erweitert. Die anderen Komponenten wie Pattern Manager beziehen Informationen vom Solution Manager welche in Form von semantischen Links in die Muster eingefügt werden. Die Suche nach dem Einstiegspunkt kann auch davon profitieren, dass die Relevanz der Muster für das beschriebene Problem auf Basis der Lösungswege, in der das Muster vorkommt, weiter verfeinert wird. Die Benutzer bekommen dadurch die Möglichkeit, ihr Feedback jederzeit einzufügen. Falls sich diese Lösungswege später als Best Practice erweisen, besteht dann die Möglichkeit, neue Muster, die in der Kernsprache übernommen werden, zu entwerfen. Findet man den Lösungsweg eines Problems mehrmals im Repository, ist dieser ein Kandidat, um als Muster in der Sprache übernommen zu werden. Dadurch leistet die Community ihren Beitrag zur Entwicklung der Sprache und diese Arbeit ist nicht einzig dem Entwickler oder Verfasser dieser Sprache überlassen.

Ein Lösungsweg soll im Unterschied zu einem Muster beschreiben, wie die Muster angewendet wurden, um ein Problem zu lösen (siehe Kap. 3.6). Dabei werden in den meisten Fällen viele zusätzliche Dateien erstellt, die beim Lösen des Problems gebraucht werden. Deswegen werden diese auch zu jeder Lösung gespeichert. Dadurch kann der Benutzer nachvollziehen, wie der konkrete Lösungsweg entstanden ist und kann ihn anhand der vorhandenen Dateien wiederholen.

4.2.4 Recommender System

Um die Suchergebnisse zu verbessern und damit die Suchgeschichte als Feedback zu finden, wird ein *Recommender System* als Teil des *Pattern Wiki* gebraucht. Die Entscheidungen, die jedes Mitglied der Community trifft, können dazu eingesetzt werden, zukünftige Suchanfragen treffsicherer zu machen. Wird nach einem bestimmten Problem immer mit den gleichen oder ähnlichen Angaben gesucht, ist die Wahrscheinlichkeit sehr groß, dass Benutzer, die in der Zukunft gleiche Angaben machen, nach dem gleichen Muster suchen. Daraus entsteht die Notwendigkeit, die Suchgeschichte zu speichern und als Grundlage für spätere Entscheidungen einzubeziehen. Dadurch entstandene Vorschläge und ihre Relevanz hängen vom Wissen der Benutzer ab. Neben den vielen Vorteilen, die das *Recommender System* bringt, darf man die negativen Charakteristiken und Nebeneffekte, die auftreten können, nicht unterschätzen. Da das System sich nur auf die Benutzer verlässt, sind Ungenauigkeiten durch falsche vergangene Entscheidungen sehr wahrscheinlich. Dazu muss man auch anmerken, dass das System eine gewisse Anzahl von Anfragen braucht, um relevante Informationen zu liefern.

Das *Recommender System* setzt auf einer Datenbank auf (*Recommender Repository*, siehe Abb. 4.1), in der die Suchgeschichte der Benutzer (alle Eingaben, die die Benutzer im Dialogmanager machen) gespeichert wird. Diese Angaben werden der Entscheidung des

Benutzers zugeordnet. Unter Entscheidung ist zu verstehen, welches Muster am Ende der Suche in der Liste mit den Vorschlägen ausgewählt wurde. Bei einer zukünftigen Suche wird auf Basis der vom Benutzer eingegeben Information die Wahrscheinlichkeit berechnet, welches Muster am besten als Lösung oder als Einstiegspunkt in der Sprache gebraucht werden kann. Das *Recommender System* kann als Ergänzung zur Standardsuche eingesetzt werden. Es stellt einen Einsatz dar, bei dem das Wissen der Benutzer zurück in das System fließt und ohne Absicht die Benutzer es bereichern und erweitern.

Das in diesem Kapitel vorgestellte Konzept und seine Komponenten wurden anhand der Anforderungen der Mustersprachen aufgestellt. Die Komponenten versuchen, alle wichtigen Aspekte einer Mustersprache abzudecken. Neben der Verwaltung und der zentralen Speicherung der Muster, soll eine Möglichkeit geschafft werden, um einen Einstiegspunkt in die Sprache zu gewährleisten. Diese Suche basiert auf den Überlegungen zum Kontext eines Musters, sowie auf der Problembeschreibung des Musters. Mit der Einführung des *Solution Managers* wird den Benutzern ein Werkzeug in die Hände gegeben, mit dem sie in der Lage sind, sich nicht nur andere Anwendungen der Muster anzusehen, sondern anderen Benutzern im Gegenzug auch eigene Anwendungen bereit zu stellen. Weil die Suche nach dem Einstiegspunkt eine der wesentlichen Aufgaben in dem vorgestellten Konzept ist, wurde sie um eine weitere Komponente ergänzt, welche Feedback aus der Suchgeschichte anderer Benutzer liefert. Dadurch wird das *Pattern Wiki* zu einem vollständigen System, das alle benötigten Funktionalitäten bietet. Eine Implementierung dieses Konzeptes wird im nächsten Kapitel vorgestellt.

5 Implementierung der Software

Im folgenden Kapitel werden zunächst die Anforderungen an die Software erläutert. Danach folgt eine Einführung in die Frameworks, die bei der Erstellung von Pattern Wiki eingesetzt wurden. Wie die einzelnen Komponenten, wie im Kapitel 4 beschrieben, in der *Pattern Wiki* umgesetzt sind, wird in Kapitel 5.3 näher erklärt.

5.1 Anforderungen

5.1.1 Funktionale Anforderungen

Verwaltung von Mustersprachen (Pattern Manager):

Beim Erstellen jeder Mustersprache brauchen die Verfasser Werkzeuge, um den Überblick über die Sprache zu behalten. Diese Werkzeuge sollen eine Reihe von Funktionalitäten anbieten, die gebraucht werden, um die Sprache zu verwalten. Deswegen sollen die Verfasser jeder Mustersprache in der Lage sein, in das Repository ein neues Muster einzufügen, dort gespeicherte Muster zu ändern oder diese zu löschen.

Einfügen von benutzter definierten Metainformationen (Pattern Manager):

In Kapitel 3 wurde deutlich, dass die Community eine wichtige Rolle spielt. Nachdem die Verfasser ein neues Muster publiziert haben, steht dieses bereit, von anderen Interessierten in bestimmten Situationen eingesetzt zu werden. Beim Erstellen des Musters wird zusätzliche (Meta) Information gebraucht. Diese kann nicht immer vollständig sein und insbesondere für die Verfasser ist das eine aufwendige Aufgabe. Deswegen sollen die Benutzer der Sprache in der Lage sein, diese Metainformationen zu jeder Zeit zu ändern und zu erweitern.

Verwaltung von Lösungswegen (Solution Manager):

Die Lösungswege stellen ein wichtiges Konstrukt im aufzubauenden System dar. Damit diese vom Benutzer oder Verfasser der Mustersprache leicht eingefügt, bearbeitet oder gelöscht werden, soll das System die Möglichkeit bieten, sie zu verwalten.

Verwaltung von Hilfsdateien der Lösungswege (Solution Manager):

Die Lösungswege bestehen nicht wie die Muster in der Kernsprache nur aus einem Textteil, sondern sind so konzipiert, dass auch zusätzliche und für die Benutzer nützliche Dateien angehängt werden können. Deswegen sollte das System den Verfassern die Möglichkeit bieten, einem Lösungsweg zusätzliche Dateien zufügen. Diese Dateien sollen dann vom System als Download bereitgestellt werden, damit alle Interessierten diese herunterladen können.

Search Manager:

Die Mustersprachen stellen eine Herausforderung nicht nur für den unerfahrenen Benutzer dar. Damit für alle Benutzer ein einfacher Anfang in der Sprache gewährleistet ist, soll das System die Möglichkeit bieten, dass sich die Benutzer leicht in der Mustersprache orientieren

und nach einem zu ihrem Problem passenden Muster suchen. Die Suche soll dabei auf die benutzerdefinierten Metainformationen setzen. Die Benutzer des Systems sollen somit in die Lage versetzt werden, mittels einfacher Schritte zu einem möglichen Einstiegspunkt geführt zu werden.

Recommender System:
Es wurde schon mehrmals erwähnt, dass das System in einer Interessensgruppe eingesetzt wird. Jede solche Gemeinschaft profitiert vom kollektiven Wissen. Wenn das Wissen auf Basis von persönlichen Entscheidungen nach dem Benutzen des Systems entstanden ist und dieses Wissen zurück in das System fließt, wird ein großer positiver Effekt erzielt. Deswegen soll das System die Möglichkeit bieten, die Entscheidungen der Benutzer des Systems zu erfassen, zu speichern und in einer geeigneten Form für spätere Auswertungen bereit zu stellen. Die so aufgefasste Information soll zukünftige Suchanfragen unterstützen und dem Benutzer auf Basis seiner Suchkriterien passende Muster neben den Suchergebnissen einblenden. Diese Vorschläge sollen auf der Suchgeschichte aller Benutzer und ihren früheren Entscheidungen basieren. Solches Feedback kann eine wichtige Rolle beim Treffen von Entscheidungen spielen, deswegen soll das System die Möglichkeit bieten, die Suchgeschichte zu analysieren und auf ihrer Basis, Vorschläge zu gängigen Suchanfragen zu machen.

5.1.2 Nichtfunktionale Anforderungen

Technische Anforderungen

- Als Programmiersprache ist PHP in der Version 5.x vorausgesetzt.
- Die Architektur soll als Client/Server konzipiert werden
- Der Client soll webbasiert und in allen gängigen Webbrowsern abrufbar sein
- Der Server soll ein Apache Server der Version 2.x sein

Benutzerfreundlichkeit

- Die Graphische Oberfläche muss zeitgemäß gestaltet werden. D.h. sie soll übersichtlich und aufgeräumt sein und alle benötigten Funktionalitäten anbieten. Die Benutzer sollen ohne großen Aufwand den Umgang mit dem System lernen.

- Die Funktionen, die bereit stehen, sollen leicht bedienbar sein. Bei falschen Eingaben sollen entsprechende Fehlermeldungen angezeigt werden.

Zuverlässigkeit
Das System unterstützt den Entscheidungsträger bei der Optimierung eines Prozesses. Dabei spielt die Richtigkeit der vom System gemachten Vorschläge eine große Rolle. Aus diesem Grund soll das System die geforderten Informationen immer korrekt berechnen.

5.2 Frameworks

Das System, das für die Zwecke dieser Diplomarbeit erstellt wurde, setzt auf Mediawiki als Repository für die Muster auf. Die semantischen Links (Relationen zwischen den Mustern) werden vom *Semantic Mediawiki Bundle* ermöglicht. Die Erweiterungen, die die

Funktionalitäten des aufgebauten Systems darstellen, beruhen auf Mediawiki und *Semantic Media Bundle* APIs. Wie diese genau implementiert sind, wird in Kapitel 5.3 näher erläutert.

5.2.1 Mediawiki

Mediawiki ist die Software, die ursprünglich für die Zwecke von Wikipedia geschrieben wurde. Diese wurde schon am Anfang als Open Source freigegeben und ist heute die bekannteste Wiki-Software. Diese wird in vielen Unternehmen, sowie Nicht-Profit ausgerichteten Organisationen verwendet, um das Wissen, das dort generiert wird, in einer leicht zugänglichen Form einem breiten Publikum bereitzustellen. Mediawiki ist eine freie Software und kann als solche nach Wunsch der Anwender frei geändert werden. Mediawiki ist so konzipiert, dass die benötigten Funktionalitäten mit einer großen Auswahl von freien Erweiterungen nachträglich angereichert werden können. Solche Erweiterungen sind mit kleinem Aufwand auch von Programmierern nach Bedarf mit Hilfe der bereitstehenden API zu erstellen.

Die Programmiersprache, die hinter Mediawiki steht, ist PHP. Dadurch kann die Software auf jedem Apache Server, der PHP unterstützt, zum Laufen gebracht werden. Die Software ist web-basiert und für ihre Nutzung wird nur ein Webbrowser vorausgesetzt.

Die Inhalte, die die Benutzer einfügen, werden als HTML-Seiten angezeigt. Sie werden aber in den speziellen *"wikitext"* eingepflegt. Dieser Text wird vom Wiki-Parser mit Hilfe von Vorlagen in HTML transformiert. Dies ist nötig, um die Konsistenz der Inhalte-Darstellung zu gewährleisten.

Die Architektur von Mediawiki ist so gestaltet, dass eine Erweiterung und Anpassung des Systems durch die Systemadministratoren oder aber auch von dem registrierten Benutzer möglich ist. Die Möglichkeiten sind:

- externe Erweiterungen (Extensions) oder Skins. Dadurch ist es möglich, die Funktionalitäten des Systems beliebig zu erweitern und an die Anforderungen anzupassen.
- Site-wide Gadgets, JavaScript und CSS. Die Systemadministratoren können die Konfiguration des ganzen Systems zentral verwalten.
- durch JavaScript und CSS sind die registrierten und angemeldeten Benutzer des Systems in der Lage, ihre eigenen Seiten anzupassen.

Mediawiki bietet jedem Programmierer die Möglichkeit, durch Einfügen von eigenen Funktionalitäten das Mediawiki zu erweitern ohne im Kern etwas ändern zu müssen. Das ist durch das System der sogenannten "Hooks" möglich. Diese "Haken" erlauben es, zu verschiedenen Ereignissen während der Bearbeitung von Benutzeranfragen Funktionalitäten, die nicht im Mediawiki implementiert sind, auszuführen. Dadurch ist z.B. das Definieren von neuen Tags, die im Wiki Text eingefügt werden, möglich. Ein Beispiel dafür ist „<*references/*>"-Tag. Die Autoren der Artikel können mit diesem Tag Referenzen in die

Artikel einfügen, die am Ende der Seite, wie das in jeder wissenschaftlichen Arbeit der Fall ist, aufgelistet werden.

Neben den Erweiterungen, die den Parser beeinflussen, ist es auch möglich, eigene spezielle Seiten (Special Pages) zu definieren. Sie erlauben die Implementierung von Zusammenhängen und eine nicht triviale Logik. Auf dieser Basis sind viele große Erweiterungen, wie Semantic Media Bundle, entstanden. Sie setzen nicht nur auf Hooks, sondern brauchen auch weitgehende Funktionalitäten und sogar eigene Tabellen in der Datenbank.

5.2.2 Semantic Media Wiki Bundle

MediaWiki hat die Erfassung von Wissen in Form von Artikeln zum Ziel. Diese Artikel lassen sich vom Benutzer in Kategorien aufteilen, damit die Suche erleichtert wird. Die Kategorien lassen sich weiter in Unterkategorien aufteilen. Der Zugang zu der Information, die sich in einem auf MediaWiki basierten Wiki versteckt, ist für externe Anwendungen sehr schwer. Das folgt aus der Tatsache, dass die Repräsentation der Information maschinell nicht bearbeitet werden kann. Dadurch ist eine artikelübergreifende Suche nach Zusammenhängen nicht möglich. Aus diesem Grund ist das Semantic MediaWiki (SMW) entstanden. SMW ermöglicht dem Benutzer das explizite Einfügen von semantischen Informationen zum Inhalt von jedem Artikel. Dadurch sind Anwendungen in der Lage, die Zusammenhänge zwischen den Artikeln in einem Wiki zu erkennen. Die Ziele von SMW sind [Km07]:

- **Konsistenz der Inhalte**. Gleiche Informationen tauchen in verschiedenen Artikeln auf und SMW versucht, ihre Konsistenz zu bewahren.
- **Zugang zum Wissen.** Wikis wie Wikipedia enthalten Tausende von Artikeln. Das Finden von Informationen und der Vergleich von mehr als einem Artikel ist eine zeitaufwendige Aufgabe.
- **Wiederverwenden von Wissen.** Das Wissen in einem klassischen Wiki ist nur in einem Browser (oder einer Anwendung mit ähnlichen Funktionalitäten) zugänglich und kann nur von Menschen interpretiert werden. Eine automatisierte Inferenz ist dadurch nicht möglich.

Durch eine einfache Syntax können die Benutzer (ähnlich der im Wiki Text benutzten Konstrukte) semantische Information in die Artikel einfügen. Annotationen stellen nicht nur Links zu einem anderen Artikel dar, sondern dienen auch der Beschreibung der Relationen zwischen den Artikeln. In einem Wiki kann man z.B. in allen Artikeln über Länder Links zu den Artikeln der Hauptstädte einfügen. Annotiert man diese Links mit “Hauptstadt von”, sind dann durch die SMW Query-Sprache Anfragen möglich, die neben einer Liste aller Hauptstädte die dazugehörigen Länder enthält. Zusätzlich können noch Information über Anwohneranzahl usw. bezogen werden. Voraussetzung dafür ist, dass diese im Voraus von den Verfassern der Artikel eingepflegt wurden. Dadurch kann auf jeder Seite zusätzliche Information eingeblendet werden, die von anderen Artikeln stammt und relevante Fakten zum gegebenen Artikel darstellt. Die Semantik dieser Annotationen beruht auf OWL DL, wobei die semantische Information nicht das HTML-Dokument beschreibt, sondern die Absicht des

Inhalts. Diese Information ist durch die zur Verfügung gestellten API von extern zum Wiki gegebenen Anwendungen zugänglich.

Erweiterungen, wie SMW Forms, setzen auf die interne Schnittstelle, die von SMW innerhalb des MediaWiki benutzt werden kann. Die Erweiterungen, die für die Zwecke dieser Diplomarbeit entwickelt wurden, setzen auch auf diese Schnittstelle.

5.3 Implementierung des Pattern Wiki

Pattern Wiki ist eine MediaWiki basierte Software, die das Basissystem um weitere speziell für die Zwecke dieser Diplomarbeit benötigte Komponenten erweitert. Diese Komponenten setzen auf die Schnittstellen, die MediaWiki und SMW bereitstellen. Sie sind auch in anderen Wikis einsetzbar, da diese die standardisierten Erweiterungsmöglichkeiten, die MediaWiki bereitstellt, nutzen. Da die Komponenten auch auf die SMW Schnittstelle zugreifen, um Zusammenhänge zwischen den Mustern zu finden, ist eine Installation der SMW Software und das Einbinden in die MediaWiki-Instanz eine Voraussetzung für das richtige Funktionieren der erstellten Software.

