Institut für Architektur von Anwendungssystemen

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Diplomarbeit Nr. 3190

Metamodell und Plattform für Mustersprachen und Musterkataloge

Philipp Grimm

Studiengang:	Informatik
Prüfer:	Prof. Dr. Frank Leymann
Betreuer:	Dipl.-Inf. Christoph Fehling Dipl.-Inf. David Schumm
begonnen am:	25. Mai 2011
beendet am:	28. Oktober 2011
CR-Klassifikation:	I.5.0, I.6.5, D.2.1, D.2.2

Inhaltsverzeichnis

Kapitel 1

Einleitung

Beim Suchen nach einer Lösung zur Behebung eines Problems wird oft auf bestehende Ansätze und Ideen zurückgegriffen. Die Denkweise des Verknüpfens von Problemstellungen und Lösungsansätzen tritt in vielen Bereichen auf, zum Beispiel in der Architektur [CAA77], in den Wirtschaftswissenschaften [Etz64] und in der Software-Technik [BJ94]. Auch im Umgang mit sozialen Problemen wird häufig auf diese Denkweise zurückgegriffen [New72]. Die Beschreibung von Problem-Lösungs-Paaren, die auf wiederkehrende Probleme in bestimmten Kontexten ein Lösungsschema bieten, werden Muster genannt. [BMR+98, S. 1]

Speziell bei der Entwicklung von großen Softwaresystemen ist es sinnvoll wiederkehrende Problemstellungen, ganze Softwarearchitekturen und deren Komponenten mit Hilfe von Mustern zu lösen und aufzubauen. In den letzten Jahren ist eine ganze Reihe von Mustern, Mustersprachen und Musterkatalogen vor allem für die Softwareentwicklung entstanden, auf die bei der Softwareerstellung zurückgegriffen werden sollte. Solch ein Musterkatalog stellt eine vernetzte Menge von Mustern dar. Seine Aufbau wird durch eine Mustersprache definiert, die das Modell des Musterkataloges darstellt. [LL10, S.429-444]

Die Vernetzung und Beziehungen der Muster in Mustersprachen wird immer wichtiger, da sie einen große Informationsgehalt bieten. Durch die Beziehungen der Muster lassen sich zum Beispiel ähnliche Lösungsansätze oder auch andere Lösungsansätze zu verwandten Problemen zuordnen. Eine Mustersprache wie die von Christopher Alexander [CAA77] weist eine hierarchische Struktur auf, die mit der Gliederung eines Buches vergleichbar ist. In der Mustersprache von Christopher Alexander werden Ähnlichkeits-, Spezialisierung- und Einschlussbeziehungen verwendet, die die Vernetzung der Mustersprache zusätzlich ausprägen. Die Vernetzung und die Beziehungen der Muster ermöglichen und vereinfachen das Kombinieren der Muster. Muster werden oft in mehrere Teile unterteilt, die wiederum von Muster beschrieben werden. Um einen Lösungsansatz zu finden, der vollständig mit Muster beschrieben werden kann, ist es wichtig die Beziehungen und Verbindungen zwischen den Mustern zu kennen. [PHBO10]

Durch mustersprachenübergreifende Vernetzungen und Beziehungen kann beispielsweise die Lösung eines Problems in eine andere Richtung gelenkt werden oder die Betrachtung von anderen Lösungsideen für das bestehende Probleme entdeckt werden. Dies kann das Finden von Lösungen in einer fachfremden Mustersprache beinhalten. Zum Beispiel fanden Dorigo und Caro einen Ansatz zur Lösung verschiedener diskreter Optimierungsprobleme [DC99], indem sie durch das Futtersuchverhalten von Ameisenkolonien inspiriert wurden. Das Verhalten der Ameisen lieferte die Grundlage für ihre Algorithmen. Sie entwickelten ein Framework, das auf ihrem Algorithmus basiert und mit dem Lösungen von diskreten Optimierungsproblemen berechnet werden können. Mit dem Framework lassen sich viele bekannte Probleme lösen, unter anderem die Berechnung des Travelling-Salesman-Problems

oder das Finden von kurzen Wegen in verbindungslosen Netzwerken unter der Berücksichtigung der Netzauslastung.

1.1 Problemstellung und Zielsetzung

Das Ziel dieser Arbeit ist es, ein Metamodell für Mustersprachen zur Strukturierung und Verwaltung zu entwickeln und eine Plattform für Mustersprachen und Musterkataloge zu planen und umzusetzen, die die Verwendung des Metamodells für Mustersprachen ermöglicht. Im Folgenden werden die Anforderungen an das Metamodell für Mustersprachen und an die Plattform für Mustersprachen und Musterkataloge vorgestellt:

Einordung der Grundlagen des Metamodells für Mustersprachen Das Metamodell für Mustersprachen stützt sich inhaltlich auf bekannte und häufig verwendete Mustersprachen, wie zum Beispiel auf das Metamodell von Meszaros und Doble sowie auf zahlreiche Analysen und Kenntnisse von Mustersprachen, die von Buschmann in *Pattern Oriented Software Architecture Volume 5: On Patterns and Pattern Languages* [BHS07] detailliert beschrieben sind.

Basis der Mustersprachen Das Metamodell für Mustersprachen stellt ein Modell für Mustersprachen dar und bietet für die Mustersprachen eine einheitliche Basis, die Beschreibungen der Struktur und der Semantik von Mustersprachen enthält. Durch diese einheitliche Basis der Mustersprachen ist es möglich, Vernetzungen zwischen Mustersprachen aufzubauen. Zusätzlich können die Strukturen und die Bedeutungen der Mustersprachen analysiert und verglichen werden.

Variable Gestaltung der Mustersprachdefinitionen Je nach Inhalt der Muster, die durch eine Mustersprache beschrieben werden, variieren die Definitionen der verschiedenen Mustersprachen. Damit Autoren von Mustern eine geeignete Mustersprachdefinition für ihre Muster finden, erstellten Meszaros und Doble ein Metamodell für Mustersprachen [MD96], das grundlegende Aufgaben für die Erstellung einer Mustersprachendefinition erfüllt. Das Metamodell für Mustersprachen, das in dieser Arbeit beschrieben wird, basiert auf den Ideen von Meszaros und Doble und ermöglicht darüber hinaus das Erstellen von Metamodellerweiterungen. Diese Metamodellerweiterungen werden durch Profilerweiterungen des Metamodells für Mustersprachen ermöglicht. Durch die Erweiterungen des Metamodellprofils für Mustersprachen wird das Metamodell an spezielle Mustersprachen angepasst, die bestimmten Mustersprachdefinitionen gerecht werden müssen.

Modellgetriebener Architekturansatz Das Metamodell für Mustersprachen aus dieser Arbeit soll dem modellgetriebenen Architekturansatz gerecht werden und durch Profilerweiterungen des Metamodells für Mustersprachen erweiterbar sein. Durch die Erstellung eines Metamodells für Mustersprachen, das dem Konzept der modellgetriebenen Architektur zu Grunde liegt, soll die Modellierung von Mustersprachen und Musterkatalogen ermöglicht werden. Aus diesem Grund wird das Metamodell für Mustersprachen in dieser Arbeit in Form eines UML-Diagramms vorgestellt.

Objektorientierter Aufbau Die Struktur des Metamodells für Mustersprachen soll die objektorientierte Betrachtung von Mustersprachen und Mustern erlauben. Die objektorientierte Betrachtungsweise von Mustersprachen und Mustern soll außerdem die Generalisierung und Spezialisierung der Muster und Mustersprachen ermöglichen.

Vernetzungsinformationen In Verbindung mit der objektorientierten Betrachtungsweise von Mustersprachen sollen explizite Strukturen und Beziehungsarten zwischen Mustersprachen und Mustern festgehalten werden, um die Vernetzungsinformationen zu erweitern und die Bedeutung der Beziehung festzuhalten. Die Bedeutungen der Beziehungen sowie die Semantik der Struktur des Metamodells wird in dem Profil des Metamodells für Mustersprachen festgehalten.

Verwendung des Metamodells für Mustersprachen Das Metamodell für Mustersprachen aus dieser Arbeit soll die Basis für Mustersprachen und Musterkataloge bieten, die durch die Modellierung mit Softwareprogrammen erstellt werden können. Eine Plattform für Mustersprachen und Musterkataloge soll das Erstellen und Verwalten von Mustersprachen und Musterkatalogen bezüglich der modellgetriebenen Softwareentwicklung ermöglichen. Anschließend sind die Anforderungen an die Plattform für Mustersprachen und Musterkataloge aufgeführt.

Plattform für Mustersprachen und Musterkataloge Im Rahmen dieser Arbeit wurde eine Plattform für Mustersprachen und Musterkataloge geplant und implementiert. Sie basiert auf dem Metamodell für Mustersprachen und stellt ein Mustersystem dar, mit dessen Hilfe Mustersprachen erstellt und verwaltet werden können. Aus den Mustersprachen, die in der Plattform für Mustersprachen und Musterkatalogen hinterlegt sind, können Musterkataloge instanziiert und verwaltet werden.

Anforderungen an ein Mustersystem Bei der Planung der Plattform für Mustersprachen und Musterkataloge sollen die folgenden Anforderungen von Buschmann an ein Mustersystem berücksichtigt werden:

- Die Plattform für Mustersprachen und Musterkataloge ermöglicht das Verwalten einer großen Anzahl von Mustern.
- Die Plattform für Mustersprachen und Musterkataloge bietet das Einpflegen von Mustern in die jeweiligen Musterkataloge an, damit die Problematik der Musterkataloge detailliert beschrieben und gelöst werden kann.
- Das Metamodell für Mustersprachen aus dieser Arbeit wird als Grundlage verwendet, um die Muster der Mustersprachen auf eine einheitliche Art und Weise beschreiben zu können.
- Die einheitliche Beschreibungsgrundlage der Muster ermöglicht es, die Inhalte der Muster übersichtlich zu präsentieren und die Muster untereinander zu vergleichen.
- Das Metamodell für Mustersprachen ermöglicht es, verschiedene Beziehungen zwischen Musterkatalogen, Mustern und deren Inhalten in der Plattform für Mustersprachen und Musterkatalogen zu erstellen.

- Damit die Muster in den Musterkatalogen geeignet angeordnet und schnell gefunden werden können, stellt die Plattform für Mustersprachen und Musterkataloge eine Verwaltung der Muster und eine Suche nach den Mustern zur Verfügung.
- Die Plattform für Mustersprachen und Musterkataloge kann mit der dynamischen Änderungen der Inhalte von den Musterkatalogen und Mustern umgehen.
- Es sind Schnittstellen für Erweiterungen der Plattform vorgesehen, damit Lösungen für neue Anforderung bereitgestellt werden können.

[BMR+98][S. 359-361]

Erweiterung der Mustersystemanforderungen Die Anforderungen von Buschmann an ein Mustersystem wurden durch zusätzliche Anforderungen erweitert und verfeinert:

- Die Musterkataloge der Plattform für Mustersprachen und Musterkataloge bieten eine komfortable und effiziente Nutzung für Autoren und Leser.
- Aufgrund der dynamischen Erstellung und Veränderung der Muster können verschiedene Versionen der Musterkataloge und deren Muster gespeichert werden. Die Leser und Autoren können durch das Festhalten der Inhaltsänderungen die inhaltlichen Entwicklungen der Lösungen nachvollziehen.
- Die Plattform für Mustersprachen und Musterkataloge ermöglicht zusätzlich eine erweiterte Suche, die nach Musterkatalogen, Mustern und Musterinhalten suchen kann. Sie stellt außerdem die Zusammenhänge des Suchergebnisses dar.
- Leser und Autoren können die Musterkataloge der Plattform für Mustersprachen und Musterkataloge durchstöbern und dabei ihren Vernetzungen folgen.
- Die Plattform für Mustersprachen und Musterkataloge stellt eine Basis für die Diskussion über die Inhalte der Musterkatalge, Muster und deren Inhalte zur Verfügung.
- Um die Daten der Mustersprachen und Musterkataloge effizient verwalten zu können, besitzt die Plattform für Mustersprachen und Musterkataloge ein Repository, das die Verwaltung der Daten übernimmt.

1.2 Aufbau der Arbeit

Die Arbeit besteht aus der Erklärung der Grundlagen, der Beschreibung des Metamodells für Mustersprachen und dessen Verwaltungssoftware, sowie der Plattform für Mustersprachen und Musterkataloge. Im Anschluss folgt eine Zusammenfassung mit Ausblick des behandelten Themas. Im Folgenden wird auf den Inhalt der einzelnen Bestandteile der Arbeit eingegangen.

Einleitung Die Einleitung dieser Arbeit führt den Leser in die Thematik ein und beschreibt diese in wenigen Worten. Sie setzt sich außerdem mit der Aufgabenstellung und der Zielsetzung auseinander und schildert den Aufbau dieser Arbeit.

Grundlagen Die Grundlagen enthalten fachspezifische Informationen über Mustersprachen, die für das Verständnis diese Arbeit nützlich sind. Es werden die wichtigsten Begriff erklärt, die als Hintergrundwissen der Arbeit benötigt werden. Die vorgestellten Arbeiten, deren Inhalt mit Mustersprachen zusammenhängt, stellt eine notwendige Grundlage für das Kapitel Metamodell für Mustersprachen dar. Da anhand deren Inhalten das Metamodell für Mustersprachen begründet und aufgebaut ist.

Metamodell für Mustersprachen Im Kapitel Metamodell für Mustersprachen wird das Metamodell für Mustersprachen aufgebaut und beschrieben. Dieses wird anhand eines erstellten UML-Diagramms erklärt.

Plattform für Mustersprachen und Musterkataloge Im dritten Kapitel wird die Plattform für Mustersprachen und Musterkataloge vorgestellt. Es werden ihre Aufgaben und Anforderungen beschrieben und eine Abgrenzung zu anderen Softwaremusterkatalogen vorgenommen. Im Anschluss wird die Plattform spezifiziert und deren Struktur vorgestellt.

Zusammenfassung und Ausblick In der Zusammenfassung wird das erstellte Metamodell für Mustersprachen und die Plattform für Mustersprachen und Musterkataloge reflektiert und einen Ausblick auf Erweiterungen vorgestellt.

Anhang Im Anhang dieser Arbeit befinden sich die vollständigen UML-Diagrammen des Metamodells für Mustersprachen, eine Beispielmustersprache und die Profilerweiterung des Metamodells für Mustersprachen für die Anpassung an die Plattform. Zusätzlich ist ein Literaturverzeichnis, ein Abbildungsverzeichnis und ein Verzeichnis der Algorithmen abgebildet.

Kapitel 2

Grundlagen

In diesem Kapitel werden die für diese Arbeit benötigten fachlichen Grundlagen und relevanten Vorarbeiten vorgestellt und diskutiert.

Es werden Mustersprachen und deren Strukturen betrachtet, um einen Ausblick zu geben, was das zu erstellende Metamodell der Mustersprachen in dieser Arbeit leisten soll. Hierzu wird die erste populäre Mustersprache von Christopher Alexander [CAA77] vorgestellt. Anschließend werden Mustersprachen betrachtet, die inhaltlich der Architektur und dem Design von Software zu zuordnen sind.

2.1 Begriffserklärung von Mustern

Die bekanntesten Definitionen von Mustern (engl. Pattern) haben alle die gleiche Kernaussage: ein Muster beschreibt einen Lösungsansatz für ein wiederkehrendes Problem und dessen Kontext. Wichtig dabei ist, dass der Lösungsansatz der Muster unendlich oft auf das beschriebene Problem anwendbar ist ohne den Lösungsansatz ändern zu müssen. Je nach Fachrichtung und Mustersprache variiert jedoch das Format der Muster. Im Folgenden werden bekannte Definitionen von verschieden Autoren vorgestellt, um ein Verständnis der Muster und ihrer Sprachen zu bieten.

2.1.1 Muster nach Christopher Alexander

Der Architekt, Professor und Gesellschaftstheoretiker Christopher Alexander legte den Grundstein für das Konzept der Muster und der Mustersprachen mit seinen Büchern: *A Pattern Language* [CAA77] und *A Timeless Way of Building* [CAA77]. [HW04, S. xli] Die Bücher von Christopher Alexander befassen sich erstmals mit der Beschreibung von Mustern, welche sich auf den Sachverhalt der Architektur, das Bauwesen und der Planung beziehen. In seinem Buch *A Pattern Language* beschreibt er eine Mustersprache. Diese Mustersprache besteht aus Mustern, die Städte, Gebäude und Bauformen beschreiben. An Hand ihrer Information können konkrete Konstrukte abgeleitet und entworfen werden. Christopher Alexander versteht unter einem Konstrukt ein architektonisches Werk, das aus der Konkretisierung mehrere Musterlösungen von zusammenhängenden Mustern ableitbar ist. Unter der Ableitung von Mustern versteht man das Anwenden der Musterlösungen durch eine Projektion der Musterlösung auf die zu entwerfende Lösung. Das Buch *A Timeless Way of Building* behandelt die fundamentale Natur der Tätigkeiten, um Städte und Gebäude zu errichten. Es beschreibt unter anderem die Vorgehensweisen der Erstellung von Architekturen anhand einer Mustersprache. [CAA77, S. ix-x]

Musterdefinition

> Each Pattern describes a problem which occurs over and over again in our environment, and then descibes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice. [CAA77, S. x]

Musteraufbau Christopher Alexander hat seine Muster in Textform festgehalten. Der Einfachheit halber und der Verständlichkeit wegen haben alle Muster das gleiches Format. Der Inhalt wird mit einem Bild eingeleitet, das ein architektonisches Beispiel des zu beschreibenden Musters darstellt. Davon gefolgt wird ein Paragraph, der die Zusammenhänge des zu beschreibenden Musters festlegt und dieses somit in einen Kontext einordnet. Anschließend folgen drei Karos, die als Symbole dargestellt werden, damit der Beginn und die Abgrenzung zur der Problembeschreibung erkenntlich ist. Danach folgen zwei hervorgehobene Sätze, die die Überschrift für die anschließenden Problemstellung darstellen und deren Inhalt kurz erläutern. In der Problemstellung wird das Problem erläutert und in einen Kontext gestellt. Dies wiederum wird von einer weiteren Überschrift, die sich auf den folgenden Lösungsansatz bezieht erweitert. Der Lösungsansatz wird von einer Grafik gefolgt, die einen Überblick der Lösung schaffen soll. Dieser Hauptteil des Musters wird erneut durch drei Karosymbole getrennt. Der folgende Paragraph enthält die Verbindung zu anderen Muster und schließt das Format der Muster ab. [CAA77, S. x-xi]

Christopher Alexander hat aus den zwei folgenden Gründen ein Format für seine Muster gewählt:

- Das Format ermöglicht es Muster miteinander zu verknüpfen. Außerdem soll bei der Wahl eines Konstruktes aus mehreren Mustern eine Vielzahl von Möglichkeiten bestehen um das gewünschte Konstrukt zu gestalten.
- Zweitens soll das Format dem Leser die Problematik und den Lösungsansatz in einer Art wiedergeben, damit dieser die Kernaussage für sich verwenden kann.

[CAA77, S. xi]

Jedes Muster von Christopher Alexander weist eine Beziehung zu einem anderen auf. Die Mustersprache ist in eine hierarchische Baustruktur unterteilt, die zugleich eine Art Inhaltsangabe seines Buches darstellt. Die Knoten des Baumes repräsentieren disjunkte Mengen von Mustern. Jede Menge ist für einen Inhaltsbereich der Mustersprache vorgesehen. Die Wurzel des Baumes stellt die Menge der Muster für Regionen dar, welche von Städten, Nachbarschaften, Gruppierungen von Gebäuden, ... bis hin zu Tischen und Stühlen gefolgt werden. Diese Struktur birgt indirekt Inklusions-, Ähnlichkeits- und Spezialisierungsbeziehungen in sich. [vv03] Christoph Alexander stellt sich seine Muster als eine fundamentale ”Welt” vor, die seine Mustersprache repräsentiert. Keines der Muster steht alleine da und ist Teil der ganzen ”Welt”. Er unterteilt die Ansicht seiner ”Welt” in drei Perspektiven:

- Muster, die kleinere enthalten.
- Muster, die gleich groß und in den gleichen Inhaltsbereich eingeordnet sind
- Muster, die klein sind und Teile von größeren Mustern bilden.

Man kann nicht einfach ein Muster isoliert betrachten und dieses in einer konkreten Lösung verwenden. Die Lösung eines Problems steckt in der Vernetzung der Mustersprache. [CAA77, S. xiii]

Den Grad der invarianten Eingenschaft zwischen Problemstellung und Lösungsansatz wird in den Mustern von Christopher Alexander explizit durch zwei, einen oder keinen Stern angegeben. Dieser Grad versucht eine Abschätzung der Allgemeingültigkeit zwischen Lösungsansatz und Problemstellung darzustellen. Eine totale Erfüllung der invarianten Eigenschaft würde bedeuten, dass der gegebene Lösungsansatz eines Musters immer auf die Problemstellung anwendbar ist. Die invariante Eigenschaft ist nicht erfüllt, wenn es alternative Lösungsvorschläge gibt die das Problem des Musters lösen könnten. [CAA77, S. x-xvii]

Anwendung der Muster Christopher Alexander hat eine Mustersprache entwickelt, die von sehr praktischer Natur ist. Die Sprache entstand durch das Festhalten von seinen Gebäude- und Plangungserrungenschaften. Sie ist dafür gedacht, um seine eigene Stadt und Nachbarschaften zu verbessern, um selber sein Haus oder ein Büro zu planen, bei öffentlichen Gebäuden mitzuwirken und vieles mehr. Es ist vor allem eine Anleitung um dem eigentlichen Prozess der Konstruktion zu folgen. Die Mustersprache ist nicht nur für Experten gedacht sondern explizit für jedermann. [CAA77, S. x]

Um eine konkrete Lösung eines Problems anhand der Mustersprachen zu finden, muss das konkrete Problem auf eine allgemeine Problemstellung eines Musters zutreffen. Man kann nun den Lösungsansatz des Musters ausarbeiten und als konkrete Lösung verwenden. Bei der Wahl des Musters ist zu beachten, dass es alternative Lösungsansätze geben kann. Dies wird in der Mustersprache von Christopher Alexander durch den invarianten Grad angegeben. Die meisten Muster sind Aggregation oder Erweiterungen mehrerer untergeordneten Muster. Bei der Referenzierung von Mustern besteht oft ein Wahlmöglichkeit des Musters, das aus einer Menge verschiedener in Frage kommender Muster gewählt werden kann. Bei der Ausarbeitung einer Lösung müssen auch die untergeordneten Muster gewählt und ausgearbeitet werden, bis schließlich die komplette Lösung vollständig ist. [CAA77, S. xiii-xv]

2.1.2 Muster nach 'The Gang of Four'

'The Gang of Four' besteht aus den Autoren Gamma, Helm, Johnson und Vlissides, welche die Väter der Entwurfsmuster (engl. Design Patterns) sind.

Entwurfsmuster sind Muster deren Inhalt sich auf Softwarekomponenten bezieht. In der Softwareentwicklung wird zwischen Architekturmuster, welche auf der Ebene der Systemarchitektur inhaltlich angesiedelt werden, und den Entwurfsmustern unterschieden. Es werden auch für spezielle Softwarekomponentenarten Mustersprachen erstellt, so wie zum Beispiel Martijn van Welie und Gerrit C. van der Veer eine Mustersprache für Benutzerschnittstellen erstellt haben. [vv03] Entwurfsmuster werden für die Lösungen gängiger Entwurfsprobleme auf der Ebene des Feinentwurfs von Softwarekomponenten angewendet. Sie werden unabhängig von Programmiersprachen entwickelt und vermitteln lediglich eine Lösungsidee. Die meisten Entwurfsmuster beziehen sich auf eine objektorientierte Lösung, was keine freie Wahl der Programmiersprache einschränkt. Das Prinzip der Wiederverwendung ist in dem objektorientierten Denken verankert und lässt sich gut kombinieren. Um einen guten Softwareentwurf zu erstellen, ist die wiederholte Verwendung von Architektur- und Entwurfsmuster essentiell. [LL10, S. 436-437]

'The Gang of Four' beschreiben in ihrem Buch *Design Patterns: Elements of Reusable Object-Oriented Software* [GHJV95] die Grundsätze der Entwurfsmuster und bieten dem Leser einen

Entwurfsmusterkatalog mit einer Fülle von Entwurfsmustern. Das Buch liefert zwei Kernbestandteile. Zum einen zeigt es welche Rolle die Entwurfsmuster in der Architektur von komplexen Systemen spielen und zum anderen bietet es ein Referenzwerk, welches eine Menge dieser ausgereiften Entwurfsmuster darstellt. [GHJV95, S. xiii]

Musterdefinition 'The Gang of Four' benötigte eine Definition für Entwurfsmuster, die es erlaubt ihren Mustern mit objektorientierten Inhalten und Beziehungen darzustellen. Ihre Lösungen stellen sie durch Objekte und Schnittstellen dar, anstatt von Wänden und Türen wie es Christopher Alexander tat. Weil in Christopher Alexanders Muster die selbe objektorientierte Idee mit Wänden und Türen verkörpert, wie 'The Gang of Four' für ihre Muster benötigte, wurde die Definition von Christopher Alexander als Grundlage ihrer Muster verwendet. [GHJV95, S. 2-3]

Musteraufbau Die Bestandteile der Muster von 'The Gang of Four' bestehen aus einem Musternamen, einer Problembeschreibung, einem Lösungsansatz und einem Kompromiss. Der Mustername beschreibt das Problem, die Lösung oder den Kompromiss des Musters. Das Muster wird auch durch seinen Namen referenziert. Das Problem beschreibt, wann die Lösung anwendbar ist und stellt sich selbst in einen Kontext. In der Problembeschreibung werden zum Beispiel Klassen oder Objektstrukturen erläutert, die einen nicht flexiblen Entwurf aufweisen. Manchmal wird eine Liste von Bedingungen angegeben, die erfüllt werden muss bevor es Sinn macht die Musterlösung zu verwenden. Die Lösung beschreibt einen Lösungsansatz bezüglich der angegeben Problemstellung. Der Lösungsansatz enthält eine abstrakte Beschreibung der Entwurfsidee, die durch Beziehungen, Verbindlichkeiten und Zusammenwirkungen dargestellt wird. Der Kompromissbereich beschreibt die Vorteile und Nachteile der gefundenen allgemeinen Lösung im Gegensatz zur einer intuitiven oder einfachen Lösung. Es werden oft Kapazitäten- und Geschwindigkeitsvergleiche getroffen. Es wird eine über die Wahl der Programmiersprache, der Implementation und über die Einflüsse der Flexibilität, Erweiterbarkeit und Portabilität eines Systems bezüglich der Musterlösung diskutiert. [GHJV95, S. 3]

Anwendung der Muster Die Anwendbarkeit der Muster von 'The Gang of Four' lehnt sich stark an der von Christopher Alexander an. Die Anwendung der Muster von 'The Gnag of Four' bringt immer eine softwaretechnische Implementierung hervor. Das besondere hier ist, dass nicht nur die Muster sondern auch die Implementierung, die aus ihnen hervorgeht, wiederverwendbar sein soll. In den Mustern sind Beispiele angegeben, die schon eine Stütze bieten sollen, um die Implementierung leichte erstellen zu können. [GHJV95, S.3-4]

2.1.3 Muster nach Frank Buschmann

In diesem Abschnitt wird die Musterdefinition und der Musteraufbau von Frank Buschmann aus seinem Buch *Pattern-orientierte Software-Architektur: Ein Pattern-System* [BMR+98] vorgestellt.

Buschmann führt in seinem Buch *Pattern-orientierte Software-Architektur: Ein Pattern-System* den Begriff des Musters ein und diskutiert die Prinzipien einer Musterbeschreibung. Anschließend präsentiert er drei Musterkataloge, die Muster zu den Themenbereichen Architektur und Entwurf von Software. Der dritte Musterkatalog beinhaltet und verweist auf Idiome. Idiome stellen

programmiersprachen spezifische Muster dar und werden eine Abstraktionsebene unter den Entwurfsmustern angesiedelt.

Buschmann beschreibt die Bedeutung und die Verwendung von Mustersystemen und deren Organisation. Die Mustersysteme helfen Muster für die richtige Situation zu finden, Lücken zwischen Muster zu schließen und Beziehungen zwischen Mustern zu bilden. Er diskutiert die Verwendung von Musterkatalogen und Musterorganisationen in dem Gebiet der Software-Architektur und erklärt die Unterstützung von der Verwendung von Mustern. [BMR+98][S. XVI-XVII, 14-16]

Musterdefinition Buschmann verwendet eine Musterdefinition um Problematiken und deren Lösungen aus dem Bereich Software-Architektur und Software-Entwurf zu beschreiben. Die Musterdefinition von Buschmann ist aus der Musterdefinition von Christopher Alexander und der Musterdefinition von 'The Gang of Four' abgeleitet und an seine drei Themen angepasst. Er unterteilt seine Muster in die Kategorien Architekturmuster, Entwurfsmuster und Idiome. Die Muster aus den drei Kategorien basieren alle auf der selben Musterdefinition. Die Verwendung der selben Musterdefinition führt zu einem einheitlichen Musterformat und soll das Vergleichen von Mustern und das Auffinden von alternativen Lösungen erleichtern. [BMR+98][S. 8-21]

Musteraufbau Im Folgenden werden die Bestandteile der Muster von Buschmann aufgelistet und deren Verwendung erläutert. Die Muster bestehen aus einem Namen, alternativen Bezeichnungen, einem Beispiel aus der Realität, einem Kontext, einem Problem, einer Lösung, einem dynamischen Aspekt, einer Beschreibung für Implementierungsrichtlinien, einer Diskussion aller wichtigen Aspekte für die Lösung des Beispiels, einer kurzen Beschreibung der Varianten und Spezialisierungen der Muster, einem Anwendungsbeispiel existierender Software, einer Beschreibung der Vor- und Nachteile der Verwendung und Verweise auf andere Muster oder auf ähnliche Probleme.

2.1.4 Muster nach Martin Fowler

Um das Erstellen von komplexen Unternehmensanwendungen (engl. Enterprise Application) zu erleichtern, stellt Martin Fowler in seinem Buch [Fow02] eine Mustersprache für die Architektur von Unternehmensanwendungen bereit. Er kann keine genaue Definition von der Unternehmensanwendung liefern und gibt deshalb Indikatoren an, die dem Leser die Bedeutung und Gestalt von Unternehmensanwendungen nahe legen sollen. Die Musterinhalte beziehen sich zum Beispiel auf Probleme, die mit folgenden Aspekten in Verbindung stehen: persistente Daten, gleichzeitiger Zugriff auf die selben Daten, eine Menge von Benutzerschnittstellen, Integrationen von anderen Unternehmensanwendungen, konzeptionelle Unstimmigkeiten von IT-Strukturen und komplexe Geschäftslogiken, ... Um ein tieferes Verständnis zu erlangen, ist der Leser eingeladen diese Erläuterung von Martin Fowler nachzulesen. [Fow02, S. 1-4]

Musterdefinition Martin Fowler nimmt ebenfalls die Definition von Christohper Alexander als Grundlage. Laut ihm sind seine Muster von Unternehmensanwendungen im Gegensatz zu den Mustern von Christopher Alexander keiner ursprüngliche Idee, sondern sehr viel eher aus Betrachtungen von Geschehnissen in bestimmten Bereichen entstanden.[Fow02, S. 9-11]

Musteraufbau Bei der Findung seines Musterformats hat sich Martin Fowler an den Büchern: *A Language Pattern* [CAA77], *Design Patterns* [GHJV95] und *Pattern-Oriented Software Architecture* [BMR+98] orientiert. Sein Musterformat setzt sich letztendlich aus dem Musternamen, der das Muster identifizieren soll, einer ein bis zwei Sätzen großer Zweckbeschreibung mit dazugehöriger Skizze. Die Skizze ist oft ein UML-Diagramm, dass einen Überblick des Musterinhaltes verschaffen soll. Nach der Skizze folgt ein Absatz, der die Motivation des Musters enthält. Die Lösungsbeschreibung beinhaltet, eine Diskussion über die Implementierungsprobleme und -variationen. Der Inhalt ist so gut wie möglich unabhängig bezüglich einer Plattform gehalten. Optional sind plattformspezifische Lösungen angefügt. Dies wird gefolgt von einem Abschnitt der beschreibt, wann die Musterlösung angewendet werden kann und wann es Sinn macht diese anzuwenden. In einem weiteren optionalen Abschnitt werden hilfreiche Referenzen zu weiteren Informationen über den Sachverhalt angeboten. Am Schluss des Musterformates erläutert ein Beispiel in Java-Code die Lösung praktisch. Der Zweck des Beispielcodes ist es, die Idee der Musterlösung verständlicher darzustellen. Nicht jeder Teil des Musterformates ist verpflichtend für die Beschreibung der Muster. [Fow02, S. 11-13]

Anwendung der Muster Martin Fowler nimmt vorweg, dass die Anwendung seiner Muster meistens auch eine Anpassung derer verlangt. Die Muster von Martin Fowler eignen sich nicht so gut für eine Wiederverwendung. Sie sind aber für die Dokumentation von Software und dem Austausch mit anderen von großer Bedeutung. Durch die Referenzierung der Muster anhand ihres Namens kann jeder die Bedeutung und den Sinn des Musters verstehen oder nachschlagen. Abgesehen von der Anpassung der Muster bezüglich des eigenen Kontextes verhält sich das Vorgehen genauso wie bei Christopher Alexander. [Fow02, S. 9-11]

2.1.5 Muster nach Hohpe und Woolf

Hope und Woolf verfassten ein Werk [HW04], das eine Mustersprache bezüglich der Integration von Unternehmensanwendungen liefert. Interessante Anwendungen werden nur selten als alleinstehende Anwendung betrieben. Die meisten Anwendungen bieten und verwenden Schnittstellen zu dritten Anwendungen. Zum Beispiel muss in einem Unternehmen eine Finanzanwendung mit der dazugehörigen Lagenbestandshaltungsanwendung, Auftragsabwicklungsanwendung und einem Webshop kooperieren. Um ein Gefühl zu bekommen, was die Muster von Hope und Woolf inhaltlich thematisieren ist eine List fundamentaler Herausforderungen der Integrationslösungen aufgeführt: nicht zuverlässige Netzwerke, langsame Netzwerke und unausweichlichen Veränderungen der Anwendungen. Um ein tieferes Verständnis für die aufgeführten Herausforderungen der Integrationslösungen zu bekommen, ist der Leser eingeladen, die Erklärungen von Hope und Woolf nachzulesen. [HW04, S. xxix-xxx]

Musterdefinition Die Definition von Hope und Woolf stützt sich ebenso vollständig auf die von Christopher Alexander. Hope und Woolf möchten die Eigenschaften der Wiederverwendung und Lösungsfindung von Christopher Alexanders Muster und deren Vernetzung in ihren Mustern verwenden. [HW04, S. xli]

Musteraufbau Hope und Woolf haben eine Musterform gewählt, die sich nah an der von Christopher Alexander orientiert. Sie wollten eine prosaische Musterform ohne Überschriften von Untergruppierungen für ihre Muster, um den Diskussionsfluss nicht zu unterbrechen. Durch Abbildungen, unterstrichene oder fett gedruckte Texte sollen wichtige Informationen hervorgehoben werden. Die Struktur der Muster besteht aus folgenden Bestandteilen: Mustername, einem eindeutig zuordnungsbaren Symbol für die Musterrepräsentation, einer Beschreibung des Kontextes, einer Problembeschreibung, einen Abschnitt, der die Schwierigkeiten der Problemlösung, Alternativlösungen und Irrwegen anspricht, einer Lösung, eine lösungsbeschreibende Skizze, einen Abschnitt, der die Anwendung der Lösung beschreibt, eine Beschreibung der Anwendung von in Verbindung stehender Muster, einen Abschnitt, der auf die technischen Details eingeht und zum Abschluss ein Beispiel, das die Lösung praktisch veranschaulichen soll. [HW04, S. xli-xlii]

Anwendung der Muster Die Anwendung der Muster von Hope und Woolf ist gleich zu der von Christoper Alexander. Die Lösungen der Muster von Hope und Woolf bieten meistens nicht die Lösung, die einem gleich in den Sinn kommt. Damit wollen sie sagen, dass die Lösungen einmal generell gehalten sind. Die festgehaltenen Lösungen sind aber darüber hinaus schon über Jahre hinweg erprobt und beinhalten keinerlei Konzeptfehler. Die Spezialisierungen von Mustern sind natürlich dem Anwender überlassen, werden ihm aber nicht empfohlen, weil eine Architektur nicht auf Performance optimiert werden soll. Die Musterlösungen sind wie die von Fowler [Fow02] für objektorientierte Programmiersprache und beliebige Plattformen ausgelegt. Eine Musterlösung muss also in den Kontext der eigenen Umgebung gestellt und bezüglich einer beliebigen Programmiersprache implementiert werden. Hope und Woolf bieten eine Vielzahl von Beispielen die unterschiedliche Arten von Implementationen vorstellen. [HW04, xxli-xxlii]

2.2 Begriffserklärung von Modell und Metamodell

Modell Modelle sind Abbildung von oder Vorbilder für etwas. Modelle, die Abbilder sind, werden deskriptiv Modelle genannt. Modelle, die Vorbilder für etwas sind, werden präskriptive Modelle genannt. Zu einem Modell kann ein Original oder Gegenstück vorhanden, geplant oder fiktiv sein. Ein Modell enthält nicht alle Attribute seines Originals. Es stellt eine Verkürzung des Originals und somit auch Raum für eine Abstraktion des Originals dar. Ein Modell kann ein pragmatisches Merkmal besitzen, in dem es sein Original durch bestimmte Eigenschaften eindeutig zugeordnet. Modelle können für viele Absichten eingesetzt werden. Bezogen auf das Einsatzgebiet von Software können Modelle zum Beispiel für die Darstellung von formalen (mathematischen) Repräsentationen, für Dokumentationszwecke oder für die Verkörperung von explorative Modelle eingesetzt werden. Modelle im Software Engineering unterteilen sich in zwei grundlegende Arten: Software-Modell und Vorgehens- und Prozessmodell. Die Software-Modelle verkörpern Modelle, die Softwaresysteme oder deren Modelle Repräsentieren sollen. Die Vorgehens- und Prozessmodell beschreiben, wie man bei der Erstellung von Software vorgehen soll. Software-Modelle werden üblicherweise durch natürlichsprachliche Spzifikationen, Entity-Relationship-Diagramme, User-Case-Diagramme, repräsentiert. [LL10, S. 3-28]

Metamodell Ein Metamodell ist die Beschreibung eines Modell. Durch die Beschreibung eines Metamodells, entsteht eine formale Sprache von Modellen. Ein Metamodell steht eine Abstraktionsebene über den Modellen, die es beschreibt. Durch die Verwendung eines Metamodells können die daraus resultierenden Modelle, einheitlich und kompatibel gehalten werden. Es ist jedoch schwierig die nötige Allgemeinheit und Erweiterbarkeit für die Modelle zu behalten. Metamodelle werden hauptsächlich in der Mathematik, Hard- und Software entwicklung eingesetzt. [Gig91, S. 255], [RJB99, S. S.105-106], [MO99, S. 305-319]

2.3 Begriffserklärung von Metamustern

Unter einem Metamuster (engl. Meta Pattern) versteht man ein Metamodell eines Musters. In dieser Arbeit wird unter einem Metamuster eine allgemeine und abstrakte Definition von Muster verstanden. Das Metamuster befindet sich eine Abstraktionsebene über dem Muster und stellt das Modell des Musters dar.

Im weiteren Verlauf soll die Definition eines Metamusters und Vorschläge für Musterformate von Meszaros und Doble [MD96] vorgestellt werden.

2.3.1 Metamuster nach Meszaros und Doble

Meszaros und Doble stellten in ihrer Arbeit [MD96] Metamuster für die Erstellung einer Mustersprache vor. Die Ideen und Informationen für dieses Metamuster stammen aus den Erfahrungen der Teilnehmer von der Konferenz ”Pattern Languages of Design” aus dem Jahre 1995. Nach dem Überarbeiten einer großen Anzahl von Mustern und Mustersprachen, begannen die Teilnehmer eine Sammlung von Musterschreibtechniken und -annährungen aufzustellen. Sie verfassten ihre Errungenschaften in einer von ihnen entworfenen Mustersprache. Diese Mustersprache bietet Vorschläge und Regeln, um ein Musterformat zu gestalten. Andere Autoren von Mustersprachen sollen das Metamuster verwenden, um schneller und einfacher ein Format für ihre Muster und eine Musterstruktur zu finden.

Im Folgenden wird eine Übersicht der Mustersprachenstruktur der Metamuster vorgestellt. Anschließend wird auf die einzelnen Bestandteile der Lösungsvorschläge für die Metamuster eingegangen. Um einen tieferen Einblick zu erhalten, ist der Leser angehalten Muster in der Arbeit [MD96] nachzulesen.

Übersicht der Metamuster In der Abbildung 2.1 ist eine Strukturübersicht der Mustersprache von Meszaros und Doble zu sehen. Die Mustersprache beschreibt die Metamuster zur Erstellung einer Mustersprache. Die Muster für die Beschreibung der Metamuster sind in fünf Bereiche unterteilt. Die Bereiche sind in der Übersicht mit den Buchstaben von A bis E gekennzeichnet. Sie sind im Gegensatz zu den Unterteilungen, die als Rechtecke in den Bereichen dargestellt sind, in Textform und nicht in einer Musterform von Meszaros und Doble festgehalten worden. Diese Texte beschreiben den Kontext, in dem die jeweilige Bereiche stehen.

Strukturbereiche in der Übersichtsabbildung 2.1:

- Bereich A (Context Setting Patterns) befasst sich mit dem Konzept der Muster (Beziehung zwischen Problem-, Kontext- und Lösungsbeschreibung), dem Konzept der Mustersprache

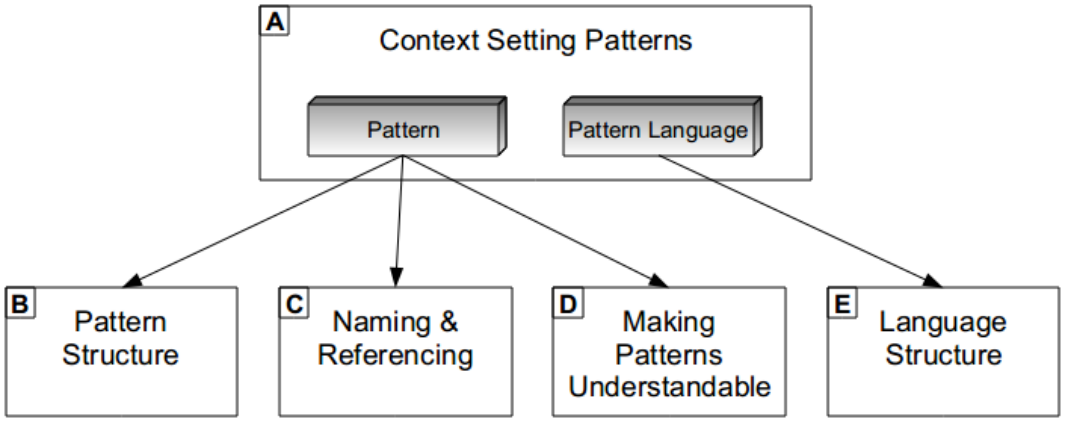


Abbildung 2.1: Strukturübersicht der Metamuster-Mustersprache von Meszaros und Doble

(Sammlung von Mustern und deren Beziehungen) und der Wiederverwendung dieser Sprache und deren Muster.

- Bereich B (Pattern Structure) beschreibt die Struktur und den Inhalt der Metamuster.
- Bereich C (Naming & Referencing) beinhaltet die Namensgebung der Muster und die Referenzierung anhand des Namens der Muster und legt somit einen Grundstein für die Beziehungen der Muster, die in Bereich E diskutiert werden.
- Bereich D (Making Pattern Understandable) liefert Techniken für das Gestaltung von leicht zu lesenden und gut zu verstehenden Mustern.
- Bereich E (Language Structure) bezieht sich auf die Struktur und Inhalt der Mustersprache, die durch die Metamuster gebildet werden.

Bereich A (Context Setting Patterns) Dieser Bereich beschäftigt sich mit der Problemstellung: Wie man wiederkehrende Problem löst und wie man Lösungen beschreibt um sie verständlich und verwendbar präsentieren kann.

Der Bereich A ist eine Überordnung der genannten Bereiche und schließt somit die anderen Bereiche inhaltlich ein. Der Bereich A besteht aus zwei Teilen, der Beschreibung wie ein Muster aufgebaut werden soll und der Beschreibung wie die Struktur der Sprache aus zu sehen hat. Der erste Teil wird durch die Bereiche B, C und D abgedeckt und der zweite Teil durch den Bereich E.

Bereich B (Pattern Structure) Der Bereich B beschreibt die Struktur eines Musters, den Zusammenhang zwischen Problemstellung und Lösungsansatz und deren Kontext. Oft reicht es nicht nur zu beschreiben wieso und warum man die Lösung verwenden muss, sondern auch wie. Die Muster in diesem Bereich befassen sich mit der Struktur der einzelnen individuellen Muster.

Es wird die Möglichkeit eingeräumt zusätzlich zu den Pflichtbestandteilen eines Musters auch optionale Bestandteile, die aber in dem Musterformat vorgesehen sind, einzufügen. Dies hilft dem Autor Muster mit zusätzlicher Information flexibel anzureichern. Damit der Leser mit einmaligen Lesen den

Inhalt des Musters verstanden hat, wird vorgeschlagen äußere Einflüsse festzuhalten. Dadurch soll der Leser die Wahl der Lösung besser nachvollziehen können. Durch die übersichtliche Gestaltung und Gliederung der Muster ist es für den Leser möglich, nur interessante Bereiche des Musters anzuschauen und anschließend beurteilen zu können, ob das Muster für ihn von Relevanz ist, um gegebenenfalls den Rest des Musters zu lesen. Dies kann vor allem sinnvoll sein, wenn der Leser zum Beispiel nach einer Problemstellung oder Lösungsansatz sucht. [MD96]

Bereich C (Naming & Referencing) Nur wenige Muster sind isoliert und haben keine Beziehung zu den anderen. Die meisten Problemen lassen sich mit mehreren kleinen Lösungen oder durch die Spezialisierung einer generellen Lösungen bewältigen. Es kann auch mehrere Lösungen für ein Problem geben. Diese Lösungen werden durch Muster dargestellt, die Alternative darstellen. In vielen Fällen müssen mehrere Muster referenziert werden, um einen Problemstellung zu lösen. Damit Muster auseinander gehalten werden können, wird die Namensgebung benötigt. Anhand der Namen können Muster mittels Referenzen verknüpft werden. Durch die Verknüpfungen lassen sich Alternativen vermerken und Problemstellungen darstellen, die durch mehrere Muster gelöst werden. Dieser Bereich befasst sich mit der Namensgebung und Referenzierung mithilfe von Musternamen, deren Darstellung, Bedeutung und einem internen Referenzkatalog.

Bereich D (Making Pattern Understandable) Je besser der Leser die Muster versteht, desto mehr kann er mit ihnen anfangen und sein Problem auf sie projizieren. Deshalb ist es von großer Bedeutung, die Muster verständlich darzustellen. Der Bereich D befasst sich mit der Wahl von klaren Zielgruppen, verständlichen und eindeutigen Bezeichnungen, Diagrammen, Skizzen und Beispielen.

Bereich E (Language Structure) Dieser Bereich beschäftigt sich mit der Zusammenstellung in Beziehung stehender Muster, um eine zusammenhängende Mustersprache zu schaffen. Damit beispielsweise dem Leser eine Übersicht der Muster bieten zu können. Dies kann in Form einer Zusammenfassung oder Gliederung der Muster bereitgestellt werden. Bei der Referenzierung von Mustern ist es wichtig, dem Leser den Problem-Lösungs-Ansatz und -Zusammenhang nahe zu legen. Damit dieser schnellstmöglich die gesuchte Problemstellung oder Lösung findet. Das Hervorheben von gemeinsamen Problemstellungen, das Übermitteln von aussagekräftigen Referenzen, das Erklären von fortlaufenden Beispielen und das Bereitstellen eines Glossars verbessert die Struktur der Sprache und deren Vernetzungsgrad und fördert somit die Ausdruckskraft des Inhaltes.

2.4 Beziehungen in Mustersprachen

Damit man sich einen Eindruck von der Bedeutung in Mustersprachen verschaffen kann, werden im Folgenden die Beziehungen zwischen Mustern vorgestellt. Sie werden bezüglich ihrer fundamentalen Unterschiede unterteilt und deren Eigenschaften und Verwendungsmöglichkeiten erläutert.

2.4.1 Beziehungen in Mustersprachen nach van Welie und van der Veer

Martijn van Welie und Gerrit C. van der Veer analysierten unter anderem in ihrer Arbeit *Pattern Languages in Interaction Design: Structure and Organization* [vv03] die Mustersprache von Christopher Alexander bezüglich ihrer Struktur und Beziehungen. In diesem Abschnitt sollen die Errungenschaften der Analyse von van Welie und van der Veer erläutert werden. Sie stellen vier verschiedene Beziehungsarten vor und erkennen fundamentale Ähnlichkeiten zu Beziehungen aus der Objektorientierung.

Hierarchische Gliederung bezüglich der Problemstellung in Mustern Wie auch in schon von Christopher Alexander beschrieben weist seine Mustersprache eine hierarchische Vernetzung der Muster auf. Diese hierarchische Vernetzung wird durch den Inhalt der Muster bestimmt. Die hierarchische Struktur der Mustersprache ist mit der Gliederung von Texten in Büchern zu vergleichen. [CAA77, S. x] [vv03] Die hierarchische Struktur der Mustersprache ist die Struktur der Unterteilung von großen Problemen in mehrere kleine. Die großen Probleme sind in der hierarchischen Ordnung oben angesiedelt und werden von ihren direkten Teilproblemen gefolgt. Wenn es erwünscht ist, kann man solch eine Hierarchie auch an anderen Inhalten der Muster aufbauen. [vv03]

Christopher Alexander beginnt auf der Ebene der Städte mit seiner Gliederung. Die ist die oberste Ebene. Sie wird gefolgt von Nachbarschaften, Gebäuden, ... bis hin zu Fenstern und Stühlen. Die Idee von Alexander war es, dass durch eine Traversierung der Mustersprache eine konkrete Lösung abgeleitet werden kann. Das Traversieren der Mustersprachen beginnt bei einem Muster, das die zu lösende Problematik beschreibt, und geht bis in die tiefste Ebene zu den Mustern, die Teillösungen der Problematik darstellen. [vv03]

Künstliche hierarchische Gliederung der Muster Die Beschreibung der hierarchischen Gliederung wird durch die Vernetzung der Muster abgeleitet. Wenn die Gliederung groß wird, wird sie auch unübersichtlich. Zusätzlich wird keine Aussage über den Zusammenhang der Muster in der Gliederung gemacht. Um eine Überblick zu schaffen und die Problemstruktur der Muster und des Kontextes der Mustersprache kenntlich zu machen, erstellen van Welie und van der Veer künstliche Gruppen von Mustern. Diese Gruppen sind auch hierarchisch geordnet und bilden eine Gliederung von den Mustern auf einer höheren Ebene. Im Gegensatz zu der vorgestellten Gliederung kann hier eine Übersicht geschaffen werden. Sie kann die Grundidee des Musterspracheninhaltes strukturiert darstellen. Außerdem bietet die künstliche Gliederung ein übersichtlicheres Zurechtfinden und Suchen in der Mustersprache. [vv03]

Vorschläge für künstliche hierarchische Gliederungskategorien Wie schon erwähnt, kann eine Mustersprache bezüglich der Problemstellungen der Muster und deren Kontext gegliedert sein. [vv03]

Muster können auch nach der Alternative gegliedert werden. Alexander hat für die Lösungen eines Problems mit einer anderen Lösung einen Vermerkt vorgesehen. Muster, die Alternativen zu einem anderen Muster liefern, könnte man Gruppieren und mit diesen Gruppen eine Gliederung aufbauen. [vv03]

Generell kann eine Gliederung mithilfe von verschiedenen Arten von Beziehungen erstellt werden.

Van Welie und van der Veer arbeiten an Benutzeroberflächen und pflegen hierfür eine Mustersprache. Sie haben zum Beispiel eine Gruppierung bezüglich Benutzeraufgaben und Benutzertypen erstellt. [vv03]

Beziehungstypen zwischen Mustern Wie schon in dem vorherigen Absatz angedeutet, sind die Beziehungen in den Mustersprachen das Herzstück. Die Beziehungen bereichern ein einzelnes Muster mit zusätzlichen Informationen. Diese zusätzliche Information macht den Zusammenhang der Muster möglich, den man benötigt um die Mustersprache zu erstellen und zu verwenden. [vv03]

Im Folgenden werden die Beziehungen und die Verbindungen zwischen den Mustern, die deren Zusammenhang schaffen näher betrachtet. Es lassen sich fundamentale Beziehungsunterschiede erkennen, nach denen die Beziehungen in Arten aufgeteilt sind. Diese Arten kennt mach auch aus der Objektorientierung. Die Analyse und Feststellungen von Welie und van der Veer aus ihrer Arbeit *Pattern Languages in Interaction Desing: Structure and Organization* [vv03] wurde in der kürzlich veröffentlichten Arbeit *Building an interaction design pattern language: A case study* [PHBO10] von Paulwels, Hübscher Bargas-Availa und Opwis bestätigt. [vv03]

Aggregation Ein Muster kann ein großes Problem in mehrere kleine Probleme unterteilen, die schon von anderen Mustern gelöst sind. Ein Muster kann somit ein großes Problem lösen, indem es durch die Aggregation mehrerer kleiner Muster eine Lösung bereitstellt. Dieser Typ von Beziehung wird Aggregationsbeziehung oder ”hat-ein”-Beziehung genannt. [vv03]

Die Aggregationsbeziehung wurde von Buschmann noch verfeinert. Buschmann unterteilte die Aggregationsbeziehung von Mustern in seinem Buch *Pattern-Oriented Software Architecture: On Patterns and Pattern Languages* in drei verschiedene Arten. Die erste Art ist die klassische Aggregation, welche die Vervollständigung eines Musterinhaltes durch mehrere andere Musterinhalte darstellt. Die zweite Art der Aggregationsbeziehung sind die Beziehungen, in denen die referenzierten Musterinhalte nicht erforderlich sind aber zum Verständnis beitragen. Die dritte Art der Aggregationsbeziehung ist das gegenseitige Ausschließen von alternativen Musterinhaltsgruppierungen. [BHS07, S.135-163]

Spezialisierung Muster können auch durch die Spezialisierung von anderen Mustern erstellt werden. Das spezialisierte Muster beinhaltet die gleiche Grundidee wie das Muster mit der generellen Lösung. Diese Art von Beziehung wird Spezialisierungsbeziehung oder ”ist-ein”-Beziehung genannt. [vv03]

Assoziation Die dritte Beziehungsart, die van Welie und van der Veer vorstellen ist eine Ähnlichkeitsbeziehung. Zwei Muster können in Beziehung stehen, wenn sie eine gleiche Eigenschaft bezüglich des Inhaltes aufweisen. Hier werden Eigenschaften betrachtet, die eine unmittelbare Verwandtschaft von Mustern bezüglich Aggregation und Spezialisierung ausschließen. Sinn dieser Beziehung ist es Querverbindungen zu schlagen. Diese Ähnlichkeitsbeziehung wird auch Assoziationsbeziehung oder ”ist-ähnlich-zu”-Beziehung genannt. [vv03]

2.5 Webbasierte Musterkataloge und -sammlungen

In diesem Abschnitt werden webbasierte Musterkataloge und -sammlungen vorgestellt. Webbasierte Musterkataloge haben mehrere Vorteil gegenüber gebundenen Musterkatalogen. Es ist beispielsweise einfacher nach Musterinhalten oder deren Ähnlichkeiten zu suchen. Muster können auch in verschiedenen Ansichten und Gruppierungen darstellt werden, die den Lesern auf ihre Bedürfnisse angepasste Übersichten bieten können. Ein webbasierender Musterkatalog kann durch die Verwendung von Hyperlinks das Durchstöbern von Mustern in einer komfortablen Art anbieten, weil viele Muster auf andere verweisen oder deren Inhalt mitverwenden.[vv03]

In der Recherche für diese Arbeit wurden einige webbasierte Musterkataloge gefunden, deren Inhalt sich auf die Erstellung von graphischen Benzutzerschnittstellen und die Webprogrammierung beziehen. Es gibt viel webbasierte Musterkataloge im Web zu finden, deren Inhalte fehlerhaft sind und deren Qualität wegen schlechte Beiträgen leidet. Muster von einigen webbasierte Musterkataloge weisen keine oder ungenügende Inhaltsstrukturen auf. Aus diesen Gründen werden die in Pauwels Arbeit *Buildung an interaction design pattern language: A case study* [PHBO10] erwähnten webbasierten Musterkataloge: Pattern library von van Welie [Van11] und Yahoo! Design Pattern Library [Yah11] beschrieben.

2.5.1 Interaction Design Pattern Library - Welie.com

Der webbasierte Musterkatalog von van Welie bietet eine Menge von Mustern, die bekannte und erprobte Konzepte enthalten. Deren Inhalte befassen sich mit Interaktionen der graphischen Oberflächenbedienung und deren Erstellung. Die Erstellung der Muster der Interaction Design Pattern Library wird nur von van Welie vorgenommen. Er lädt jedoch dazu ein ihm Vorschläge für weitere Muster zu schicken.

Van Welie biete seinen Lesern eine Übersicht, in der er alle Muster in eine zweistufigen Struktur einordnet sind. Beim Durchstöbern der Muster wird einem immer die Kategorie angezeigt, in der sich das abgebildete Muster befindet.

Die Musterstruktur besteht immer aus einer Problembeschreibung, einer Lösung, einem Kontext, einer Beschreibung der Umsetzung und einer Beschreibung der Notwendigkeit und Beispiele. Zusätzlich können Leser Diskussionsbeiträge an die Muster anfügen und über die Musterinhalte debattieren.

Die Muster von van Welie sind durch Hyperlinks, die sich in den Musterinhalten befinden und auf andere Muster verweisen, miteinander vernetzt.

2.5.2 Yahoo! Design Pattern Library

Der webbasierte Musterkatalog von Yahoo! wurde am 13. Februar 2006 veröffentlicht. Yahoo! wollte durch die Veröffentlichung ihrer Designmuster ihr Wissen und ihre Ansichten teilen und bittet zugleich um sinnvolle Rückmeldungen und Beiträge. Sie sehen die Kommunikation und die Wissensverbreitung durch die Veröffentlichung von Designmustern als wichtig und sinnvoll an. [Sco06]

Trotz so vieler vorhandener Musterkataloge entwickelte Yahoo! seinen eigenen Musterkatalog. Sie wollen ihr Wissen, das in vielen kleinen Abteilungen der Firma verteilt war, in dem Musterkatalog vereinen, um einheitliche Designs und Softwarelösungen für die ganze Firma bereit zu stellen. Durch

einheitliche Designs und Softwarelösungen sollen Standards definiert, werden mit denen Yahoo! seine Produkte verbessert und erweitert. Um dies zu erreichen kreierte Yahoo! ein Repository, das Designmuster für Benutzeroberflächen enthält und einen Prozess, der für das Erstellen von qualitativ hochwertigen Muster verantwortlich ist. [LMW05]

Die Musterstruktur von Yahoo! enthält einen Titel, eine Problembeschreibung, eine Beschreibung des Kontextes, eine Lösungsbeschreibung, Beispiele, die durch Abbildungen verdeutlicht werden und eine Beschreibung von Grundsatzgedanken und Verwendbarkeit. Zusätzlich enthält ein Muster einen Blog, in dem Beiträge und Diskussionen über das Muster stattfindet. [Yah11]

Jedes Muster, das in dem Musterkatalog von Yahoo! veröffentlicht wird, muss einen Erstellungsprozess durchleben. Das Yahoo! "Design Reasearch Team" prüft, überarbeitet und verfeinert die Muster. Ein Muster muss drei Stadien durchlaufen: die Betaphase, Lösungserarbeitung und die Einsatzphase. Zusätzlich können Muster bewertet werden, um die Qualität der Lösung und die Flexibilität der Verwendung einzuschätzen. [Yah11]

Die Ideen und Anregungen der Muster für die Veröffentlichung des webbasierten Musterkataloges von Yahoo! basiert auf den früheren Werken:

- *A Pattern Language* von Christopher Alexander [CAA77]
- *Design Patterns: Elements of Reusable Object-Oriented Software* von Erich Gamma, Richard Helm, Ralph Johnson und John M. Vlissides [GHJV95]
- *COMMON GROUND: A Pattern Language for Human-Computer Interface Design* von Jenifer Tidwell [Tid99]
- *Interaction Design Pattern Library - Welie.com von* von Martijn van Welie [Van11]
- *User Interface Design Patterns* von Sari A. Laakso [Laa03]
- *Ajax Patterns* von Michael Mahemoff [Mah08]
- *The Elements of a Design Pattern* von Jared M. Spool [Spo06]

  [Sco06]

2.6 UML-Diagramm

Die vereinheitlichte Modellierungssprache (engl. Unified Modeling Language (UML)) ist eine allgemeine und graphische Modellierungssprache. Sie wird verwendet um Teile von Softwaresystemen zu spezifizieren, visualisieren, konstruieren und zu dokumentieren. Es werden mit UML statische Strukturen und dynamische Verhaltensweisen von Softwaresystemen beschrieben. Die statische Struktur definiert eine Art von Objekten und Beziehungen zwischen diesen. Sie sind für die Entwicklung des Systems und für dessen Implementierung wichtig. Das dynamische Verhalten definiert eine Historie von Objektinteraktionen und Kommunikationen zwischen Objekten. Diese Historie beschreibt das Verhalten des Systems und durch welche Vorgänge dessen Ziele und Absichten erreicht werden. Diese Systeme werden aus verschiedenen aber trotzdem ähnlichen Blickwinkeln modelliert, damit das Verständnis für die verschiedenen Absichten vorhanden ist. UML sieht für die Umsetzung dieser

verschiedenen Blickwinkel mehrere Arten von Diagrammen vor. In dieser Arbeit wird hauptsächlich das Klassendiagramm verwendet, weil in erster Linie eine Datenstruktur und deren Beziehungen vorgestellt wird. [RJB99, S.3-11]

Ein Klassendiagramm ist eine graphische Repräsentation einer statischen Abbildung, die eine Menge von deklarativen Modellelementen zeigt. Modellelementen werden durch Klassen, Arten und deren Inhalt und Beziehungen konkretisiert. Ein Klassendiagramm beinhaltet Elemente, die Verhaltensweisen darstellen. Diese nennt man Operationen. Ihre Abläufe werden jedoch in anderen Diagrammen beschrieben. Diese Abläufe können in dem Zustandsdiagramm oder Kommunikationsdiagramm dargestellt werden. [RJB99, S. 190] Falls weitere und detailliertere Informationen benötigt werden, ist der Leser angehalten in der UML-Spezifikation [Obj10] nach zu lesen.