5.3.1 Pattern Manager

Pattern Manager ermöglicht die Verwaltung der Muster innerhalb des Wikis. Diese stellt ein frei konfigurierbares Formular, mit dem sich Muster nach einer vorgelegten Struktur im Wiki einpflegen lassen. Die Verwaltung der Muster umfasst dabei das Einfügen von neuen Mustern und das Bearbeiten von bestehenden Mustern. Das Löschen von Mustern wird durch die Standardfunktion des MediaWiki “Delete Article” übernommen und nicht explizit in der Komponente implementiert.

Der Pattern Manager, sowie die drei nachfolgenden Komponenten, sind wie Wiki-Erweiterungen aufgebaut. Dieser Aufbau folgt immer der gleichen Vorgehensweise und ist in [Bd09] ausführlich beschrieben. Deswegen werden hier nur die Hauptaspekte zusammengefasst.

Die Erweiterungen in MediaWiki sind im Ordner *“Extensions”* zu finden. Diese sind in Unterordnern organisiert und der Eingangspunkt jeder einzelnen Erweiterung ist der Name des Ordners benannten “php”-Datei. So ist z.B. der Pattern Manager im Ordner *„PatternManager“* und dann in der darin enthaltenen *PatternManager.php*-Datei. Hierin sind alle globalen Variablen, die in der Funktion der Erweiterung gebraucht werden, definiert. Zudem sind hier auch die Hooks definiert, die aussagen, zu welchem Zeitpunkt die Funktionalitäten der Erweiterung gebraucht und ausgeführt werden sollen. Die weitere Organisation innerhalb dieses Ordners ist allein dem Entwickler überlassen. Er kann selbst bestimmen, welche Klassen er braucht und wie diese aufgebaut werden sollen. Einzig notwendig ist es, diese in der Hauptdatei der Erweiterung zu registrieren (Abb. 5.2), damit sie zugänglich sind. Nach diesem Schema ist auch der Pattern Manager aufgebaut.

```
$wgExtensionMessagesFiles['PatternManager'] = $dir . 'PatternManager.i18n.php';
$wgAutoloadClasses['PatternManager'] = $dir.'PatternManager_Body.php';
$wgAutoloadClasses['PatternForm'] = $dir.'PatternForm.php';
```

Abbildung 5.2 Auszug aus der PatternManager.php

Um Muster im Repository einzufügen oder bestehenden zu bearbeiten, wird ein Formular aufgestellt. Dieses lässt sich wie folgt konfigurieren:

```
$wgPatternManagerFormDescriptor =  array(
                'patternTitle' => array(
                        'section' => 'section0',
                        'label'=> 'Pattern title:',
                        'class' => 'HTMLTextField',
                        'validation-callback' => array('PatternForm', 'validatePatternTitle'),
                        'required' => true,
                        'semantic' => true,
                        'linkType' => 'Has title',
                ),
…
```

Abbildung 5.3 Konfiguration des Formulars

Das Formular beinhaltet alle Datenfelder, die in der festgelegten Struktur des Musters gebraucht werden. MediaWiki hat in der API vorimplementierte Klassen und Funktionen, die die Arbeit mit einem Formular übernehmen sollen. Was dabei nicht standardmäßig berücksichtigt wird, ist die semantische Bedeutung einiger der Felder (z.B. *"linkType"* ist *"Has question"*). Diese zusätzliche Information wird vom Pattern Manager benötigt, um die vom Benutzer eingegebenen Daten als semantische Annotationen in den Wiki Text einzufügen. Das erleichtert die Arbeit der Musterverfasser und somit entfällt die Notwendigkeit, das SMW Syntax zu lernen. Diese Aufgabe wird von der Software übernommen und passiert im Hintergrund. Die Abspeicherung der Benutzereingabe als semantische Information zu jedem Muster ist für den Benutzer transparent. Die von Media Wiki zur Verfügung gestellten Funktionalitäten befassen sich nur mit Standard HTML Formfeldern wie: Text, Auswahllisten, Optionsfelder usw. Das Erstellen eines neuen Musters setzt zusätzliche Informationen über andere Muster, Linktypen usw. voraus. Diese sind dann nicht mehr einfache HTML Felder, sondern Kombinationen von solchen, die nur zusammen einen Sinn ergeben. Ein solches Feld ist das Definieren einer Relation zu einem anderen Muster. Dazu braucht man den Typ der Relation und den Namen des Musters. Aus diesem Grund wurden zwei zusätzliche Klassen von Feldern eingefügt. Das sind *TagField* und *PatternRow*.

TagField dient dem Einfügen von Schlagwörtern zu jedem Muster. Beim Erstellen eines neuen Musters sind Metainformationen über seinen Kontext einzufügen. Damit der Benutzer schon vorhandene und in anderen Mustern angegebene Tags schnell herausfindet, wird beim Eintippen in ein solches Feld eine Liste mit Vorschlägen gemacht, die sich ständig aktualisiert. Dadurch bekommt der Benutzer fortlaufend Feedback vom System, ob ein Schlagwort schon eingesetzt wurde oder nicht. Eine solche Funktionalität erfordert auch eine benutzerdefinierte Überprüfung der angegebenen Information und eine andere Darstellung als

die von HTML standardmäßig angebotene. Zusätzlich werden auch auf der Client Seite (im Browser) noch Funktionalitäten gebraucht, die diese Darstellung und das Verhalten des Felds steuern (Abb. 5.4). Das ist durch das Einbinden von JavaScript möglich, welches diese Aufgaben im Browser übernimmt.

Abbildung 5.4 Feld für Schlagwörtereingabe

Das *PatternRow* Feld ist eine Kombination aus zwei Auswahllisten (Drop-Down-List). In diesem Fall werden wieder dynamische Informationen gebraucht. Zum einen wird der Linktyp gebraucht. Dieser wird in der Konfigurationsdatei festgelegt:

```
----------------------------------------------------------------------------------------------------
//pattern link types
$wgGlobalPatternManagerPatternLinkTypes = array('Can be used with','Is similar to');
----------------------------------------------------------------------------------------------------
```

Abbildung 5.5 Auszug aus PatternManager.php

Zum anderen wird die Liste mit den Musternamen gebraucht, die aus der Wiki- Datenbank bezogen wird (Abb. 5.6).

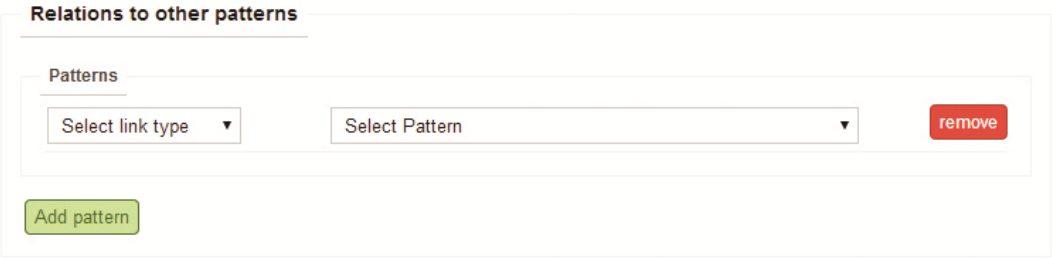

Abbildung 5.6 Einfügen von Relationen

Eine weitere Besonderheit hier ist, dass es möglich ist, mehrere Relationen zu jedem Muster einzufügen. Solche Fälle sind nicht Gegenstand einer API und wurden zusätzlich erstellt.

Die Konfiguration des Formulars ist in der Konfigurationsdatei der Erweiterung festgelegt und leicht zugänglich. Die Administratoren können diese leicht an die Anforderungen jeder Mustersprache anpassen[10]. Zu beachten ist, dass die Felder, die semantische Annotation darstellen, den "*linkType*" und den dazugehörigen Wert brauchen. Der Relationen-Teil des Formulars lässt sich nicht ausblenden. Die Administratoren haben weder Einfluss auf die Darstellung noch auf die Position in der Struktur des Formulars, weil für die Existenz jeder Mustersprache Relationen zwischen den Mustern eine notwendige Bedingung sind. Was sich an dem Feld ändern lässt, ist die Liste der *Link Typen* (siehe Abb. 5.3) Das *TagField* dagegen lässt sich oft einsetzen oder auch entfernen.

[10] Ausführliche Information über die Beschreibung der einzelnen Felder findet man auf der Media Wiki Seite: http://www.mediawiki.org/wiki/HTMLForm

Mittels dieser Komponente lassen sich neue Muster in das Repository mit semantischer Annotation bereichern und einfügen. Die Komponente ermöglicht auch die Bearbeitung von schon im Repository eingefügten Mustern. Aus dem Wiki Text der Seite (des Musters) werden alle relevanten Informationen extrahiert und an die richtigen Stellen in das Formular eingefügt. Nachdem die Änderungen vom Verfasser (Benutzer) gemacht worden sind, wird das Muster wieder im Repository gespeichert. Bei einer seitens der Verfasser der Sprache und der Systemadministratoren sorgfältig und genau überlegten Struktur des Musters wird dem Benutzer ein bequemes Werkzeug in die Hände gelegt, mit dem sich mit wenigen Klicks ein neues Muster im Repository ablegen lässt oder schon bestehende geändert werden können.

5.3.2 Search Manager and Recommender system

Der Search Manager ist als MediaWiki Erweiterung erstellt und folgt dem in 5.3.1 beschriebenen Konzept beim Aufbau. Die benötigten Dateien befinden sich im Ordner *"PatternSearch"*. Diese Erweiterung hat eine komplexere Aufgabe zu bewältigen und vereint das Konzept der Mustersuche, sowie das Konzept des Recommender Systems. Die Suche nach dem Einstiegspunkt kann als eigenständiges System funktionieren, wobei zu bemerken ist, dass das Konzept der Suche „Problembeschreibung", "Kontextbeschreibung" und „Absichtsbeschreibung" voraussetzt. Diese drei Komponenten werden mittels „PatternManager" zu jedem Muster als semantische Information eingefügt und sind Pflicht. Sie werden auch vom Recommender System vorausgesetzt. Die Informationsabhängigkeit von der Suchgeschichte des Benutzers erlaubt keine Existenz außerhalb des Suchmanagers ohne eine neue Implementierung oder Anpassung der Datenbankstruktur.

Wie schon in Kap. 4.2.2 erläutert wurde, ist diese Suche in mehrere Schritte aufgeteilt. Diese Schritte benötigen verschiedene semantische Informationen und werden mittels SMW API anhand der Eingabe des Benutzers dynamisch von dem im Repository abgelegten Muster extrahiert. Die Logik, die jeden einzelnen Schritt begleitet, ist spezifisch und deswegen von der Logik der anderen Schritte getrennt gehalten.

Die einfachste Aufgabe übernimmt Schritt 1 (Abb. 5.7), weil nur eine Auflistung der Kategorien der Muster erwartet wird. Beim Bearbeiten der vom Benutzer angekreuzten Kategorien wird eine Instanz der Klasse *PatternCollection* erstellt (Abb. 5.7) und mit der Liste der Kategorien gefüllt. *PatternCollection* (Abb. 5.7) ist die Klasse, die als Datenstruktur zum Aufbewahren der Informationen dient, die vom Benutzer eingegeben und zusätzlich von der Datenbank gefordert werden. Damit diese Information in allen Schritten benutzt werden kann, wird sie mit Hilfe von Session-Variablen über alle Requests übergeben. Wählt der Benutzer keine Kategorie, werden in den nachfolgenden Schritten alle Kategorien betrachtet. Sonst werden nur die Muster in den ausgewählten Kategorien in den nächsten Schritten in Betracht gezogen (Abb. 5.7).

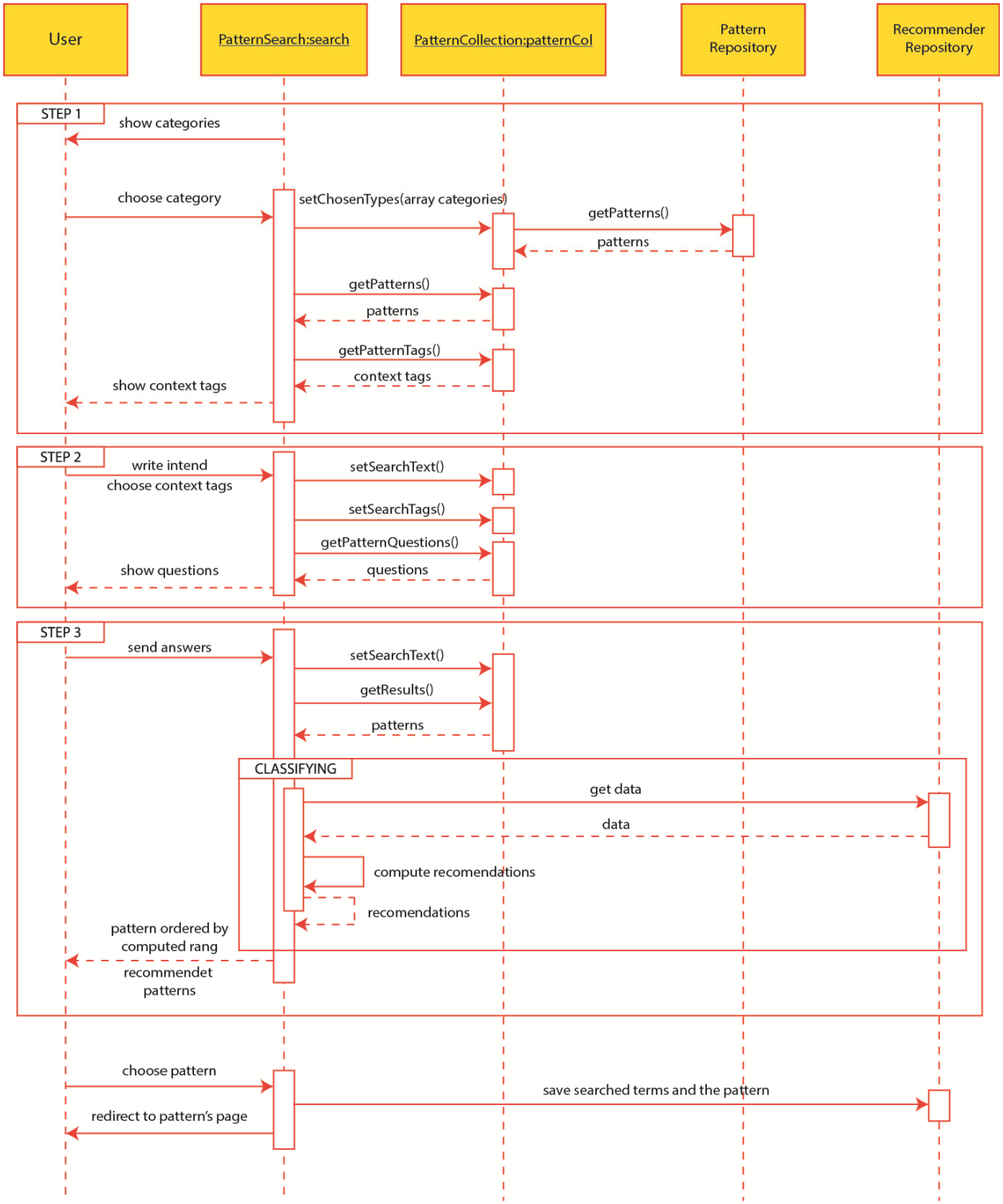

Abbildung 5.7 Sequenzdiagramm des Search Managers

Im zweiten Schritt wird die semantische Information in den Mustern in ähnlicher Weise zum Generieren der Inhalte des jeweiligen Formulars benutzt. In Schritt drei werden dafür die Fragen benötigt, die als Annotation zum Property *"Has question"* stehen. Nachdem alle drei Schritte vom Benutzer absolviert wurden, werden die Koeffizienten der einzelnen vorausgewählten Muster anhand der erhaltenen Informationen berechnet.

Die Muster werden dann nach dem Rang geordnet und für den Benutzer zur Auswahl aufgelistet. Entscheidet sich dieser für ein Muster und klickt auf den gegebenen Link, werden die im Schritt zwei ausgewählten Tags und der angegebene freie Text, im Schritt drei die ausgewählten Fragen und die in der Browse-Ansicht am Ende der Suche angeklickten Muster im Recommender Repository gespeichert (Abb. 5.8). Diese Information dient als Grundlage der Berechnungen, die vom Recommender System durchgeführt werden (Abb. 5.10).