Im Folgenden wird die Erweiterungsmöglichkeit von UML durch Profile erklärt. Diese spielen in dem Metamodell für Mustersprachen eine große Rolle.

2.6.1 Erweiterung von UML

Seit der UML Version 2 können Profile zur Erweiterung von modellierten Metamodellen eingesetzt werden. Durch die Mechanismen der Profile können die Metamodelle auf ihre verschiedenen Absichten angepasst werden. Dies bietet die Möglichkeit UML-Metamodelle auf verschiedene Plattformen oder Bereiche Maß zu schneidern. [Obj10, S. 685]

Die Erweiterungsmechnismen der Profile bestehen aus Einschränkungen (engl. Constraints), Eigenschaftswerten (engl. tagged Values) und Stereotypen (Stereotypes). Eine Einschränkung ist eine semantische Einschränkung. Diese können einfach durch Text oder durch formale Sprachen, wie der Object Constraint Language (OCL) eingeschränkt werden. Eigenschaftswerte sind Paare von Attributbezeichner und deren Werten. Diese können verwendet werden um beliebige Informationen über deren Elemente fest zu halten. Sie vervollständigen die Attributdefinition in den Elementen, die durch diese Attribute beschrieben werden. Einige Modellierer möchten die Modellierungssprache für ihren speziellen Anwendungsbereich anpassen. Ein Stereotyp erweitert ein Modellelement und stellt eine spezielle Art des Modellelementes dar. Der Informationsgehalt und die Form werden von dem Modellelement übernommen und in dem Stereotyp erweitert. Dies führt automatisch dazu, dass die Bedeutung und die Verwendung des Stereotyps unterschiedlich zu dem Modellelement ist. [RJB99, S. 101-104] Stereotypen stellen außerdem Standardelemente (engl. standard element) für die Instanz des Modells dar. [Obj10, S. 169]

2.7 Grundlagen modellgetriebener Softwareentwicklung

2.7.1 Modellgetriebene Softwareentwicklung

Das Ziel der modellgetriebenen Softwareentwicklung (engl. Model-Driven Software Development) ist es aus formalen Modellen und Modelltransformationen automatisch Code zu generieren. Modelle werden nicht nur für die Dokumentation von Software, sondern als Quelle für automatische Code-Generation großer Systemteile verwendet. Die modellgetriebene Softwareentwicklung soll den Wert ihrer Modelle steigern und den Entwicklern zeitaufwändige und plattformspezifische Programmierarbeiten ersparen. Dieses Bedürfnis entstand, weil aus UML-Diagrammen nur

Code-Rümpfe generiert werden können und infrastrukturabhängige Programme an verschiedene Plattformen angepasst werden müssen, obwohl sehr starke Ähnlichkeiten untereinander bestehen. Die modellgetriebene Softwareentwicklung wird in zwei Schritte unterteilt:

- Die Beschreibung von Modellen, die fachliche und funktionale Anforderungen an eine Software stellen und die von der Zieltechnologie unabhängig sind.
- Die automatische Code-Generierung durch Transformation der beschriebenen Modelle.

Die Initiative "Modell-Driven Architecture" der OMG [MM03] wurde die modellgetriebene Softwareentwicklung bekannt. Die Idee aus formalen Spezifikationen Programmcode halbautomatisch, aber korrektheitserhaltend generieren zu lassen oder der Traum des automatischen Programmierens sind die Vorläufer der modellgetriebenen Softwareentwicklung. [LL10, S. 340-341]

2.7.2 Modellgetriebene Architektur

Modellgetriebene Architekturen (engl. Model-Driven Architekture) sind eine spezielle Ausprägung von modellgetriebenen Softwareentwicklungen. Sie befassen sich mit der Modellierung von Softwarearchitekturen und deren Transformationen in Code. Modellgetriebene Architekturen sind Modelle die zusammengehörige Komponenten, Schnittstellen und Technologien als eine Plattform repräsentieren. Eine modellgetriebe Architektur besteht aus vier verschiedenen Modellen:

- Das *Computation-Independent Model* legt fest, was eine Software leisten soll und beschreibt diese auf fachlicher Ebene.
- Das *Platform-Independent Model* beschreibt die fachlichen Funktionalitäten von Programmen oder Komponenten. Sie werden plattformunabhängig und in einer formalen Modellierungssprache beschrieben.
- Das *Platform-Model* beinhaltet Eigenschaften und Informationen bezüglich einer Plattform.
- Das *Platform-Specific Model* wird durch eine Transformation von dem Platform-Model und dem Platform-Independent Modell gewonnen. Das Platform-Specific Model ist eine Realisierung des Platform-Independent Model bezüglich einer bestimmten Plattform, die durch das Platform-Model beschrieben wird.

Aus dem Computation-Independent Model kann das Platform-Independent Model abgeleitet werden. Durch die beschriebene Transformation kann ein Platform-Specific Model erstellt werden, das für eine weitere Transformation relativ gesehen ein Platform-Independent Model darstellt. Diese kann dann in ein noch spezielleres Platform-Specific Model transformiert werden. Es können beliebig viele Verkettungen von Transformationen hintereinander geschaltet werden. Die Abhängigkeiten der Platform-Models geben dabei die Reihenfolge der Transformationen an. Häufig werden die Modelle mit UML beschrieben und mithilfe von Profilen an bestimmte Plattformen angepasst, damit eine automatische Transformation möglich wird. Auch die Transformationen werden durch formale Quell- und Zielmodelle beschrieben. Hierfür müssen deren Metamodelle bekannt sein, um die Transformationen ausführen und aus dem Quellmodell ein Zielmodell erzeugen zu können. Weitere zusätzliche Informationen, die zusätzlich zum Quellmodell für die Transformation benötigt werden,

können Marks angegeben werden. Die resultierenden Modelle, speziell plattformspezifischer Code, können außerdem noch von Hand verfeinert und angepasst werden. [LL10, S.340-343]

Kapitel 3

Metamodell für Mustersprachen

In diesem Kapitel wird das Metamodell für Mustersprachen beschrieben. Der erste Teil dieses Kapitels stellt den Bezug zwischen den vorgestellten Grundlagen von Mustersprachen und dem Metamodell für Mustersprachen her. Anschließend wird die Form und der Aufbau, sowie die Aufgaben und Ziele des Metamodells für Mustersparchen vorgestellt. Eine Übersicht über die Modellebene soll die Abhängigkeit und Verwendung der Modelle und Instanzen erläutern. Der Schwerpunkt dieses Kapitels liegt auf der Beschreibung des Aufbaus und der Struktur des Metamodells für Mustersprachen. Das Metamodell für Mustersprachen wird mit einem Profil erweitert, das eine Grundlage für die Charakterisierung von Mustersprachen darstellt.

3.1 Einordnung der Grundlagen des Metamodells für Mustersprachen

Das Metamodell für Mustersprachen wurde anhand von Vergleichen und Analysen, der in Kapitel 2 *Grundlagen* beschriebenen wissenschaftlichen Arbeiten erstellt. In erster Linie basiert das Metamodell der Mustersprachen auf der Musterdefinition von Christopher Alexander [CAA77, S. x] und verwendet die Eigenschaften und Absichten seiner Musterstruktur und Mustersprachstruktur als Grundlage. Meszaros und Doble erstellten bereits ein Metamodell für Mustersprachen [MD96] anhand bekannter Mustersprachen, die alle auf der Definition von Christopher Alexander aufbauen. Aus dem Metamodell von Meszaros und Doble sind viele Ideen und Anregungen in diese Arbeit eingeflossen. Einen weiteren starken Einfluss, nahm das Werk *Pattern-Oriented Software Architecture: On Patterns and Pattern Languages* [BHS07] von Buschmann, Henney und Schmidt, das viele Mustersprachen aus unterschiedlichen Bereichen analysiert und deren Bedeutung darstellt.

3.2 Form und Aufbau des Metamodells für Mustersprachen

Meszaros und Doble beschrieben ihr Metamodell für Mustersprachen [MD96] durch Metamuster, die wiederum als Mustersprache vorliegt und durch ihren Inhalt definiert sind. Diese Arbeit löst sich von dieser rekursiven Darstellung des Metamodells und stellt diese Darstellung stattdessen in Form eines UML-Diagramms dar. Das UML-Diagramm lässt sich bezüglich der modellgetriebenen Softwareentwicklung anwenden und bietet somit Autoren die Möglichkeit, Mustersprachen und deren Modelle anhand des Metamodells zu entwerfen. Die Modellabstraktionen für die Erstellung von Mustersprachen werden im folgenden Abschnitt 3.4 *Übersicht der Modellhierarchie* in einer Modellhierarchie erläutert.

Das Metamodell für Mustersprachen besitzt zusätzlich zu den Beschreibungen für Mustersprachen

ein Profil, in dem Elemente für die Mustersprachen vordefiniert werden können. Das Profil des Metamodells für Mustersprachen wird im Anschluss der Metamodellbeschreibung vorgestellt. Während der Vorstellung des Metamodells werden dessen einzelnen Bestandteile erklärt, deren Notwendigkeit erläutert und deren Bezug zu bestehende Mustersprachen oder Mustersprachmodellen, die in dem Kapitel 2 *Grundlagen* vorgestellt wurden, hergestellt. Das UML-Diagramm des Metamodells für Mustersprachen befindet sich in Anhang A. In der Erklärungen des Metamodells für Mustersprachen werden nur Ausschnitte des UML-Diagramms vorgestellt, wegen der Größe des UML-Diagramms und des damit verbunden Darstellungsproblems. Die Bezeichner in dem Diagramm sind in Englisch und werden in den Erklärungen in runden Klammern angegeben.

3.3 Aufgaben und Ziele des Metamodells für Mustersprachen

Dieser Abschnitt erläutert die Aufgaben des Metamodells für Mustersprachen, die im Rahmen dieser Arbeit gestellt worden. Das Metamodell für Mustersprachen erfüllt die folgende Aspekte:

Viele Autoren passen die Struktur ihrer Mustersprachen an deren Inhalt und Problematik an. Daher gibt es keine eindeutige Definition von Mustersprachen. Das Metamodell für Mustersprachen dieser Arbeit bewältigt diese Problematik. Durch die Verwendung des Metamodells für Mustersprachen können Anpassungen von Mustersprachen vorgenommen werden. Zusätzlich wird die Kompatibilität der Mustersprachen untereinander gewährleisten. Die Anpassung der Mustersprachen an die Inhalte der Muster erlaubt das Gestalten von verschiedenen Mustersprachdefinitionen. Es kann somit die Frage nach der Definition von Mustersprachen offen gelassen werden.

Die Autoren der Mustersprachen können durch eine Mustersprachererweiterung in Form einer Profilerweiterung, die Teil des Metamodellprofils für Mustersprachen ist, die Mustersprachen erweitern. Durch diese Profilerweiterung wird das Metamodell verfeinert und bezieht sich auf speziellere Mustersprachen. Die Mustersprachen, die dem Metamodell zu Grunde liegen, bleiben wegen der strukturellen Abgrenzung des Profils untereinander kompatibel. Das Profil beschreibt die Charakteristik der Mustersprache, die die Struktur der Mustersprachen verfeinert. Diese Struktur wird von dem Metamodell für Mustersprachen beschrieben. Die Erweiterung des Profils wird durch vordefiniert Mustersprachelement erstellt, die eine Basis für Mustersprachen bietet.

Die Anpassung des Metamodells bezieht sich in diesem Punkt auf vordefinierte Bestandteile und deren Strukturen, die zum Beispiel in der Struktur der Muster oder der Mustersprachen vorkommen. Weil viele Mustersprachdefinitionen sich stark ähneln und auf der Musersprachdefinition von Alexander [CAA77, S. x] basieren, wird in dieser Arbeit ein Profil für das Metamodell für Mustersprachen vorgestellt. Mit diesem Profil sollen möglichst viele Mustersprachdefinitionen abgedeckt werden. Durch das Profil wird das Metamodell für Mustersprachen vervollständigt. Zudem bietet es eine Basis für die Erstellung von Mustersprachen.

Die Kompatibilität der Mustersprachen wird verwendet um Beziehungen darzustellen. Das Metamodell für Mustersprachen unterstützt die mustersprachübergreifende Referenzierung von Mustern, unabhängig von deren Art und Darstellung des Inhaltes. Dies erlaubt Referenzierungen zwischen Mustersprachen und somit die Erweiterung der Vernetzungen von Mustern. Verwandte und ähnliche Muster aus anderen Mustersprachen können somit referenziert werden, um deren Inhalte zu erweitern und um eine Brücke zwischen ihnen zu bauen.

Die Beziehungen zwischen Mustern werden explizit in dem Metamodell für Mustersprachen in Form von Typen vordefiniert. Dadurch wird die Vernetzung zwischen Mustern den Lesern und Autoren explizit dargestellt. Dies verstärkt die Bedeutungen der Verbindungen zwischen Mustern und verleiht auch den Inhalten der Muster mehr Aussagekraft. Durch die Typisierung von Beziehungen kann beispielsweise die Generalisierung und Spezialisierung von Mustern und Mustersprachen unterstützt werden. Die vordefinierten Beziehungstypen sind ein Bestandteil des Profils des Metamodells für Mustersprachen und können von den Autoren erweitert werden.

Buschmann vergleicht die Beziehung zwischen Mustern und Mustersprachen mit der Beziehung, die zwischen Klassen und Frameworks herrscht. [BHS07, 349-350] Das Metamodell für Mustersprachen soll zusätzlich Aspekte einbringen, die Brücken zwischen Mustersprachen bauen. Ziel ist es, eine Grundlage für eine Plattform zu schaffen, auf der mehrere Mustersprachen in Kooperation verwendet werden können. Wenn man Buschmanns Vergleich weiterführt, würde dies bedeuten, dass mehrere Frameworks in Kooperation verwendet werden könnten. Dies würde eine Art Betriebssystem als grundlegende Plattform für kooperierende Frameworks darstellen.

3.4 Übersicht der Modellhierarchie

Dieser Abschnitt liefert eine Übersicht über die Modellhierarchie. Sie gliedert das Metamodell der Mustersprachen, das Modell der Mustersprachen und die Musterkataloge in die Modellhierarchie ein. Die Modellhierarchie beschreibt außerdem die Abhängigkeiten zwischen diesen Modellen und dem Musterkatalog. In Abbildung 3.1 ist die Modellhierarchie mit ihren drei Ebenen zu sehen.

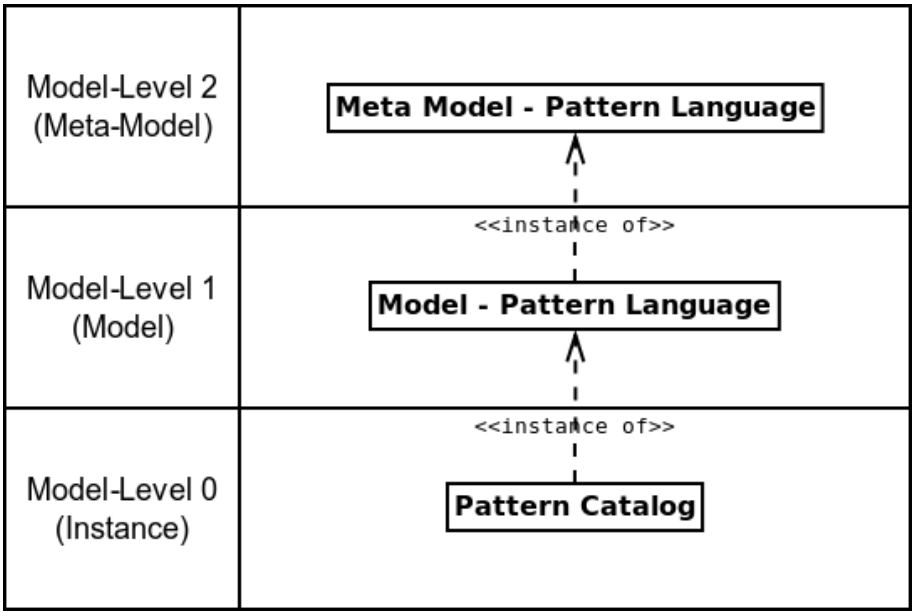


Abbildung 3.1: Modellhierarchie der Musterkataloge

Auf der obersten Ebene, der Metamodell Ebene (Model-Level 2) befindet sich das Metamodell der Mustersprachen (Meta Model - Pattern Language). Das Metamodell für Mustersprachen beinhaltet Modelle von der Mustersprachenstruktur, der Musterstruktur und der mustersprachenübergreifenden

Struktur. Das Metamodell für Mustersprachen beschreibt die Gemeinsamkeiten der Mustersprachen und deren Zusammenhänge untereinander. Es definiert eine Menge von Mustersprachen. Durch die Instantiierung des Metamodell erhält man ein Modell, das eine bestimmte Mustersprachen beschreibt. Die Instanziierungen sind in der Abbildung durch die gestrichelten Beziehungen mit Pfeil und der Aufschrift ”instance of ” dargestellt. Für die Erstellung des Mustersprachenmodells müssen die Strukturen der Muster und der Mustersprache definiert werden.

Auf der zweiten Ebene (Model-Level 1) ist das Mustersprachenmodelle (Model - Pattern Language) angesiedelt. Das Mustersprachenmodell beschreibt den Aufbau der Musterinhalte und definiert die Beziehungsarten zwischen den Mustern. Durch das Erstellen eines Musterkataloges anhand eines Mustersprachenmodells wird das Mustersprachenmodell instantiiert. Auf der untersten Ebene (Model-Level 0) befinden sich die Instanzen des Mustersprachenmodells.

Die unterste Schicht der Modellhierarchie enthält keine Modelle sondern nur Instanzen des darüber-”liegenden Modells. Diese aus den Mustersprachmodellen instantiierten Objekte sind die Musterkataloge (Pattern Catalog). Ein Musterkatalog stellt eine vernetzte Menge von Mustern dar. Diese bilden den Kontext des Musterkataloges durch ihre Inhalte. Ein Musterkatalog kann außerdem weitere Funktionalitäten, wie ein Stichwortverzeichnis oder eine Inhaltsübersicht bereitstellen. Ein Musterkatalog, in dieser Arbeit, ist vergleichbar mit den vorgestellten Mustersprachen in dem Kapitel 2 Grundlagen.

In diesem Abschnitt wird die strukturelle Abhängigkeit der vorgestellten Modelle und Instanzen erläutert. Es wird eine Baumstruktur vorgestellt, die die Abhängigkeiten zwischen Modellen und Instanzen verkörpert und eine ergänzende Erklärung zu der Abbildung 3.1 liefert. Das Metamodell für Mustersprachen stellt die grundlegende Beschreibung für die Menge von Mustersprachen dar, die Instanzen des Metamodells sind. Eine Mustersprache ist wiederum ein Modell für eine Menge von Musterkataloge. Das Metamodell stellt die Wurzel der Instanzableitungen dar. In der zweiten Ebene der Baumstruktur befindet sich die Menge der instaniierten Mustersprachen. Die dritte Ebene enthält die instanziierten Musterkataloge. Die Ebenen der Baumstruktur entsprechen der vorgestellten Ebenen aus der Abbildung 3.1. Es ist zu beachten, dass das Metamodell in Kombination mit einem Profils verwendet wird, das die Charakteristik der Mustersprachen definiert. Die Betrachtung der Baumdarstellung und Veranschaulichung der Instanzabhängigkeiten, bezieht sich auf das Metamodell für Mustersprachen mit einem konkreten Profil.

3.5 Aufbau und Bestandteile des Metamodells für Mustersprachen

Das Metamodell für Mustersprachen beschreibt eine mustersprachenübergreifende Struktur für Mustersprachen. Dies wird durch die Beschreibung der Mustersprachstruktur, der Musterstruktur und deren Beziehungen beschrieben. Die mustersprachenübergreifende Struktur ist unabhängig von der Charakterisierung der Mustersprachen, die von dem Profil beschrieben werden. Sie stellt die Grundlage der Mustersprachen dar. Mithilfe dieser einheitlichen Struktur werden die Mustersprachen untereinander kompatible gehalten. Sie ermöglicht das Vergleichen und das Erstellen von Beziehungen zwischen gleichen Elementarten, die in der mustersprachenübergreifende Struktur für Mustersprachen definiert sind.

Im Folgenden wird die Strukturbeschreibung des Metamodells für Mustersprachen erklärt. Zunächst wird eine Übersicht über die Strukturen des Metamodell vorgestellt, um einen Einstieg in die Thematik

zu erlangen. Anschließend werden die Grundlagen der Strukturen, aus denen sich das Metamodell für Mustersprachen zusammensetzt erklärt. Im weiteren Verlauf werden diese Strukturen und ihre Zusammenhänge detailliert erläutert.

3.5.1 Übersicht der Strukturen in dem Metamodell für Mustersprachen

Die Struktur für Mustersprachen, die durch das Metamodell für Mustersprachen beschrieben wird, besteht aus einer Struktur für Muster und Mustersprachen, aus Beziehungsbeschreibungen von Mustersprachen, Muster und deren Inhalten, sowie einer grundlegenden Definition von der Strukturelemente.

In diesem Kapitel wird zunächst die Grundlagen der Struktur und die Definition der Strukturelemente vorgestellt, die die Grundlage für die weiteren Strukturen der Muster und Mustersprachen liefern. Anschließend wird die Musterstruktur und die Mustersprachenstruktur vorgestellt. Die Beschreibung der Beziehungen in den Strukturen sind in den Strukturbeschreibungen enthalten.

Im Anhang A befindet sich die Abbildung des vollständigen UML-Diagramms des Metamodells für Mustersprachen. Die Musterstruktur (Pattern Structure) und die Mustersprachstruktur (Pattern Structure Language) sind zentral in dem UML-Diagramm angeordnet. Sie bilden die Struktur des Metamodells für Mustersprachen und enthalten Beziehungen (Structure Element Relation). Die Musterstruktur ist aus einer hierarchischen Anordnung von Inhaltselement (Content Element) für Muster zusammengesetzt. Die Mustersprachstruktur besteht aus einer Musterstruktur und den Beziehungen die zwischen den Musterinhalten (Content) herrscht. Außerdem enthält die Mustersprachstruktur Organisationen von Mustern (Pattern Organisation), in denen die Muster organisiert sind und zusätzliche Funktionalitäten (Pattern Language Feature), die die Inhalte der Muster und deren Beziehungen erweitern können. Die vorgestellten Elemente bilden im Groben eine mustersprachenübergreifende Struktur für Mustersprachen, die durch das Metamodell beschrieben wird.

3.5.2 Grundlagen der Strukturen des Metamodells für Mustersprachen

Im Folgenden werden die Elemente vorgestellt, die Bestandteile der Struktur des Metamodells für Mustersprachen sind und die eine Grundlage für die Mustersprachstruktur und Musterstruktur liefern. Die Elemente der Strukturgrundlagen sind Elemente, die Gemeinsamkeiten der konkreten Strukturen darstellen und eine Beschreibungsgrundlage bieten. Durch die Verwendung der Strukturgrundlagen ist die Struktur des Metamodells für Mustersprachen übersichtlich strukturiert und enthält keine redundante Beschreibungen.

In Abbildung 3.2 ist ein Ausschnitt des Metamodells für Mustersprachen zu sehen. Dieser enthält auf der linke Seite die Grundlagen der Struktur und auf der rechten Seite die konkreten Strukturen des Metamodells für Mustersprachen. Die Mustersprachstruktur (Pattern Language Structure), die Musterstruktur (Pattern Structure) und der Musterinhalt (Content Element) sind Elemente, die die Strukturen des Metamodells für Mustersprachen darstellen. Sie stellen Spezialisierungen des Strukturelements (Structure Elements). Diese ist ein abstraktes Element, das die Grundlage der konkreten Strukturen darstellt. Ein Strukturelement besteht aus einem eindeutigen Bezeichner (Identifier), der das Strukturelement identifizieren kann, ein Symbol (Icon), das das Strukturelement

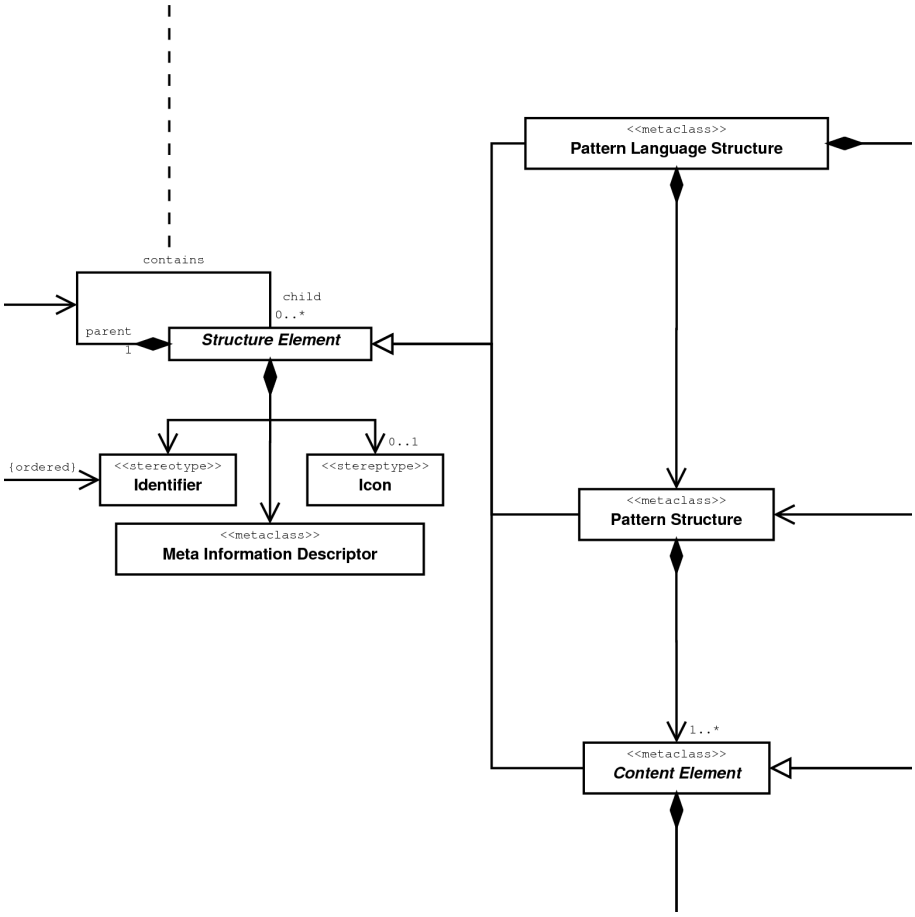


Abbildung 3.2: Strukturen und Strukturelemente des Metamodells für Mustersprachen

repräsentieren soll und eine Metainformationsbeschreibung (Meta Information Descriptor), die Informationen über den Inhalt der Strukturelement enthält.

Der Bezeichner und das Symbol der Strukturelemente sind durch Stereotypen definiert worden, weil sie Standardelemente eines Strukturelements darstellen.

Die Metainformationsbeschreibung liefert weitere Auskünfte über den Inhalt der Strukturelemente. Zusätzlich zu den drei Elementen kann genau ein Strukturelement beliebig viele Beziehungen zu anderen Strukturelementen enthalten. Durch die Angaben der Kardinalitäten 1 und 0..* wird eine azyklische und gerichtete Struktur definiert, damit ein Strukturelement mit mehrere Strukturelemente in Beziehung stehen kann.

Die Strukturen, die mit dem Strukturelement erstellt werden, stellen alle eine hierarchische Baumstruktur dar.

Die Beziehung von Strukturelementen (Structure Element Relation) wird in dem Metamodell für Mustersprachen explizit als Metaklasse dargestellt. Diese Metaklasse ist in Anhang A dem UML-Diagramm des Metamodells für Mustersprachen in der rechten oberen Ecke zu sehen. Eine Mustersprache kann durch die Verfeinerung der Beziehungstypen aussagekräftigeren Bedeutungen für

die Beziehungen darstellen. Vordefinierte Typen und Semantiken von Beziehungen sind Bestandteile des Profils von dem Metamodell für Mustersprachen und werden im Anschluss der Strukturerklärung vorgestellt.

Die Beziehungen der Strukturelemente werden durch Referenzen, die sich in den Inhaltselementen befinden, dargestellt. Anhand der Bezeichner der Strukturelemente kann jedes Strukturelement referenziert werden. Das Metamodell für Mustersprachen sieht diese Beschreibung vor, damit Mustersprachen, Muster und Musterinhalte einheitlich und somit mustersprachenübergreifend referenziert werden können. Der Aufbau und die Verwendung von Referenzen auf Strukturelemente wird in der Beschreibung der Musterstruktur detailliert beschrieben.

Bevor auf die konkreten Strukturen eingegangen wird, werden zunächst die Beziehungen zwischen den Strukturelement anhand der Abbildung 3.2 erläutert. Die drei Strukturelemente besitzen zusätzlich untereinander Aggregationsbeziehungen. Ein Mustersprachstruktur enthält somit eine Musterstruktur, die wiederum mehrere Inhaltselemente enthält. Die Rollen und Absichten der Aggregationsbeziehungen zwischen den Strukturelemente werden in den folgenden Abschnitten der Musterstrukturbeschreibung und der Mustersprachstrukturbeschreibung erklärt.

3.5.3 Modell der Musterstruktur des Metamodells für Mustersprachen

Die Struktur der Muster einer Mustersprache ist das zentrale Element der Mustersprachen. Aus diesem Grund wird mit der Erklärung des Modells der Musterstruktur des Metamodells für Mustersprachen begonnen. Das Modell der Musterstruktur des Metamodells beschreibt die Musterstrukturen der Mustersprachen. Im folgenden wird der Aufbau der Musterstruktur und die dafür notwendigen Elementen vorgestellt. Anschließend werden die Zusammenhänge der Musterstrukturen und der Mustersprachstrukturen erläutert.

Die Musterstruktur ist eine hierarchische Ordnung der Inhaltselemente von Muster in einer Mustersprache. In dem Metamodell für Mustersprachen wird eine Metaklasse definiert, die die Struktur der Musterstruktur beschreibt. Diese Beschreibung schreibt jede Mustersprache, die dem Metamodell für Mustersprachen zu Grunde liegt, die Struktur ihrer Mustersturktur vor. Die grundlegende Struktur aller Musterstrukturen, die in den vorgestellten Mustersprachen in dem Kapitel 2 *Grundlagen* enthalten sind, ist in diesem Modell für Musterstrukuren berücksichtigt. Eine einheitliche Strukturbeschreibung dient nicht nur der Übersichtlichkeit der Muster, sie bietet auch eine Grundlage um Beziehungen und Vergleiche zu erstellen.

Bevor das Modell der Musterstruktur erklärt wird und dieses in das Metamodell für Mustersprachen eingeordnet wird, werden die dafür benötigten Elemente vorgestellt. Für die Erklärung der Musterstruktur werden zunächst die Strukturelemente der Musterstruktur, die Musterinhalte und der Referenzierungsmechanismus anhand der Abbildung 3.3 vorgestellt.

Modell des Inhaltselements von Mustern Eine Musterstruktur besteht aus mehreren Inhaltselementen (Content Element), die die Strukturelemente der Musterstruktur darstellen. Abbildung 3.3 stellt einen Ausschnitt des Metamodells für Mustersprachen dar, in dem das Modell des Inhaltselements zu sehen ist. Das Inhaltselement besteht aus einem Inhalt (Content) und einer Inhaltselementreferenz (Structure Element Referenz). Außerdem erbt das Inhaltselement alle Elemente und Beziehungen, die das Strukturelementes enthält.

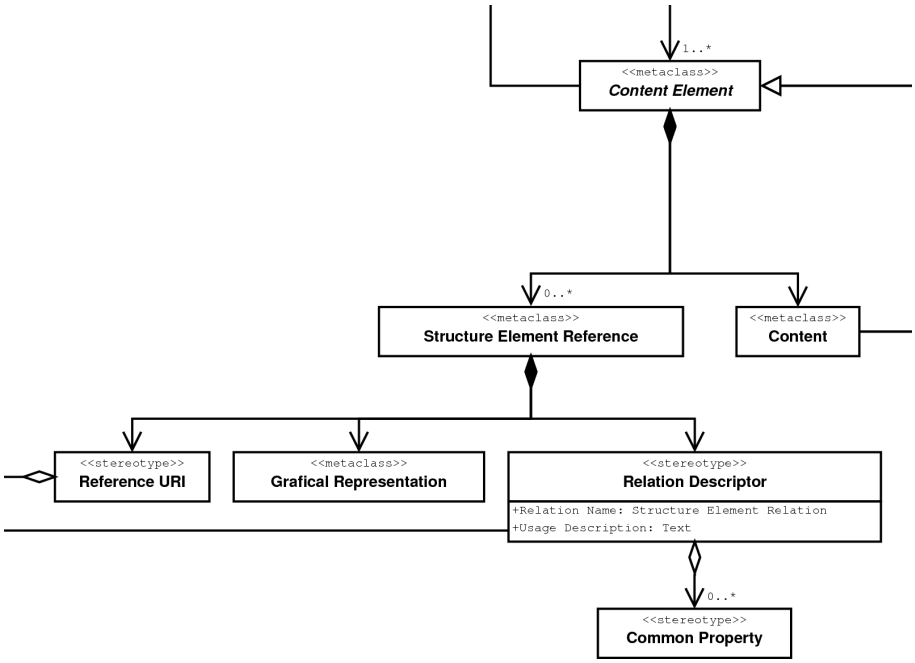


Abbildung 3.3: Modell des Inahltselements

Modell des Inhaltes Die Metaklasse des Inhaltes sieht einen Inhalt vor, der sich im Inhaltselement befindet und noch keine Aussagen über Art und Darstellung des Inhaltes macht. Die Konkretisierung des Inhaltes ist ein Bestandteil der Musterspracherstellung. Die Autoren der Mustersprachen können die Art des Inhaltes und dessen Darstellung frei gestalten. Es können mehrere Definitionen von der Art eines Inhaltes und dessen Darstellung in einer Mustersprache existieren und verwendet werden. Eine Problematik könnte zum Beispiel durch einen Text und eine Abbildung beschrieben werden. Somit hätte man zweit Arten des Inhaltes: eine Textform und eine Graphik.

Modell der Strukturelementreferenz Die Strukturelementreferenz verkörpert die Beziehungen zwischen den Strukturelementen. Durch die Verwendung einer Strukturelementreferenz kann eine Beziehung zu einem Strukturelement dargestellt werden. Konkret heißt das, dass auf andere Inhaltselemente, Muster oder Mustersprachen durch einen Referenz verwiesen werden kann. Die Strukturelementreferenzen sind als Bestandteil der Strukturelemente in Abbildung 3.3 zu sehen. Die Strukturelementreferenzen befinden sind in den Inhaltselementen (Content Element) der Muster und können im Inhalt (Content) dargestellt werden. Diese Designentscheidung ist jedoch dem Autor der Mustersprache überlassen. Eine Strukturelementreferenz besteht aus einem einheitlichen Quellenanzeiger (Reference URI), einer graphischen Repräsentation (graphical Representation) und aus einer Beziehungsbeschreibung (Relation Descriptor), die wiederum gemeinsame Eigenschaften (Common Property) der Strukturelemente enthält.

Der einheitlichen Quellenanzeiger wird für das Auffinden des verwiesenen Strukturelements benötigt. Er ist nach dem Prinzip des Uniform Resource Locator (URL) von Tim Berners-Lee [Ber94] aufgebaut. Der einheitliche Quellenanzeiger besteht aus einer Bezeichnerliste von Strukturelementbezeichnern. Die Bezeichnerliste enthält den Bezeichner des referenzierten Strukturelements und alle Bezeichner

der Strukturelemente, die in den Strukturebenen vor dem referenzierten Strukturelement erscheinen. Die Bezeichnerliste ist absteigend sortiert und beginnt mit dem Strukturelement aus der obersten Strukturebene. Ein einheitlicher Quellenanzeiger ist ein Standardelement, dass in jeder Strukturelementreferen vorkommen muss. Aus diesem Grund wurde hierfür ein Stereotyp erstellt.

Die graphische Repräsentation der Strukturelementreferenz ist wie der Inhaltstyp Bestandteil der Mustersprachen. Auch hier ist es dem Autor der Mustersprache frei gestellt eine beliebige Darstellungsart der Strukturelementreferenz zu wählen.

Die Beziehungsbeschreibung soll Aufschluss über die Art und Bedeutung der Beziehung geben, die durch die Strukturelementreferenz verkörpert wird. Es wird der Typ der Beziehung, die Verwendung und die Eigenschaften, die sich die zwei in Beziehung stehenden Strukturelemente teilen, vermerkt. Die Beziehungrelation der Strukturelementen wird in dem UML-Diagramm des Metamodells für Mustersprachen in Anhang A durch Strukturelementbeziehung (Strukture Elemente Relation) typisiert. Somit können verschieden Arten von Beziehungen zwischen Strukturelementen in der Beziehungsbeschreibung verwendet werden. In der Beschreibung des Profils des Metamodells für Mustersprachen werden Arten von Beziehungen vorgestellt, die sich zwischen den Strukturelementen befinden können. Ebenso wie der einheitliche Quellenanzeiger ist die Beziehungsbeschreibung ein Standardelement der Strukturelementreferenz, das in jeder Strukturelementreferenz enthalten sein muss. Deshalb wird es auch durch ein Stereotyp definiert.

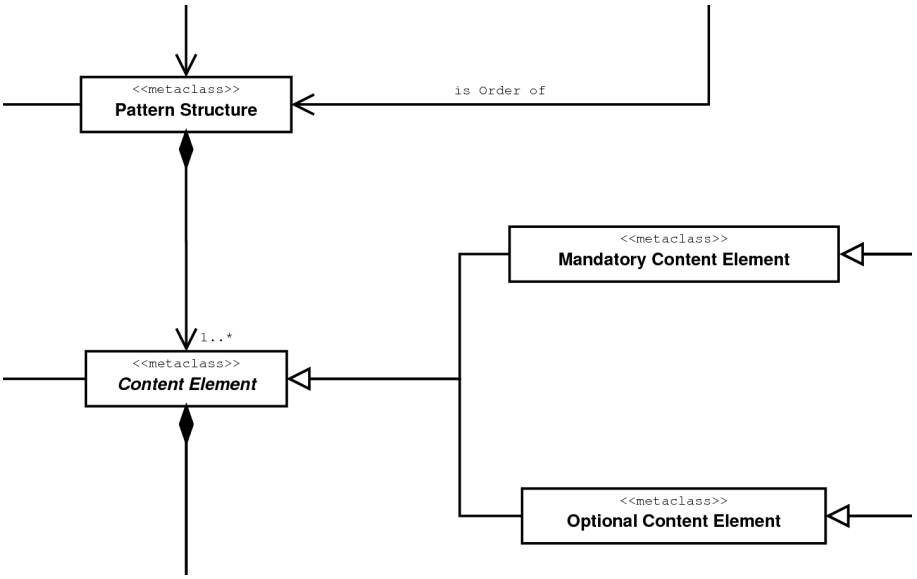


Abbildung 3.4: Modell der Musterstruktur

Die Abbildung 3.4 zeigt den Ausschnitt des Modells der Musterstruktur in dem Metamodell für Mustersprachen. Das obligatorische Inhaltselement (Mandatory Content Element) und das optionale Inhaltselement (Optional Content Element) sind Spezialisierungen des abstrakten Inhaltselements (Content Element) und verkörpern dieses.

Das obligatorische Inhaltselement und das optionale Inhaltselement erbt außerdem die Elemente und die Beziehung des abstrakten Inhaltselements. Das obligatorische Inhaltselement beschreibt Musterinhalte, die in jedem Muster einer Mustersprache vorkommen müssen. Das optionale

Inhaltselement beschreibt Musterinhalte, die optional in einem Muster einer Mustersprache vorkommen können. Der Autor muss anhand des Inhaltes entscheiden, ob es hilfreich ist die jeweiligen optionalen Inhaltselemente der Musterstruktur zu verwenden. Durch das Vererben der Beziehung des Inhaltselement können mit den obligatorischen und den optionalen Inhaltselementen eine hierarchische Struktur aufgebaut werden. Es ist außerdem durch die vererbte Beziehung möglich und erwünscht, dass obligatorische Inhaltselemente optionale enthalten und umgekehrt.

In einer Mustersprache soll es zum Beispiel Muster geben, deren Inhalt durch ein Beispiel erläutert werden kann. Das Inhaltselement der Beispielbeschreibung stellt ein optinales Inhaltselement dar. Wenn sich jedoch der Autor dazu entschließt dieses Inhaltselement zu verwenden, soll es durch eine Überschrift, einer Beschreibung in textueller Form und eine Skizze beschrieben werden. Die drei Elemente, die das Inhaltselement der Beispiele enthält wären dann obligatorische Inhaltselemente. Somit kann man vorschreiben, was ein optionales Inhaltselement enthalten muss.

Meszaros und Doble stellten in ihrer Arbeit *MetaPattern: A Pattern Language for Pattern Writing* [MD96] die Unterteilung von Musterinhalten in obligatorische und optionale Inhaltselemente vor. Zum Beispiel besteht das Inhaltselement äußere Einflüsse aus einem Bezeichner und einer Aufzählung der äußeren Einflüsse in Textform. Zusätzlich wird noch vorgegeben was genau äußere Einflüsse inhaltlich repräsentieren. Wie an diesem Beispiel zu sehen ist, besteht keine Trennung zwischen der Metamodellierung, der Profilerstellung und den semantischen Aussagen der Elemente. Hier wird der Fokus auf die Struktur der äußeren Einflüsse gelegt. Abstrahiert man das Beispiel auf die Metaebene, dann stellen die äußeren Einflüsse ein obligatorisches Inhaltselement dar, das wiederum weiteren obligatorischen Inhaltselementen enthält.

3.5.4 Modell der Mustersprachstruktur des Metamodells für Mustersprachen

Eine Menge zusammenhängender Muster in einer Mustersprache stellen mehr als nur die Summe der Muster dar. Durch die Vernetzung der Muster erhält man weitaus mehr Information als aus den einzelnen Musterinhalten. Die Mustersprache stellt die Muster zusammen in einen Kontext. Durch verschiedene Organisationen von Mustern und zusätzlichen Funktionalitäten, die die Musterinformationen aufwerten, erlangt die Mustersprache ihren Mehrwert. [MD96]

Der Ausschnitt 3.5 des Metamodells für Mustersprachen zeigt das Modell der Mustersprachstruktur. Die Mustersprachstruktur (Pattern Language Structure) besteht aus einer Musterstruktur (Pattern Structure), mehreren Organisationen von Muster (Pattern Organisation) und mehreren Zusatzfunktionalitäten (Pattern Language Feature).

Bevor auf die Beschreibung der Organisationen von Muster und der Zusatzfunktionalitäten eingegangen wird, werden die Beziehungen zwischen Musterstruktur und Mustersprachstruktur erklärt. Die Musterstruktur die Mustersprachenstruktur erben die Eigenschaften des Strukturelement und dessen Unterelemente. Sie können somit auch Beziehungen auf Elemente des selben Typs besitzen. Diese Beziehungseigentschaften von Musterstrukturen und Mustersprachstrukturen werden im Folgenden eingeschränkt und deren Bedeutungen erklärt.

Die Mustersprachstruktur enthält eine Mustersturktur, die eine Erweiterung der Mustersprachstruktur ist. Bei den Beziehungen zwischen Musterstrukturen sind jedoch Einschränkungen zu machen. Es ist zum Beispiel nicht vorgesehen, dass eine Musterstruktur eine andere enthält. Die Beziehungen zwischen Musterstrukturen werden erst auf der Ebene der Mustersprachmodelle konkretisiert. Denkbar

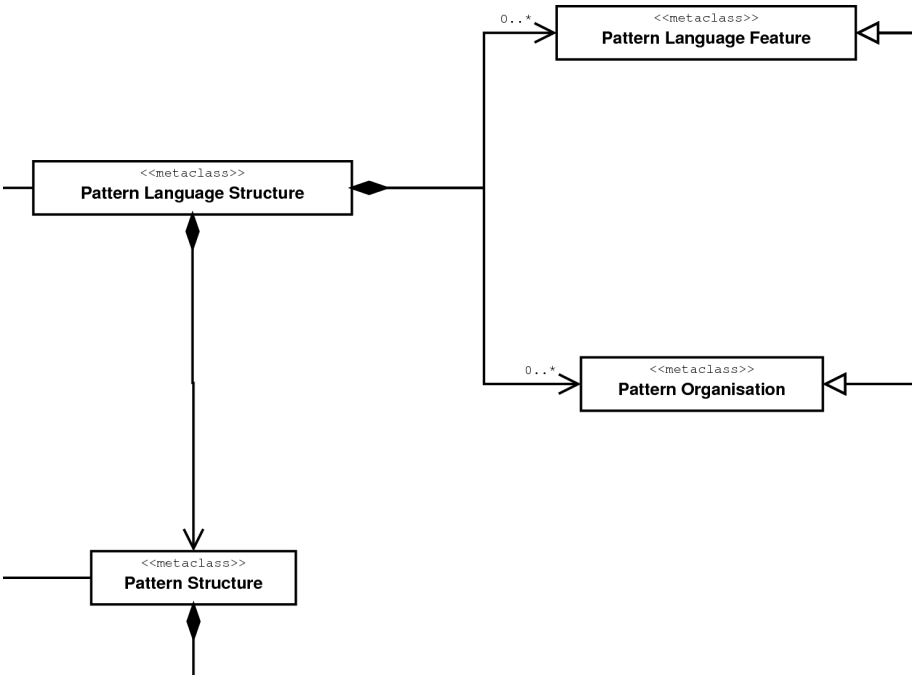


Abbildung 3.5: Modell des Inhaltselements

wäre zum Beispiel, dass eine Musterstruktur spezialisiert wird. Beide Musterstrukturen befinden sich zwar in unterschiedlichen Mustersprachen, haben aber eine Spezialisierungsbeziehung und Ähnlichkeitsbeziehungen zueinander. Die Mustersprachen, die den Musterstrukturen zu Grunde liegen, haben ebenfalls solche Beziehungen zu einander, da diese die jeweilige Musterstruktur enthalten. Die Mustersprachen könnten je nach Gestaltung als Dialekte oder Mustersprachgenerationen angesehen werden.

Buschmann schildert verschiedene Möglichkeiten wann eine Mustersprache einen Dialekt darstellen kann. [BHS07, S. 215-217] Es könnte zum Beispiele eine Mustersprache geben, die allgemeine Lösungen einer Problematik bietet. Möchte man diese Lösungen aufgreifen und für einen bestimmten Kontext konkretisieren und anpassen, dann könnte eine weitere Mustersprachen erstellt werden, die angepasste und spezialisierte Lösungen enthält. Um die Mustersprache mit den speziellen Lösungen zu erstellen, kann die Musterstruktur von der Mustersprache mit den allgemeinen Lösungen erweitert werden. Die spezialisierte Mustersprache stellt auf Grund der Ähnlichkeit der Musterstrukturen einen Dialekt der Mustersprache mit den allgemeinen Lösungen dar.

Organisationen von Muster als Bestandteile der Mustersprachstruktur Die meisten Kollektionen von Mustern sind relativ klein und überschaubar. Sie zielen meistens auf einen bestimmte Problematik ab. Ziel ist es gute Darstellung von Musterkollektionen zu finden, die auch mustersprachübergreifend sind. Das finden von Musterkollektionen ist ein grundlegender Bestandteil eines Musterkataloges und sollte durch Repositories verwaltet werden. [BHS07, S. 209-210]

Die Musterkollektionen werden je nach Bedarf und Interesse gebildet. Es ist nicht die Aufgabe eines

Autor diese Kollektionen explizit zu bilden. Der Autor versieht seine Muster mit Referenzen, mit denen er den Inhalt seiner Muster vervollständigt und auf andere Strukturelemente verweist. Implizit entsteht die Vernetzung der Strukturelemente, in der sich die Musterkollektionen befinden. Der Musterkatalog stellt den Lesern, die für ihre Interessen zugeschnittenen Musterkollektionen bereit. Dieser Ansatz motiviert zu der Erstellung eines Musterkataloges, der durch Software implementiert wird. [BHS07, S. 210-211]

Die Musterkollektionen sind in Musterorganisationen organisiert. Diese Organisationen bilden die Musterkollektionen und fokusieren deren Struktur, die in der Vernetzung des Musterkataloges enthalten ist. Dabei gibt es verschiedene Möglichkeiten nach denen die Organisationen der Muster aufgebaut werden können. Die Selektion der Muster der Musterkollektionen wird durch den Themenbereich der Musterorganisation bestimmt. [BHS07, S. 211] Das Metamodell für Mustersprachen sieht eine Metaklasse für die Organisationen von Mustern vor, damit verschiedene Musterorganisationen realisiert werden können. Auf die verschiedenen Möglichkeiten, wie Organisationen von Mustern aufgebaut werden können, wird in der Beschreibung des Profils eingegangen.

Zusatzfunktionalitäten der Mustersprachstruktur Zusatzfunktionalitäten von Mustersprachen verleihen der Mustersprache, ebenso wie die Organisationen von Muster einen Mehrwert. Zusatzfunktionalitäten sind ebenfalls vernetze Mengen von Mustern wie die Musterkollektionen der Organisationen von Mustern. Im Gegensatz zu den Musterorganisationen können Zusatzfunktionalitäten für Mustersprachen die Beziehungen zwischen den Muster und deren Informationsgehalt erweitern.

Eine Zusatzfunktionalität könnte zum Beispiel ein Wörterindex sein. Dieser gibt an, in welchen Muster sich ein bestimmtes Wort befindet. Dadurch wird die Beziehung zwischen den Mustern dargestellt, die davor noch nicht explizit existierte. Zusätzlich könnten die Begrifflichkeiten im Wörterindex erklärt werden. Dies wäre eine Erweiterung des Informationsgehaltes der Muster, die sich in dieser Musterkollektion befinden.

Ein Autor, der solche eine Zusatzfunktionalität einer Mustersprache bereitstellen will, muss die Musterinhalte kennen und diese Zusatzfunktionalität manuell erstellen. Da das Metamodell für Mustersprachen für Musterkataloge verwendet werden soll, die durch Softwareprogramme realisiert werden können, wird in dieser Arbeit zwischen Organisation von Mustern und Zusatzfunktionalitäten von Mustersprachen unterschieden.

In der Beschreibung des Profils von dem Metamodell für Mustersprachen werden Zusatzfunktionalitäten für Mustersprachen vordefiniert. Diese Zusatzfunktionalitäten stellen eine einheitliche Basis für alle Mustersprachen dar, die aus dem Metamodell für Mustersprachen abgeleitet sind.

3.6 Profil des Metamodells für Mustersprachen

Mit dem Profil des Metamodells für Mustersprachen wird die Charakteristik der Mustersprachen dargestellt. Das Profil des Metamodells für Mustersprachen wird von seiner Struktur getrennt. Auf Grund der Trennung von Struktur und Profil ist es möglich das Profils anzupassen und zu verändern. Durch die Anpassung und Veränderung des Profils werden andere Charakteristik von Mustersprachen beschrieben. Mit Hilfe von Profiländerungen kann das Metamodell für spezielle Mustersprachen

angepasst werden, ohne die Struktur des Metamodells zu verändern. Mustersprachen, die aus dem Metamodell für Mustersprachen instanziiert wurden, besitzen immer die gleiche Struktur und sind trotz verschiedener Profile zu einander kompatible. Dies ermöglicht zum Beispiel das Erstellen von mustersprachübergreifenden Verbindungen. Es wird davon ausgegangen, dass das Metamodell für Mustersprachen immer in Verbindung mit einem Profil verwendet wird. Aus diesem Grund wird das Profil als notwendiges Bestandteil des Metamodells betrachtet. Das Profil aus dieser Arbeit versucht einen allgemeinen Standard für Mustersprachen festzulegen. Dies geschieht durch die Definition und Beschreibung von Standardelementen und Einschränkungen, die die Bestandteile des Profils darstellten. Die Zusammenstellung der Standardelemente und Einschränkungen wurde anhand der im Kapitel 2 *Grundlagen* vorgestellten Mustersprachen und den MetaPatterns von Meszaros und Doble [MD96] ausgewählt.

Im Folgenden wird das Metamodell für Mustersprachen durch das Profil erweitert. Das Profil befindet sich auf der rechten Seite in dem UML-Diagramm des Metamodells für Mustersprachen, das in der Abbildung *Metamodells für Mustersprachen* von Anhang A zu sehen ist. Das Profil wird in drei Bestandteile unterteilt, die sich auf die Strukturen in dem Metamodell für Mustersprachen beziehen. Sie werden mit Hilfe von Ausschnitten des UML-Diagramms beschreiben. Die Bestandteile des Profils werden in das Beziehungsprofil, Musterprofil und in das Mustersprachprofil des Metamodells für Mustersprachen aufgeteilt und anhand dieser Aufteilung vorgestellt.

3.6.1 Beziehungsprofil des Metamodells für Mustersprachen

Das Beziehungsprofil beinhaltet vordefinierte Beziehungstypen. Diese Beziehungstypen konkretisieren Beziehungen, die sich zwischen den Strukturelementen befinden und können als Instanzen in den Mustersprachen verwendet werden. Die Beziehungstypen und deren Beschreibungen dienen zum Beispiel der Erstellung von Organisation von Mustern und den Zusatzfunktionalitäten der Mustersprachen. Sie bieten außerdem ein zusätzliches Verständnis und Informationen für die Zusammenhänge der Muster. Die Leser der Mustersprachen erhalten durch die Typisierung und Beschreibungen der Beziehungen zwischen Mustern einen größeren Informationsgehalt.

In den Mustersprachen können diese Beziehungstypen zusätzlich erweitert werden. Im Gegensatz zu den Instanzen der vordefinierten Beziehungstypen des Beziehungsprofils sind die erweiterten Beziehungen nicht in allen Mustersprachen bekannt. Beziehungstypen, die aus dem Beziehungsprofil instanziert wurden, sind zu den Beziehungstypen anderer Mustersprachen kompatibel. Wenn die Beziehungstypen den Profilvorgaben entsprechen, ist es einfacher, Analysen zwischen Mustersprachen durch zu führen.

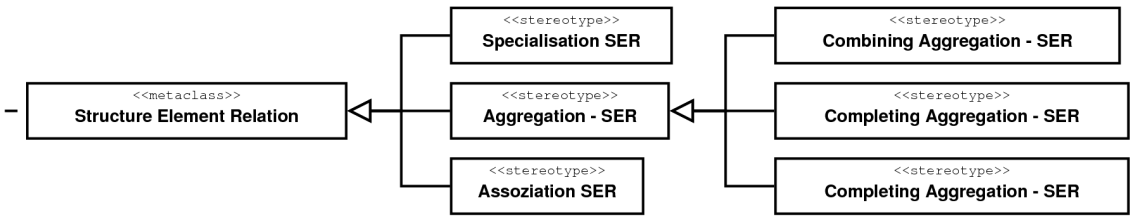


Abbildung 3.6: Beziehungsprofil

Die Abbildung 3.6 zeigt den Ausschnitt des Metamodells für Mustersprachen, auf dem die

Beziehungstypen des Profils zu sehen sind. Die Beziehungen der Strukturelemente (Structure Element Relation) werden durch eine Metaklasse beschrieben. Das Profil beinhaltet eine hierarchische Struktur von konkreten Beziehungstypen. Diese Beschreibungen von Beziehungstypen sind Spezialisierungen der Metaklasse und Stereotypen der Beziehungstypen für Strukturelemente in Mustersprachen. Sie erhalten zusätzlich durch die Beschreibung des Beziehungsprofils ihre Semantik.

Die Unterteilung von Beziehungstypen wurde bereits in dem Kapitel 2 *Grundlagen* unter der Rubrik 2.4.1 *Beziehungen in Mustersprachen* vorgestellt. Im folgenden werden die benötigten Erkenntnisse, die für das Beziehungsprofil des Metamodells für Mustersprachen erforderlich sind, erneut aufgeführt.

Van Welie und van der Veer unterteilten die Beziehungen zwischen Muster in drei Beziehungstypen [vv03]. Diese Beziehungstypen sind in die Typen Spezialisierung, Aggregation und Assoziation unterteilt. In dem Metamodell für Mustersprachen werden diese Beziehungstypen und deren Semantik übernommen. Die Verallgemeinerungen der Beziehungstypen werden als Spezialisierungen der Metaklasse von Strukturelementsbeziehung dargestellt. Die Spezialisierungsbeziehung (Specialisation SER), Aggregationsbeziehung (Aggreagation SER) und Assoziationsbeziehung (Association SER) bilden somit die erste Strukturebene der Beziehungstypen in Mustersprachen.

Buschmann verfeinerte die Aggreagtionsbeziehung weiter in die Untertypen: Vervollständigungsaggregationen (Completing Aggregation - SER), Erweiterungsaggregationen (Combining Aggregation - SER) und in Aggregationen die sich gegenseitig ausschließen (Compteting Aggregation - SER). Diese drei Verfeinerungen bilden die zweite Strukturebene der Beziehungstypen in Mustersprachen. Deren Beschreibung ist in der Rubrik 2.4.1 *Beziehungen in Mustersprachen* detaillierter vorgestellt. Das Beziehungsprofil beinhaltet die vollständige Struktur mit ihren beiden Ebenen.

In dem Metamodell für Mustersprachen und seinem Profil werden keine Aussagen über das Erstellen von qualitativ hochwertigen Inhalten gemacht. Der Einsatz und die Verwendung der Beziehungstypen nimmt Einfluss auf die Qualität des Inhaltes, da diese einen Teil des Inhaltes repräsentieren. Deshalb ist es den Autoren der Muster und Mustersprachen überlassen die richte Wahl der Verwendung von den Beziehungstypen zu treffen. Die Beziehungstypen besitzen Semantiken, an denen die Autoren herausfinden können in welchen Fällen der jeweilige Beziehungstyp verwendet werden kann. Ein Beispiel des Einsatzes von Beziehungen wurde bereits in der allgemeinen Erklärung der Zusammenhänge zwischen Muster und Mustersprachen in dem Abschnitt 3.5.4 *Modell der Mustersprachstruktur des Metamodells für Mustersprachen* geschildert.

3.6.2 Musterstrukturprofil des Metamodells für Mustersprachen

Das Musterstrukturprofil beschreibt die allgemeine Charakteristik der Muster. Die Charakteristik der Muster wird durch die Vorgabe der Musterstruktur und deren Einschränkungen gebildet. Im Gegensatz zu dem Beziehungsprofil, bei dem die Verwendung der vorgegebenen Arten von Beziehungstypen optional ist, werden für das Musterstrukturprofil Einschränkungen definiert, die eine Vorgabe der Musterstruktur erzwingen. Im Folgenden werden anhand der vorgestellten Mustersprachen aus dem Kapitel 2 *Grundlagen* die Inhaltsbestandteile der Musterstruktur ausgewählt und Vorschriften für deren Verwendung formuliert.

Um das Musterstrukturprofil erstellen zu können, werden zunächst die Muster und deren Charakteristik betrachtet. Muster sind Beschreibungen von Lösungen bezüglich ihrer

Problemstellungen, die sich in einem bestimmten Kontext befindet. Dieser Satz ist jedoch nicht die ganze Wahrheit, was ein Muster darstellt und enthält. Es stellt aber die Kernaussage dar, auf der das Musterstrukturprofil aufgebaut werden soll. Wie schon in dem Kaptitel 2 *Grundlagen* angerissen, gibt es noch weitere Anforderungen an Muster. Zum Beispiel sollen sie korrekte und anpassungsfähige Lösungen bieten und eine konsistente Menge von Muster darstellen. [BHS07, S. 30] Meszaros und Doble definierten in ihren MetaPatterns [MD96] eine Musterstuktur, die auch wie die Musterstuktur des Metamodells für Mustersprachen aus obligatorischen und optionalen Inhaltselementen besteht. Sie schreiben die Verwendung von bestimmten obligatorischen Inhaltselementen vor und empfehlen die Verwendung von bestimmten optionalen Inhalteselementen. Diese Auswahl der Inhalteselemente von Meszaros und Doble werden als Grundlage des hier vorgestellten Musterstrukturprofils verwendet. Im Folgenden werden die Fakten der Musterstrukturen von Meszaros und Doble vorgestellt und als Grundlage für das Musterstrukturprofil des Metamodells für Mustersprachen verwendet.

Meszaros und Doble beschreiben in dem ersten MetaPattern *A.1 Pattern: Pattern*, das sie in ihrer Arbeit [MD96] vorstellen, dass ein Muster eine Problembeschreibung, eine Lösungsbeschreibung und eine Beschreibung, wieso diese Lösung anwendbar ist, enthalten muss. Sie verweisen in diesem Kontext auf die obligatorischen Inhaltselemente. Es wird allerdings nicht vorgeschrieben, dass alle drei Inhaltselemente obligatorische Inhaltselemente sind.

In den folgenden Mustern wird oft auf die Beschreibung der Anwendbarkeit der Lösung verzichtet. Aus diesem Grund enthält das Musterstrukturprofil des Metamodells für Mustersprachen Spezialisierungen des obligatorischen Inhaltselement, die die Problem- und die Lösungsbeschreibung der Muster darstellen. Die Beschreibung der Anwendbarkeit der Lösung wird als optionales Inhaltselement in das Musterstrukturprofil aufgenommen.