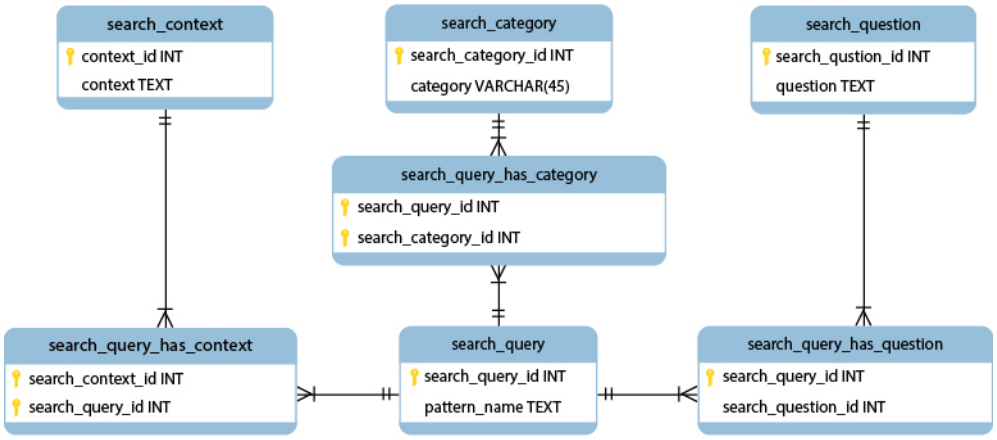

Abbildung 5.8 Recommender System Repository-Struktur

Die so gespeicherte Information dient dazu, nach dem dritten Schritt beim Generieren der Liste mit den Mustern, unten auf die Seite eine Liste mit Empfehlungen einzublenden. Das Recommender System implementiert ein Bayes Model und berechnet die Wahrscheinlichkeit, welches Muster für den Benutzer anhand der in den drei Schritten angegebenen Information am relevantesten im Vergleich zu den in der DB gespeicherten Suchgeschichte ist. Der Algorithmus ist in der Datei PatternSearch_Body.php implementiert:

```
-----------------------------------------------------------------------------------------------------------
function classify(){
                $table = '_search_query';
                $classes = array();
                $classesCounts = array();
                $probClasses = array();
                $argmax = array();
                //get the db object
                $dbr = wfGetDB( DB_SLAVE);
                //get all query results
                $query = array('pattern_name');
                $var = array('') ;
                $result = $dbr->select($table, 'pattern_name', '', '', 'DISTINCT');
                foreach($result as $row){
                        array_push($classes, $row->pattern_name);
                }

                //get count of the rows in the search_query table
                $count = $dbr->select($table, 'count(search_query_id) as num');
                foreach($count as $c){
                        $rowCountclasses = $c->num;
                }

                // find total number of every class
                foreach($classes as $class){
                        $options = ("pattern_name LIKE '{$class}%'");
                        $classCount = $dbr->select($table, 'count(pattern_name) as num ', $options);

                        foreach($classCount as $c){
                                $classesCounts[$class] = $c->num;
                        }
```

```
                }

                //calculate the probability of the classes
                foreach($classesCounts as $class=>$count){
                        $probClasses[$class] = round($count/$rowCountclasses, 4);
                        $argmax[$class] = 1;

                }

                $checkedItems = array();
                $chosenTypes = $this->patternCol->getChosenTypes();
                foreach($chosenTypes as $type=>$value){
                        foreach($value as $category){
                                $checkedItems['category'][] = $category;
                        }
                }

                $patterns = $this->patternCol->getPatterns();
                foreach($patterns as $pattern){
                        //get the tags
                        $tags = $pattern->getCheckedTags();
                        if(count($tags) > 0){
                                foreach($tags as $t){
                                        $checkedItems['kontext'][] = $t;
                                }
                        }
                        //get the questions
                        if($pattern->isQuestionChecked()){
                                $checkedItems['question'][] = $pattern->getQuestion();
                        }
                }
                foreach($classesCounts as $c=>$p){
                        foreach($checkedItems as $item => $value){
                                //$item is tag, category or question
                                //$value is array
                                $tableSearch = '_search_query';
                                $tableName = '_search_'.$item;
                                $tableHasItem = '_search_query_has_'.$item;
                                foreach($value as $i){
                                        $query = 'count(s.'.$item.'_id) as num';
                                        $tables = array( 's' => $tableName, 'sq' => $tableSearch,
'sqh' => $tableHasItem);
                                        $options = array( "sq.pattern_name LIKE '{$c}%'",
                                        "s.{$item} LIKE '{$i}%'",
                                        "s.".$item."_id = sqh.".$item."_id",
                                        "sq.search_query_id = sqh.search_query_id"
                                                        );
                                        $resultItem = $dbr->select($tables, $query, $options);

                                        foreach($resultItem as $r){
                                                $itemProb = $r->num;
                                        }
                                        $pI[$c][$item] = round($itemProb/$p, 2);

                                        if( $pI[$c][$item] !=0 ){
                                                $argmax[$c] = $pI[$c][$item];
                                        }


                                }
                        }
                        $argmax[$c] *= $probClasses[$c];

                }
                $sum = 0;
                foreach($argmax as $key=>$value){
                        $sum += $value;
                }
                return $argmax;
        }
```

Abbildung 5.9 Klassifikationsverfahren

Dadurch fließt die Information, die bei jeder Suche generiert wird, als Feedback zurück und ergänzt die Suche. Die Interessierten können dann entweder einen Vorschlag von der tatsächlichen Suche auswählen oder sich an der Auswahl, die bei einer ähnlichen Eingabe andere Benutzer getroffen wurden, bedienen.

5.3.3 Solution Manager

Der Solution Manager implementiert die letzten Funktionalitäten, die das System komplett machen. Mit ihm sind die Benutzer des Pattern Wiki in der Lage, neue Lösungswege zu erstellen. Der Solution Manager ist nur für registrierte Benutzer des Systems zugänglich. Alle Anderen sind in der Lage, sich die erstellten Lösungen anzusehen, aber das Erstellen und das Bearbeiten von Lösungen ist ihnen nicht erlaubt.

Bei der Implementierung wurden die Anforderungen an den Solution Manager, wie sie am Anfang des Kapitels beschrieben wurden, berücksichtigt. Beim Konzept des Managers wurde großer Wert auf die Bedienbarkeit und die Benutzerfreundlichkeit des Moduls gelegt. Die Natur des Erstellens einer Lösung unterscheidet sich im Wesentlichen von dem eines Musters dadurch, dass die Benutzer sich zuerst die passenden Muster aussuchen müssen. Dann werden diese in einer bestimmten (oder mehrerer) Reihenfolge(n) angewandt. In dieser Version des Solution Managers besteht keine Möglichkeit, diese Reihenfolge abzuspeichern und dies wird somit Objekt künftiger Versionen des Managers. Nachdem der Benutzer sich entschieden hat, eine neue Lösung zu erstellen und den Solution Manager aktiviert, erscheint auf jeder Muster-Seite die Leiste aus Abb. 5.10.

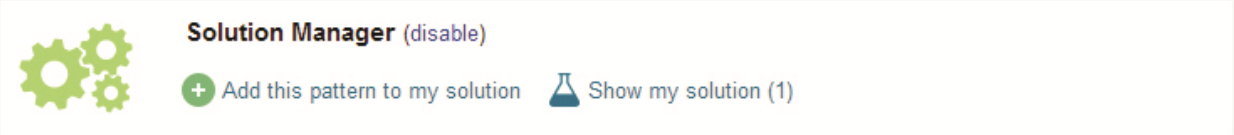

Abbildung 5.10 Solution Manager Leiste

Dadurch kann der Benutzer des Systems zu jeder Zeit ein neues Muster zu seiner Lösung hinzufügen oder entfernen. Es ist auch möglich direkt zur Lösung zu gehen und diese dann zu vervollständigen und zu speichern.

Der Solution Manager ist wie die im Kapitel 5.4 und 5.5 beschriebenen Erweiterungen aufgebaut. Er ist als spezielle Media Wiki Seite aufgebaut und hat zusätzlich ein Plug-In zum Einblenden der Leiste (Abb. 5.10) und die dazugehörige Logik zum Steuern der Eingaben des Benutzers. Dabei wird ein “noticeAfter” Haken (Abb. 5.11, Abb. 5.12), vom Media Wiki zur Verfügung gestellt und beim Rendern der HTML-Seite ausgeführt.

```
-----------------------------------------------------------------------------------------------------
static function siteNoticeAfter( &$siteNotice, $skin = null){
                //global variable with the namespace
                global $wgRequest,$wgTitle, $wgSolutionPortletForLoggedUsersOnly, $wgUser;
                $title = $wgTitle;
                $request = $wgRequest;

                if( !$wgSolutionPortletForLoggedUsersOnly || !$wgUser->isLoggedIn()){
                                        return true;
                        }
```

```
                $wiki = new WikiPage($title);
                //if there are no categories in the page exit
                if(count($title->getParentCategories()) == 0 ){
                        return true;
                }
                //if the page is in more than one catogory
                //walk through the categories and check if Category:Pattern is there also
                $inArray = false;
                foreach($title->getParentCategories() as $key=>$value){
                        if($key == "Category:Pattern"){
                                $inArray = true;
                        }
                }

…
                $patterns = $session->getPatterns();
                //if the patter is in the solution already, then show the remove button
                if(in_array($title->getPrefixedText(), $patterns)){
                        $ajaxHint = 'removePattern';
                }
                //render the html
                $siteNotice .= self::renderSolutionBox($ajaxHint, $title);

                return true;

        }
-------------------------------------------------------------------------------------------------
```

Abbildung 5.11 NoticeAfter Hook in PatternSolutions_Hooks.php

Sobald alle Muster, die zu einer Lösung gehören, eingefügt wurden, kann der Benutzer diese noch mit einer Beschreibung und einem eindeutigen Namen versehen und im Repository speichern. Da die Namen der Muster gleichzeitig Identifikator der Seiten sind, dürfen diese sich nicht wiederholen. Damit eine solche Wiederholung der Namen leicht umgegangen wird, wird per AJAX-Anfrage auf die spezielle Solution Manager Seite beim Eintippen im Feld "Title" auf dem Server überprüft, dass der Name nicht existiert und einzigartig ist. Die Beschreibung des Musters nach Konzept im Kap. 3.3 hat keine eindeutige Struktur. Deswegen wird hier nur ein Textfeld bereitgestellt in dem Wiki Text eingefügt wird. Im Feld ist Wiki Text erlaubt, weil die Lösungen dann als Wiki Seiten im Repository angelegt werden.

Nachdem die Informationen, die vom Verfasser des neuen Musters auf dem Server überprüft und die neue Seite angelegt wurden, erfolgt eine Weiterleitung zu dieser neuen Seite. Da in mehreren Situationen eine Reihe von zusätzlichen Informationen und Daten gebraucht werden, benötigt der Benutzer eine Speichermöglichkeit. Wegen des allgemeinen Charakters dieser Funktionalität wurde auf eine schon im offiziellen Erweiterungsrepository vom Media Wiki vorhandene Erweiterung Bezug genommen. Sie heißt *"FileList"*. Diese kann so konfiguriert werden, dass nur bestimmte Benutzer die hochgeladenen Dateien verwalten dürfen. In Pattern Wiki dürfen nur die Verfasser der Lösungen diese Dateien ändern. Alle anderen Benutzer haben nur Lesezugriff. Das ist nötig, da nur derjenige, der die Lösung erstellt hat, sich mit dem Problem auskennt, von welchem die angehängten Dateien stammen.

Diese letzte Komponente macht das System vollständig und nach einer Konfiguration auch für jede Mustersprache einsatzbereit. Beachtet man die technischen Anforderungen, kann der Pattern Manager für jede beliebige Mustersprache angepasst werden und mit dieser wird

immer eine Konsistenz der Struktur und Darstellung der Muster gewährleistet. Der Search Manager und das Recommender System sind an neue Mustersprachen anpassbar. Das Konzept der Suche spielt eine wichtige Rolle und muss immer beachtet werden. Die Kontextbeschreibung mittels Tags, Problembeschreibung mittels Fragen sowie die Klassifizierung der Muster in den Sprachen sind Konstrukte, die in jeder Mustersprache eine wichtige Rolle spielen. Dadurch lässt sich das abstrakte Problem, das in jedem Muster dargestellt wird, in einer einfachen und von jedermann nachvollzierbaren Weise ableiten. Deswegen liegt diese Tatsache im Kern der Suchmanager und kann in jeder beliebigen Mustersprache angewendet werden. Das gilt auch für den Solution Manager. Im nächsten Kapitel wird anhand der Green Business Patterns Language ein praxisbezogenes Beispiel gezeigt, wie eine Mustersprache als Basis für die gerade vorgestellte Software vorbereitet werden kann. Im Anschluss wird die Konfiguration der Software gezeigt und einsatzbereit gemacht.

6 Erstellung des Green Pattern Wiki

In diesem Kapitel wird anhand der Green Business Process Pattern (GBPP) [NL11] [NL12] [NL14] [NaL13] – Mustersprache gezeigt, wie eine Mustersprache für das Pattern Wiki vorbereitet werden kann. In Kap. 6.1 wird die Analyse der Sprache anhand der in Kap. 3 gesetzten Anforderungen durchgeführt. Dann wird in Kap. 6.2 gezeigt, wie diese Information bei der Konfiguration eines Pattern Wiki eingesetzt werden kann.

6.1 Analyse der GBPP Mustersprache

Die Analyse, wie in Kapitel 3 gezeigt wurde, dient dazu, die Mustersprache und seine wesentlichen Teile zu identifizieren. Diese werden dann als Basis für die Konfiguration der Green Pattern Wiki Software gebraucht, um diese voll funktionsfähig zu machen.

6.1.1 Struktur der Green Business Process Patterns

Die Muster in jeder Sprache brauchen eine strikt definierte Struktur. Wie in Kapitel 2 gezeigt wurde, werden die Muster in der Sprache konsistent aufgebaut, was die Arbeit mit den Mustern erleichtert. Die Green Business Process Patterns folgen einer festgelegten Struktur und jedes Muster besteht aus:

- **Bild.** Bild, das die Grundidee des Musters visuell erfasst
- **Absicht.** In einem Satz wird die *Absicht* des Musters erläutert
- **Kontext.** In welchem *Kontext* kann das Muster angewendet werden
- **Problemstellung.** Die *Problemstellung* beschreibt die Ausgangssituation im Unternehmen und die existierenden Probleme
- **Lösung.** Die *Lösung* zeigt, wie das Muster im gegebenen Kontext passt und wie sich das Muster anwenden lässt
- **Lösungsskizze.** Die *Lösungsskizze* stellt einen Vergleich der Ausgangssituation und der geänderten Situation dar, nachdem das Muster angewendet wurde.
- **Ergebnis**. Beschreibt die neue verbesserte Situation, nachdem das Muster angewendet wurde
- **Beziehungen zu anderen Mustern**. In diesem Teil werden die Relationen zu den anderen Mustern beschrieben.

Die Definitionen der Muster findet man in [NL11] [NL12] [NL14] [NaL13]. Die Sprache bestand im Moment des Schreibens dieser Diplomarbeit aus 23 Mustern. Diese sind: Green Client Session, Green Compensation, Green Server Session, Green Eventual Consistency, Green Lazy Load, Green Data Transfer Object, Common Process Improvement for Environmental Aspects, Green Control Flow, Green Feature, Human Process Performance, Insourcing, Outsourcing, Green Cancel Activity, Green Explicit Termination, Green External Choice, Green Batch Processing Component, Green Shared Component, Green Loose Coupling, Green Public Cloud.

6.1.2 Klassifizierung der Muster

Die im vorherigen Kapitel angegebenen Muster sollten für ihren Einsatz im Green Wiki

kategorisiert werden. Die GBPP sind Muster, die im Kontext der Geschäftsprozessoptimierung entstanden sind. Die Geschäftsprozesse bestehen aus Aktivitäten, die in einer festgelegten Reihenfolge ausgeführt werden, um ein bestimmtes Ziel zu erreichen [Oa96]. Die einzelnen Aktivitäten können auch andere Prozesse sein. Die Aktivitäten haben als Input Daten, die sie bearbeiten und dann an die nächste Aktivität in der Prozesskette weitergeben. Sind die Aktivitäten als Web Services realisiert, laufen sie in der Regel auf verschiedenen Rechnern. Dieser Aufbau und das Verhältnis der Aktivitäten zwischen einander führt bei den Workflow Patterns, die auch als Grundlage der GBPP dienen, zur folgenden Kategorisierung: Control, Ressource, Data, Exception Handling und Präsentation [WP]. Die Workflow Patterns stellen verschiedene Konstrukte dar, die in den meisten BP Modelling Languages vorkommen. Bei der Klassifizierung der Muster in der GBPP Mustersprache wurden drei Gruppen von Mustern identifiziert: *Data, Activity und Service* (Abb. 3.2). Sie umfassen die Hauptaspekte jedes Geschäftsprozesses.

***Data*.** Unter Data sind alle Informationen zu verstehen, die während der Ausführung des Business Prozesses innerhalb der Aktivitäten bearbeitet und untereinander austauscht werden. Wichtig dabei ist, wie auf diese Daten zugegriffen wird, wo die Daten sich befinden und wie diese Daten strukturiert sind. Diese drei Aspekte haben einen großen Einfluss auf die Rechenleistung bei der Bearbeitung und beeinflussen dadurch den Stromverbrauch und den Umwelteinfluss des Prozesses. Diese drei Aspekte sollen die Hauptkategorie verfeinern. Daraus wurden die folgenden drei Unterkategorien identifiziert: *Access*, *Source* und *Structure*. Sie sind wie folgt definiert:

- **Access.** Der Zugriff auf die Daten, die eine Aktivität bearbeitet, kann auf verschiedenste Weise erfolgen. Falls zu oft auf die Daten zugegriffen wird, erhöhen sich z.B. der Stromverbrauch und dadurch die Ökobilanz des Prozesses. Die Muster, die dieser Gruppe zugeordnet sind, versuchen, den Zugriff so zu gestalten und umzubauen, dass dieser optimal verläuft und dadurch der Energiebedarf minimiert wird. Die Muster, die diesen Zugriff regeln, sind: *Green Client Session State, Green Compensation, Green Gateway und Green Server Session State*
- **Source.** Neben dem Zugriff ist die Bereitstellung der Daten ein wichtiger Aspekt. Die Muster in dieser Unterkategorie gehen der Frage nach, wie die Datenquellen sich umbauen lassen, damit der Zugriff auf sie verbessert wird. Sie haben als Ziel, die Verwaltung der Datenquellen zu regeln. Darin enthalten sind die folgenden Muster: *Green Eventual Consistency, Green Lazy Load*
- **Structure.** Neben dem Zugriff und der Quelle spielt die Struktur der Daten eine große Rolle bei ihrer Bearbeitung. Die Muster in dieser Untergruppe beschäftigen sich mit der Frage, wie sich die Struktur der Daten verbessern lässt, damit ihre Bearbeitung schneller erfolgen kann. Dieser Untergruppe sind alle Muster zugeordnet, die die Struktur der Daten verbessern: *Green Data Transfer Object*