In der Beschreibung der Musterstruktur von Meszaros und Doble werden deren obligatorische Inhaltselemente durch den Kontext und des Problems erweitert. Es wird außerdem beschrieben, dass die Anwendbarkeit der Lösung ein notwendiges Inhaltselement ist. Es ist jedoch keine obligatorisches Inhaltselement, da dessen Existenz abhängig von dem Lösungsinhalt ist. Die Beschreibung der Musterstruktur wird durch das Muster *Visible Forces* erweitert. Diese Erweiterung ist ein obligatorisches Inhaltselement, das die äußeren Einflüsse der Problematik und der Lösungsfindung beschreibt. Meta Informationen, die Informationen über die Muster enthalten werden in dieser Arbeit nicht als eigenständige Inhaltselemente angesehen. Dies betrifft zum Beispiel den Musterbezeichner. Die vorgestellten Inhaltselemente werden wie von Meszaros und Doble beschrieben in das Musterstrukturprofil des Metamodells für Mustersprachen übernommen. Meszaros und Doble gehen auch auf die Gestaltung von qualitativ hochwertigen Musterinhalten ein. Dies ist zwar nicht Bestandteil des Metamodells für Mustersprachen, es wird aber die Idee der Inhaltserweiterung durch ein Beispiel aufgenommen. Sehr viele Muster beinhalten Beispiel, die deren Lösungsansätze veranschaulichen sollen. Diese Charakteristik soll durch die Erweiterung mit einem optimalen Inhaltselement, das ein Beispiel der Lösung eines Musters darstellt, in das Musterstrukturprofil einbezogen werden.

Das Musterstrukturprofil des Metamodells für Mustersprachen ist in der Abbildung 3.7 zu sehen. In diesem Absatz wird die Struktur des Musterstrukturprofil und dessen Eingliederung in das Metamodell für Mustersprachen detailliert erklärt. Das Musterstrukturprofil besteht aus den Inhaltselementen Metainformationsbeschreibung (Meta Information Descriptor), Problembeschreibung (Problem Description), Lösungsbeschreibung (Solution Description), Kontextbeschreibung (Context Description) und Beschreibung der äußeren Einflüsse (Visible Forces Description). Diese Inhaltselemente sind Spezialisierungen des obligatorischen Inhaltselements und stellen damit feste Bestandteile von Mustern

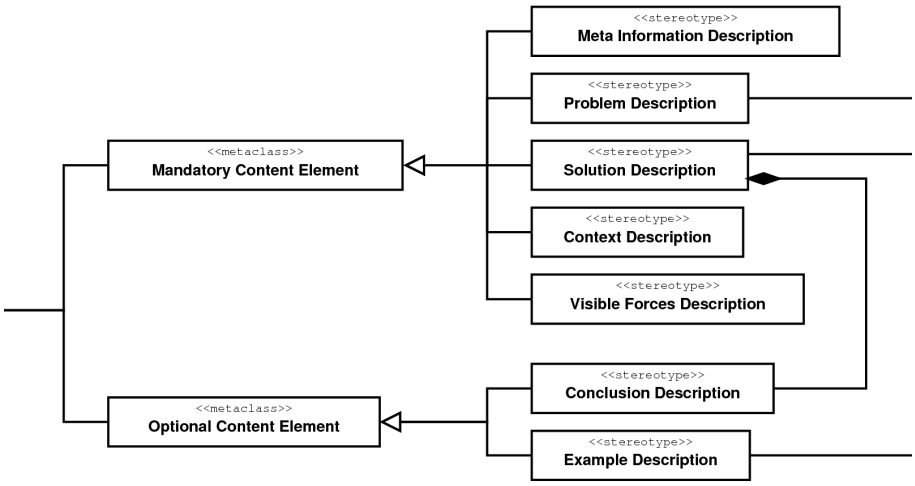


Abbildung 3.7: Profil der Musterstruktur

dar. In dieser Aufzählung wurde die Metainformationsbeschreibung als obligatorisches Inhaltselement hinzugenommen. Es soll die Informationen der Metainformationsbeschreibung der Musterstruktur als Inhaltselement repräsentieren. In der Metainformationsbeschreibung können sich zusätzlich zu den Metainformationen des Musters auch Anmerkungen zu verwandten Themen, Mustern oder zu anderen Namen, die das Muster tragen könnte gemacht werden. Alle Metainformationen, die die Muster betreffen soll in diesem Inhaltselement eingepflegt werden. Zusätzlich zu der Definition dieser fünf obligatorischen Inhaltselemente wird die Einschränkung gemacht, dass sie in jedem Muster aller Mustersprachen, die aus dem Metamodell für Mustersprachen abgeleitet worden sind, enthalten sein müssen. Das Musterstrukturprofil wird durch die optionalen Inhaltsbestandteile Schlussfolgerungs- und Beispielbeschreibungen erweitert. Zu beachten ist, dass die Schlussfolgerungsbeschreibung ein optionales Inhaltselement der Lösungsbeschreibung ist, weil sie erklärt wieso die Lösung am geeignetsten für das beschriebene Problem bezüglich des angegebenen Kontextes passt.

Es ist noch zu erwähnen, dass das Musterstrukturprofil nicht auf die Qualität des Inhalts der Muster eingeht. Die Konsistenz, Allgemeinheit, Absichten und weitere Aspekte, die bezüglich der Qualität des Inhalts der Muster abzielen, werden dem Autor der Muster überlassen. Meszaros und Doble veröffentlichten unter anderem in ihrem Werk *MetaPatterns: A Pattern Language for Pattern Writing* [MD96] Tipps und Ratschläge für Autoren von Mustersprachen. Diese bieten den Autoren eine Stütze für das Erstellen von qualitativ hochwertig Musterinhalten. Den Autoren wird empfohlen sich an den Ratschlägen zu orientieren und das Metamodell für Mustersprachen aus dieser Arbeit in Kombination zu verwenden.

3.6.3 Mustersprachprofil des Metamodells für Mustersprachen

Die Musterorganisationen und Zusatzfunktionalitäten für Mustersprachen wurden in dem Abschnitt 3.5.4 *Modell der Mustersprachstruktur des Metamodells für Mustersprachen* vorgestellt. Das Mustersprachprofil definiert verschiedene Standardelemente für Mustersprachen, die Spezialisierungen von Musterorganisationen und Zusatzfunktionalitäten für Mustersprachen sind. Im Folgenden wird das

Mustersprachprofil in zwei Teile unterteilt. Der eine beschreibt die Organisationen von Mustern und der andere die Zusatzfunktionalitäten für Mustersprachen.

3.6.3.1 Typen von Musterorganisationen im Mustersprachprofil

Die hier vorgestellten Typen von Musterorganisationen beruhen auf den Analysen von Buschmann. [BHS07, S. 209-246] Es werden die geläufigsten Musterorganisationen vorgestellt, die eine Grundlage für die meisten Mustersprachen bieten können.

Die Autoren der Muster haben viele Möglichkeiten um Muster darzustellen und diese anhand ihres Inhaltes in eine Vernetzung von Muster einzupflanzen. Leser verwenden Muster meistens, wegen ihrer Anwendbarkeit. Durch diese können sie ein bestimmtes Vorhaben in seinem Kontext umsetzen. Um Muster zu finden, benötigen sie Organisationen, die ihre Situation und Absichten darstellen können. [BHS07, S. 225]

In den folgenden Abschnitten werden Organisationen vorgestellt, die auf allgemeine Muster angewendet werden können und häufige von den Leser der Mustersprachen benötigt werden. Im Allgemeinen ist es vorgesehen, dass die Organisationen in Kombination verwendet werden können, um komplexere Organisationen darzustellen und mehr Ausdruck verleihen können.

Organisation von Musterebenen Viele Mustersprachen besitzen eine hierarchische Struktur, in der die Muster bezüglich ihrer Aggregationsbeziehung in verschiedene Ebenen unterteilt sind. Die Muster dieser Struktur stellen in den niedrigeren Ebenen Erweiterungen und Verfeinerungen der Muster aus den höheren Ebenen der Struktur dar. Umso höher die Ebene, desto abstrakter oder allgemeiner ist der Inhalt dessen Muster. [BHS07, S. 213-218]

Christpher Alexanders Mustersprache *A Pattern Langauge: Towns Buildungs Construction* [CAA77] enthält eine hierarchische Struktur, die bei Muster für Städten anfängt und gefolgt wird von Nachbarschaften, Gebäuden, Zimmer, ... bis hin zu Sitzgelegenheiten. An diesem Beispiel ist zu sehen, dass die Hierarchische Organisation von Mustern Umsetzungsregeln in ihren Traversierungswegen beinhaltet. [vv03]

Die Ebenen der Organisation von Musterebenen können auch durch die Art der Mustertypen benannt werden, die in der jeweiligen Ebene zu finden sind. Eine hohe Ebene beinhaltet zum Beispiel Muster, die architektonische Inhalte repräsentierten. Eine Stufe weiter unten könnten sich Designmuster befinden, die die Architekturmuster erweitern und verfeinern. Die Designmuster könnten wiederum von konkreteren bereichsabhängigen Mustern erweitert und verfeinert werden. Damit die Hierarchische Organisation von Mustern, die Mustertypen eindeutig zuordnen kann, könnte man zum Beispiel die Informationen des Mustertyps in dessen Metainformationen mitaufnehmen. [BHS07, S. 213-218] Bezüglich des Metamodells für Mustersprachen aus dieser Arbeit, bietet es sich an, in dem vorgestellten Szenario von Buschmann, drei verschiedene Mustersprachen zu erstellen. Dies hat den Vorteil, dass die Inhaltselemente für die Musterstruktur bezüglich der Kategorien der Ebenenen zusammengestellt werden können. Durch den Mechanismus der mustersprachenübergreifenden Referenzierung können Beziehungen zwischen den Mustersprachen und somit zwischen den Musterebenen erstellt werden. Anhand dieser Beziehungen kann die Organisation von Musterebenen die Muster der drei Mustersprachen darstellen.

Organisation von Musterbereichen Unter Musterbereichen versteht man Themenbereiche, die durch die Problem- und der Kontextbeschreibung der Muster angesprochen werden. Die Muster werden in Organisationen von Musterbereichen bezüglich ihrer Themenbereiche in Kollektionen unterteilt. Die Kollektionen können zum Beispiel nach folgenden Themenbereichen unterteilt werden: Telekommunikation, Finanzwesen, Gesundheitswesen, Luftfahrtelektronik, Logistik, Lehre, ... In der Organisation von Musterbereichen können auch Schnittmengen von Themenbereichen gebildet werden, um Überschneidungen der Themenbereiche darzustellen. [BHS07, S. 218-219]

Organisation von Musterpartitionen Musterpartitionen stellen einen engeren Bereich als Musterbereiche dar. Sie beziehen sich ursprünglich auf verschiedene Bereich der Architekturlösungen von Muster. Die Musterbereiche stellen somit verschiedene Technologien und Konzepte dar. Sie können beispielsweise folgendes darstellen: verschiedene Architekturebenen, Schichten, Komponenten, Pakete oder Rollen. [BHS07, S. 218-221] Die Organisation von Musterpartitionen soll in dem Metamodell für Mustersprachen für allgemeine Musterbereiche eingesetzt werden. Die Organisation von Musterpartitionen könnte so zum Beispiel als Verfeinerung der Musterbereiche dienen.

Organisation von Musterabsichten Eine weitere häufige Organisation, die in Mustersprachen verwendet wird, ist die Organisation von Musterabsichten. Muster werden dabei bezüglich ihres Inhaltes in Kollektionen unterteilt. Solche Kollektionen haben zum Beispiel eine gleiche architektonische Charakteristik oder verfolgen ein gleiches Ziel. Oft beinhalten Mustersprachen Muster, die solch eine Struktur darstellen. Zum Beispiel stellt das Muster der Beschreibung einer Stadt von Christopher Alexander eine Kollektion von weiteren Mustern dar. Diese Kollektion bildet eine Struktur, die als Teil einer Organisation von Musterabsichten angesehen werden kann. [CAA77, S. 3-7] Organisation von Musterabsichten sind sinnvoll für das Auffinden von Musterproblembeschreibungen oder deren Lösungsbeschreibungen, da die Muster nach den Absichten ihrer Problemstellung oder Lösungsbeschreibung organisiert sind. [BHS07, S. 221-224]

Profil der Musterorganisationen Die Abbildung 3.8 zeigt das Profil der Mustersprachorganisationen. Es ist die Metaklasse der Musterorganisationen (Pattern Organisation) und seine Spezialisierungen zu sehen. Die Spezialisierungen sind die vorgestellten Musterorganisationen des Profils der Musterorganisationen: Organisation von Musterebenen (Pattern Organisation by Level), Organisation von Musterbereichen (Pattern Organisation by Domain), Organisation von Musterpartitionen (Pattern Organisation by Partition) und Organisation von Musterabsichten (Pattern Organisation by Intent).

3.6.3.2 Zusatzfunktionalitäten für Mustersprachen im Mustersprachprofil

Die Ideen und Anregungen der Auswahl von Zusatzfunktionalitäten für das Mustersprachprofil stammen aus der Beschreibung der Mustersprachstruktur von Meszaros und Doble. Sie sollen durch die Erweiterung der Mustersprachstruktur die nicht triviale Vernetzung der Muster übersichtlicher darstellen. Sie erstellen zusätzliche Strukturen, die Musterinhalte erweitern und auf die entsprechenden Muster verweisen. Umsetzung solcher konkreten Mechanismen werden hier als Zusatzfunktionalitäten vorgestellt. [MD96]

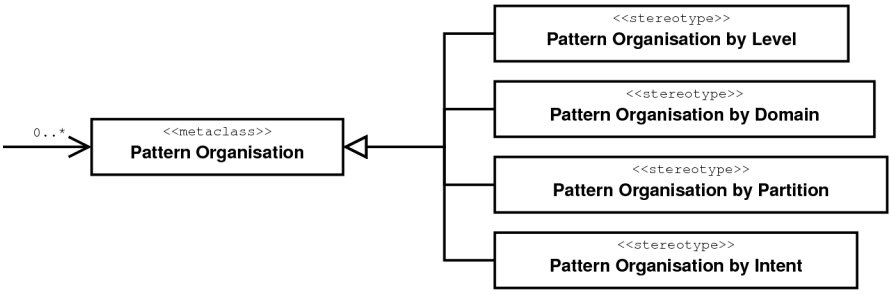


Abbildung 3.8: Arten von Musterorganisationen im Mustersprachprofil

Zusammenfassung der Musterinhalte Die Zusammenfassung der Musterinhalte spiegelt die Themenbereich der Mustersprache wieder. Sie soll einen Überblick über die wichtigsten Muster der Mustersprache liefern, deren Probleme und Lösungen schildern und ihre Zusammenhänge und Anwendbarkeit erläutern. Es können auch die Absichten der Mustersprache als Ganzes und deren Ausblicke auf die Erweiterung von Mustern und Sammlungen von Lösungen festgehalten werden. [MD96]

Problem-/Lösungstabelle der Muster Die Zusatzfunktionalitäten Problem-/Lösungstabelle der Muster erweitert eine Mustersprache durch eine Tabelle, die Probleme und Lösungen ausgewählter Muster in verkürzter Form gegenüberstellt. Diese Tabelle stellt auch ein Zusammenfassung der Muster dar. Im Gegensatz zu der Zusammenfassung der Musterinhalte werden die Inhalte der Muster getrennt dargestellt. Es stellt somit keine Beschreibung der Mustersprache dar sondern eine detaillierte Übersicht über die Probleme und Lösungen der Muster. Die Tabelle bietet den Lesern der Mustersprache ein Nachschlagewerk der wichtigsten Muster einer Mustersprache. [MD96]

Musterübergreifende Beispiele Ein musterübergreifendes Beispiel ist eine Darstellung eines Beispiels, das über mehrere Muster fortgeführt wird. Muster, die ein Teil eines fortlaufenden Beispiels enthalten, bilden eine Musterkollektion bezüglich dieses Beispiels. Oft stellen diese Musterkollektionen auch Mustersequenzen dar. [MD96] Eine Mustersequenzen ist die Anordnung von Mustern, in der ihre Lösungen angewendet werden. In den meisten Fällen werden mehre Muster benötigt, um ein Problem zu lösen. Die Mustersequenzen ist in diesen Fällen die Reihenfolge der anzuwendenden Muster. [BHS07, S. 192-193]

Wörterverzeichnis der Musterinhalte Um Fachbegriffe zu erklären, soll ein Wörterverzeichnis die Musterpsprachen erweitern können. In einem Wörterverzeichnis werden die Fachbegriffe alphabetisch geordnet, erklärt und auf die Muster verwiesen, in denen dieser Begriff enthalten sind. Implizit wird durch die Erklärung der Fachbegriffe und die Referenzierung auf die entsprechenden Muster Verbindungen zwischen Muster hergestellt. Diese Verbindungen sind anhand der Fachbegriffe entstanden und erlangen sogar durch die Erklärung der Fachbegriffe eine Repräsentation ihrer Bedeutung. [MD96]

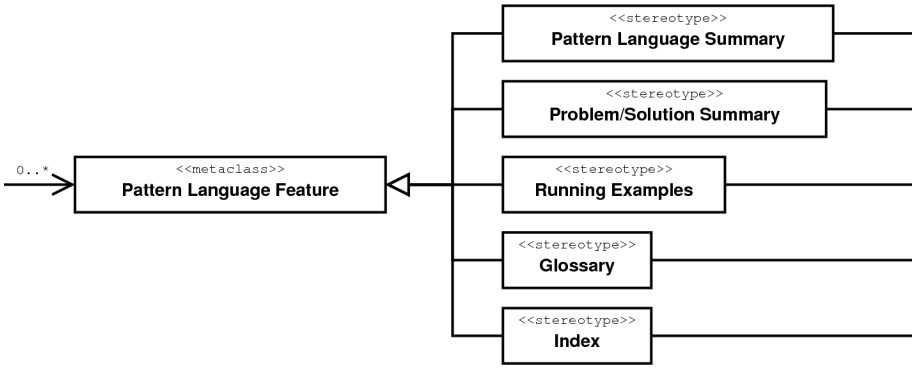


Abbildung 3.9: Profil der Mustersprachorganisationen

Inhaltsverzeichnis für Mustersprachen Musterorganisationen können die Muster abhängig von deren Inhalten und Vernetzungen darstellen. Große Mustersprachen oder mehrere Mustersprachen, die von einer Musterorganisation dargestellt werden, können sehr unübersichtlich werden. Das Inhaltsverzeichnis für Mustersprachen ist eine hierarchische Struktur, die manuell angelegte Kategorien enthält. Autoren sortieren ihre Muster entsprechend ihrer Inhalte und Abhängigkeiten in die jeweilige Kategorie des Inhaltsverzeichnisses ein. Somit wird eine übersichtliche Struktur geschaffen, die den Inhalt der Mustersprache wiedergibt und die Muster in einem groben Raster organisiert. [MD96]

Profil der Zusatzfunktionalitäten für Mustersprachen In der Abbildung 3.9 ist die Metaklasse der Zusatzfunktionalitäten und die vordefinierten Arten von Zusatzfunktionalitäten des Mustersprachprofils zu sehen. Die vorgestellten Zusatzfunktionalitäten Zusammenfassung der Musterinhalte (Pattern Language Summary), Problem-/Lösungstablle der Muster (Problem/Solution Summary), musterübergreifende Beispiele (Running Example), Wörterverzeichnis der Musterinhalte (Glossary) und das Wörterverzeichnis der Musterinhalte (Index) sind Spezialisierungen der Metaklasse für Zusatzfunktionalitäten von Mustersprachen. In diesem Ausschnitt sind die Beziehungen der Zusatzfunktionalitäten zu den einzelnen Bestandteilen der Inhaltselementen von Mustern zu erkennen. In der Abbildung *Metamodells für Mustersprachen* in Anhang A ist die Darstellung der Beziehungen zwischen den Zusatzfunktionalitäten und den Inhaltselementen der Muster auf der rechten Seite zu sehen. Durch diese Beziehungen werden inhaltliche Abhängigkeiten dargestellt. Je nach Gestaltung der Zusatzfunktionalitäten für Mustersprachen können die Beziehungen erweitert werden.

Kapitel 4

Plattform für Mustersprachen und Musterkataloge

In diesem Kapitel wird eine Plattform für Mustersprachen und Musterkataloge vorgestellt. Zunächst werden deren Aufgaben und Ziele erläutert. Anschließend werden die bestehende Musterkataloge vorgestellt und die Abgrenzung der Plattform für Mustersprachen und Musterkataloge erklärt. Der Hauptteil dieses Kapitels ist die Spezifikation und der Aufbau der Plattform für Mustersprachen und Musterkataloge. Die Spezifikation dient zur Beschreibung der Plattform und der Definition ihrer Anforderungen.

Im Rahmen dieser Diplomarbeit wurde eine Software erstellt, die die Spezifikation der Plattform für Mustersprachen und Musterkataloge erfüllt. Ein weiterer Bestandteil der Softwareplanung ist die Beschreibung der Struktur der Plattform für Mustersprachen und Musterkataloge, die anschießend nach der Spezifikation vorgestellt wird. Die Struktur der Plattform erfüllt die Anforderungen der Spezifikation und stellt das Bindeglied zwischen Spezifikation und Implementierung dar. Die Implementierungsbeschreibung enthält die Umsetzung und die Verwendung der Spezifikation und der Struktur von der Plattform für Mustersprachen und Musterkatalogen. Die Inbetriebnahme der Plattform für Mustersprachen und Musterkataloge beschreibt die Installation der Plattform. [LL10, S. 353-356, 399-400]

4.1 Aufgaben und Ziele der Plattform für Mustersprachen und Musterkataloge

In diesem Abschnitt werden die Aufgaben und Ziele der Plattform für Mustersprachen und Musterkataloge vorgestellt. Sie werden als Anforderungen an die Plattform formuliert. Die Anforderungen werden in verschiedene Bereiche unterteilt, um sie übersichtlich darzustellen. Es werden die Bereiche Anforderungen der Modellabhängigkeiten des Metamodells für Mustersprachen, Anforderungen der Autoren, Anforderungen der Leser von Muster und technische Anforderungen vorgestellt. Die Inhalte der Anforderungsbereiche sollen außerdem eine Vorschau für die benötigten Bestandteile der Plattform liefern. [LL10, S. 353-355, 366-374]

Anforderungen der Modellabhängigkeiten des Metamodells für Mustersprachen Die Plattform für Mustersprachen und Musterkataloge soll für die Autoren von Mustersprachen eine webbasierte Verwaltungssoftware für die Erstellung und Verwaltung von Mustersprachen und deren Musterkataloge darstellen. Die Mustersprachen der Plattform sind Instanzen des vorgestellten Metamodells für

Mustersprachen. Autoren sollen mit Hilfe der Plattform Mustersprachen in diese einpflegen können. Diese Mustersprachen dienen als Modell der Musterkataloge, die die Autoren ebenfalls in die Plattform einpflegen und erweitern können.

Anforderungen der Autoren Die Plattform soll zusätzlich eine Verwaltung von verschiedene Musterversionen unterstützen, damit den Autoren eine komfortable Mustererstellung geboten wird. Die Autoren und Leser der Muster sollen zudem Diskussionen über die Inhalte der Muster führen können. Für diesen Zweck soll die Plattform ein Forum für Diskussionsbeiträge über die Musterinhalte bereitstellen.

Anforderungen der Leser von Muster Für die Leser der Muster sollen verschiedene Musterorganisationen und Zusatzfunktionalitäten der Mustersprachen zur Verfügung stehen. Die Musterorganisationen und Zusatzfunktionalitäten sollen für mustersprachenübergreifende Suchen und Verwaltungen von Muster verwendet werden. Die Musterorganisationen sollen ein dynamisch erzeugtes Suchergebnis liefern, das aus mehreren ausgewählten Kriterien zusammengestellt wird und zusammenhängende Muster aus verschiedenen Musterkatalogen darstellt. Für diese Musterkollektionen sollen Darstellungen bereitgestellt werden, die die Vernetzung der Muster anschaulich darstellen.

Technische Anforderungen Damit die Daten der Plattform für Mustersprachen und Musterkataloge effizient verwaltet werden, soll ein Repository die Datenaufbereitung und -verwaltung für die Plattform übernehmen. Die Plattform soll für Erweiterungen konzipiert sein und sie soll die Integration von weiteren Bestandteilen ermöglichen.

4.2 Abgrenzung zu bestehenden webbasierten Musterkatalogen

In diesem Abschnitt werden die in dem Kapitel 2 *Grundlagen* vorgestellten webbasierten Musterkataloge mit der Plattform für Mustersprachen und Musterkataloge verglichen und deren Unterschiede präsentiert.

Die vorgestellten webbasierten Musterkataloge basieren alle auf einer Musterstruktur. Außerdem bezieht sich der Inhalt der Muster nur auf einen Themenbereich. Die hier vorgestellte Plattform für Mustersprachen und Musterkataloge bietet im Gegensatz zu den vorgestellten webbasierten Musterkatalogen die Erstellung mehrerer Mustersprachen für verschiedene Themenbereiche an. Je nach Themenbereich können die Mustersprachen unterschiedliche Musterstrukturen enthalten, die der Autor der Mustersprache selbst definieren kann. Außerdem ist es möglich Verknüpfungen zwischen den Mustersprachen herzustellen.

Die vorgestellten webbasierten Musterkataloge werden nur durch ausgewählten Autoren und Experten der jeweiligen Themenbereiche gepflegt und sind für Dritte lesbar. Es gibt noch weitere Musterkataloge, wie zum Beispiel das *Patternry* [LVH11], das sich thematisch mit Benutzeroberflächenmuster befasst. Es basiert auf der Masterarbeit [Lam07] von Janne Lammi, die über dynamisch erzeugte Konzepte und Techniken und über die Erstellung und den Aufbau eines webbasierten Musterkatalog berichtet. In der Arbeit von Janne Lammi wird detailliert auf die Benutzerschnittstelle des Musterkataloges und den Prozess der Mustererstellung eingegangen.

In dem Musterkatalog Patternry [LVH11] kann jeder eigene Muster erstellen und diese für andere Leser und Autoren zur Verfügung stellen. Autoren der Plattform für Mustersprachen und Musterkataloge sollten Experten bezüglich der Themengebiete ihrer Muster sein und qualifizierte Beiträge bezüglich der Mustersprachen erbringen. Die Muster der Plattform sind allen Autoren zugänglich. Es ist dem Betreiber der Plattform für Mustersprachen und Musterkataloge überlassen, ob die Muster auch für Dritte zugänglich sind. Im Gegensatz zu Patternry können keine Muster erstellt werden, die für andere Autoren nicht zugänglich sind. Muster, die sich in der Plattform befinden, sollen für alle Autoren öffentlich sein und von diesen in anderen Mustern referenziert werden können.

Die vorgestellten Musterkataloge besitzen fast alle Kategorien, in denen die Muster unterteilt sind. Diese Unterteilungen bestehen oft aus zwei Hierarchieebenen. Zusätzlich werden oft Suchfunktionen nach Musternamen oder deren Textinhalten angeboten. Oft wird auch ein Index mit den Musternamen zur Verfügung gestellt. Die Plattform für Mustersprachen und Musterkataloge enthält erweiterte Suchmechanismen und Darstellungen, die die Vernetzung der Muster darstellen. Die Verwendung von Musterorganisationen und Zusatzfunktionalitäten der Mustersprachen wurde in dem Abschnitt 4.1 *Aufgaben und Ziele der Plattform für Mustersprachen und Musterkataloge* bereits vorgestellt. Eine detaillierte Spezifikation des Aufbaus der Musterorganisationen und Zusatzfünktionalitäten der Mustersprachen wird in dem Kapitel 4.3 *Spezifikation der Plattform für Mustersprachen und Musterkataloge* beschrieben.

Einige webbasierte Musterkataloge bieten die Möglichkeit über Muster Diskussionen zu führen. Die Plattform für Mustersprachen und Musterkataloge stellt auch die Möglichkeit, Diskussionen über Muster zu führen, zur Verfügung.

4.3 Plattformspezifikation für Mustersprachen und Musterkataloge

In diesem Bereich der Arbeit wird eine Spezifikation durch die Verfeinerung der Anforderungen erstellt. Diese Anforderungen stellt die Aufgaben und die Ziele der Plattform für Mustersprachen und Musterkataloge dar. Die Spezifikation enthält Funktionalitäten, die die Funktionsweise, die Verwendungen und die Handhabung der Plattform beschreiben. Sie enthält die allgemeine Anforderungen, die Absichten, die Funktionalitäten und die Handhabung mit der Plattform für Mustersprachen und Musterkatalogen. Für die Bewältigung der Anforderungen an die Plattform wird das Metamodell für Mustersprachen als grundlegendes Datenmodell verwendet. Die Spezifikation erweitert das Profil des Metamodells für Mustersprachen. Damit kann das Metamodell für Mustersprachen in Kombination mit einer Verwaltungssoftware verwendet werden. Die Spezifikation bietet zusätzlich die Grundlage für den Entwurf und die Struktur der Plattform für Mustersprachen und Musterkataloge. [LL10, S. 353-356, 399-400]

Im Folgenden wird zunächst die Erstellung und die Verwaltung von Mustersprachen in der Plattform für Mustersprachen und Musterkataloge sowie die dafür notwendige Anpassung des Metamodells für Mustersprachen beschrieben. Anhand einer gepflegten Mustersprache können Musterkataloge instanziiert werden. Im Anschluss wird die Erstellung und die Verwaltung von Musterkatalogen in der Plattform für Mustersprachen und Musterkatalogen vorgestellt. Zusätzlich zu den weiteren Anpassungen des Metamodells für Mustersprachen, wird der Suchmechanismus der Plattform vorgestellt.

4.3.1 Erstellung und Verwaltung von Mustersprachen

Die Plattform für Mustersprachen und Musterkataloge ermöglicht es, dass Mustersprachen angelegt werden können. Die Mustersprachen basieren alle auf dem Metamodell für Mustersprachen. Durch das Anlegen einer Mustersprache wird eine Instanz des Metamodells für Mustersprachen gebildet. Die Musterkataloge sind wiederum Instanzen der Mustersprachen. Die Musterkataloge liegen den Beschreibungen der entsprechenden Mustersprachen zu Grunde. Die Informationen über den Aufbau eines Musterkataloges befindet sich in der entsprechenden Mustersprache. Diese Beschreibungen für den Aufbau der Musterkataloge, in Form von Mustersprachen, sind in der Plattform für Mustersprachen und Musterkataloge hinterlegt. Bei der Erstellung von Mustersprachen müssen nicht alle Elemente des Metamodells für Mustersprachen instanziert werden. Im Folgenden wird das Profil des Metamodells für Mustersprachen durch vordefinierte Elemente, die bei der Erstellung einer Mustersprache verwendet werden, erweitert. Zunächst wird die Erstellungsphase einer Mustersprache in der Plattform für Mustersprachen und Musterkataloge vorgestellt.

Erstellung von Mustersprachen Bei der Erstellung einer Mustersprache liegt der Schwerpunkt auf der Definition der Musterstruktur. Die Musterstruktur wird durch Inhaltselemente definiert. Die Inhaltselemente enthalten wiederum Inhaltsarten, die die konkreten Inhalte der Muster enthalten. Die Arten der Inhalte werden von der Plattform für Mustersprachen und Musterkataloge vordefiniert. Es können nur die Arten von Inhalten verwendet werden, die die Plattform für Mustersprachen vorsieht. Für die Erstellung einer Mustersprache müssen auch die obligatorischen und optionalen Inhaltselemente, die durch das Metamodell für Mustersprachen vorgesehen sind, angegeben werden. Mit ihnen wird die Musterstruktur der Mustersprache vervollständigt. Die Arten der Inhaltselemente sind in dem Profil des Metamodells der Mustersprachen in der Plattform hinterlegt. Die Plattform enthält ebenso ein vorgefertigtes Konstrukt für Referenzen auf Strukturelementen. Diese können in Kombination mit den entsprechenden Beziehungstypen verwendet werden. Diese Beziehungstypen zwischen Strukturelementen sind ebenfalls in der Plattform für Mustersprachen und Musterkataloge vordefiniert. Um die Erstellung einer Mustersprache fertigzustellen, muss noch deren Mustersprachstruktur definiert werden. Sie enthält Musterorganisationen und Zusatzfunktionalitäten. Die Musterorganisationen sind Strukturen, die in der Plattform für Mustersprachen und Musterkataloge erstellt werden können. Das Profil des Metamodells für Mustersprachen bietet die optionale Verwendung von den bereits vorgestellten Musterorganisationen. Zusätzlich können weitere Musterorganisationen in die Plattform eingepflegt und verwendet werden. Die Zugehörigkeit und Einordnung der Muster in Musterorganisationen wird im folgenden Abschnitt 4.3.2 *Erstellung und Verwaltung der Musterkataloge* geklärt. Die Zusatzfunktionalitäten einer Mustersprache sind ebenfalls durch das Profil des Metamodells für Mustersprachen vorgegeben. Sie stellen spezielle Funktionalitäten dar, die optional verwendet werden können. Das Erstellen von weiteren Zusatzfunktionalitäten wird von der Plattform für Mustersprachen und Musterkataloge nicht unterstützt, weil die Zusatzfunktionalitäten sehr spezielle und unterschiedliche Konstrukte darstellen. Nach dem Anlegen eines Musterkataloges kann dessen Mustersprachen nicht mehr geändert oder gelöscht werden, da die Mustersprachen Beschreibungen und Vorschriften der Musterkataloge darstellen und Veränderungen der Mustersprachen Inkonsistenzen hervorbringen könnten. Es wird lediglich die Erweiterung der Musterorganisationen zur Verfügung gestellt.

Beispiel einer Mustersprache In Anhang B ist eine Abbildung zu sehen, die ein Beispiel einer Mustersprache darstellt. Dieses Beispiel soll helfen, die Erstellung einer Mustersprache mit vordefinierten und neu erstellten Elementen zu erklären. In der Abbildung sind vordefinierte Elemente der Plattform für Mustersprachen und Musterkataloge und auch neue Elemente enthalten. Beispielsweise sind die Inhaltselemente (Content Pattern Element) der Musterstruktur für diese Beispielmustersprache verfeinert worden. Ein neu erstelltes Inhaltselement ist zum Beispiel das Diskussionselement (Discussion). Es ist von dem Typ der optionalen Inhaltselemente (optional Pattern Content Element). Das Diskussionselement enthält einen neuen Inhaltstyp (Discussion Representation), der den Inhalt und die Darstellung der Diskussionen definiert. Das Diskussionselement wird weiter verfeinert, in dem es wiederum Beitragselemente (Contribution) enthält, die von dem Typ der optionalen Inhaltselemente abgeleitet sind und die den bereits vordefinierten textbasierenden Inhaltstyp (Text Content) enthalten. Es wurden auch konkrete Inhaltselemente aus dem Profil des Metamodells für Mustersprachen übernommen, wie zum Beispiel das Beispielelement.

Erweiterung des Metamodellprofils für Mustersprachen In diesem Abschnitt wird das Profil des Metamodells für Mustersprachen an die Plattform für Mustersprachen und Musterkataloge angepasst. Das Profil des Metamodells für Mustersprachen wird durch vordefinierte Elemente erweitert. Diese Erweiterungen, die sich auf die Erstellung der Mustersprachen in der Plattform für Mustersprachen und Musterkatalogen beziehen, werden im Folgenden vorgestellt und sind in der Abbildung im Anhang C dargestellt. Die Abbildung enthält das angepasste Profil für Mustersprachen der Plattform, das mit der folgenden Erklärung vervollständigt wird. Die Bezeichner der Profilerweiterungen aus der Abbildung sind englisch dargestellt und werden in den folgenden Profilerweiterungen in Klammern hinter den deutschen Begriffen aufgeführt.

Zunächst werden die Arten der Inhaltselemente, deren Inhaltsarten und ihre Abhängigkeiten vorgestellt. Danach werden die Beziehungsarten und Referenzierungen von Strukturelementen präsentiert. Anschließend werden die Arten von Musterorganisationen und Zusatzfunktionalitäten erklärt.

Inhaltselement (Content Element) Die Inhaltselemente von Muster werden aus dem Profil des Metamodells für Mustersprachen übernommen und erweitert. Damit sie als Datenstruktur für die Plattform für Mustersprachen und Musterkatalogen verwendet werden können. Die bereits bestehenden Inhaltselement Metainformation (Meta Information Description), Problembeschreibung (Problem Description), Beschreibung der äußere Einflüsse (Visible Forces Description), Kontextbeschreibung (Context Description), Lösungbeschreibung (Solution Description), deren Schussfolgerungsbeschreibung (Conclusion Description) sowie das Beispiel (Example Description) der Musterstruktur wurden aus dem Profil des Metamodells für Mustersprachen übernommen. Im Folgenden werden die erweiterten Inhaltselemente erläutert.

> **Diskussion (Discussion)** Das Diskussionselement wird in der Plattform für Mustersprachen verwendet. Damit können Diskussionen über die jeweiligen Musterinhalte geführt werden.
>
> **Dikussionsbeiträge (Contribution to the Discussion)** Die Diskussionbeiträge sind unterelemente des Diskussionselementes. Sie enthalten die Beiträge der Autoren.

Anhang (List of Attachments) In das Anhangselement können Dateien angefügt werden. Diese Informationen helfen die Muster zu vervollständigen.

Anhangsdatei (Attachment Element) Eine Anhangsdatei repräsentiert eine Datei, die sich in dem Anhang des Musters befindet.

Arten von Inhalten (Content) In dem Inhaltselement der Muster wird ein Teil des Musterinhaltes beschrieben. Die Arten der Inhalte werden benötigt, um zu unterscheiden was für einen Inhalt es sich handelt und wie dieser beispielsweise verwendet oder repräsentiert wird. Im Folgenden werden die Inhaltselement der Plattform für Mustersprachen und Musterkataloge vorgestellt:

Metainformationsinhalt (Meta Data Content) Die Inhaltsart Metainformationsinhalt enthält Informationen, die den Inhalt eines Musters beschreiben. Durch die Definition dieser Art ist es möglich die Metainformationen zu verändern und dem Benutzer zu präsentieren. Diese Inhaltsart wird dem Inhaltselement der Metainformation zugeordnet.

SimpleHTML (Textual Content) Die Inhaltsart SimpleHTML enthält hauptsächlich Text, der graphisch in den Musterkatalogen repräsentieren wird. Dieser Text wird durch einfache HTML-Tags formatiert. Die Formatierung mit HTML bietet sich sehr gut an, da die Textinhalte in einem Webbrowser angezeigt werden. In dem Text können zusätzliche Verknüpfungen auf anderen Mustern enthalten sein. Auch Hyperlinks können zur Beschreibung verwendet werden. Es können ebenso Bilder in den Text eingebettet werden, um den Text verständlicher zu gestalten. Die Inhaltsart SimpleHTML wird den Inhaltselementen Problembeschreibung, Lösungsbeschreibung, Schlussfolgerung, Kontextbeschreibung, Beschreibung der äußeren Einflüsse und dem Beispiel zu geordnet.

Diskussion (Discussion Content) Die Inhaltsart Diskussion stellt eine Liste von Diskussionsbeiträgen bereit und wird hauptsächlich für die Repräsentation und deren Diskussionsbeitrag in der Plattform benötigt. Die Diskussionsinhaltsart wird dem Inhaltselement Diskussion zugeordnet.

Diskussionsbeitrag (Contribution Content) Die Inhaltsart Diskussionsbeitrag ist eine Spezialisierung der Inhaltsart SimpleHTML. Es wird zwischen den beiden Arten unterschieden, damit eine andere Darstellungsart für die Diskussionsbeitrag gewählt werden kann. Die Inhaltsart Diskussionsbeitrag wird dem Inhaltselement Dikussionsbeiträge zugeordnet.

Anhänge (Attachment List) Die Inhaltsart Anhänge bietet die Möglichkeit beliebige Dateien an ein Muster anzufügen. Somit können zum Beispiel Bilder, Konfigurationsdateiten, Beispielcode und viele weitere Informationen an ein Muster angefügt werden. Die Anhänge stellen zusätzlich Musterinformationen dar, die nicht notwendigerweise in dem Musterkatalog graphisch Repräsentation werden müssen. Diese Inhaltsart wird dem Inhaltselement Anhang zugeordnet.

Anhangsdatei (Attachment File) Die Inhaltsart Anhangsdatei repräsentiert eine angefügte Datei, die sich in dem Anhang befindet. Sie wird dem Inhaltselement Anhangsdatei zugeordnet.

Arten von Beziehungen zwischen Mustersprachen, Muster und Musterinhalten (Structure Elemente Relation) Um Referenzierungen zu erstellen, wird die Beschreibung der Beziehung zu den referenzierten Elementen benötigt. Die Plattform für Mustersprachen und Musterkataloge sieht dafür die in den folgenden aufgelisteten Arten von Beziehungen zwischen Mustersprachen, Muster und Musterinhalten vor. Die Beziehungsarten wurden aus dem Metamodell für Mustersprachen übernommen und werden im Folgenden für die Plattform für Mustersprachen und Musterkataloge verfeinert.

Spezialisierungsbeziehung (Specialisation SER) Es können Spezialisierungen von Mustersprachen, Muster und Musterinhalten erstellt werden. Die Spezialisierungen von diesen Elementen besitzen eine Spezialisierungsbeziehung zu ihren Generalisierungen.

Vervollständigende Aggregationsbeziehung (Completing Aggregation - SER) Die Beziehung einer vervollständigenden Aggregationsbeziehung besteht zwischen zwei Elementen, wenn das eine Element den Inhalt des anderen durch Einschluss seines Inhalts erweitert.

Alternative Aggregationsbeziehung (Competing Aggregation - SER) Die alternative Aggregationsbeziehung besteht auch zwischen zwei Elementen, wenn das eine Element den Inhalt des anderen durch Einschluss seines Inhaltes erweitert. Zusätzlich stellt der Einschluss des Inhaltes eine Alternative zu einem anderen Einschluss eines Inhaltes dar.

Erweiternde Aggregationsbeziehung (Combining Aggregation - SER) Die erweiternde Aggregationsbeziehung wird auch verwendet, um den Inhalt eines Elementes mit dem eines anderen zu erweitern. Im Gegensatz zu der vervollständigenden und der alternativen Aggregationsbeziehung stellen die Inhalte der Elemente mit der erweiternden Aggregationsbeziehung optionale Zusatzinformationen dar.

Assoziationsbeziehung (Assoziation SER) Eine Assoziationsbeziehung wird verwendet, wenn eine Beziehung vorliegt, die nicht durch die bereits vordefinierten Beziehungen abgedeckt ist. Die Assoziationsbeziehung stellt eine universale Beziehung dar.

Referenzen von Mustersprachen, Muster und Musterinhalte Die Plattform für Mustersprachen und Musterkataloge bietet einen Mechanismus an, um Mustersprachen, Muster und Musterinhalte zu referenzieren. Dabei wird das Modell der Strukturelementreferenz aus dem Metamodell für Mustersprachen verwendet. Eine Referenz der Plattform für Mustersprachen und Musterkataloge wird durch einen Hyperlink verkörpert. Die Referenz enthält zusätzlich zu dem Namen (Identifier) und der Adresse (Reference URI) des Hyperlinks eine Beschreibung der Beziehung, die zwischen dem referenzierten Element und dem Element, das die Referenz enthält besteht. Optional kann die Referenz auch mit dem Symbol (Icon) des Elements dargestellt werden. Die Beschreibung der Beziehungen verwendet die vordefinierte Art von Beziehungen zwischen Mustersprachen, Muster und Musterinhalten, um deren Semantik auszudrücken. Zusätzlich können Gemeinsamkeiten von Elementen aufgezählt werden, die durch ihre Beziehung dargestellt werden sollen. Die Gemeinsamkeiten können beschreiben aus welchen Gründen die Beziehung zwischen den Strukturelementen von dem Autor gewählt wurde und welchen Schwerpunkt sie darstellt.

Arten von Musterorganisationen (Pattern Organisation) Die Plattform für Mustersprachen und Musterkataloge bietet die Möglichkeit die Muster der Musterkataloge in Musterorganisationen

einzuordnen. Durch diese Einteilung wird die Vernetzung der Muster verstärkt. Die Musterorganisationen stellen Hierarchien dar, die durch Kategorien aufgebaut sind. Die ursprüngliche Idee des Metamodells für Mustersprachen ist es die Musterorganisationen anhand der Inhalte von Mustern aufzubauen. Der Einfachheit halber werden die Musterorganiation nicht automatisch von der Plattform für Mustersprachen und Musterkataloge erstellt und deren Muster eingepflegt. Autoren können bei der Erstellung von Mustersprachen vorhandene Musterorganisationen auswählen oder neue erstellen. Den Autoren wird die Erstellung der Struktur von Musterorganisationen und das Einpflegen der Muster überlassen. Durch das hinzufügen von Informationen in die Metainformationen der Muster, können die Muster den Musterorganisationen zugeordnet werden. Im Folgenden werden vordefinierte Arten von Musterorganisationen vorgestellt. Diese Musterorganisationen basieren alle auf dem Metamodell für Musterspachen. Sie wurden in dem Abschnitt 3.6.3.1 *Typen von Musterorganisationen im Mustersprachprofil* vorgestellt und werden für den Einsatz in der Plattform für Mustersprachen und Musterkataloge verfeinert. Zusätzlich ist es möglich weitere Musterorganisation in der Plattform für Mustersprachen und Musterkatalogen zu erstellen und deren Struktur zu definierten.

Organisation von Musterebenen (Pattern Organisation by Level) Die Organisation von Musterebenen ist eine mustersprachübergreifende Menge von vernetzten Mustern. Die Vernetzungen der Muster werden durch deren Referenzen dargestellt. Die von den Autoren bei der Erstellung der Muster angelegt werden.

Organisation von Musterbereichen (Pattern Organisation by Domain) Die Organisation von Musterbereichen stellt eine mustersprachübergreifende Hierarchie von Bereichen jeglicher Art dar. Es können sich mehrere Muster aus unterschiedlichen Mustersprachen in den selben Bereichen befinden. Zudem können auch Muster mehreren Bereichen zugeordnet werden. Damit die Organisation von Musterbereichen aufgebaut werden kann, müssen die Autoren eine Hierarchie von Musterbereichen festlegen. Die Metainformationen (Meta Information) der Muster werden erweitert, damit man ihnen die Musterbereiche (Domain) zuordnen kann. Aus den angehefteten Musterbereichsinformationen, kann die Organisation von Musterebenen aufgebaut werden.

Organisation von Musterpartitionen (Pattern Organisation by Partition) Die Organisation von Musterpartitionen ist eine mustersprachabhängige hierarchische Unterteilung in Partitionen. Die Partitionen stellen Themenbereiche dar, die sich gegenseitig ausschließen. Somit kann zum Beispiel eine Mustersprache bezüglich ihres Lösungsansatzes unterteilt werden. Genau so wie bei der Organisation von Musterbereichen werden die Musterpartitionen (Partition) vordefiniert und in den Metainformationen (Meta Information) der Muster angegeben, zu welcher Partition das Muster gehört.

Organisation von Musterabsichten (Pattern Organisation by Intent) Die Organisation von Musterabsichten ist eine mustersprachübergreifende hierarchische Einteilung der Muster in deren Absichten. Genauso wie bei der Organisation von Musterbereichen und der Organisation von Musterpartitionen kann die Organisation von Musterabsichten durch vordefinieren der Musterabsichten (Intent) und der Angabe der Musterabsichten in den Metainformationen (Meta Information) der Muster, aufgebaut werden.

Arten von Zusatzfunktionalitäten von Mustersprachen (Pattern Language Feature) Die Arten von Zusatzfunktionalitäten von Mustersprachen, die von der Plattform von Mustersprachen und Musterkatalogen unterstützt werden, basieren alle auf dem Metamodell für Mustersprachen. Die Zusatzfunktionalitäten von Mustersprachen wurden in 3.6.3.2 *Zusatzfunktionalitäten für Mustersprachen im Mustersprachprofil* beschrieben und werden im Folgenden für die Verwendung innerhalb der Plattform für Mustersprachen und Musterkataloge angepasst. Jede Mustersprache enthält die Zusatzfunktionalitäten. Sie können bei der Erstellung der Musterkataloge optional angelegt werden.

Zusammenfassung der Musterinhalte (Pattern Language Summary) Die Zusammenfassung der Musterinhalte stellt einen Text dar, der den Kern des Mustersprachenin haltes beschreibt. Für jede Mustersprache wird eine eigene Zusammenfassung der Musterinhalte erstellt.

Problem-/Lösungstabelle der Muster (Problem/Solution Summary) Die Problem-/Lösungstabelle der Muster ist eine Tabelle, die Kurzfassungen der Musterprobleme und -lösungen übersichtlich darstellt. Die Plattform für Mustersprachen und Musterkataloge ermöglicht es, die Tabellen aller Mustersprachen zusammenzufassen, damit die Autoren und Leser in der Zusammenfassung die Problem-/Lösungspaare suchen können.

Musterübergreifende Beispiele (Running Examples) In dieser Zusatzfunktionalität musterübergreifender Beispiele werden fortlaufende oder ähnliche Beispiele in Beschreibungen zusammengestellt, um den Mustersprachen weitere Zusammenhänge und Vernetzungen bezüglich der demonstrativen Umsetzung ihrer Muster zu verleihen.

Wörterverzeichnis der Musterinhalte (Glossary) In dem Wörterverzeichnis der Musterinhalte werden Fachbegriffe der Musterinhalte erklärt und auf die Muster verwiesen, in denen diese Fachbegriffe vorkommen.

Inhaltsverzeichnis für Mustersprachen (Index) Eine Mustersprache kann eine von Autoren angelegte Gliederung enthalten. Bei der Erstellung der Muster werden die Muster in diese Gliederung eingefügt.

4.3.2 Erstellung und Verwaltung von Musterkatalogen

Mustersprachen, die in der Plattform für Mustersprachen und Musterkataloge erstellten worden sind, stellen Typen von Musterkatalogen dar. Die Plattform ermöglicht es, dass Musterkataloge anhand bereits definierten Mustersprachen angelegt und verwaltet werden können. Die Musterkataloge sind Instanzen der Mustersprachen, die in der Plattform hinterlegt sind. Sie werden erzeugt, in dem sie von einer Mustersprache abgeleitet werden. Der neu erzeugte Musterkatalog der Mustersprache erhält deren Eigenschaften und einen Namen. Durch das Anlegen von Muster wird der Inhalt der Musterkataloge erstellt. Die Muster des Musterkataloges müssen nach der Musterstruktur der Mustersprache aufgebaut sein, aus der der Musterkatalog abgeleitet wurde. Zusätzlich dazu können die Zusatzfunktionalitäten und die Musterorgnisationen der Mustersprache angelegt und verwendet werden. Die Verwaltung der Musterkataloge beinhaltet unter anderem das Verwenden und das stetige Pflegen und Erweitern der Zusatzfunktionalitäten und Musterorganisationen. Muster können auch nach der Erstellung eines Musterkataloges eingepflegt werden. Ebenso können bestehende Musterinhalte geändert werden, in dem eine neue Version des Musters angelegt wird. Für diesen Zweck wird ein

Mechanismus für die Verwaltung der verschiedenen Musterversionen vorgestellt. Die Verwendung der Musterkataloge beinhaltet zusätzlich das Stöbern in den Musterkatalogen. Dies wird durch das Verfolgen der hinterlegten Referenzen auf Musterkataloge, Muster und deren Inhalten möglich. Außerdem wird ein Suchmechanismus vorgestellt, der es ermöglicht nach bestimmten Inhalten der Musterkataloge, Muster und deren Inhalten zu suchen. Damit den Autoren und Lesern der Muster eine Diskussionsplattform geboten werden kann, ermöglicht die Plattform für Mustersprachen und Musterkatalogen das Führen von Diskussionen über die Inhalte der Muster. Im Folgenden wird auf die Erstellung und Verwaltung von Mustern eingegangen. Anschließend werden die Möglichkeiten der Verwendung der Muster und deren Hilfsmittel, die die Plattform für Mustersprachen und Musterkataloge zur Verfügung stellt, spezifiziert.

Erstellung der Muster Um ein Muster für eine Mustersprache zu erstellen, wählt der Autor einen Namen für das Muster. Dieses identifiziert das Muster in dem Musterkatalog eindeutig. Optional kann ein Muster ein Symbol enthalten, das es graphisch repräsentiert. Es können noch weitere Angaben der Metainformation über das Muster gemacht werden. Dies sind beispielsweise Autoren der Musterinhalte, Suchwörter oder Informationen für die Musterorganisationen, in denen der Musterinhalt einzuordnen ist. Das Hauptaugenmerk bei der Erstellung von Mustern liegt auf der Erstellung und Formulierung des Musterinhaltes. Dazu müssen die Inhaltselemente der zu Grunde liegenden Musterstruktur instanziiert werden. Der Autor entscheidet, welche optionalen Inhaltselemente für die Beschreibung des Musterinhaltes benötigt werden. In den Inhaltselementen sollten Referenzen auf andere Mustersprachen, Muster oder Musterinhaltselemente eingefügt werden, falls diese für die Erklärung des Sachverhaltes beitragen.

Veränderung und Erweiterung der Musterinhalte Die Plattform für Mustersprachen und Musterkataloge ermöglicht ihren Autoren die Veränderung von Mustern. Die Musterinhalte eines Musters können verändert oder erweitert werden. Durch eine Veränderung oder Erweiterung der Musterinhalte wird eine neue Version des Musters erstellt. Die Autoren können dadurch auf alte Versionen der Muster zurückgreifen und die Entstehungs- und Verbesserungsprozesse der Musterinhalte einsehen. Außerdem wird es ermöglicht Muster zu löschen. Damit die Plattform für Mustersprachen und Musterkataloge dies ermöglichen kann, ist eine Erweiterung von den Metainformation von Mustersprachen, Muster und deren Inhaltselementen, die dem Metamodells für Mustersprachen zu Grunde liegen, erforderlich. Die Anpassung und die Erstellung der Metainformationsmodelle wird im Folgenden detailliert vorgestellt. Bei der Erstellung der Modelle für die Metainformationen der Mustersprachen, Muster und deren Inhaltselementen wird bereist auf Bestandteile, die für die Suche benötigte werden, vorweg gegriffen.

Erweiterung der Metainformationen von Mustersprachen, Mustern und deren Inhaltselementen Damit die Anforderungen der Spezifikation der Plattform für Mustersprachen und Musterkataloge erfüllt wird, werden zunächst Modelle der Metainformationsbeschreibung für Mustersprachen, Muster und Musterinhalte erstellt. Durch die Definition der Modelle für die Metainformationsbeschreibungen wird es möglich, dass Musterorganisationen aufgebaut, die Suche von Mustern erweitert und die verschiedenen Musterkatalogen, Mustern und Musterversionen verwaltet werden können.

Die Abbildung 4.1 zeigt die Metainformationsbeschreibung für Mustersprachen (Pattern Language Meta Information Descriptor). Sie ist eine Instanz der Metaklasse für die Metainformationsbeschreibung

des Metamodells für Mustersprachen und wird einheitlich in den Mustersprachen der Plattform für Mustersprachen und Musterkataloge verwendet. Die Metainformationsbeschreibung enthält ein Erstellungs- (Created Date) und Änderungsdatum (Changed Date), eine Liste von Autoren (Author) und eine Liste von Schlüsselwörtern (Keyword). Die Daten der Mustersprachen sollen den Lesern und Autoren chronologische Informationen bezüglich der Erstellung und Änderung der Musterkataloge liefern. In der Metainformationsbeschreibung der Mustersprachen werden alle Autoren festgehalten, die an der Erstellung der Musterkataloge beteiligt sind. Die Liste der Schlüsselwörter ist eine Zusammenstellung von Stichpunkten, die den Inhalt der Mustersprache stichpunktartig wiedergeben soll. Die Werte der Metainformationen von Mustersprachen werden aus den Metainformationen ihrer Muster abgeleitet.

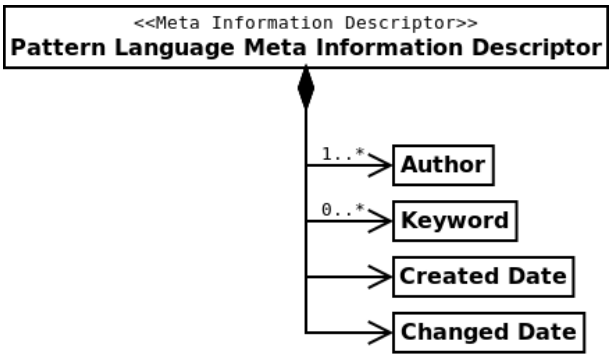


Abbildung 4.1: Metainformationsbeschreibung für Mustersprachen

Die Metainformationsbeschreibung für Muster (Pattern Meta Information Descriptor) ist in der Abbildung 4.2 dargestellt. Sie ist genauso wie die Metainformationsbeschreibung für Mustersprachen eine Instanz der Metaklasse für Metainformationsbeschreibung des Metamodells für Mustersprachen und wird einheitlich in den Muster der Plattform für Mustersprachen und Musterkataloge verwendet. Die Metainformationsbeschreibung besteht aus einer Liste von Autoren (Author), einer Liste von Schlüsselwörtern (Keyword), einer Liste von Musterbereichen (Domain), einer Liste von Musterpartitionen (Partition), einer Liste von Musterabsichten (Intent) und einer Versionsbeschreibung. Im Gegensatz zu der Metainformationsbeschreibung für Mustersprachen wird hier auf das Erstellungs- und Änderungsdatum verzichtet, da diese in der Versionsbeschreibung enthalten sind. Die Liste der Autoren enthält die Autoren des Musters und dessen Inhalte. Die Liste der Schlüsselwörter stellt eine stichpunktartige Inhaltsangabe des Musters dar. Die Liste der Musterbereiche beschreibt stichpunktartig die Zuordnung des Musterinhaltes in Themenbereiche. Die Liste von Musterabsichten stellt stichpunktartig die Absichten des Musterinhaltes dar. Die Versionsbeschreibung ist für die Verwaltung der unterschiedlichen Versionen der Muster verantwortlich. Im weiteren Verlauf wird genauer auf das Modell der Versionsbeschreibung eingegangen.

Für die Inhaltselemente der Muster wird das selbe Metainformationsbeschreibungsmodell verwendet, wie für die Muster. Die Mustermetainformationen von Mustern stellen die Menge der Mustermetainformationen ihrer Inhaltselemente dar.

In Abbildung 4.3 ist das Modell der Versionsbeschreibung (Version Descriptor) für Muster und deren Inhaltselemente zu sehen. Die Versionsbeschreibungen der Muster, die aus dem selben Anfangsmuster entstanden sind, werden in ein Gruppe (Version Collection) unterteilt. In den Gruppen werden alle verschieden Versionen eines Musters vermerkt. Eine Versionsbeschreibung enthält den Autor (Author),

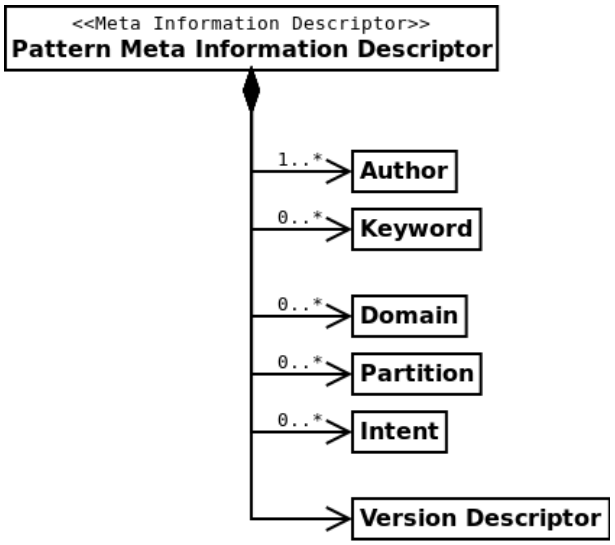


Abbildung 4.2: Modell der Metainformationsbeschreibung für Muster

der die neue Version des Musters angelegt hat. Es wird das Erstelldatum (Created Date) der ersten Musterversion und das Datum der aktuellen Version des Musters (Changed Date) vermerkt. Jede Version enthält eine Versionsnummer (Version Number), die bei der Erstellung einer neuen Musterversion hochgezählt wird. Wird eine ältere Musterversion verändert, dann wird die daraus erzeugte neue Musterversion die aktuellste und erhält die höchste Versionsnummer. Muster können nicht gelöscht werden, weil auf diese verwiesen wird. Aus diesem Grund kann ein Muster als gelöscht markiert werden. Es ist ein Element vorgesehen, das diese Information enthält (Removal). Damit die Leser und Autoren wissen, wieso das Muster oder ein Musterinhalt entfernt wurde, kann dafür eine Begründung (Reason) hinterlegt werden. Ein Muster kann aus mehreren Gründen entfernt werden. Das Problem eines Musters kann durch die Erweiterung des Musterkataloges gelöst werden. Ebenso kann ein neues Muster eine bessere alternative bieten, die ein altes Muster in jeder Hinsicht überflüssig macht. Die Entwicklung eines neuen Paradigmas, einer neuen Programmiersprache, eines neuen Programmierstiles oder einer Verlagerung der Art der Systeme, die entwickelt werden, kann dazu führen, dass Muster entfernt werden. [BMR+98][S. 376-377]

Suchen nach Mustersprachen, Mustern und Musterinhalten Die Plattform für Mustersprachen und Musterkatalogen sieht einen Suchautomatismus vor, damit die Leser und Autoren nach Inhalten der Musterkatalogen, Muster oder Musterinhalten suchen können. Das Ergebnis der Suche kann spezielle Musterlösungen oder bestimmte Vernetzungen von Mustern darstellen.

Die Plattform bietet für die Suche mehrere Filter an. Ein Filter stellt eine Menge von Strukturelementen aus einer Eingabemenge von Strukturelementen zusammen, die dem Kriterium des Filters genügen. Die Eingabemengen und die Kriterien der Filter können den Filtern übergeben werden. Damit möglichst detaillierte Selektionen erstellt werden können, können die Filter in einer Suchanfrage kombiniert angewendet werden.