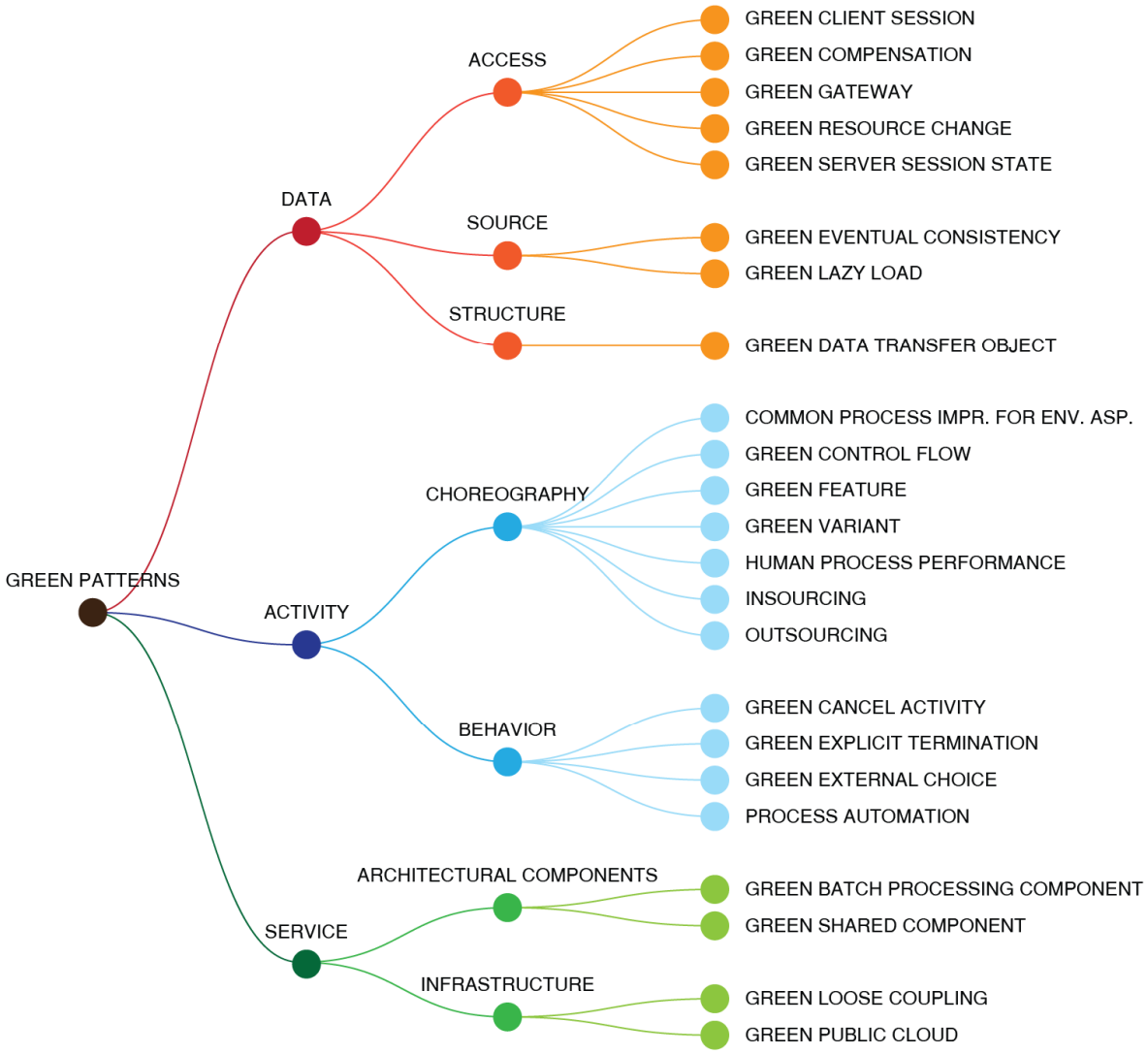

Abbildung 6.1 Kategorisierung der Muster in der GBPP Mustersprache

Activity. Die Aktivitäten sind die einzelnen Bausteine jedes Geschäftsprozesses. Die Workflow Pattern definieren Muster, die die Zusammenhänge zwischen den Aktivitäten regeln. Sie sind auf einer sehr feinen Granularität ausgelegt, die für die Zwecke der Mustersprache nicht geeignet ist. Die Verfeinerung im Fall der GBPP Mustersprache macht nur von zwei Hauptaspekten Gebrauch: Choreography und Behaviour. Sie sind wie folgt definiert:

- **Choreography.** Die Kategorie *Choreographie* enthält alle Muster, die die mögliche Neuordnung der Aktivitäten innerhalb des Business Prozesses beschreiben. Dabei können die Reihenfolge optimiert, Aktivitäten zusammengefasst, entfernt oder durch andere, die eine bessere Ökobilanz ausweisen, ersetzt werden. Die Muster in dieser Kategorie sind: *Common Process Improvement for Enviromental Aspects*, *Green Control Flow*, *Green Feature, Green Variant, Human Process Performance, Insourcing, Outsourcing*
- **Behavior.** Das Verhalten der Aktivitäten kann auch eine wesentliche Rolle bei der

Optimierung spielen. Zum Beispiel können die Aktivitäten, die ihre Aufgabe erledigt haben, einfach ausgeschaltet werden und dadurch wird Strom gespart. All diese Aufgaben übernehmen die Muster *Green Cancel Activity, Green Explicit Termination, Green External Choice, Process Automation*

Service. Nicht nur die Daten und die Aktivitäten tragen dazu bei, ein Prozess „grüner" zu gestalten, sondern auch die Infrastruktur, auf der die Prozesse laufen. Verschiedene Ansätze, wie z.B. Cloud Computing, haben sich in den letzten Jahren als wichtige Technologien zum Reduzieren des CO_2-Ausstoßes bewiesen. Deswegen orientieren sich die Muster in dieser Kategorie eng an Cloud Computing. Sie beschäftigen sich mit dem Optimieren der Architektur der Software sowie der Infrastruktur, auf der die Geschäftsprozesse laufen.

- **Architectural Components**. Die Unternehmen, die Anwendungen an mehrere Kunden zur Verfügung stellen, sind immer mit dem Problem konfrontiert, wie sich der Zugriff auf die Anwendung besser verteilen lässt. Cloud Computing stellt diesbezüglich eine Lösung dar, indem Ressourcen on Demand bereitgestellt werden. Nur dadurch ist eine dynamische Anpassung an große Nachfragen möglich. Die Muster sind: *Green Shared Component, Green Batch Processing Component*
- **Infrastructure.** Die Infrastruktur, auf der die Prozesse laufen, ist eine der wichtigsten Voraussetzungen, um die Prozesse „grüner" zu gestalten. Ist die Infrastruktur nicht richtig ausgelegt, dann sind die oben genannten Muster auch nicht in der Lage, den negativen Umwelteinfluss des Prozesses zu verbessern. Die bessere Auslastung der Rechenkapazitäten spielt in fast jedem Unternehmen eine wichtige Rolle und ist auch bei der Optimierung von großer Bedeutung. Diese Unterkategorie enthält die Muster, die sich bei der Verbesserung der Auslegung der Infrastruktur anwenden lassen: *Green Loose Coupling, Green Public Cloud*

6.1.3 Beziehungstypen

Die GBPP Mustersprache setzt auf allen Grundprinzipien, die von Alexander definiert wurden, auf, die Einfluss auf die Relationen zwischen den Mustern ausüben. Geschäftsprozesse unterscheiden sich in jedem Unternehmen und man kann sie nicht standardisieren. In der Softwareentwicklung wird z.B. bei der GUI-Entwicklung das MVC-Muster in jeder Anwendung gebraucht. Die Softwareentwickler kennen das Muster und setzen es immer ein. Dieses Muster kann man nicht nur in einer Programmiersprache anwenden, sondern es lässt sich in fast allen existierenden Sprachen implementieren. Beim Optimieren von Geschäftsprozessen dagegen ist nicht zu erwarten, dass ein solches Muster existieren kann, weil sich die Anforderungen an die Geschäftsprozesse in jedem Unternehmen unterscheiden. Die Geschäftsprozesse bringen Wettbewerbsvorteile für das jeweilige Unternehmen und sind speziell zugeschnitten. Wenn versucht wird, diese Prozesse „grüner" zu gestalten, kann eine exakte Lösung des Problems nicht vorgegeben sein. Die Muster in der GBPP Mustersprache weisen aus diesem Grund keine konkrete Reihenfolge auf, was die Anwendung dynamischer macht, aber die Suche nach dem Einstiegspunkt erschwert.

Beim Studieren der Muster und der Relationen zwischen ihnen wurden zwei Typen identifiziert: *Can be used with (kann auch mit ... eingesetzt werden)* (Abb. 6.2) *und Is similar to (Alternative)* (Abb. 6.3).

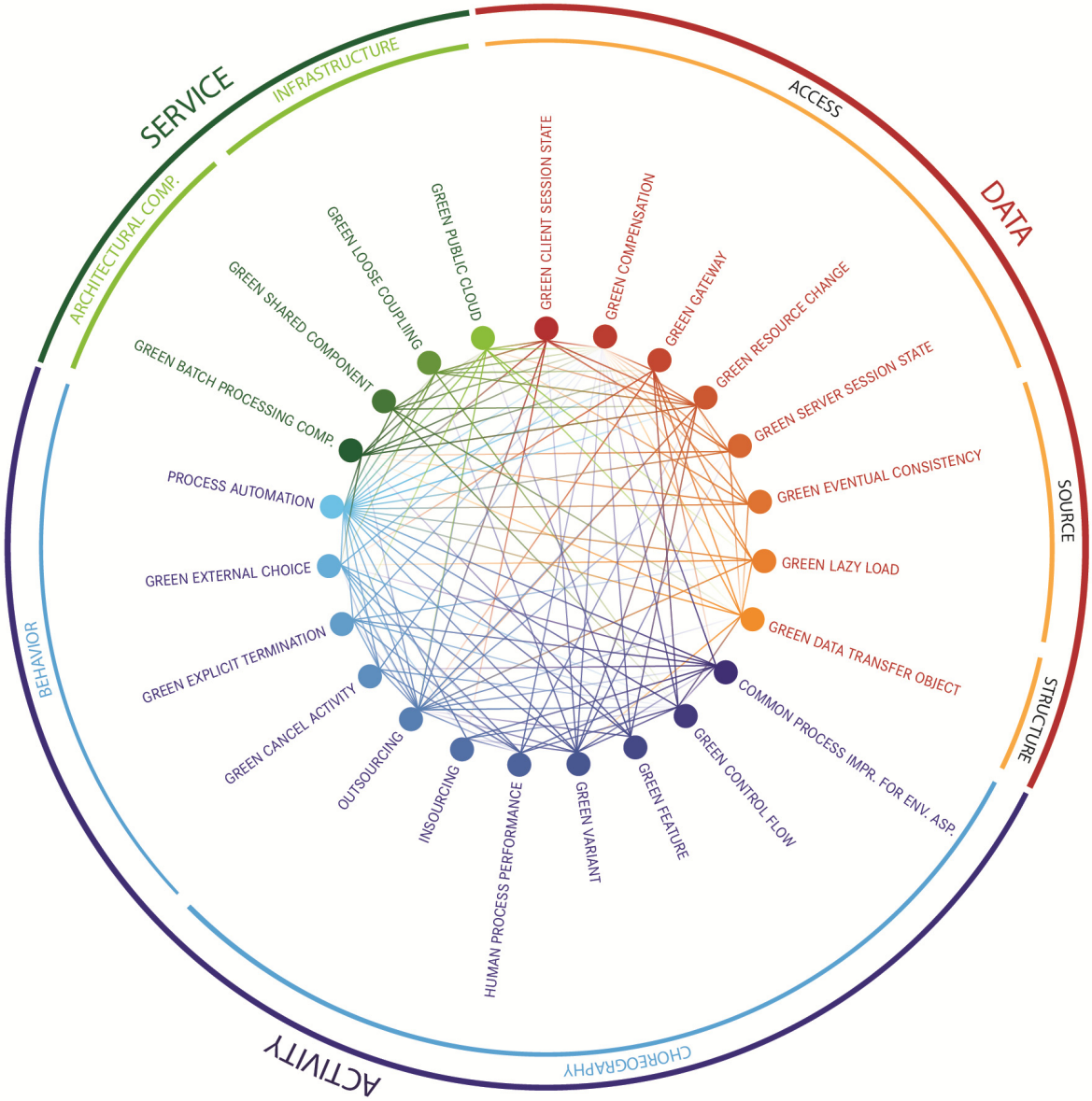

Abbildung 6.2 „Can be used with“ - Relationen zwischen den Mustern in GBPP Mustersprache

Can be used with

Die Relation “Can be used with”, wie auch die Relation “Alternative”, wurden aus den “Relationen zu anderen Mustern” abgeleitet. Abb. 6.2 zeigt eindeutig, dass diese Relation eine große Rolle innerhalb der Mustersprache spielt. Das Fehlen einer strikten Hierarchie in der Art und Weise, wie sie bei Alexander beschrieben ist, macht die Sprache sehr dynamisch und vielseitig. Der Name der Relation “Can be used with” verdeutlicht diese Tatsache. Ein Muster muss nicht unbedingt von einem anderen gefolgt werden, um ein Problem zu lösen. Man kann immer wieder verschiedene Wege nehmen und die Restriktionen, die bei der GoF-

Mustersprache zu sehen sind, fallen hier aus. Die Natur der Probleme, die die Mustersprache abdeckt, erlaubt es nicht, dass solche Restriktionen existieren. Um den negativen Umwelteinfluss zu mildern, kann sich der Interessierte genau die Muster auswählen, die die Situation im Unternehmen betreffen. In einem Unternehmen kann man die Chorgeographie der Aktivitäten ändern, in einem anderen kann das unerwünscht oder sogar verboten sein. Deswegen wird von der Sprache sehr viel mehr Flexibilität erwartet als das bei den Mustersprachen von Alexander oder GoF der Fall ist.

Alternative

Die Relation *"Is similar to (Alternative)"* ist von den "Relationen zu anderen Mustern" in der Musterstruktur abgeleitet. Sie zeigt, dass zwei Muster das gleiche oder ähnliche Problem lösen. Die Muster "Green Compensation" und "Green Variant" lösen z.B. die gleiche Problemgruppe und können ausgetauscht werden. Diese Verbindung ist aber nicht bidirektional, sondern zeigt nur von Muster X zu Muster Y. Es existieren mehrere "Alternative zu" Verbindungen zwischen verschiedenen Mustern. Die Muster, die sich aber auf der linken Seite der Relation befinden, kann man nicht austauschen. Wäre eine solche Möglichkeit gegeben, dann hätte das zu bedeuten, dass von "X ist Alternative zu Z", "Y ist Alternative zu Z" und "Z ist Alternative zu Y" auch "X ist Alternative zu Z". In den Mustern, die dieser Diplomarbeit zugrunde liegen, existiert eine solche verkettete Verbindung nicht (Abb. 6.3). Das bedeutet aber nicht, dass bei einer zukünftigen Erweiterung der Sprache eine solche Relation ebenfalls nicht möglich wäre.

6.1.4 Kontext eines Musters

In der von Nowak vorgeschlagenen Struktur enthält jedes Muster nach dem Ikon und der Absicht einen Teil mit dem Namen "Kontext". In diesem Teil wird der Kontext beschrieben, in dem das jeweilige Muster eingesetzt werden kann. Diese Information kann beim Identifizieren des gesuchten Kontexts helfen, ist aber nach Meinung des Autors dieser Diplomarbeit nicht ausreichend. Deswegen sollte man sich die zwei Teile „Problemstellung" und „Lösung" genauer anschauen. Dort finden sich wichtige Hinweise, die die Abgrenzung des Kontexts veranschaulichen können.

Problemstellung

Die Problemstellung befasst sich mit einer allgemeinen Beschreibung des Problems, das das Muster löst. So ist z.B. im Muster "Green Compensation" in diesem Teil zu lesen:

"...Bei der Integration ökologischer Zielvorgaben kann es jedoch Prozesse geben, welche ***aufgrund interner Unternehmensrichtlinien, gesetzlichen Regulierungen oder sonstigen Beschränkungen nicht angepasst oder geändert werden dürfen****... den Umwelteinfluss dieser Geschäftsprozesse verbessert,* ***ohne jedoch den eigentlichen Prozess zu verändern****. Ohne eine nach außen sichtbare Veränderung des Geschäftsprozesses können die Auswirkungen der Optimierung auf Kunden und andere Akteure jedoch* ***sehr limitiert sein****."*

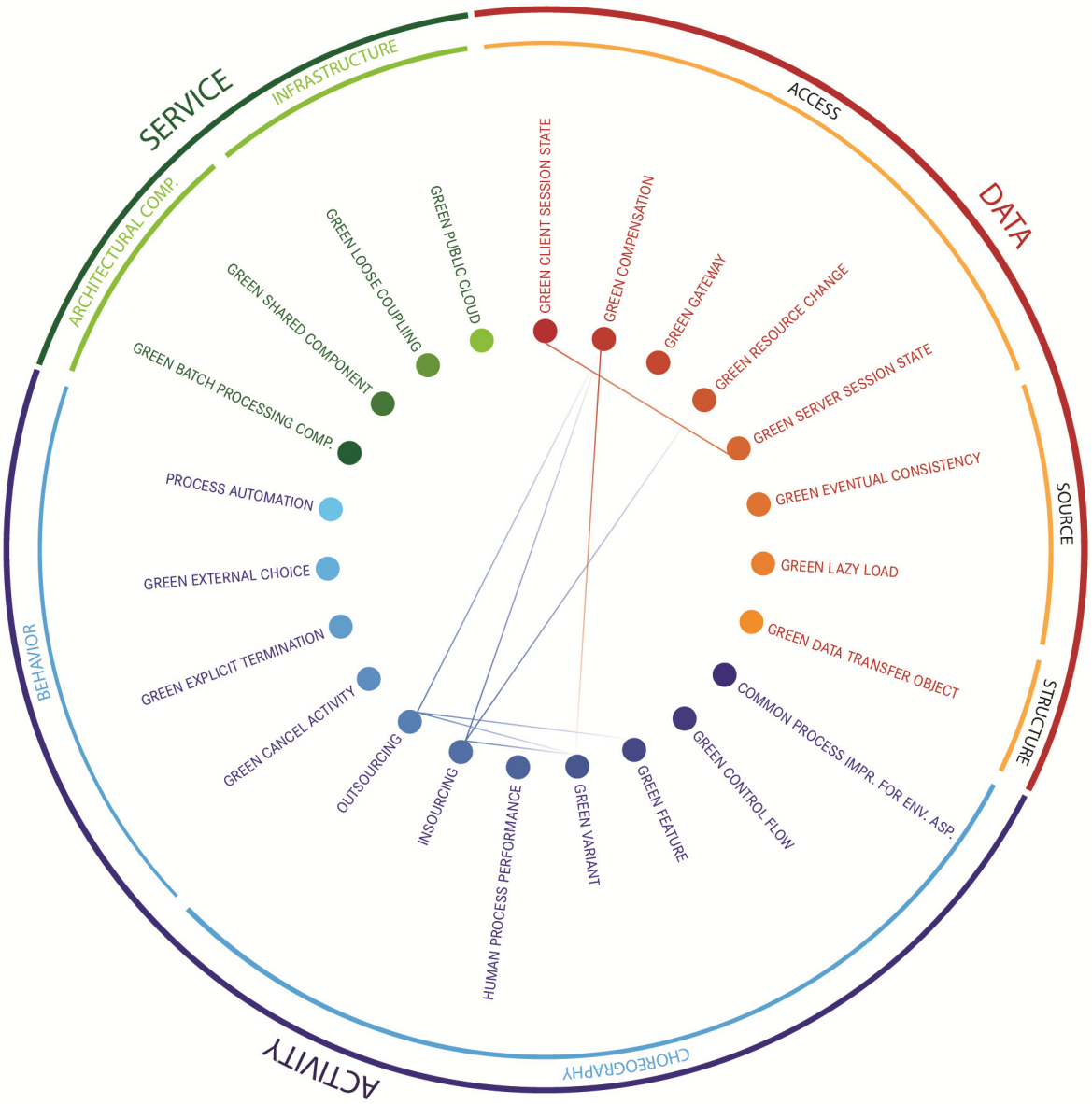

Abbildung 6.3 „Is similar to“ - Relationen zwischen den Mustern in GBPP Mustersprache

Die wichtigsten Aspekte in diesem Fall sind, dass eine Änderung der Geschäftsprozesse nicht möglich ist und dass eine Optimierung des Prozesses ohne nach Außen sichtbare Änderungen von den Kunden als solche nicht wahrgenommen werden kann.