Jede Musterorganisation und jede Zusatzfunktionalität einer Mustersprache muss einen Filter für die Suche bereitstellen, damit deren Strukturen nach Mustern durchsucht werden können. Zusätzlich

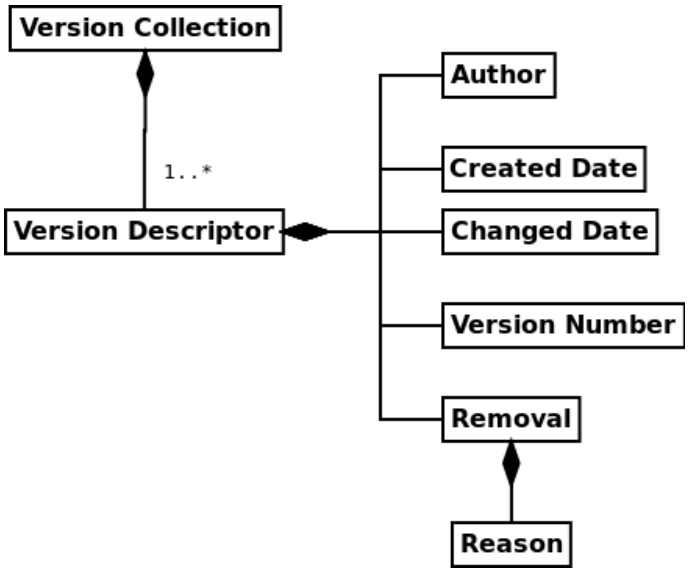


Abbildung 4.3: Modell der Versionsbeschreibung

können die Inhaltsarten Filter für die Suche von Inhaltsbestandteile bereitstellen. Darüberhinaus bietet die Plattform für Mustersprachen und Musterkataloge Filter an, die nach folgenden Kriterien die Musterkataloge, Muster und deren Inhaltselemente selektieren können: Erstellungs-, Änderungsdaten, Autoren oder Schlüsselwörter.

Im Folgenden wird die Form der Suchanfragen definiert, die Anordnung der Filter bezüglich der Suchanfragen erklärt, die Abarbeitung der Filter vorgestellt und das Verfahren an einem Beispiel erklärt.

Um ein Suchergebnis der Suchfunktion zu erhalten, muss eine Suchanfrage erstellt und deren Filterprozess aufgebaut werden, bevor die Suchanfrage durchlaufen werden kann. Da sich die Inhalte der Musterkataloge ständig ändern und die Suchanfragen unterschiedlich und neu formuliert werden können, wird die Struktur der Suchanfrage und die Menge der benötigten Daten für jede Suche erneut aufgebaut. Zunächst wird das Modell der Suchanfrage vorgestellt, damit der Filterabarbeitungsprozess erstellt werden kann.

Grammatik der Suchanfragensprache:

G = (V, Σ, P, QP)

V = { QP }

Σ = { *filter*, $\mathbb{SE}\setminus$, $\cap$, $\cup$, *(*, *)* }

P = {

1. QP $\rightarrow$ $\mathbb{SE}\setminus$QP
2. QP $\rightarrow$ QP $\cup$ QP
3. QP $\rightarrow$ QP $\cap$ QP
4. QP $\rightarrow$ *(* QP *)*
5. QP $\rightarrow$ *filter*

}

Alle gültigen Suchanfragen für den Suchdienst der Anwendungslogik sind Elemente der kontextfreien Sprache L(G). Die Suchanfragensprache L(G) wird durch die angegebenen Grammatik G gebildet. In der Definition der Grammatik werden Filter und Mengenoperatoren verwendet, um Suchanfragen zu formulieren. Das Terminalsymbol $\mathbb{SE}\backslash$ stellt das Komplement einer Menge von allen Strukturelementen dar. Die anderen Terminalsymbole werden für beliebige Filter, Klammern und die Mengenoperatoren Vereinigung und Schnittbildung eingesetzt. Ein Filter enthält eine Funktion, die durch Hinzunahme eines Filterkriteriums eine Menge von Strukturelemente filtert und eine gefilterte Menge von Strukturelementen zurück liefert. Die Filter repräsentieren in der Suchanfrage Mengen von Strukturelementen. Es ist zum Beispiel die Suchanfrage $filter_1 \cap (filter_2 \cup filter_3)$ in der Suchanfragensprache L(G) enthalten. Der $filter_1$ könnte beispielsweise alle Muster auswählen, die ein bestimmter Autor erstellt hat. Damit die Filter eine Menge liefern, werden diese auf die Menge $\mathbb{SE}$, aller zu durchsuchenden Strukturelemente, angewendet. Die Beispielsuchanfrage kann somit in die folgende Gleichung umgeformt werden: $\mathbb{SE}_1 \cap (\mathbb{SE}_2 \cup \mathbb{SE}_3)$. Die Menge $\mathbb{SE}_i$ wird durch die Selektion der jeweiligen Filter $filter_i$ gebildet. Durch die Berechnung der erstellten Gleichung erhält man das Suchergebnis. Im folgenden Verlauf wird der Aufbau der Filter, der Suchanfragenstruktur und deren Traversierung erklärt. Anhand dessen wird die Einsparung von Rechenoperationen vorgestellt, die durch die Verwendung von vorselektierten Eingabemengen der Filter gewonnen wird. [Sch01, S. 11-26, 51-79], [Goo97, S. 77-78, 94-140]

In Abbildung 4.4 ist das Modell eines Filters in Form einer UML-Klasse abgebildet. Das Modell ist durch eine abstrakte Klasse (Filter) dargestellt, die eine Funktion (filterStructureElements) enthält, um eine Mengen von Strukturelementen (Set<StructureElements>) anhand vordefinierter Filterkriterien (Criterion) zu filtern. Eine Suchanfrage ist ein Element der Suchanfragensprache, es wird durch eine Suchanfragestruktur dargestellt. Die Suchanfragestruktur wird der Suche als Eingabe über geben. Anhand der Suchanfragestruktur kann die Suche das Suchergebnis berechnen. Die Suchanfragestruktur stellt eine Objektstruktur dar, das die Suchanfrage durch ein Datenmodell repräsentiert. Die Suchanfragestruktur verknüpft Filter durch Mengenoperatoren. Die Struktur wird durch einen Baum dargestellt, dessen Konten die Mengenoperatoren Vereinigung und Schnittbildung enthalten. Zusätzlich wird noch die Komplementbildung von der Menge aller Strukturelementen verwendet. Die Komplementbildung stellt eine unäre Mengenoperation dar.

In der Abbildung 4.4 sind die Klassen der Suchanfragestruktur zu sehen. Die Suchanfragestruktur ist ein binärer Baum, bis auf die Ausnahme des Komplementbildungsknotens (SetOperatorComplementation). Der Baum besteht aus Knoten (TreeNode), die ein oder zwei weitere Konten enthalten können. Die Filter (Filter) werden als Blätter in dem Baum verwendet. Sie enthalten keine weiteren Konten. Die Knoten werden durch die drei Mengenoperatoren Vereinigung (SetOperatorUnion), Schnittbildung (SetOperatorIntersection) und Komplementbildung (SetOperatorComplementation) darstellt. Diese Suchanfragestruktur repräsentiert außerdem den Syntaxbaum der Suchanfrage.

Die logische Abarbeitung der Baumstruktur der Suchanfrage beginnt an den Blättern bzw. Filtern. Die Filter selektieren die Ausgangsmenge von Sturkturelementen und reichen sie weiter an deren Vorgängerknoten. Die Knoten empfangen die gefilterten Mengen und führen ihren Mengenoperatoren auf diese aus. Sie geben die daraus resultierende Menge ebenfalls weiter an deren Vorgängerknoten. Die Wurzel der Suchanfragestruktur enthält schließlich das Ergebnis der Suche.

Im Folgenden wird der Ablauf der Traversierung der Suchanfragestruktur anhand eines Algorithmus erläutert. Der Algorithmus ist in Pseudocode festgehalten und in Abbildung 4.1 zu sehen.

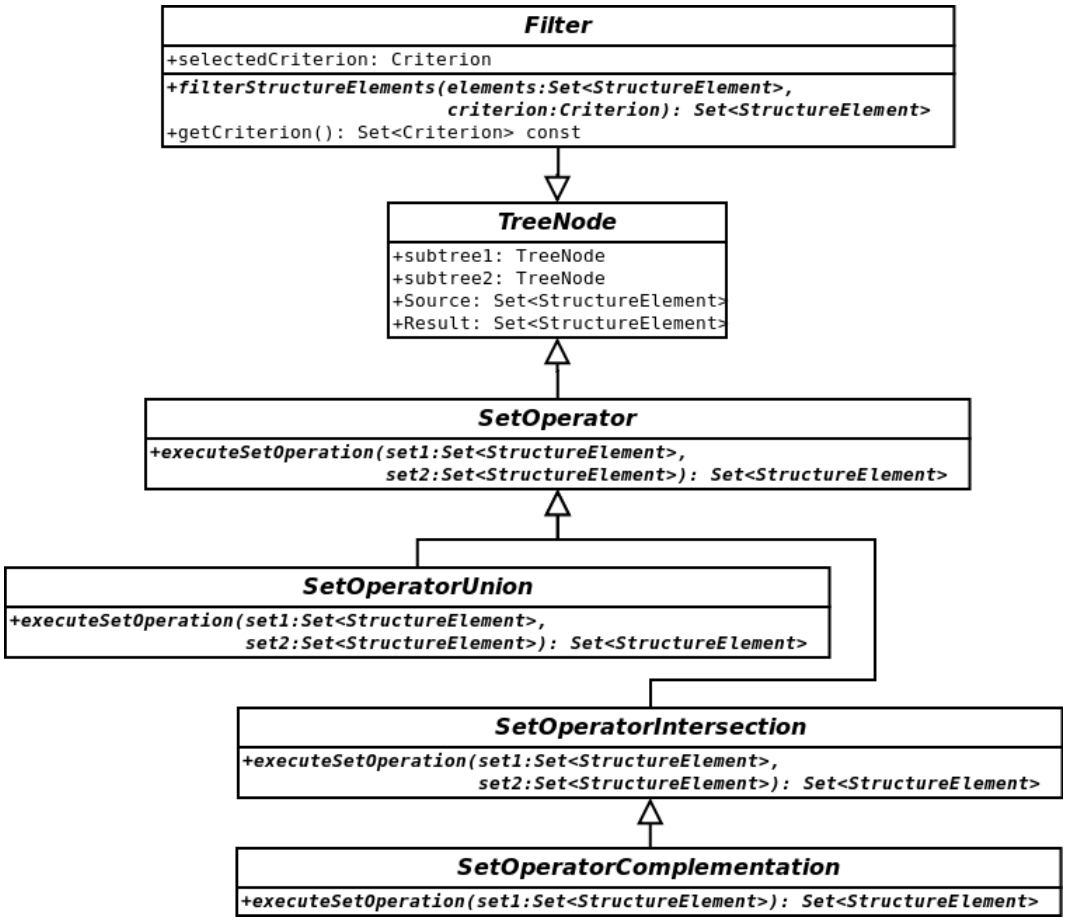


Abbildung 4.4: Modell der Suchanfragestruktur des Suchdienstes der Anwendungslogik

Es sind zwei Funktionen für die Traversierung der Suchanfragestruktur enthalten, die Funktion *processSearchRequest* und die Funktion *processNode*. Die Funktion *processSearchRequest* startet initial die Abarbeitung der Suchanfrage und legt die Ausgangsmenge der zu durchsuchenden Strukturelementen fest. Die Funktion *processNode* definiert die Traversierung der Suchanfragestruktur rekursiv. Sie berechnet jeweils das Ergebnis des Knoten, der als Parameter übergeben wurde. Bevor der Parameterknoten berechnet werden kann, werden durch rekursive auf Rufe der selben Funktion mit den Kinderknoten die Unterbäume berechnet. In der Traversierung der Baumstruktur werden zunächst die benötigten Ausgangsmengen an die Mengenoperatoren/Knoten und Filtern/Blätter verteilt. Um Rechenschritte einzusparen, werden die Schnittoperationen nicht durchgeführt. Stattdessen werden Teilergebnisse als Ausgangsmengen verwendet. Dies ist möglich, weil die Hintereinanderausführung der Filter die Mengenoperation Schnittbildung ersetzen kann. Es muss trotzdem die Reihenfolge der Operatoren eingehalten werden. Die Komplementbildungsoperatoren und die Vereinigungsoperatoren müssen vor den Schnittbildungsoperatoren ausgeführt werden, damit statt der Durchführung von Schnittbildungsoperatoren die Teilergebnisse der Komplementbildungsoperatoren und der Vereinigungsoperatoren verwendet werden können. Die zu filternden Mengen werden verkleinert, durch die Ersetzung der Ausgangsmengen durch Zwischenergebnisse. Dadurch werden zusätzliche Rechenoperationen gespart.

```
1
2  // computes the result of the search request
3  //
4  procedure processSearchRequest
5        ( TreeNode rootNode,
6          Set<StructureElement> wholeSet )
7        return Set<StructureElement>
8
9  begin
10    processNode( rootNode );
11    return rootNode.result;
12 end processSearchRequest
13
14
15 // Traverses the tree and computes the intermediate result of each node.
16 // The traversation is a deepth frist search. The intermediate results
17 // will be computed by its traversation order.
18 //
19 procedure processNode
20       ( TreeNode node )
21       return Set<StructureElement>
22
23 begin
24    if node isTypeOf SetOperatorComplementation then
25
26       SetOperatorComplementation compOp = (SetOperatorComplementation) node;
27
28       compOp.subtree1.source = compOp.source;
29       compOp.subtree1.result = processNode( compOp.subtree1 );
30
31       compOp.result = compOp.executeOperation( compOp.subtree1.result );
32       return compOp.result;
33
34    else if node isTypeOf SetOperatorIntersection then
35
36       if node.subtree1 isTypeOf SetOperatorUnion then
37
38          node.subtree1.source = node.source;
39          node.subtree1.result = processNode( node.subtree1 );
40
41          node.subtree2.source = node.subtree1.result;
42          node.subtree2.result = processNode( node.subtree2 );
43
44          //the result of subtree2 is the result of the node
45          node.result = node.subtree2.result;
46
47       else //if node.subtree1 isTypeOf SetOperatorIntersection then
48
49          node.subtree2.source = node.source;
50          node.subtree2.result = processNode( node.subtree2 );
51
52          node.subtree1.source = node.subtree2.result;
53          node.subtree1.result = processNode( node.subtree1 );
54
55          //the result of subtree1 is the result of the node
```

```
56            node.result = node.subtree1.result;
57
58        end if
59
60        return node.result;
61
62    else if node.subtree1 instanceof SetOperatorUnion then
63
64        SetOperatorUnion unionOp = (SetOperatorUnion) node;
65
66        unionOp.subtree1.source = unionOp.source;
67        unionOp.subtree1.result = processNode( unionOp.subtree1 );
68
69        unionOp.subtree2.source = unionOp.source;
70        unionOp.subtree2.result = processNode( unionOp.subtree2 );
71
72        unionOp.result = unionOp.executeOperation(
73                                                unionOp.subtree1,
74                                                unionOp.subtree2 );
75
76        return compOp.result;
77
78    else //if node instanceof Filter then
79
80        Filter filter = (Filter) node;
81        node.result = filter.filterStructureElements(
82                                                    filter.source,
83                                                    filter.selectedCriterion );
84        return node.result;
85
86    end if
87
88
89 end processNode
```

Listing 4.1: Algorithmus für die Traversierung der Suchanfragenstruktur

Beispiel für die Traversierung einer Suchanfragestruktur Der Algorithmus für die Traversierung der Suachanfragenstruktur wird anhand eines Beispiels erläutert. Zunächst wird eine Suchanfrage formuliert und deren Strukturbaum aufgestellt. Anschließend wird die Traversierung anhand des Pseudocodes erklärt.

Dabei sollen Muster gefunden werden, in denen nicht das Wort Pattern und nicht das Wort Language vorkommt. Die Filter $filter_a$ und $filter_b$ finden jeweils Muster mit den beiden Worten. Zusätzlich sollen diese Muster gefunden werden, die von den Autoren Meyer und Müller oder von Fischer erstellt worden sind. Die Filter $filter_c$, $filter_d$ und $filter_e$ finden jeweils die Muster mit den entsprechenden Autoren. Die Suchanfrage wird durch folgenden logische Formel dargestellt: $(\mathbb{SE} \backslash filter_a \cap \mathbb{SE} \backslash filter_b) \cap ((filter_c \cap filter_d) \cup filter_e)$.

In Abbildung 4.5 ist die Suchanfragestruktur der Beispielsuchanfrage zu sehen. Die Suchanfragestruktur in der Abbildung stellt einen Syntaxbaum der logischen Formel dar. Die Knoten innerhalb des Baumes stellen die logischen Operatoren der Formel dar. Die Blätter werden durch die einzelnen Filter repräsentiert. Die Bezeichnung ”Comp.” in den Knoten soll die Komposition der

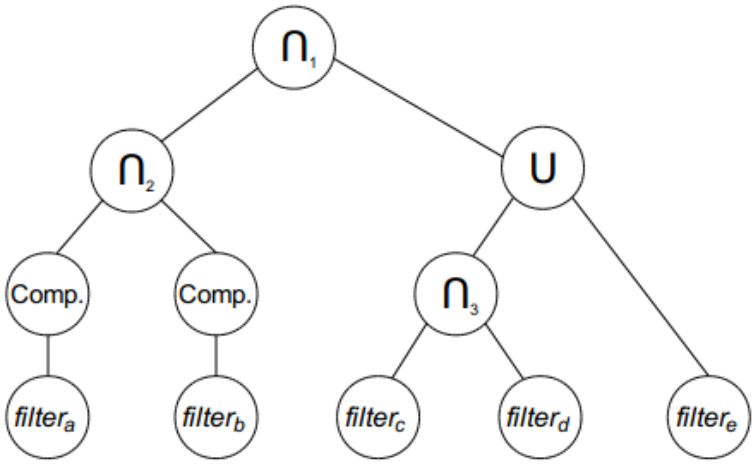


Abbildung 4.5: Beispiel einer Suchanfragestruktur

darunter hängenden Blätter darstellen. In den übrigen Knoten sind die Mengenoperatoren Vereinigung und Schnittbildung mit den gleichen Symbolen wie in der Formel abgebildet.

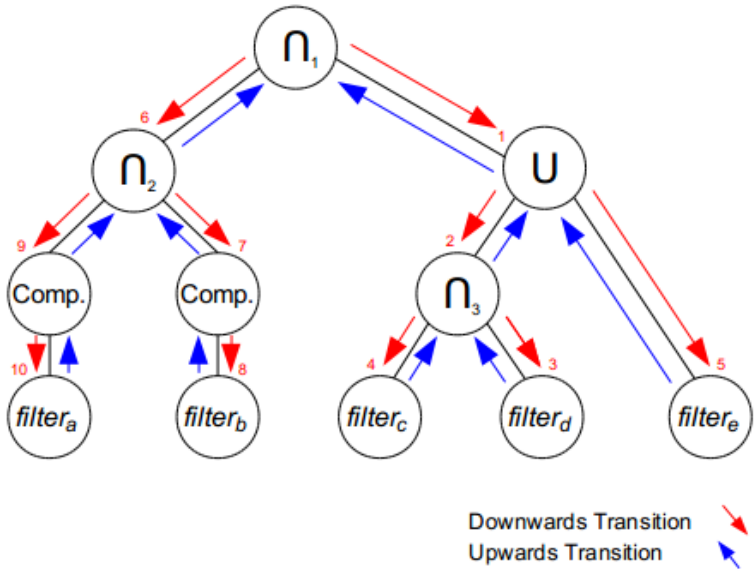


Abbildung 4.6: Beispielsuchanfragestruktur mit Traversierungsschritten

Die Abbildung 4.6 stellt den Syntaxbaum der Beispielabfrage erneut dar. Er enthält Pfeile, die die einzelnen Traversierungsschritte darstellen. Die Reihenfolge der Traversierungschritte entsprechen denen aus dem Algorithmus des Pseudocodes. Die roten Pfeile stellen die Abwärtstraversierung in der Baumstruktur dar. Sie enthalten eine Nummer, die Aufschluss über die Reihenfolge der einzelnen Traversierungsschritte gibt. Die blauen Pfeile stellen den Rückweg des Traversierungsschrittes dar. In einem Abwärtsschritt wird eine Menge von Strukturelementen von oben nach unten gereicht, die als Ausgangsmenge für die Ergebnisberechnung der einzelnen Teilbäume verwendet wird. In den Rückwärtsschritten wird das Ergebnis der gefilterten und verknüpften Mengen nach oben

propagiert. Diese Teilergebnisse sind wiederum Ausgangsmengen für die weitere Berechnung anderer Geschwisterknoten oder des darüber liegenden Vaterknoten. Der Algorithmus beginnt mit dem Aufruf der *processSearchRequest* Funktion an dem hierarchisch als höchsten angeordneten Operator. In der Abbildung ist dies der oberste Vereinigungsoperator, der den Index 1 trägt. Dieser Knoten stellt die Wurzel des Syntaxbaumes dar. Er enthält zu Beginn die Menge der zu durchsuchenden Strukturelemente. Am Schluss der Berechnung enthält der Knoten schließlich das Ergebnis der Suchanfrage. Bei der Abarbeitung der Unterbäume wird immer der Knoten als erstes berechnet, der keinen Vereinigungsoperator als Wurzel enthält, damit den Vereinigungsoperatoren vorselektierte Mengen übergeben werden können. Somit werden viele Vergleiche in den Berechnungen der Filter gespart und die Berechnungen der Vereinigungsoperatoren entfallen, durch das Weiterreichen bereits vorselektierter Mengen.

Diskussionen über Musterinhalte Die Plattform für Mustersprachen und Musterkataloge bieten den Autoren und Lesern von Musterkatalogen eine Plattform bezüglich der fachlichen Lösungen, die die Muster der Musterkataloge enthalten. Zusätzlich sollen Diskussionen in diesen Fachbereichen ermöglicht werden. Die Plattform für Mustersprachen und Musterkataloge unterstützt das Anfügen von Kommentaren und Bemerkungen an Muster, damit deren Inhalte diskutiert werden können. Die Inhaltselemente der Muster werden um ein Diskussionselement und ein Beitragselement erweitert. Die Beitragselemente stellen einen Bestandteil der Diskussion dar und sind dem Diskussionselement untergeordnet. Die beiden Elemente stellen eine zusätzliche Erweiterung des Metamodellprofils für Mustersprachen dar. Sie sind in der Abbildung im Anhang C zu sehen. Durch diese Erweiterung können Autoren und Leser fachliche Beiträge zu den Muster liefern und beispielsweise Erfahrungen, Erweiterungsvorschläge oder Fehler melden, damit das betroffene Muster verbessert und überarbeitet werden kann.

4.4 Struktur der Plattform für Mustersprachen und Musterkataloge

In diesem Abschnitt wird die gewünschte Struktur und der Entwurf der Plattform für Mustersprachen und Musterkataloge vorgestellt. Damit kann die zuvor beschrieben Spezifizierungen umgesetzt werden. [LL10, S. 399-400] Der Schwerpunkt dieses Abschnittes liegt auf der Erklärung der Struktur und der Architektur der Plattform und speziell auf der Funktionsweise der Suche nach Musterkatalogen, Mustern und deren Inhaltselemente. Im Folgenden wird die Architektur der Plattform für Mustersprachen und Musterkataloge beschrieben.

Die Plattform für Mustersprachen und Musterkataloge ist nach dem Prinzip der Drei-Schichten-Architektur organisiert. Sie ist in eine Präsentationsschicht, in eine Anwendungsschicht und in eine Datenhaltungsschicht unterteilt, um die Anforderungen der Plattform für Mustersprachen und Musterkataloge in Aufgabenbereiche zu unterteilen. Die Aufgabenbereiche stellen in sich abgeschlossen Bereiche dar. Die Unterteilung strukturiert die Software und reduziert deren Grad an Komplexität. Sie ist eine Grundlage für dynamische, skalierbare und interaktive Softwarelösungen. Solch eine Softwarelösung kann einfach abgeändert werden, um neuen Bedürfnissen gerecht zu werden. Ein weiterer Grund der Unterteilung der Software in unterschiedliche Schichten, ist der Ort, an dem die Software ausgeführt wird. Damit dies möglich ist, müssen die Schichten Schnittstellen anbieten und über Protokolle kommunizieren. Die Präsentationsschicht wird auf dem Computer des

Benutzers ausgeführt und realisiert die Benutzeroberfläche, damit der Benutzer die Software bedienen kann. Die anderen beiden Schichten werden serverseitig ausgeführt. In der Anwendungsschicht sind fachliche Komponenten und die Anwendungslogik der Software enthalten. Die Anwendungsschicht ist unabhängig von der Präsentationsschicht und stellt diese Dienste zur Verfügung. Die fachlichen Komponenten und die Anwendungslogik können durch die Verwendung der Dienste beansprucht werden. Die Anwendungsschicht greift wiederum auf Dienste der Datenhaltungsschicht zu, damit diese Daten manipuliert oder gelesen werden können. Diese Datenhaltungsschicht sorgt dafür, dass die manipulierten Daten dauerhaft gespeichert werden. Aufgrund der Aufteilung können die Softwarekomponenten, die durch die Anwendungsschicht und die Datenhaltungsschicht verkörpert werden, repliziert werden. Dies erhöht deren Ausfallssicherheit und Verfügbarkeit. Außerdem kann eine größere Bandbreite von Benutzern bedient werden, die die Softwarekomponenten der beiden Schichten als Dienste in Anspruch nehmen. [Tho11], [BCHP03, S. 11-13], [Fow02, S. 17-24], [LL10, S. 431-432]

Zwischen den logischen und physikalischen Schichten der Plattform für Mustersprachen und Musterkataloge wird kein Unterschied gemacht. Durch die Planung der Plattform wurde dafür gesorgt, dass die Aufgabenbereiche der logischen und physikalischen Schichten deckungsgleich sind. Im Allgemeinen wird in dieser Arbeit von den logischen Schichten der Plattform gesprochen. [Fre09]

In Abbildung 4.7 ist die Unterteilung der Plattform für Mustersprachen und Musterkataloge zu sehen. Die Plattform ist in das vorgestellte Drei-Schichten-System unterteilt. Eine webbasierte Benutzeroberfläche (Web User Interface) stellt die Präsentationsschicht der Plattform dar. Eine Anwendungslogik (Domain Logic) bietet Dienste an, die die spezifizierten Aufgaben der Plattform für Mustersprachen und Musterkataloge bewältigt. Die Anwendungslogik repräsentiert die Anwendungsschicht in dieser Drei-Schichten-Architektur. Die Datenhaltungsschicht wird durch ein Repository umgesetzt. Das Repository (Repository) verwaltet die Daten der Mustersprachen und der Musterkataloge. Die drei Bestandteile der Plattform sind bezüglich ihrer Abhängigkeiten zu einander dargestellt. Der Applikationsserver ist abhängig von den Daten, die das Repository bereitstellt und die webbasierte Benutzeroberfläche ist wiederum abhängig von den Diensten, die der Applikationsserver bereitstellt.

Die Bestandteile der Plattform für Mustsersprachen und Musterkataloge, deren Schnittstellen und deren Kommunikationen werden in den folgenden Abschnitten detaillierter vorgestellt.

4.4.1 Repository der Plattform für Mustersprachen und Musterkataloge

Das Repository der Plattform für Mustersprachen und Musterkataloge verkörpert die Datenhaltungsschicht und enthält eine Datenzuordnung für eine Datenstruktur, die für die Verwendung von darüber liegenden Schichten konzipiert ist. Software, die sich in der darüber liegenden Schicht befindet, kann Objekte in das Repository speichern, diese auf dem Repository lesen und löschen. In größeren Systemen mit mehren Softwarekomponenten, welche auf das Repository zugreifen und die vordefinierten Objektstrukturen und vorgefertigten Datenabfragen verwenden, wird viel Quellcode gespart. Außerdem liefert die Schnittstelle des Repositorys reine objektorientierte Datensätze und gewährt eine konsistente Sicht der Datenquellen, aus denen das Repository die Datensätze zusammenstellt. Das Repository verhält sich wie eine Objektsammlung, die sich in dem Speicher befindet. Durch das Ansprechen dieser Schnittstellenoperationen, werden die entsprechenden Funktionalitäten des Repositorys im Inneren aufgerufen. Sie sind dafür verantwortlich, dass die Daten

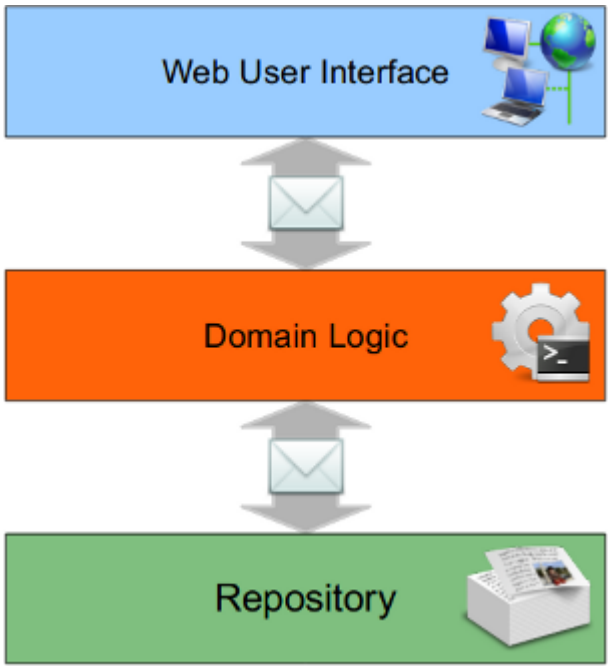


Abbildung 4.7: Architekturübersicht der Plattform für Mustersprachen und Musterkataloge

in der richtigen Datenstruktur und dem richtigen Format vorliegen, gespeichert, gelöscht oder gelesen werden können. [Fow02, S. 133-142, 322-327]

In Abbildung 4.8 ist die Architektur des Repositorys zu sehen. Sie enthält eine Web-Service-Schnittstelle (Web Service Interface), damit Softwarekomponenten aus höheren Sichten auf die Daten des Repositorys zugreifen können. Damit die gewünschten Daten geliefert werden können, müssen diese zuerst zusammengestellt und in eine Objektstruktur transformiert werden. Diese Aufgabe übernimmt die Komponente Datenmapper (Data Mapper), die unter der Web-Service-Schnittstelle zu sehen ist. Diese Komponente greift direkt auf das Datenbankmanagementsystem (Database Management System) und auf das Anlageverzeichnis (Attachment Storage - File System) zu, welches sich auf dem Dateisystem, das sich auf der gleichen Maschine wie das Repositorys befindet, zu. Im Folgenden wird auf die einzelnen Komponenten des Repositorys genauer eingegangen.