Lösung

Der Lösungsteil des Musters liefert weitere Hinweise, die zur Abgrenzung des Kontexts führen können. In “Green Compensation” ist zu lesen:

*“... **alternative Ressourcen** oder auch **alternative Partner zur Ausführung der Services eines Prozesses verwenden** ...”*

Die Rede ist hier über den Einsatz von “*alternativen Ressourcen*” und “*alternativen*

Partnern". Für den Kontext sind diese zwei Tatsachen wichtig, weil in einem Unternehmen, in dem eine Prozessoptimierung angestrebt wird, durch interne Regeln oder gesetzliche Bestimmungen ein solcher Einsatz unerwünscht oder sogar verboten sein kann.

Diese Überlegungen führen zu den folgenden Schlagwörtern, die die Rahmenbedingungen beim Anwenden des Musters festlegen:

- *Funktionale Ausrichtung*
- *Gesetzliche Regulierungen*
- *Interne Unternehmensrichtlinien*
- *Sonstige Beschränkungen*
- *Keine Änderung des Prozesses*
- *Keine Anpassung des Prozesses*
- *Setzen auf alternative Ressourcen*
- *Setzen auf alternative Partner*
- *Setzen auf externe Dienste*

Die restlichen Teile des Musters betrachten das Problem aus einer anderen Perspektive und liefern nicht immer Information über den Kontext. Daraus muss nicht automatisch geschlossen werden, dass sie nicht zu studieren sind. Sie haben aber einen anderen Schwerpunkt und spielen bei der Kontextbeschreibung eine eher geringere Rolle.

6.1.5 Beschreibung des Problemfelds

Bei der Bestimmung des Problemfelds gilt, wie im vorherigen Kapitel beschrieben wurde, dass für die Verfasser der Sprache im Gegensatz zu den Benutzern, die eine Erweiterung und Verfeinerung der Sprache anstreben, das Stellen der richtigen Fragen eine relativ einfache Aufgabe wird. Im Fall der GBPP Mustersprache sind z.B. der Autor der Muster und der Verfasser der Sprache zwei verschiedene Personen. Deswegen werden hier Hinweise über das Herausfinden passender Fragen aus Sicht eines Benutzers der Sprache gegeben. Die Teile in der Struktur des Musters, die als Referenz benutzt werden können, sind die *Absicht* und die *Lösung*.

Absicht. Die *Absicht* ist die entscheidende Stelle in der Textbeschreibung des Musters und soll zusammenfassen, was das Ziel der Muster ist. Dabei geht es um Schwachstellen im Prozess, die gezielt verbessert werden sollen. Eine mögliche Frage, die vom Absicht-Teil des „*Green Client Session State*" Musters abgeleitet wurde, lautet:

" Soll die Anzahl von Ressourcen und ihre Effizienz durch Nutzung zustandsloser Serverknoten optimiert werden?"

Die Suche nach möglichen Fragen kann im *Lösung-Teil* dann fortgesetzt werden.

Lösung. Die Problemstellung kann auch dazu eingesetzt werden, passende Fragen ausfindig zu machen. Die Absicht dieses Teils ist es, das Problem im Detail vorzustellen und die Ausgangsituation des Problemfelds, in dem das Muster gebraucht wird, herzuleiten. Aus der

Problemstellung des "*Green Client Session State*" Muster lassen sich folgenden Fragen ableiten:

" Soll ein Mechanismus zur Ausführung von Anfragen auf beliebigen Ressourcen bereitgestellt werden?"
und
" Soll ein dynamischer und flexibler an den aktuellen Bedarf angepasster Ressourcenpool zur Verfügung gestellt werden?"

6.1.6 Beschreibung der Lösungswege

Die Lösungswege in GBPP haben zum Ziel, verschiedene Anwendungen der Muster beim Lösen von Problemen zu dokumentieren. Dadurch können bestimmte Lösungsvorschläge von mehreren Benutzern eingesetzt werden. Die vorgeschlagene Struktur dieser Lösungswege ist von der Struktur der Muster in GBPP abgeleitet. Eine Besonderheit dabei ist es, dass die Benutzer neben einer reinen Textbeschreibung zusätzliche Materialien einfügen können. Diese zusätzlichen Materialien sind verschiedene Dateien, die beim Lösen des Problems gebraucht wurden z.B. BPEL-Dateien, UML Diagrammen, PDF-Dokumente mit den Anforderungen an den Geschäftsprozess usw.

Die vorgeschlagene Struktur eines Lösungsweges ist, wie folgt, definiert:

- **Name.** Eindeutige Kennzeichnung des Lösungswegs.
- **Absicht.** Ähnlich, wie bei den Green Patterns, beschreibt die Absicht den Lösungsweg: Was wurde beabsichtigt, als die Muster gewählt wurden und was war das Ziel.
- **Beschreibung.** Für die Beschreibung ist wichtig, dass die Reihenfolge beim Anwenden der Muster angegeben ist. Wichtige Information sind die Hintergründe, die diese Reihenfolge bestimmt haben. Somit sind die Interessierten an diesem Lösungsweg in der Lage, ihn entweder anzuwenden, zu ändern oder als Basis für eigene Interpretationen zu benutzen.
- **Verwendete Muster.** Die Muster, die im Lösungsweg gebraucht wurden.
- **Anhänge.** Verschiedene Dateien, die für den Benutzer von Interesse sein können, damit er leichter nachvollziehen kann, wie die Lösung entstanden ist.

Dadurch werden alle wesentlichen Teile eines Lösungswegs im Rahmen der Mustersprache abgedeckt.

6.2 Konfiguration des Green Pattern Wiki

Im vorherigen Kapitel wurden die Green Business Process Pattern (GBPP) Mustersprache analysiert und die wichtigen Konstrukte, die für das Konfigurieren des Pattern Wiki gebraucht werden, extrahiert. In diesem Kapitel wird anhand dieser Information gezeigt, wie sich ein Pattern Wiki einstellen lässt und was dabei zu beachten ist. Im Anhang A sind alle benötigten externen Erweiterungen aufgelistet, sowie wie sich diese im Media Wiki

einbinden lassen. Dieser Schritt ist notwendig, weil Pattern Wiki Erweiterungen wie Semantic Media Wiki als Voraussetzung zum Funktionieren des Systems vorgesehen sind. Nachdem diese installiert und in Media Wiki eingebunden sind, kann man die Komponenten *Pattern Manager*, *Search Manager* und *Solution Manager* in das System einfügen und konfigurieren.

Pattern Manager

Der *Pattern Manager* ist die einzige Komponente, die konfiguriert werden sollte. Die Konfiguration befindet sich in der *PatternManager.php*-Datei. Hier müssen die Klassen der Klassifizierung eingefügt werden. Für die GBPL Mustersprache sieht die Klassifikation, die im Kap. 6.1.2 gemacht wurde, wie folgt aus:

```
//categorization of the pattern language
$wgPatternManagerCategorization =
        array( 'Data' => array( 'Access', 'Source','Structure'),
                 'Activity' => array('Choreography', 'Behavior'),
                 'Service' => array('Architectural Components','Infrastructure')
);
```

Abbildung 6.4 Auszug aus PatternManager.php – Konfiguration der Musterkategorien

Ein weiterer wichtiger Punkt sind an dieser Stelle die Relationen zwischen den Mustern. Sie werden wie folgt definiert:

```
//pattern link types
$wgGlobalPatternManagerPatternLinkTypes = array('Can be used with','Is similar to');
```

Abbildung 6.5 Auszug aus PatternManager.php – Konfiguration der Musterrelationen

Die Muster benötigen neben der Struktur, die von [NL11] erfordert wird (siehe Kap. 6.1.1), Felder, die von den Benutzern fürs Einfügen des Kontexts und die Beschreibung des Problemfelds gebraucht werden. Dazu kommen zwei zusätzliche Felder, die keine Bedeutung für das richtige Funktionieren des Systems haben, aber Informationen, die für die Benutzer wichtig sind, tragen. An erster Stelle steht das Feld „improves“. Die Schlagwörter, die die Benutzer hier eingeben können, dienen als zusätzliche Hilfe bei der Entscheidung, ob ein Muster passend zum Problem ist oder nicht. Die in diesem Feld gespeicherten Schlagwörter zeigen, was das Muster in einem Business Prozess verbessert. Das zweite Feld ist genau das Gegenteil. Hier werden die Schlagwörter eingegeben, die beschreiben, welche Aspekte des Geschäftsprozesses beim Einsetzen des Musters negativ beeinflusst werden.

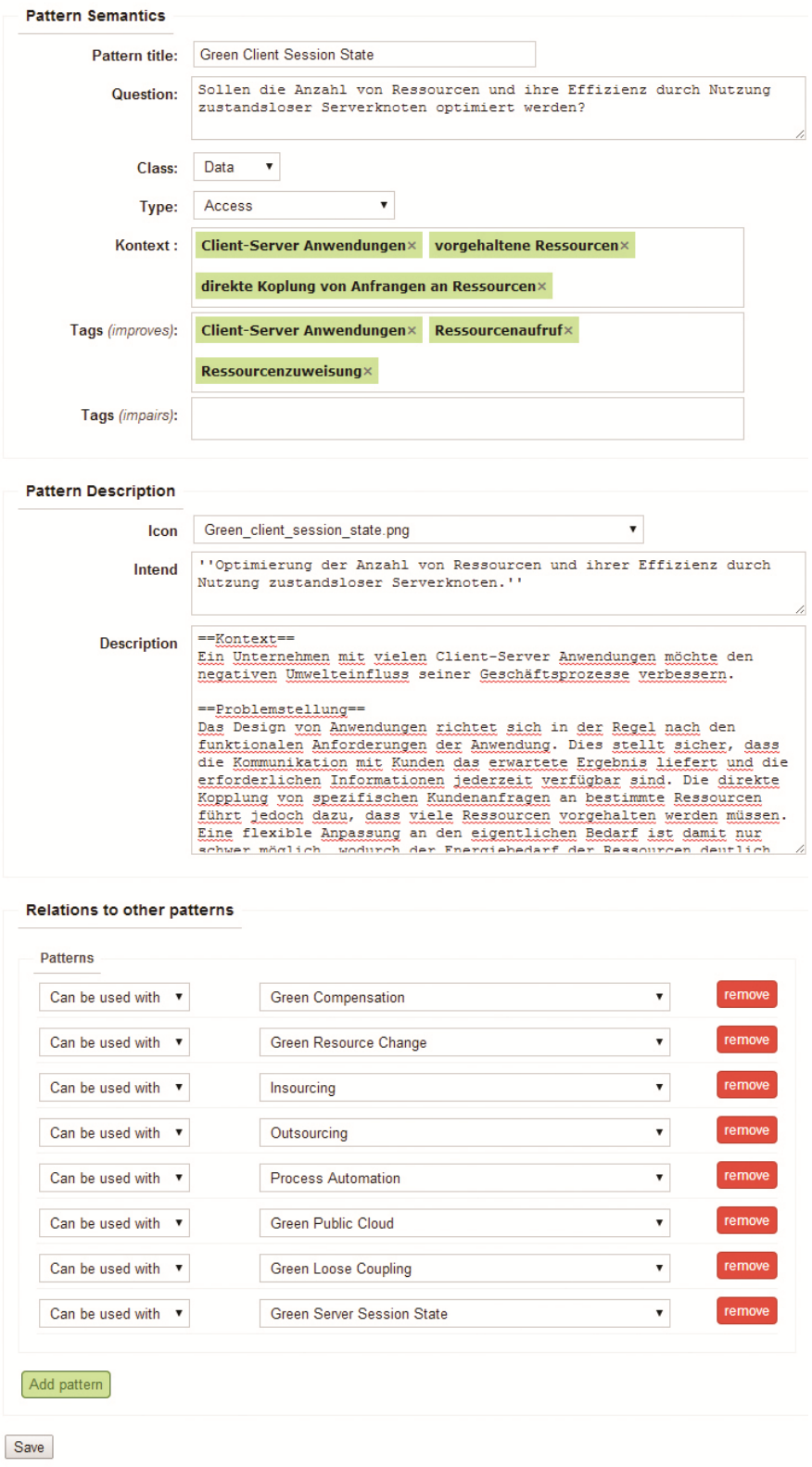

Abbildung 6.6 Pattern Manager Frontend

Das Formular wurde in drei Teile aufgeteilt (Abb. 6.5). Im Teil *„Pattern Semantics“* werden die semantischen Informationen erfasst. Das sind neben dem Namen des Musters, die Frage (Problembeschreibung), die Anordnung in der Klassifizierung, der Kontext (Abgrenzung des Kontexts der Problembeschreibung), sowie die Felder, die die Aspekte, die sich beim

Benutzen des Musters verbessern oder verschlechtern lassen. Im Teil *„Pattern Description“* sind alle Informationen gegliedert, die die Repräsentation des Musters, wie diese in der Sprache beschrieben sind, darstellen. Das sind: das Bild, die Absicht und die Beschreibung. Im letzten Teil sind die Relationen zu den anderen Mustern zusammengefasst.

Die Konfiguration des Eingabeformulars ist wie folgt definiert:

```
---------------------------------------------------------------------------------------------------------
$wgPatternManagerFormDescriptor = array(
        'patternTitle' => array(
                        'section' => 'section0',
                        'label'=> 'Pattern title:',
                        'class' => 'HTMLTextField',
                        'validation-callback' => array('PatternForm', 'validatePatternTitle'),
                        'required' => true,
                        'semantic' => true,
                        'linkType' => 'Has title',
        ),
        'patternQuestion' => array(
                        'section' => 'section0',
                        'label' => 'Question:',
                        'class' => 'HTMLTextAreaField',
                        'cssclass' => 'smallTextArea',
                        'rows' => 1,
                        'cols' => 30,
                        'required' => true,
                        'semantic' => true,
                        'linkType' => 'Has question'
        ),
        'patternClass' => array(
                        'section' => 'section0',
                        'label'=> 'Class:',
                        'class' => 'HTMLSelectField',
                        'validation-callback' => array('PatternForm', 'validatePatternTitle'),
                        'linkType' => 'Is from class',
                        'required' => true,
                        'semantic' => true,
                        'options' => array(
                                'Data' => 'Data',
                                'Activity' => 'Activity',
                                'Service' => 'Service',
                        ),
        ),
        'patternClassType' => array(
                        'section' => 'section0',
                        'label'=> 'Type:',
                        'class' => 'HTMLSelectField',
                        'validation-callback' => array('PatternForm', 'validatePatternTitle'),
                        'linkType' => 'Is from type',
                        'required' => true,
                        'semantic' => true,
                        'options' => array(
                                'Access' => 'Access',
                                'Source' => 'Source',
                                'Structure' => 'Structure',
                                'Choreography' => 'Choreography',
                                'Behavior' => 'Behavior',
                                'Achitectural Components' => 'Architectural Components',
                                'Infrastructure' => 'Infrastructure',
                        ),
        ),
        'patternTagsKontext' => array(
                        'section' => 'section0',
                'label' => 'Kontext <span style="font-style:italic;color:#464646;font-
weight:normal;"></span>:',
                        'class' => 'HTMLTagField',
                        'semantic'=> true,
                        'linkType'=> 'Kontext',
        ),
        'patternTagsImproves' => array(
                        'section' => 'section0',
```

```
                        'label' => 'Tags <span style="font-style:italic;color:#464646;font-
weight:normal;">(improves)</span>:',
                        'class' => 'HTMLTagField',
                        'semantic'=> true,
                        'linkType'=> 'Improves',
        ),
        'patternTagsImpairs' => array(
                        'section' => 'section0',
                        'label' => 'Tags <span style="font-style:italic;color:#464646;font-
weight:normal;">(impairs)</span>:',
                        'class' => 'HTMLTagField',
                        'semantic' => true,
                        'linkType' => 'Impairs',
                        'cssId' => 'patternTagsImpairs',
        ),
        'patternIcon' => array(
                        'section' => 'section2',
                        'label' => 'Icon',
                        'class' => 'HTMLSelectField',
                        'options' => array(
                                'Option 1' => 'option1',
                                'Option 2' => 'option2'
                        ),
                        'required' => true,
                        'validation-callback' => array('PatternForm', 'validatePatternIcon'),
                        'mimeType' => 'image',
        ),
        'patternIntend' => array(
                        'section' => 'section2',
                        'label' => 'Intend',
                        'type' => 'textarea',
                        'rows' => 1,
                        'cols' => 30,
                        'required' => true,
        ),
        'patternDescription' => array(
                        'section' => 'section2',
                        'label' => 'Description',
                        'type' => 'textarea',
                        'rows' => 1,
                        'cols' => 30,
                        'required' => true,
        ),
); //end form descriptor;
```

Abbildung 6.7 Auszug aus PatternManager.php – Konfiguration des Formulars

Das Formular deckt somit alle Aspekte der Struktur eines Musters ab, die vom Pattern Wiki gebraucht werden. Wie schon in Kap. 5 einführend erklärt wurde, haben die Administratoren des Wiki keinen Einfluss auf den Teil, der die Relationen zu den anderen Mustern definiert. Deswegen ist dieser intern im Pattern Manager abgebildet und definiert und von außen nicht zugänglich. Diese Information ist Pflicht für jede Mustersprache und es wird angenommen, dass sie immer existiert.

Auf der Browse-Seite lassen sich alle Muster im Wiki auflisten. Diese Seite lässt sich wie folgt konfigurieren:

```
$wgGlobalPatternManagerBrowseConfig = array(
                'patternParts' => array('patternIntend', 'patternIcon'),
                'classification' => $wgPatternManagerCategorization,
                'classificationMeta' => $wgPatternManagerCategorizationMeta,
);
//how many patterns per row to be shown
$wgPatternManagerPatternsRowCount = 4;
```

Abbildung 6.8 Auszug aus PatternManager.php – Konfiguration der Browse-Seite

An erster Stelle wird dabei angegeben, welche Teile des Musters in der Browse-Seite gezeigt werden (*„patternParts“*). Dann wird die Klassifikation, sowie die Semantik der Klassifikation erwartet.

Damit auf jeder Muster-Seite rechts ein Block mit einer Zusammenfassung der semantischen Information des Musters angezeigt wird, wurden zusätzlich Wiki Vorlagen erstellt. Ihre Definitionen sind in Anhang C aufgelistet.

Search Manager

Der Search Manager muss nur als Erweiterung im Media Wiki eingebunden werden. Damit dieser richtig funktioniert, sollten die Muster der Sprache in das Wiki eingefügt werden. Die Muster müssen unbedingt die semantische Information über Fragen und Kontext speichern, weil diese die Grundlage des Konzepts der Suche nach dem Einstiegpunkt in der Sprache sind. Das Recommender System ist im Search Manager enthalten und hier wird keine Konfiguration benötigt.

Solution Manager

Der Solution Manager erfordert keine spezielle Konfiguration. Das Formular zum Erstellen der Lösungswege besteht aus allen benötigten Teilen. Diese sind, sowie die Beschreibung eines Lösungswegs, allgemein gehalten. Somit ist das Einfügen von allen möglichen Informationen über einen Lösungsweg gewährleistet. Der einzige administrative Aufwand neben dem Einbinden der Erweiterung im Wiki System ist das Einbinden der Erweiterung „Files“, die im offiziellen MediaWiki Repository gefunden werden kann. Diese ermöglicht das Einfügen von zusätzlichen Materialien zu jedem Lösungsweg.

Nachdem die Analyse der Mustersprache durchgeführt wurde und die Muster in das konfigurierte System eingefügt wurden, ist dieses System bereit, in der Praxis eingesetzt zu werden. Wie eine Suche mit diesem System abläuft, wird im nächsten Kapitel gezeigt.

7 Anwendungsfall

Nachdem im vorherigen Kapitel das Green Pattern Wiki konfiguriert wurde und einsatzbereit ist, wird in diesem Kapitel seine Funktionsweise anhand der Green Business Process Patterns (GBPP) Mustersprache veranschaulicht. Die Muster der Sprache sind schon im Pattern Wiki eingepflegt und den Interessierten zur Verfügung gestellt.

Als Beispiel wird dabei ein Geschäftsprozess optimiert, der einen Bestellprozess [WK10] innerhalb eines Onlineshops abbildet. Er besteht aus mehreren Schritten, die notwendig sind, um eine Warenbestellung innerhalb eines Onlineshops zu tätigen (Abb. 7.1). Dabei geht es im ersten Schritt um das Überprüfen der Lagerbestände und darum, ob die bestellten Waren im Lager des Unternehmens vorhanden sind. Falls das nicht der Fall ist, versucht der Prozess, in einem weiteren Schritt die Waren in Partnerunternehmen zu bestellen und von dort ins eigene Lager zu liefern. Sobald die Waren vorhanden sind, kann die Bestellung fortgesetzt werden. Der Käufer wird informiert und aufgefordert, die Zahlungsart zu wählen. Nachdem die Kreditkarte des Benutzers überprüft wird oder die Banküberweisung seitens des Benutzers getätigt wurde, werden die Waren zum Verschicken vorbereitet. Diese werden vom Speditionsunternehmen an den Kunden geliefert. Der Geschäftsprozess kann zu jeder Zeit, wegen Mangel an Waren im Lager nicht nur des Onlineshops, sondern auch beim Partner, abgebrochen werden.

Der Geschäftsprozess muss so optimiert werden, dass dadurch eine Verbesserung der Ökobilanz erzielt wird. Um die Ökobilanz zu untersuchen, lohnt es sich, einen Blick auf das KEI Dashboard (KEIDA)[11] [LD13] zu werfen, falls das KEI Framework, sowie KEIDA im Unternehmen eingesetzt werden. KEIDA ist das Frontend des Datawarehouses, das vom KEI Framework generierte oder aufgefasste Daten bei der Simulation der Ausarbeitung eines Prozesses oder bei der Ausführung von Prozessen in einer produktiven Umgebung enthält. Diese Daten werden auf dem Dashboard grafisch dargestellt und ermöglichen die Erkennung von Zusammenhängen oder Trends beim Ausführen des Geschäftsprozesses. Dadurch lässt sich der Energieverbrauch jeder Aktivität innerhalb des Prozesses ablesen und während einer längeren Periode beobachten. Die Interessierten an einer Optimierung dieses Energieverbrauchs sind in der Lage, Problemstellen zu erkennen und Gegenmaßnahmen zu ergreifen.

Das Ziel einer solchen Optimierung der Prozesse kann vielseitig sein. Als Hauptgründe kann man die Verbesserung des Workflows des Prozesses und die damit verbundene Effektivität nennen, sowie die Steigerung der Kundenzufriedenheit, eine vollständige Auslastung der vorhandenen Rechenkapazitäten und eine Reduzierung der Kosten, die beim Ausführen des Prozesses entstehen [MSA]. Dadurch werden eine Nachhaltigkeit beim Betreiben des

[11] Für die Zwecke des Anwendungsfalls wird in diesem Kapitel angenommen, dass KEIDA im Unternehmen aktiv eingesetzt wird und dass eine vollständige Statistik des Energieverbrauchs des Prozesses und seiner einzelnen Aktivitäten vorliegen.

Geschäftsprozesses und eine langfristige Verbesserung der Ökobilanz, die der Prozess aufweist, verfolgt.

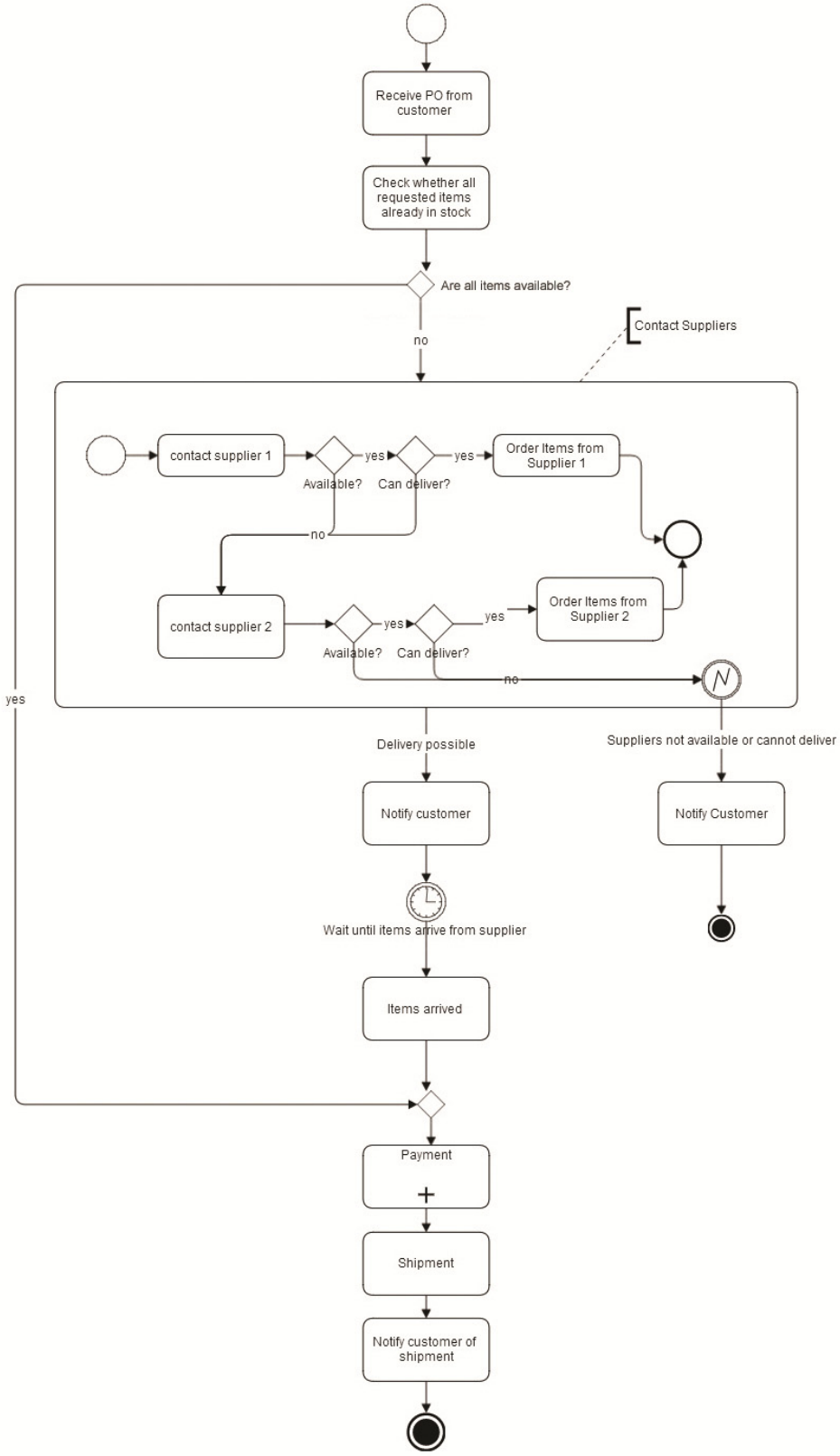

Abbildung 7.1 Bestellprozess

Im Fall des Bestellprozesses des Online Shops liefert KEIDA erste Hinweise über mögliche Problemstellen. Der Energieverbrauch von Aktivitäten „Order items from supplier“ und „Check availability in stock“ ist im Gegensatz zu den anderen viel höher, verursacht einen Großteil der Kosten und ist verantwortlich für die negative Ökobilanz. Aber nicht nur KEIDA, sondern auch das Wissen über den Aufbau und die Struktur des Prozesses und wie dieser implementiert ist, ist wesentlich und liefert wichtige Hinweise für das Optimieren des Prozesses. Für die Zwecke dieses Anwendungsfalls wird angenommen, dass eine vollständige und ausführliche Information über Aufbau und Struktur des Prozesses vorliegt. In der Realität kann das nicht immer der Fall sein, weil z.B. für mehrere Services externe Anbieter eingesetzt werden, deren Quellencode nicht vorhanden ist oder die Information über den Energieverbrauch der Services nicht vorliegt. Das bedeutet für den Bestellprozesses, dass sich jeder Aspekt bei dem Aufbau und der Struktur ändern lässt.

Um die Optimierung zu vereinfachen, wird im nächsten Schritt das Einsetzen vom Green Pattern Wiki empfohlen. Im gegebenen Fall lässt sich jeder Aspekt ändern. Für die Suche nach einem Einstiegspunkt in der Mustersprache bedeutet das, dass Muster aller Kategorien in Betracht gezogen werden können. Diese Auswahl wird aber in der weiteren Analyse beschränkt, weil sich nur bestimmte Aspekte beim Aufbau des Geschäftsprozesses sinnvoll ändern lassen. Im Geschäftsprozess, der hier betrachtet wird, ist z.B. eine Änderung der Daten, ihrer Quelle und Struktur nicht nötig. Eine Änderung an der Infrastruktur, auf der die Aktivitäten des Prozesses laufen, ist aber dringend nötig.

Es wird angenommen, dass die Webservices, die den Prozess abbilden, auf der eigenen Infrastruktur des Unternehmens laufen (Server im eigenen Rechenzentrum). Weiter wird angenommen, dass neben den zwei Partnershops, bei denen im eigenen Lager nicht vorhandene Waren nachbestellt werden, weitere Partnerunternehmen einbezogen werden können. Die Spedition der Waren erfolgt über ein externes Unternehmen. Mit dieser Information und mit der Information, die mit Hilfe von KEIDA gewonnen werden kann, ist der Interessierte in der Lage, mit Hilfe von Green Pattern Wiki einen passenden Einstiegspunkt in der Mustersprache zu finden.

Die Ausgangsituation zeigt, dass eine Änderung der Infrastruktur zu empfehlen ist. Es existieren schon Technologien, wie Cloud Computing, die eine Verringerung des Energieverbrauchs ermöglichen und die in Betracht gezogen werden sollen. Aber nur eine Änderung der Infrastruktur allein wird nicht zum gezielten Effekt führen. Eine Verbesserung des Workflows des Geschäftsprozesses ist nötig, um z.B. weitere Lieferanten zum Bestellen fehlender Waren einzubeziehen. Dabei gibt es eine wesentliche Anforderung, dass seitens dieser externen Unternehmen Informationen über den Umwelteinfluss ihrer Dienstleistungen vorgelegt werden sollten. Es ist auch nötig, weitere Speditionsunternehmen in Perioden größerer Nachfragen zu engagieren, damit die Waren rechtzeitig und möglichst umweltfreundlich geliefert werden. Diese Anforderungen zeigen, dass ein schonender Umgang mit den Ressourcen, die zur Verfügung stehen, unumgänglich ist. Nachdem diese Information vorliegt und es eine klare Vorstellung über den neuen Prozess gibt, kann die Suche nach dem passenden Muster beginnen.

Optimierung des Bestellprozesses mittels Green Pattern Wiki

Im ersten Versuch wird nach einer Möglichkeit gesucht, die Infrastruktur, auf der der Prozess läuft, zu verbessern. Deswegen wird im ersten Schritt in der Suche die Kategorie "*Service // Infrastruktur*" ausgewählt. Im nächsten Schritt wird im freien Textfeld "*Verbesserung der Infrastruktur, auf der der Prozess läuft*" eingegeben sowie die "*Ressourcenbelastung*" sowie "*Ressourcennutzung*" und im letzten Schritt wird "*Sollen externe Cloud Computing Ressourcen zur Verbesserung der Ressourceneffizienz verwendet werden?*" ausgewählt. Als Ergebnis der Suche werden Muster vorgeschlagen, aus denen nach einem genauen Blick auf die Einleitungstexte das "Green Public Cloud"-Muster als das passendste ausgewählt wird. Liest man die "Problemstellung" des Musters, sieht man sofort, dass das Übertragen des Geschäftsprozesses des Onlineshops in eine Cloud-Umgebung notwendig ist. Dadurch ist an erster Stelle eine On-Demand Ausdehnung der Rechenkapazitäten möglich, was in Perioden größerer Nachfrage zu keinen Ausfällen oder anderen negativen Effekten führen würde. Weiter besteht die Möglichkeit, auf der Webseite ein Öko-Siegel oder „CO_2-Frei" zu platzieren [12] . Dadurch wird dem Benutzer bewusst, dass das Unternehmen nachhaltig ausgerichtet ist, wodurch die Kundenzufriedenheit verbessert werden könnte.

Das Übertragen des Prozesses auf die Cloud reicht aber nicht aus und weitere Schritte werden benötigt, die die Anwendung weiterer Muster erfordern. Wie oben beschrieben wurde, ist die Einbindung weiterer Zulieferer und Speditionsunternehmen, die die Waren an die Kunden liefern, das Ziel. Die Anforderungen an den Prozess, die dadurch entstehen, sind eine bessere Skalierbarkeit und eine Optimierung des Zulieferer- und Speditionsauswahlablaufs. Da für die Verbesserung der Infrastruktur Schritte schon vorgenommen wurden, wird jetzt nur nach einer Lösung gesucht, die die Zusammenarbeit und die Choreographie der Aktivitäten verbessert. Eine neue Suche wird gestartet und im ersten Schritt werden nur die Kategorien "*Activity//Choreography*" und "*Activity//Behaviour*" ausgewählt. Im nächsten Schritt wird als Beschreibung des Problems folgender Text eingegeben: *"Verbesserung der Zusammenarbeit der Aktivitäten, Änderung der Reihenfolge der Aktivitäten"*. Im Kontext-Teil stehen mehrere zum Problem passende Terme, die am besten in der Situation entsprechen: "*Einsatz von Optimierungstechniken*", "*Komplexitätserhöhung durch Einführung von KEI*", "*Auftragsausführung*", „P*rozessoptimierung aus Kundensicht*", *„Kostensteigerung durch Prozessumstrukturierung", „ineffiziente funktionalorientierte Ressourcennutzung", „Aktivitätsalternativen", „Prozesspfadalternativen"* (Abb. 7.2).

[12] Im Jahr 2008 stellt der deutsche Cloud Anbieter "Strato" den Rechenzetrumsbetrieb auf CO_2 –neutralen Strom um. Die Webseiten, die dort gehostet sind, haben die Möglichkeit das so genannte Öko-Siegel zu platzieren. Quelle: http://www.strato.de/ueber-uns/

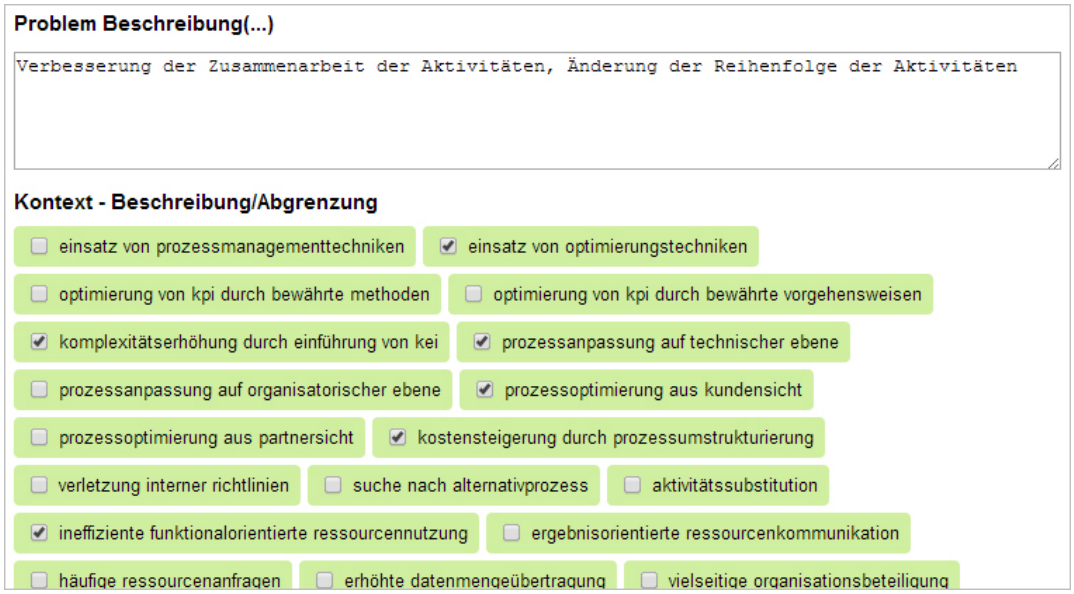

Abbildung 7.2 Schritt zwei in der Suche

Im nächsten Schritt wären die passenden Fragen (Abb. 7.3):

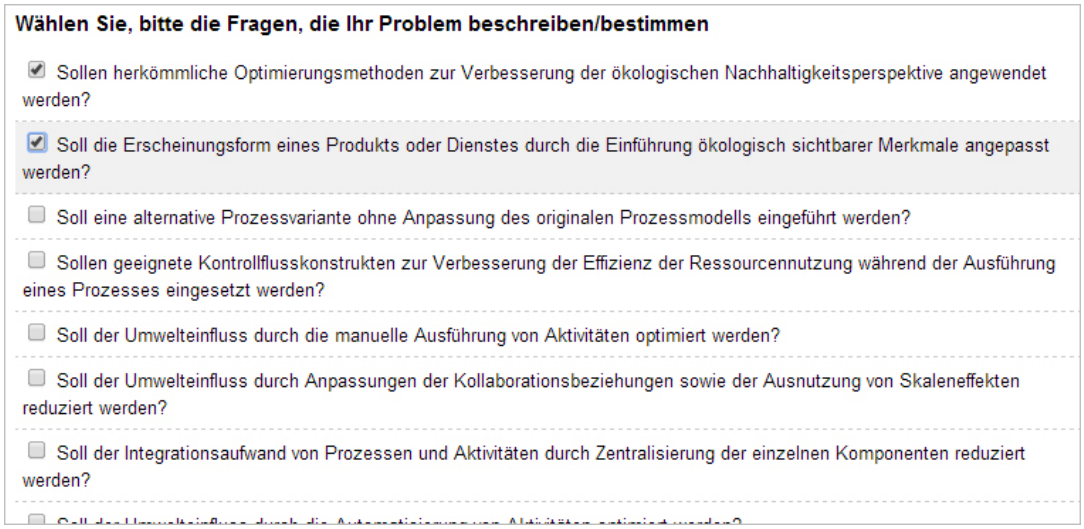

Abbildung 7.3 Schritt drei in der Suche

Als Ergebnis (Abb. 7.4) werden mehrere Muster vorgeschlagen, und als das Passende wird „*Common Process Improvement for Environmental Aspects“* ausgewählt. Dieses besagt, dass beim Optimieren eines Prozesses die Aktivitäten parallel ausgeführt oder in Teilaufgaben zusammengefasst werden sollen. Für den Bestellprozess des Onlineshops bedeutet es, dass eine neue und bessere Struktur (Workflow) gebraucht wird. Die Beispiele im “Beispiele”-Teil des Musters zeigen, dass andere Unternehmen auf technische Mittel setzen, um den Umgang mit den Ressourcen zu optimieren. Für eine Verbesserung der Auswahl der Zulieferer, die hier verfolgt wird, bietet dieses Muster keine Hilfe mehr.

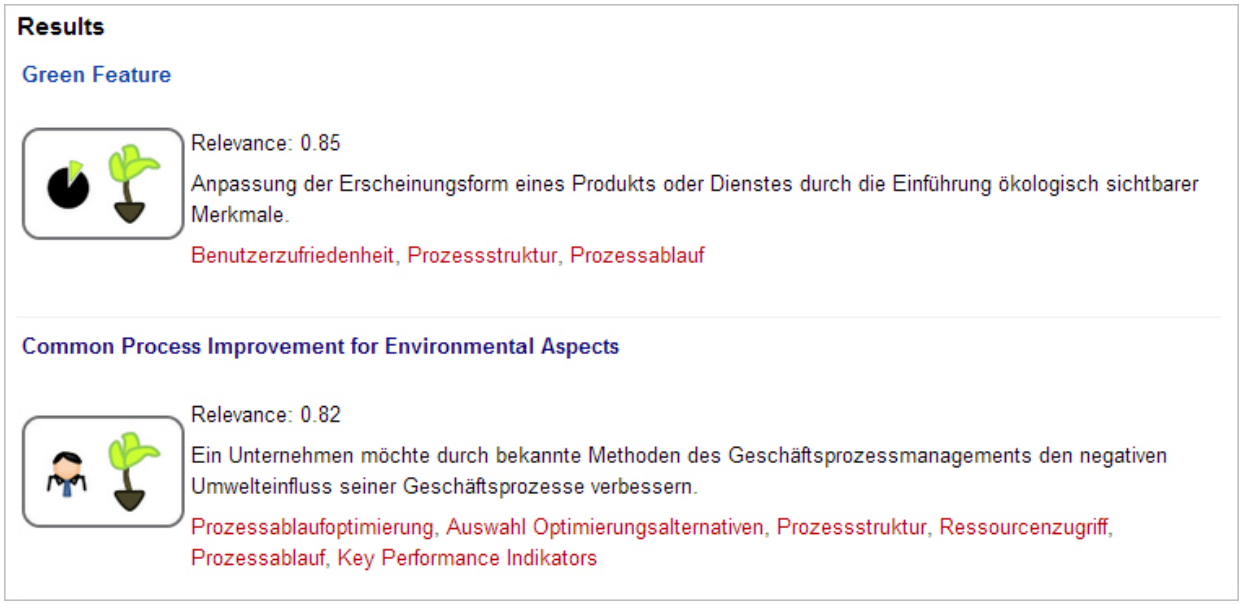

Abbildung 7.4 Ergebnis der Suche

Liest man aber die Zusammenfassungen der Muster, die in Relation "*can be used with*" stehen, fällt das Muster *"Green External Choice"*. Dieses sieht externe Information vor, die den Kontrollfluss in Verzweigungen im Prozess verbessern kann. Im Geschäftsprozess des Onlineshops entsteht dadurch die Möglichkeit, dass beim Vorliegen solcher Informationen über die Zulieferer, z.B. ob diese Kleintransporter mit Elektromotoren für kürzere Strecken einsetzen, ob die Transporter des Unternehmens mit Biokraftstoff betankt werden usw., eine Entscheidung über den Zulieferer mit dem niedrigsten Umwelteinfluss getroffen werden kann. Eine solche Komponente (Abb. 7.5) im Geschäftsprozess kann zu einer enormen Verbesserung der Ökobilanz des ganzen Prozesses führen. Die neue Komponente vereinfacht die Struktur des Geschäftsprozesses und ermöglicht gleichzeitig eine Erweiterung der Liste der mögliche Zulieferer.

Die gleichen Überlegungen führen zu einer Verbesserung der *Shipment*-Komponente.

Nachdem alle Muster identifiziert und eingesetzt wurden ergibt sich die neue Struktur des Geschäftsprozesses (Abb. 7.5). Die Verbesserungen sehen an erster Stelle eine Übertragung des Geschäftsprozesses in eine Cloudumgebung vor. Die Infrastruktur, die vom großen Cloud-Anbieter zur Verfügung gestellt wird, ist so optimiert, dass diese viel weniger Strom verbraucht und bei der Produktion des Stroms auf erneuerbare Energie gesetzt wird. Neben der Infrastruktur wird vorgeschlagen, dass die Struktur des Prozesses geändert wird. Die neuen Aktivitäten „choose supplier" und „choose shipping agent" erfordern Information über die externen Partner und insbesondere Information über den Umwelteinfluss dieser Unternehmen. Dadurch wird der Prozess neu gestaltet und muss in der Praxis überprüft werden.

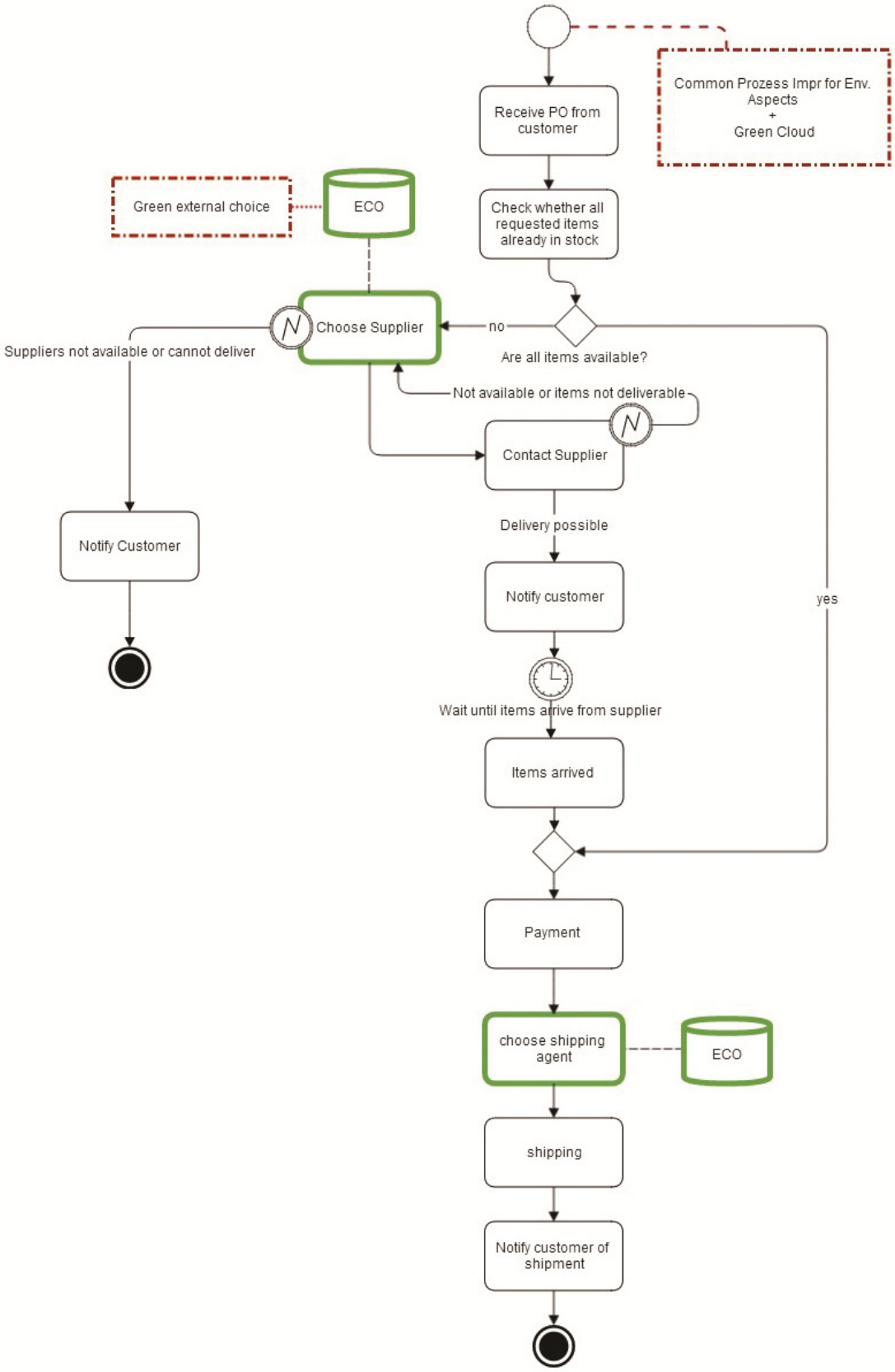

Abbildung 7.5 Bestellprozess nach der Optimierung

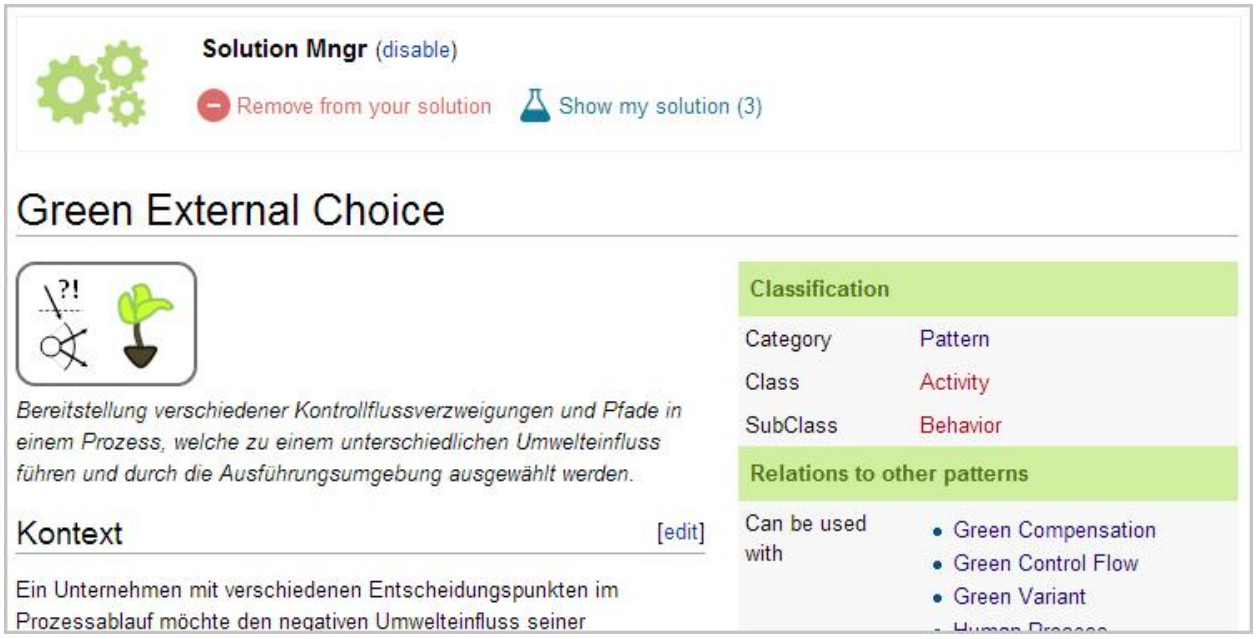

Abbildung 7.6 Solution Manager Leiste

Erstellung eines Lösungswegs

Nachdem der Geschäftsprozess geändert wurde, kann diese Lösung als solche im Repository abgelegt werden. Im ersten Schritt muss dafür der Lösungsmanager eingeschaltet werden. Die Muster "*Green Public Cloud*", "*Common Process Improvement for Enviromental Aspects*" und "*Green External Choice*" werden zur Lösung mittels Kontrollleiste (Abb. 7.6) des Solution Managers eingefügt. Die Beschreibung des Problems, wie und mit welchem Muster dieses gelöst wurde, werden ebenfalls hinzugefügt. Sobald die Lösung gespeichert wurde, können zusätzliche Dateien eingefügt werden und somit ist die Lösung fertig. Sie ist für jeden Benutzer des Pattern Wiki sofort zugänglich und kann in ähnlichen Situationen wie der gerade beschriebenen angewendet werden.

In diesem Kapitel wurden die einzelnen Schritte vorgestellt, die benötigt werden, um ein passendes Muster als Einstiegspunkt in einer Mustersprache (nach dem Konzept in Kapitel 4) zu finden. Der Geschäftsprozess, der dabei vorgeschlagen wurde, ist sehr vereinfacht dargestellt worden und veranschaulicht den Umgang mit einer Implementierung des Pattern Wiki-Konzepts, das in dieser Diplomarbeit erstellt wurde. Die Muster in der GBPP Mustersprache [NL11] lassen sich mit Hilfe des Pattern Wiki Systems leicht bestimmten Problemen zuordnen. Dadurch wird eine Optimierung des Umwelteinflusses der Geschäftsprozesse gewährleistet. Damit diese Lösungen (mehrere Muster, die an einem Problem angewendet wurden) von anderen Benutzern des Pattern Wiki als Vorlage (Lösungsweg) für spätere Optimierungen dienen können, bietet das System die Möglichkeit, sie als solche zu speichern. Die Benutzer sind danach in der Lage, sich beim Lesen des Musters auch die Lösungen, bei denen es verwendet wurde, anzuschauen. Die schon vorbereiteten Lösungen stehen, wie alle Muster der Sprache, danach bereit, in ähnlichen oder gleichen Situationen angewendet zu werden. Das Wissen, das von allen Benutzern im Pattern Wiki abgelegt wird, entwickelt einerseits die Sprache und bereitet gleichzeitig das Wissen auf und stellt es somit allen anderen Benutzern zur Verfügung.

Zusammenfassung

Wissen wird ständig in allen Wissensdomänen generiert. Die Notwendigkeit, dieses Wissen für einen zukünftigen Gebrauch zusammenzufassen und zu speichern, ist eine der größten Herausforderungen, vor der die Industrie, die Wissenschaft und alle anderen Sphären des Lebens stehen. Christopher Alexander [Ac72] hat diese Notwendigkeit erkannt und mit seinem Vorschlag, das Wissen in Form von Mustern in eine Mustersprache zusammenzufassen, den ersten großen Schritt in Richtung Systematisierung von Lösungen zu immer wieder vorkommenden Problemen gemacht. Seitdem findet dieser Ansatz eine breite Verwendung und dient als Grundlage für viele wissensbasierte Datensystemen. Nach dem Vorbild der Mustersprache von Alexander sind mehrere Mustersprachen entstanden, wobei für die Benutzer der Sprachen immer wieder das gleiche Problem vorkommt und zwar, wie das richtige Muster gewählt werden soll. Für Benutzer, die wenig Erfahrung mit der Sprache haben, wie schon in Kap. 2 angesprochen, wird die Lösung dieses Problems zu einer langwierigen Suche. Diese Diplomarbeit hat zum Ziel, diese Suche zu vereinfachen und generell die Arbeit mit einer Mustersprache zu erleichtern. Das Konzept, das in Kap. 3 vorgeschlagen wurde, setzt auf eine Analyse der Mustersprachen und darin erkannten immer wieder vorkommenden Strukturen. Dadurch wurden die Hauptteile eines Systems ausgearbeitet, die dann auch im Kap. 5 anhand einer Beispielimplementierung veranschaulicht wurden. Die Komponenten des Konzepts *Pattern Manager, Solution Manager, Search Manager* und *Recommender System* decken alle Aspekte der Arbeit mit einer Mustersprache ab. Dabei ist es nicht nur wichtig, dass eine leichte Verwaltung der Sprache möglich ist, sondern auch, dass eine Suche nach einem Einstiegspunkt gewährleistet wird.

Das System ist so konzipiert, dass nicht nur die Verfasser der Mustersprache, sondern auch die Community, die die Mustersprache verwendet, dafür verantwortlich sind, dass das System richtig funktioniert und die gesuchte Information liefert. Die Verfasser der Mustersprache liefern die Lösungen und sind in einem ersten Schritt für das Einfügen dieser Lösungen ins System zuständig. Da aber die Muster von einem breiten Publikum gebraucht werden, soll dieses Publikum in der Lage sein, die vom Verfasser eingefügten zusätzlichen Informationen zu ändern. Die Metainformationen sind ein wesentlicher und notwendiger Bestandteil des Suchkonzepts. Dabei wurden drei wichtige Teile identifiziert: Klassifikation der Muster, Abgrenzung des Kontexts der Problemstellung und die Problembeschreibung. Wie in Kapitel 3 erläutert wurde, sind diese drei Teile in jeder Mustersprache zu finden und können dazu eingesetzt werden, in den unstrukturierten Informationen, die die Muster darstellen, zu suchen. Werden diese drei Teile richtig und ausführlich beschrieben, werden die Benutzer des Systems in der Lage sein, unkompliziert und in wenigen Schritten das zu ihrem Problem passende Muster zu finden. Die Suche nach dem richtigen Muster wird durch das Feedback, das heißt durch Auffassung und Auswertung der Suchgeschichte jeder einzelnen Benutzersuche, erweitert.

In Kapitel 6 wurde dann gezeigt, wie sich eine Mustersprache analysieren lässt, und wie ein Pattern Wiki für die Praxis vorbereitet werden könnte. Dabei wurden alle Schritte vorgenommen, die auch in Kapitel 3 erläutert wurden. Das Beispiel in Kapitel 7 zeigt auch wie einfach der Umgang mit dem System ist und dass ein Benutzer ohne große Vorkenntnisse über die Muster schnell die richtigen Muster finden kann.

Das in dieser Diplomarbeit vorgeschlagene Konzept hat gegenüber schon bestehenden Konzepten den Vorteil, dass es sich in jeder Mustersprache einsetzen lässt. Es setzt einerseits auf das Wissen der Verfasser der Mustersprache und andererseits auf das Engagement der Community, die die Mustersprache benutzt. Die drei Haupteile der Suche werden mit Hilfe von Metadaten beschrieben und können jederzeit leicht angepasst oder verbessert werden. Die Konstrukte, die dabei eingesetzt werden, sind leicht zu verstehen und jeder, der Erfahrung mit der Mustersprache hat, soll in der Lage sein, sie zu benutzen und dadurch das Funktionieren des Systems zu verbessern. Dadurch entsteht ein System, das leicht zu konfigurieren und zu erweitern ist und die Aufgabe übernimmt, dem Benutzer der Mustersprache auf Basis von eingegebenen Informationen die Richtung zu zeigen und ihn beim Treffen von Entscheidungen zu unterstützen.

Anhang A

Technische Anforderungen an die Software

Server:

Software	Version	Mehr Informationen
Apache	2.2	www.apache.org
PHP	5.3.8	www.php.net
MySQL	5.5	www.mysql.com

Media Wiki:

Die MediaWiki Version der Software, die für das Erstellen des Wiki Pattern eingesetzt wurde, ist 1.21.3

Erweiterungen für Media Wiki:

Erweiterung	Version	Download
Semantic media Bundle		http://goo.gl/AKwEUk
File list		http://goo.gl/JfVahT
Cite		http://goo.gl/QB5bQi

Die Semantic media Bundle enthält folgende Erweiterungen:

- SemanticBundle master
- SemanticMediaWiki 1.8.x
- SemanticResultFormats 1.8.x
- SemanticForms 2.5.3
- SemanticFormsInputs REL_0_7
- SemanticCompoundQueries 0.3.4
- SemanticDrilldown 1.3
- SemanticMaps 2.0.1
- SemanticTasks master
- SemanticImageInput master
- SemanticInternalObjects 0.7.5
- SemanticWatchlist 0.2.1
- AdminLinks 0.1.8
- ApprovedRevs 0.6.5
- Arrays master
- DataTransfer 0.4.1
- ExternalData 1.6.2
- HeaderTabs 0.9.2
- Maps 2.0.1
- PageSchemas 0.3.1
- ReplaceText 0.9.7
- Validator 0.5.1
- Widgets 1.0

Anhang B

Installation der Erweiterungen im Pattern Wiki

Nachdem die Software in Anhang A erfolgreich konfiguriert und zum Laufen gebracht wurde, können die *PatternManager*, *PatternSearch* und *SolutionManager* installiert werden. Die Installation erfolgt wie folgt:

1. Die Ordner mit den Dateien der jeweiligen Erweiterung müssen in den „Extensions“-Ordner des Wiki Installation kopiert werden.

2. In die Datei LocalSettings.php im Hauptordner der Wiki-Installation müssen folgende Zeilen eingefügt werden:

```
//pattern search extension
require_once( "$IP/extensions/PatternSearch/PatternSearch.php" );
//solution manager
require_once("$IP/extensions/PatternSolutions/PatternSolutions.php");
//green form
require_once("$IP/extensions/PatternManager/PatternManager.php");
```

---

3. Die zusätzlichen Erweiterungen, die gebraucht werden, sollen wie folgt eingebunden werden:

```
//array functions
require_once( "$IP/extensions/Arrays/Arrays.php" );
//references
require_once "$IP/extensions/Cite/Cite.php";
//file uploads
require_once("$IP/extensions/FileList/FileList.php");
// set this to true if uploads need to be anonymous
$wgFileListConfig['upload_anonymously'] = false;
// parser functions/
require_once("$IP/extensions/ParserFunctions/ParserFunctions.php");
# Semantic Bundle
require_once( "$IP/extensions/SemanticBundle/SemanticBundleSettings.php" );
require_once( "$IP/extensions/SemanticBundle/SemanticBundle.php" );
```

---

4. Das PatternManger-Formular muss an der Struktur der Muster der jeweiligen Mustersprache angepasst werden

Anhang C

Pattern Wiki Templates

Damit auf die Musterseiten der Block mit der semantischen Information angezeigt wird, muss im Wiki Repository folgendes Template angelegt werden:

```
---------------------------------------------------------------------------------------------------------
<includeonly>
{| class="semanticBlock"
|-
! colspan="2" | Classification
|-
| Category || [[:Category:{{#ask:[[-Is from category::{{PAGENAME}}]]|link=none}}|{{#ask:[[-Is from
category::{{PAGENAME}}]]|link=none}}]]
|-
| Class || {{#ask:[[-Is from class::{{PAGENAME}}]]}}
|-
| SubClass || {{#ask:[[-Is from type::{{PAGENAME}}]]}}
|- style=""
! colspan="2" | Relations to other patterns
|- style=""
{{#if: {{#ask:[[-Can be used with::{{PAGENAME}}]]|format=ul}}
| {{!}}Can be used with {{!}}{{!}}{{#ask:[[-Can be used with::{{PAGENAME}}]]|format=ul}}
{{!-}} style=""
|}}
{{#if: {{#ask:[[-Is similar to::{{PAGENAME}}]]|format=ul}}
| {{!}}Is similar to {{!}}{{!}}{{#ask:[[-Is similar to::{{PAGENAME}}]]|format=ul}}
{{!-}}
|}}
! colspan="2" style="" | Tags / This pattern improves
|- style=""
| colspan="2" class="patternRelations" | {{#ask:[[-
Improves::{{PAGENAME}}]]|link=none|format=template|template=Tag Result Template}}
|-
! colspan="2" style="" | Used in solutions
|- style=""
{{#if: {{#ask:[[Solution uses::{{PAGENAME}}]]|format=ul}}
| {{!}}Solution{{!}}{{!}}{{#ask:[[Solution uses::{{PAGENAME}}]]|limit=3|format=ul}}
{{!-}}
|}}
|}
</includeonly>
---------------------------------------------------------------------------------------------------------
```

Zusätzlich werden folgenden zwei Templates:

```
---------------------------------------------------------------------------------------------------------
|
---------------------------------------------------------------------------------------------------------
```

und

```
---------------------------------------------------------------------------------------------------------
|-
---------------------------------------------------------------------------------------------------------
```

In diesem Template sind die verschiedenen semantischen Informationen beschrieben. Weiter wird angegeben, wann diese gezeigt werden sollen. Wie in Kapitel 6 eingeführt, wurden in der Struktur der Muster in Green Business Processes Patterns zwei zusätzliche semantische Felder eingefügt. Diese sind auch bei Erstellung des Templates ausschlaggebend. Sie werden

genau dann angezeigt, wenn diese im Backend nicht leer gelassen worden sind. Diese Information ist die einzige semantische Information neben der Klassifizierung und den Relationen zu den anderen Mustern, die dem Benutzer im Frontend präsentiert wird. Alle anderen semantischen Informationen dienen der Suche und aus diesem Grund sollen sie für den Benutzer versteckt bleiben.

Ausführliche Information über das Anlegen von Templates ist auf dieser Seite zu finden:
http://www.mediawiki.org/wiki/Help:Templates/de

Literaturverzeichnis

[AC11] Agarwal, D., Chen, B, ICML'11 Tutorial on Machine Learning for Large Scale Recommender Systems, Yahoo! Research, 2011

[Ac72] Alexander, C., The Timeless Way of Building, Oxford University Press, 1972

[Bd09] Barrett, D., MediaWiki, O'Reilly Media, 2009

[Bf98] Buschmann, F., Pattern-orientierte Software-Architektur: ein Pattern-System, Pearson Deutschland GmbH, 1998

[BHS07] Buschmann, F., Henney, K., Schmidt, D., Pattern-Oriented Software Architecture, On Patterns and Pattern Languages, John Wiley & Sons, 2007

[Bj01] Brochers, J., A Pattern Approach to Interaction Design, John Wiley & Sons, 2001

[BKM97] Ben-David, S., Kushilevitz, E., Mansou, Y., Online Learning versus Offline Learning, in Machine Learning, Kluwer Academic Publishers, 1997

[BLD12] Brazen, J., Leymann, F., Schumm, D., Ein Ansatz zur Unterstützung des Kostümmanagements im Film auf Basis einer Mustersprache, in Proceedings of Modellierung 2012, Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bamberg, Germany, 2012

[BLR14] Bloching, B., Luck, L., Ramge, T., Data Unser, in http://goo.gl/RXo7Lx, 19.01.2012, Besucht am 22.02.2014

[Bp11] Baumgartner, P., Taxonomie von Unterrichtsmethoden. Ein Plädoyer für didaktische Vielfalt, Waxmann Verlag, 2011

[COR] http://www.exsys.com/exsyscorvid.html

[CS] Classifier showdown, http://blog.peltarion.com/2006/07/10/classifier-showdown/, besucht am 02.02.2014

[DF02] Druzdel, M., Flynn, R., Decision Support Systems, University of Pittsburgh, 2002

[FM13] Figura, M., Gross, D., Die Qual der Wiki-Wahl - Wikis für Wissensmanagement in Organisationen, Pumacy Technologies AG, url: http://goo.gl/OIIn1P, besucht am 10.03.2014, 2013

[Fw12] Fritsch, W., Analytics macht Big Data nutzbar, http://goo.gl/byJzrn, Besucht am 03.03.2014

[FWW06] Fesenmaier, D., Wöber, K., Werthner, H., Destination Recommendation Systems: Behavioral Foundations and Applications, CABI, 2006

[Gi03] Graham, I., A Pattern language for Web Usability, Addison-Wesley, 2003

[GR11] Gamma, E., Helm, R., Johnson, R., Vlissides, J., Design Patterns: Elements of Reusable Object Oriented Software, Addison-Wesley, 2011

[HA06] Heninger, S., Ashokkumar, P., An Ontology-Based meta model for Software Patterns, Univ. of Nebraska-Lincoln Computer Science and Eng., Lincoln, 2006

[Hd06] Huntington, D., "From Information to Answers: Transferring Expertise at the SBA", November 3, 2006, at URL DSSResources.COM.

[HZ09] Hentrich, C., Zdun, U., A Pattern Language for Process Execution and Integration Design in Service-Oriented Architectures, Transaction on Pattern Languages of Programming, Springer, Seiten 136-191, 2009

[KK07] Kim, D., Khawand, C., An approach to precisely specifying the problem domain of design patterns, Journal of Visual Languages & Computing, Nr. 18, Seite 560, Elsevier, 2007

[Km07] Krötzsch, M., Vrandecic, D., Völkel, M., Haller, H., Studera, R., Semantic Wikipedia, in Web Semantics: Science, Services and Agents on the World Wide Web archive, Vol. 5 Issue 4, December, 2007, Seiten 251-261, Elsevier Science Publishers B. V. Amsterdam, The Netherlands, 2007

[KP10] Kumar, K., Prabhakar, T., Pattern-Oriented Knowledge Model for Architecture Design, in Proceeding PLOP '10 Proceedings of the 17th Conference on Pattern Languages of Programs Article No. 23, ACM New York, USA, 2010

[KZ07] Kampffmeyer, H., Zschaler, S., Finding the Patter You Need: The Design Pattern Intent Ontology, Proceeding MODELS'07 Proceedings of the 10th international conference on Model Driven Engineering Languages and Systems, Seiten 211-225, Springer-Verlag Berlin, Heidelberg, 2007

[LA] http://lucene.apache.org/core/

[LD13] Davidkov, L., Analyse und Prognose von Umweltdaten in Geschäftsprozessen, Institut für Architektur von Anwendungssystemen, Universität Stuttgart, 2013

[LMM04] Little, J., Models and Managers: The Concept of a decision calculus,in Management Science, Seite 1843, Dec 2004

[MD96] Meszaros, G., Double, J., Meta Patterns: A Pattern Language for Pattern Writing, in Proceedings of the Conference on Pattern Language of Programming, Illinois, Sept. 4-6, 1996.

[Mf03] Fowler, M., Patterns of Enterprise Application Architecture, Addison-Wesley Professional, 2003

[Mh98] Maurer, H., Web-based knowledge management, in Computer Vol.31, Issue 3, 1998

[MM05] Gupta, A., Malik, A., Management Information Systems, Firewall Media, 2005

[MM06] Mahemoff, M., Ajax Design Patterns, O'Reilly Media, 2006

[MMD12] Minelli, M., Chambers, M., Dhiraj, A., Big Data, Big Analytics: Emerging Business Intelligence and Analytic Trends for Today's Businesses, John Wiley & Sons, 2012

[MRS08] Manning, C., Raghavan, P., Schütze, H., Introduction to Information Retrieval, Cambridge University Press, 2008

[MS03] Matsatsinis, N., Siskos,Y., Intelligent Support Systems for Marketing Decisions, Springer, 2003

[MSA] Manufacturing Skills Australia, Process Improvement, url: http://goo.gl/yWspSh, besucht am 10.04.2014

[NaL13] Nowak, A., Leymann, F., An Overview on Implicit Green Business Process Patterns, Technischer Bericht Nr. 2013/05

[Nj98] Nobble, J., Classifying Relationships Between Object-Oriented Design Patterns, Software Engineering Conference, 1998. Proceedings. 1998 Australian, Seiten 98 - 107, Adelaide, SA, 1998

[NL11] Nowak, A., Leymann, F., Schleicher, D., Schumm, D., Wagner, S., Green Business Process Patterns. In: Proceedings of the 18th Conference on Pattern Languages of Programs PLoP 2011

[NL13] Nowak, A., Leymann, F., Green Business Process Patterns - Part II. In: Proceedings of the 6th IEEE International Conference on Service Oriented Computing & Applications (SOCA 2013)

[NL14] Nowak, A., Leymann, F., Green Enterprise Patterns (to appear). In: Proceedings of the 20th Conference on Pattern Languages of Programs (PLoP), October 23-26, Allerton, IL, USA, 2014

[Oa96] Oberweis, A., Modellierung und Ausführung von Workflows mit Petri-Netzen, Teubner, Stuttgart, 1996

[Pd02] Power, D., Decision Support Systems: Concepts and Resources for Managers, Greenwood Publishing Group, 2002

[Pd06] Power, D., What are examples of decision support systems in global enterprises?, http://goo.gl/pRdY6h, Besucht am 10.02.2014

[Pi09] Peters, I., Folksonomies. Indexing and Retrieval in Web 2.0.,in Knowledge and Information, De Gruyter, Saur: Berlin. 2009

[Rf11] Ricci, F. et al., Recommender Systems Handbook, Springer, 2011

[SB11] Sabherwal, R., Becerra-Fernandez, I., Business Intelligence, John Wiley & Sons, 2011

[SDH98] Sahami, M., Dumais, S., Heckerman, D., Horvitz, E., A Bayesian approach to filtering junk email, *AAAI Workshop on Learning for Text Categorization*, July 1998, Madison, Wisconsin. AAAI Technical Report WS-98-05

[SNK12] Suresh,S., Naidu,M., Kiran, S., An XML Based Knowledge-Driven Decision Support System For Design Pattern Selection, International Journal of Research in Engineering and Technology (IJRET) Vol. 1, No. 3, 2012

[SS05] Schiemenz, B., Schönert, O. Entscheidung und Produktion - Lehr- und Handbücher der Betriebswirtschaftslehre, Oldenbourg Verlag, 2005

[TMW02] Turban, E., McLean, E., Wetherbe, J., Information technology for management: transforming business in the digital economy, J. Wiley, 2002

[UW12] Urbanski, J., Weber, M., Big Data im Praxiseinsatz – Szenarien, Beispiele, Effekte, http://goo.gl/tcxa08, BITKOM, 2012

[Vc09] Vercellis,C., Business Intelligence – Data Mining and Optimization for Decision Making, Wiley 2009,

[WK10] Wetzstein, B., Karastoyanova, D., Kopp, O., Leymann, F., Zwink, D., Cross-Organizational Process Monitoring based on Service Choreographies. In: Proceedings of the 25th Annual ACM Symposium on Applied Computing (SAC 2010); Sierre, Switzerland, 21-26 March, 2010

[WP] http://www.workflowpatterns.com/

[Wt06] Wal, T., Understanding Folksonomy (Tagging that Works), Brighton, England, 2006

[Wt07] Wal, T., Folksonomy, http://vanderwal.net/folksonomy.html, besucht am 10.02.014, 2007

[YT] YouTube, https://www.youtube.com/yt/press/en-GB/statistics.html, besucht am 10.02.2014

[Zh10] Zhang, H., Naïve Bayes Classification, State Key Lab of CAD&CG, 2010

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 übernommene 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.

Stuttgart, den 16. Mai 2014 _____