Web-Service-Schnittstelle des Repositorys Die Web Service Description Language (WSDL) [CCMS01] wurde als Industriestandard entwickelt, um Schnittstellen für Dienste zu beschreiben. Die Dienstbeschreibung definiert Metadaten, die vollständig die Charakteristik der Dienste beschreibt, die in Netzen eingesetzt werden. Diese Metadaten sind die Grundlage, um lose Kopplung zwischen den Dienstverwendern und den Dientanbietern herzustellen. Mit WSDL werden Informationen zur Verfügung gestellt, die benötigt werden, um Dienste verfügbar zu machen und um diese Diensten ansprechen zu können. [WCL+05, S. 40]

Das Repository der Plattform für Mustersprachen und Musterkataloge bietet eine Web-Service-Schnittstelle an, damit eine Softwarekomponente auf dessen Daten zugreifen und diese verwenden kann. Diese Schnittstelle stellt die Funktionalitäten des Repositorys als Dienste nach

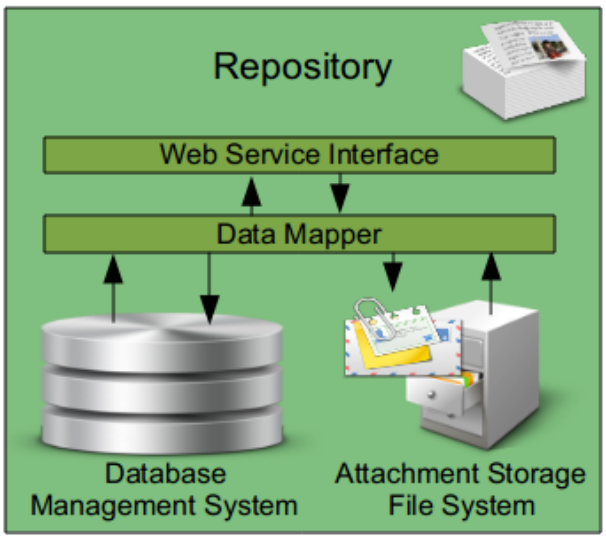


Abbildung 4.8: Arichitektur des Repository für Plattform für Mustersprachen und Musterkataloge

außen zur Verfügung. Das vorgestellte Prinzip der Schnittstelle basiert auf dem *Gateway* Muster. [Fow02, S. 133-142, 466-472]

Datenmappter des Repositorys Ein Datenmappter ist eine Softwarekomponte, die die Brücke zwischen der Anwendungslogik und den Daten darstellt. Viele Bestandteile der Objektorientierung, wie beispielsweise Kollektionen und Vererbung, sind nicht in relationalen Datenbanken hinterlegt. Wenn ein Objektmodell erstellt wird, das viel Anwendungslogik besitzt, ist es sinnvoll objektorientierte Konzepte und Mechanismen zu verwenden, damit die Daten der Anwendungslogik besser organisiert werden können und damit deren Verhalten besser ausgedrückt und verwendet werden kann. Um dies zu bewältigen muss das relationale Datenbankschema in eine Objektstruktur transformiert werden. Die Aufgabe des Datenmappers ist es, die Strukturen der Daten zu trennen und zwischen diesen Transformationen durchzuführen. Er isoliert somit das relationale Datenbankschema von der Anwendungslogik und stellt eine Objektstruktur zur Verfügung, die auf einer höheren Abstraktionsebene angesiedelt ist. [Fow02, S. 165-179]

Der Datenmapper des Repositorys für Mustersprachen und Musterkataloge übernimmt eine weitere Aufgabe im Gegensatz zu der allgemein vorgestellten Beschreibung des Datenmappers. Da das Repository der Plattform für Mustersprachen und Musterkataloge zwei Datenquellen enthält, muss er die Daten beider Quellen zur Verfügung stellen und transformieren. Je nach Operation kann dies bedeuten, dass Selektionen von den Daten gebildet werden müssen. Diese werden wiederum in Objektstrukturen transformiert und anschließend an die Web-Service-Schnittstelle weitergeleitet. Der Datenmapper muss aber auch Daten der Web-Service-Schnittstelle entgegennehmen, diese transformieren und sie auf die beiden Datenquellen aufteilen. Er bezieht und speichert Daten aus einem Datenbankmanagementsystem und einem Anlageverzeichnis. Alle Daten, die nicht in Form einer Datei vorliegen, werden in die Datenbank des Datenbankmanagementsystem geschrieben. Die Dateien werden in das Anlageverzeichnis geschrieben und deren Metainformationen in der Datenbank

festgehalten. In den folgenden Abschnitten wird dabei näher auf das Datenbankmanagementsystem und das Anlageverzeichnis eingegangen.

Datenbankmanagementsystem des Repositorys Das Datenbankmanagementsystem des Repositorys ist ein relationales Datenbankmanagementsystem, das eine Datenbank enthält. In dieser Datenbank werden die Daten und Informationen der Mustersprachen und Musterkataloge hinterlegt. Um einen konsistenten und sicheren Datenaustausch zu gewährleisten, werden die Daten über Transaktionen ausgetauscht. Damit alle Daten und Informationen der Plattform für Mustersprachen und Musterkataloge gespeichert werden können, wird das angepasste Metamodell für Mustersprachen als Grundlage für ein relationales Datenbankschema verwendet. [SE07, S. 1-3]

Anlageverzeichnis des Repository Das Anlageverzeichnis des Repositorys der Plattform für Mustersprachen und Musterkataloge ist für die Speicherungen von Dateien, die Anhänge der Muster darstellen, verantwortlich. Es stellt eine einfache Ablage auf dem Dateisystem dar. Das Anlagenverzeichnis läuft auf dem selben Computersystem wie auch das Repository. Die Musteranhänge, die in Dateien des Anlageverzeichnis liegen, können von dem Datenmapper gelesen und gespeichert werden.

4.4.2 Anwendungslogik der Plattform für Mustersprachen und Musterkataloge

Die Anwendungslogik besteht aus Softwarekomponenten, die die Daten der Datenhaltungsschicht manipulieren, transformieren und in Daten für die Präsentationsschicht konvertieren. Diese Softwarekomponenten stellen den Kern der Anwendungsschicht dar und enthalten die Logik der Anwendungsprozesse. Die Funktionalitäten der Anwendungslogik werden als Dienste bereitgestellt und können von der Präsentationsschicht verwendet werden. [McL02, S. 21-22], [Tar09, S. 116]

In Abbildung 4.9 ist die Anwendungslogik (Domain Logic) der Plattform für Mustersprachen und Musterkataloge zu sehen. Sie enthält eine Schnittstelle (Service Interface) und eine Sammlung von Funktionalitäten (Service Pool). Die Funktionalitäten können über die Schnittstellen angesprochen werden. Die Sammlung von Funktionalitäten enthält zwei Funktionalitäten, einen Dienst, um Daten der Plattform für Mustersprachen und Musterkataloge zu manipulieren (Data Manipulation Service) und einen Dienst, um nach Musterkatalogen, Mustern oder deren Inhaltselementen zu suchen (Data Search Service). Im weiteren Verlauf werden die Bestandteile der Anwendungsschicht näher erläutert.

Schnittstelle der Anwendungslogik Die Schnittstelle der Anwendungslogik definiert das Kommunikationsprotokoll und die Operationen, damit die Funktionen der Anwendungsschicht als Dienste verwendet werden können. Speziell die Präsentationsschicht soll auf die Funktionalitäten der Anwendungsschicht zugreifen können.

Anwendungsdienste Die Dienste der Anwendungsschicht der Plattform für Mustersprachen und Musterkataloge sind in der Sammlung von Funktionalitäten enthalten. Sie werden von der Präsentationsschicht in Anspruch genommen und bereiten die Daten des Reposiotrys auf oder können diese durchsuchen und anschließend das Suchergebnis bekannt geben. Die Dienste können gegenseitige

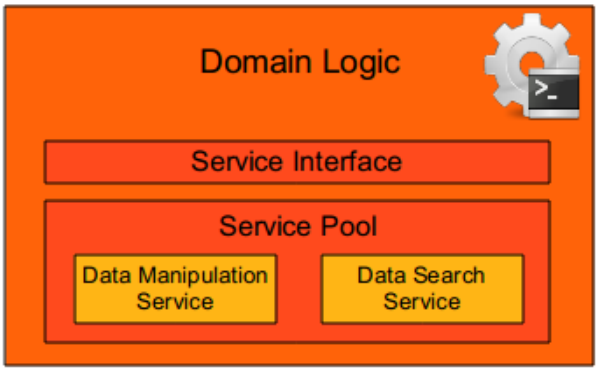


Abbildung 4.9: Architektur der Anwendungslogik der Plattform für Mustersprachen und Musterkataloge

Abhängigkeiten darstellen. Beispielsweise kann der Suchdienst die Datenmanipulationsdienst verwenden, um Daten zu laden. In den folgenden beiden Abschnitten wird auf diese zwei Dienste der Anwendungsschicht eingegangen. Der Schwerpunkt liegt dabei auf dem Entwurf des Suchmechanismuses.

Datenmanipulationsdienst Der Datenmanipulationsdienst ermöglicht es, über den Aufruf von Operation, Daten der Plattform zu lesen und zu schreiben. Je nach Anwendungszweck können von der Präsentationsschicht Daten angefordert werden. Der Datenmanipulationsdienst lädt die Daten aus dem Repository und gibt sie weiter an die Präsentationsschicht. Die Daten können nach Bedarf zusammengestellt und aufbereitet werden, bevor sie an die Präsentstionsschicht übergeben werden. Außerdem bietet der Datenmanipulationsdienst die Speicherung und Änderung von Daten an. In der Anwendungslogik können zum Beispiel eine Korrektheits- und Konsistenzprüfung oder Konvertierungen der geänderten oder zu speichernden Daten durchgeführt werden. Die geänderten oder zu speichernden Daten werden anschließend an der Repository weiter gereicht. Der Datenmanipulationsdienst stellt einen weiteren Datenmapper dar, der sich zwischen der universalen Schnittstelle des Repository und der Schnittstelle der Anwendungsschicht befindet. Auf die Einzelheiten der Schnittstellen wird nicht weiter eingegangen.

Suchdienst In der Funktionssammlung ist ein Suchdienst enthalten, der nach Musterkatalogen, Muster und Musterinhalten suchen kann. Dieser stellt einen großen Teil der Anwendungslogik dar und ist ebenfalls als Dienst über die Schnittstelle der Anwendungslogik für die Präsentationsschicht zugänglich. Der Suchdienst enthält das in der Spezifikation vorgestellte Prinzip der Suche nach Musterkatalogen, Mustern und deren Inhalten. Um nach Musterkatalogen, Mustern und Musterinhalten zu suchen, wird zunächst eine Suchanfrage benötigt. Diese enthält Einschränkungen in Form von Filtern, nach denen die Daten der Musterkatalogen, Mustern und deren Inhalten selektiert werden. Anschließend werden Daten aus dem Repository geladen, die bezüglich der Suchanfrage durchsucht werden sollen. Die geladenen Daten werden aufbereitet und mittels mehreren Filtern selektiert, um somit das Ergebnis der Suche zu erhalten. Das Suchergebnis kann durch die Präsentationsschicht abgerufen und den Lesern und Autoren repräsentiert werden.

4.4.3 Webbasiertes Frontend der Plattform für Mustersprachen und Musterkataloge

Die Präsentationsschicht der Plattform für Mustersprachen und Musterkataloge wird durch eine Rich Internet Application verkörpert. Diese bietet dem Benutzer eine einfache und universale Art und Weise auf sie zu zugreifen und eine ähnlich interaktive Bedienung und Darstellung wie eine Desktopanwendung. Außerdem ist sie unabhängig von Betriebssystemen und kompatibel zu allen modernen und gängigen Webbrowsern, die eine spezielle und abgeschottete Laufzeitumgebung darstellen und dadurch die Systemsicherheit erhöhen. Vorteile einer Rich Internet Application sind beispielsweise, dass sie nicht auf dem Computer des Benutzers installiert werden müssen, dass Updates automatisch mit dem Laden der Anwendungen verfügbar sind und dass URLs für die Referenzierung bestimmter Inhalte verwendet werden kann. [Sim07], [Fow02, S. 55-61], [SBB08, S. 1-19]

Das webbasiertes Frontend der Plattform für Mustersprachen und Musterkataloge in Form einer Rich Internet Application stellt eine Bedienanwendung für die Verwaltung und Verwendung der Mustersprachen und Musterkataloge der Plattform dar. Sie kann über die Schnittstelle der Anwendungslogik mit dieser kommunizieren und somit deren Dienste in Anspruch nehmen. Im weiteren Verlauf wird die Architektur der Bedienanwendung erläutert.

Herkömmliche Webseiten werden serverseitig aufgebaut und clientseitig repräsentiert. Im Gegensatz zu herkömmlichen Webseiten sind clientseitige Repräsentationen von Rich Internet Applications interaktiv. Das heißt der Webseitenaufbau wird clientseitig geändert und eventuell auch clientseitig erstellt. Viele Rich Internet Applications bestehen aus einer Kombination dieser beiden Konzepte. Oft wird das Grundgerüst der Webseite in der Anwendungsschicht aufgebaut und in der Präsentationsschicht dynamisch ergänzt und verändert. Um die Kombination dieser beiden Konzepte umzusetzen, werden serverseitige und clientseitige Skripte benötigt, die in den jeweiligen Bereichen ausgeführt werden.

In der Bedienanwendung der Plattform für Mustersprachen und Musterkatalogen wird der Aufbau der Webseite und deren interaktive Veränderung ausschließlich clientseitig durchgeführt. Die Architektur der Bedienanwendung trennt strikt die Anwendungsschicht von der Präsentationsschicht. Die Anwendungsschicht übernimmt somit nur Aufgaben, die sich auf fachliche Inhalte beziehen und unabhängig von der Präsentationsschicht sind. Da die Anwendungsschicht und Präsentationsschicht klar voneinander getrennt sind, könnte die Bedienanwendung zum Beispiel leicht durch eine andere ersetzt werden.

Die Bedienanwendung wird außerdem komplett clientseitig aufgebaut und verändert. Dadurch müssen ausschließlich für die Clientseite Quellcode entwickelt werden. Es besteht nicht die Notwendigkeit Serverskripte auszuführen, die clientseitigen Skriptcode erstellen, um einen server- und clientseitigen Webseitenaufbau zu ermöglichen. Die Entscheidung nur Code für die clientseitige Skriptengine zu liefern, soll die Struktur der Bedienanwendung übersichtlich halten. Außerdem soll es die Entwicklung vereinfachen, in dem man nur für eine Skriptengine Quellcode entwickelt. Zusätzlich nimmt die Bedienanwendung nicht die übliche Rolle einer Webseite ein. Es muss zum Beispiel nicht mit Suchmaschinen nach Inhalten gesucht werden, da die Suche nach Inhalten von einer programmeigenen Suche abgewickelt werden kann. Um Inhalte mit Suchmaschinen zu finden, müsste man diese statisch zur Verfügung stellen. Statische Webseiteninhalte müssen allerdings serverseitig aufgebaut werden. Da dies nicht benötigt wird und die Bedienanwendung eher einer Desktopanwendung ähnelt, bietet es sich an diese komplett clientseitig zu gestalten. [dwo], [BCFC06]

Die Bedienanwendung der Plattform für Mustersprachen und Musterkataloge benötig viele Ansichten, die deren Inhalte darstellen. Um dies verwalten zu können wird das Model-View-Presenter

(MVP) Muster angewendet. Die Umsetzung des MVP Musters soll auf der einen Seite einen Mechanismus darstellen, mit dem die Ansichten (View) verwaltet werden können. Auf der anderen Seite bietet das MVP Muster weitere technische Vorteile für die beschriebene Verwendung und Art der Bedienanwendung. Sie teilt die Logik der Oberfläche von der Ansicht, was zu einer klaren Struktur der Oberfläche führt. Auf Grund der eventbasierten Benutzeroberflächenumgebung des Browser bietet es sich an, die Steuerung der Ansichten durch ein extra Element abzuwickeln. Dieses Steuerelement wird in dem MVP Muster Presenter genannt. Es übernimmt die vollständige Steuerung der Ansichten. Wenn sich der zu Grunde liegende Datensatz ändert, dann wird der Presenter benachrichtigt, der wiederum die Ansicht aktualisiert. Falls eine Interaktion mit der Ansicht statt findet, dann werden die geänderten Daten und empfangenen Browserevents dem Presenter weitergeleitet. Dieser reagiert anschließend auf diese Veränderungen. Er kann die Ansicht wieder dem entsprechend anpassen oder der Datensatz aktualisieren. In Abbildung 4.10 ist das MVP Muster und das beschriebene Szenario zu sehen. [Fow06], [BM00], [Pot96]

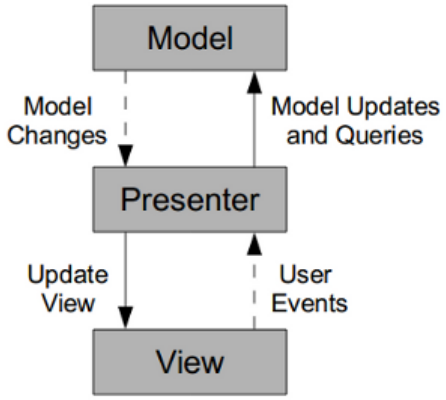


Abbildung 4.10: Model-View-Presenter Muster

Die Presenter des MVP Musters werden durch Ereignisse benachrichtigt, wenn eine Interaktion ihrer Ansichten stattfindet. Auch die Verwaltung mehrerer Presenter-Ansicht-Paare wird ebenfalls über eine Ereignisarchitektur gesteuert. Diese ereignisgesteuerte Benachrichtung liegt der ereignisgesteuerten Architektur zu Grunde. Sie wird durch ein Architekturmuster dargestellt, das die Kommunikation zwischen mehreren Komponenten durch das Versenden von Ereignissen beschreibt. Dabei steht speziell die Erstellung, die Erkennung und die Verarbeitung von Ereignissen im Vordergrund. Ereignisse werden meist bei Zustandsänderungen an Interessierte verschickt, die auf die empfangenen Ereignisse individuell reagieren.

Die Grundidee der ereignisgesteuerten Architektur basiert auf dem Publish-Subscribe-Konzept. [Jos08, S. 165-168] Dieses Konzept wurde in der Bedienanwendung durch das Mediator Muster umgesetzt, damit mehrere Presenter einfach verwaltet werden können. Das Mediator Muster stellt das Konzept für das Medium der Kommunikation in der Bedienanwendung der Plattform für Mustersprachen und Musterkataloge dar. Das Konzept des Mediator ist sehr verwandt mit dem Publish-Subscribe-Konzept, das auch Observer Muster [GHJV95, S. 293-304] genannt wird. In dem Konzept des Mediators gibt es ein Mediatorobjekt, dem beliebige Objekte Ereignisse übergeben können. Die Ereignisse werden anschließend von dem Mediatorobjekt an dessen Interessenten verteilt. In der

Architektur der Bedienanwendung wird das Mediatorelement zur Verwaltung der Presenter Event-Bus genannt. Der Event-Bus der Bedienanwendung kann eine komplexe Kommunikation durch einen lose gekoppelten Nachrichtenaustausch einfach bewältigen. Außerdem können mehrere Presenter auf eine einfache Art und Weise erreicht werden. [GHJV95, S. 273-282] Damit die Presenter gesteuert und kontrolliert werden können, verfügt die Bedienanwendung über eine Anwendungssteuerung. Die Anwendungssteuerung enthält die Logik der Benutzeroberfläche. Sie entscheidet, wann welche Benutzeroberfläche zu sehen ist und wie diese gestaltet ist. Zusätzlich kann die Anwendungssteuerung das Ausführen von benutzeroberflächenabhängigen Befehlen veranlassen. Oft wird solch eine Anwendungssteuerung durch das Application Controller Muster realisiert. Das Application Controller Muster wird allerdings in Verbingung mit dem Model-View-Controller Muster verwendet. Der vorgestellte Verwendungszweck ist zwar der selbe, aber durch die Verwendung des MCP Musters muss das Application Controller Muster leicht angepasst werden. In der klassischen Anwendungssteuerung des Application Controller Musters werden beispielsweise die Interaktionen der Benutzer durch Anfragen entgegengenommen. [Fow02, S. 379-386], [ACM03, S. 205-208] Dieser Ablauf wird in der Architektur der Bedienanwendung in den Presenter der Ansichten erledigt. Da die Presenter nicht für ansichtsübergreifende Aufgaben zuständig sind, übernimmt in diesem Fall die Anwendungssteuerung der Bedienanwendung die Abarbeitung dieser Aufgaben. Die Anwendungssteuerung erstellt oder ersetzt unter anderem die benötigten Presenter und deren Ansicht und verbindet diese mit dem Event-Bus. [Ram10]

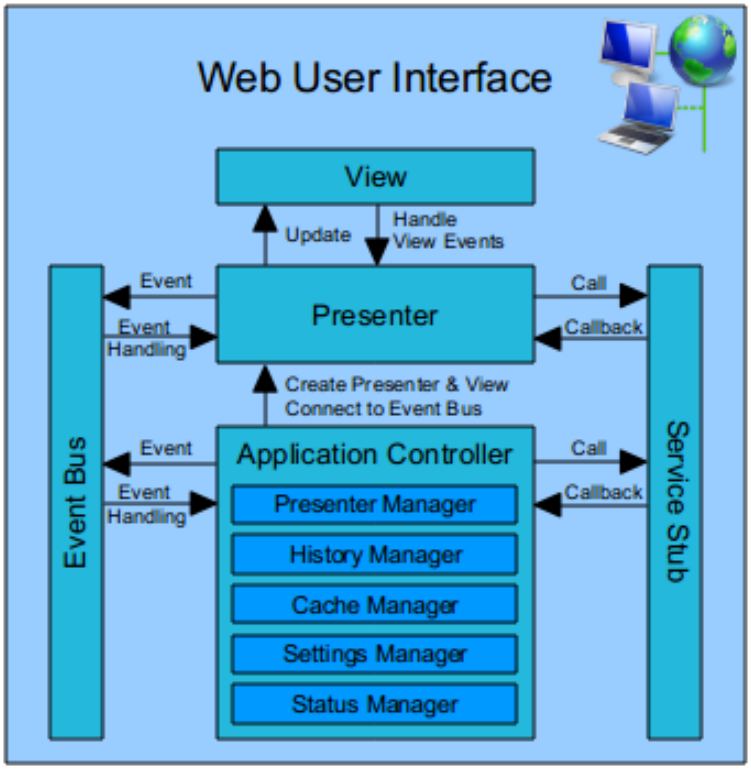


Abbildung 4.11: Architektur der Bedienanwendung der Plattform für Mustersprachen und Musterkataloge

In Abbildung 4.11 ist die Architektur der Bedienanwendung der Plattform für Mustersprachen und

Musterkataloge zu sehen. Sie enthält Ansichen (View) und Presenter (Presenter), die die beschriebenen Elemente des erwähnten MVP-Musters darstellen. Die Repräsentation des Datensatzes, der auch einen Element des MVP-Musters darstellt, ist nicht explizit in der Abbildung aufgeführt. Teile des Datensatzes werden in der Anwendungssteuerung verwaltet. Es ist außerdem der Event Bus, der das Kommunikationsmedium zwischen Presenter und Anwendungssteuerung (Application Controller) darstellt, zu sehen. Die Aufgaben der Anwendungssteuerung wird in der Abbildung in ihre Aufgaben unterteilt, die von Managern ausgeführt werden. Die Manager kümmern sich um die Einstellungen der Anwendung (Settings Manager), deren Status (Status Manager), die Webbrowsernavigation (History Manager), die Zwischenspeicherung von Teilen des Datensatzes (Cache Manager) und die Ansichten- beziehungsweise Presenterlogiken (Presenter Manager). Die Anwendungssteuerung und die Presenter können auf Funktionen (Service Stubs) zurückgreifen, die es erleichtern Dienste der Anwendungsschicht aufzurufen. Durch die Verwendung der Hilfsfunktionen und das Verwenden der entfernten Dienste, können für die Darstellung benötigte Datensätze geladen werden. Diese Datensätze können, falls dies erforderlich ist, in der Anwendungssteuerung zwischengespeichert werden. Die Hilfsfunktionen basieren auf dem Service Stub Muster [Fow02, S. 504-507]. Es erleichtert und vereinfacht das Aufrufen und Ausführen von entfernten Diensten.

Die Presenter und Ansichten der Bedienanwendung sind in unterschiedliche hierarchische Gruppen eingeordnet. Dies soll helfen eine Übersicht zu schaffen und eine Navigation für den Benutzer zu erstellen. Außerdem sollen Muster durch eine URL referenziert werden können. Damit die entsprechenden Ansichten und deren Inhalte angezeigt werden können, werden die Presenter und ihre Ansichten strukturiert. Die Bedienanwendung besteht aus fünf Gruppen: Presenter und Ansichten für anwendungsspezifische Benutzeroberflächen, für die Verwaltung von Mustersprachen, für die Betrachtung von Mustersprachen, für die Verwaltung von Musterkatalogen und für die Betrachtung von Musterkatalogen. Die Gruppen sind bezüglich ihrer Aufgabenbereiche erstellt. Die Gruppe der Presenter und Ansichten für anwendungsspezifische Benutzeroberflächen enthält Presenter und Ansichten, die für die Verwendung der Bedienanwendung notwendig sind oder dem Benutzer zusätzliche Funktionalitäten zur Verfügung stellt. Die Presenter und Ansichten präsentieren unter anderem die Anwendungsnavigation, die Werkzeugleiste oder die Statusanzeige. Die anderen Gruppen enthalten Presenter und Ansichten, die bezüglich der Verwaltung und Betrachtung der Mustersprachinhalte oder Musterkataloginhalte benötigt werden. Durch sie können die Inhalte der Mustersprachen und Musterkataloge dargestellt und verwaltet werden.

4.5 Implementierung der Plattform für Mustersprachen und Musterkataloge

Die Implementierungsbeschreibung enthält die Umsetzung der Struktur der Plattform für Mustersprachen und Musterkataloge. Dabei wird die Verwendung verschiedener Frameworks und der Einsatz benötigter Software geschildert.

Die Software ist in der Entwicklungsumgebung Eclipse [Ecl11] entwickelt worden. Jede Softwarekomponente, die eine Schicht der Plattformstruktur darstellt, ist in der Programmiersprache Java [Ora11] geschrieben.

Weitere Einzelheiten der Umsetzung von der Plattformstruktur sind im folgenden Verlauf aufgelistet.

Umsetzung des Repositorys Das Repository besteht aus einer in Java [Ora11] geschriebenen Programmlogik und wird serverseitig ausgeführt. Es enthält eine Web-Service-Schnittstelle mit der das Repository angesprochen werden kann und einen Datenmapper, der Objektstrukturen zuordnet und konvertiert. Die Web-Service-Schnittstelle ist in Verbindung mit Axis [Axi11] entwickelt. Das Repository lief während der Entwicklungs- und Testphase auf einem Apache Tomcat Server 7 [Apa11]. Die Daten der Mustersprachen und Musterkataloge werden von dem Repository in einen Datenbankserver gespeichert und aus diesem gelesen. Der Datenbankserver wird durch einen MySQL Server [MyS11] verkörpert. Die Datenbankstruktur ist mit dem Softwarewerkzeug phpMyAdmin [Php11] erstellt worden. Zusätzlich zu der Datenbank wurde eine Anlageverzeichnis erstellt, damit Dateien abgelegt werden können. Es stellt einen einfachen Ordner des Systems dar, in dem der Datenmapper die Anlagen speichert. Metainformation über die abgelegten Anlagen werden in der Datenbank festgehalten und können dadurch Mustern zugeordnet werden.

Umsetzung des Anwengunsservers Die Software der Anwendungsschicht ist ebenfalls in Java [Ora11] geschrieben und enthält zwei Dienste. Ein Dienst ist für das Aufbereiten, Laden und Speichern von Daten verantwortlich. Der andere Dienst wickelt die Suche nach Musterkataloginhalten ab. Die Dienste können die Web-Service-Schnittstelle des Repositorys ansprechen und anschließend ihre Berechnungen durchführen. Das Ergebnis der Berechnungen wird der Präsentationsschicht weitergereicht. Die Software der Anwendungsschicht ist während der Entwicklungs- und Testphase auf einem Jetty-Webserver Version 1.6 [Jet11] ausgeführt worden.

Umsetzung des webbasierten Frontends Das webbaisierte Frontend der Plattform für Mustersprachen und Musterkataloge wurde mit dem Framework Google Web Toolkit (GWT) [Goo11] realisiert. Das Framework GWT wurde gewählt um die Anforderungen und der Spezifikation der Plattform für Mustersprachen und Musterkataloge gerecht werden. Mit GWT lässt sich außerdem die gewählte Struktur der Bedienanwendung der Plattform für Mustersprachen und Musterkatalogen umsetzen, ohne eine Browserplugin verwenden zu müssen. Die Bedienanwendung ist ebenfalls in Java geschrieben und ist anschließend in einen Mix aus JavaScript, Html und CSS übersetzt worden. Dieses aus JavaScript, Html und Css bestehende Codekonstrukt stellt die Bedienanwendung der Plattform dar. Über die Java-Servlet-Technologie [Gow10] kann die Schnittstelle der Anwendungsschicht ansprechen werden. Somit können für die Darstellung und die Verwaltung von Mustersprachen und Musterkatalogen Daten anfordern oder manipulieren werden. [Dwy08, S. 5-14], [CC08, S. 30-31]

Kapitel 5

Zusammenfassung und Ausblick

5.1 Zusammenfassung

In diesem Abschnitt werden die in dieser Arbeit gefundenen Errungenschaften und konfrontierten Probleme in Kürze zusammengefasst:

Grundlagen des Metamodells für Mustersprachen Zu Beginn dieser Arbeit wurden bekannte Mustersprachen und deren Strukturen vorgestellt, die Modelle für Musterkataloge darstellen. Die Musterkataloge enthalten jeweils eine Menge von Mustern, die Lösungsvorschläge für Probleme eines bestimmten Kontextes zur Verfügung stellen.

Metamodell für Mustersprachen Die vorgestellten Mustersprachen enthalten Strukturen und Charakteristiken anhand derer ein Metamodell für Mustersprachen erstellt wurde. Das Metamodell für Mustersprachen besteht aus einer Struktur, die den Aufbau von Mustersprachen beschreibt und die deren Kompatibilität untereinander gewährleistet. Die Struktur des Metamodells für Mustersprachen wird durch ein Profil erweitert, das die grundlegende Charakteristik der Mustersprachen vorgibt. Das Profil des Metamodells für Mustersprachen ist ebenso wie die Struktur des Metamodells für Mustersprachen ein Bestandteil des Metamodells. Das Profil formt die grundlegende Charakteristik der Mustersprachen. Es verfeinert die Struktur der Mustersprachen und ordnet deren Bestandteilen Bedeutungen zu. Das Metamodell für Mustersprachen ist ein mustersprachenübergreifendes Modell, das eine Menge von Mustersprachen beschreibt. Mustersprachen, die dem Metamodell für Mustersprachen zu Grunde liegen, werden durch die gemeinsame Struktur kompatibel gehalten. Die Kompatibilität der Mustersprachen beinhaltet die Möglichkeit in und zwischen Mustersprachen und deren Musterkatalogen Beziehungen aufzubauen. Darüber hinaus können in dem Metamodellprofil für Mustersprachen die Bedeutungen der Beziehungen und Strukturelementen der Mustersprachen festgehalten werden. Durch die Definition und durch Erweiterungen des Metamodellprofils für Mustersprachen können mustersprachenübergreifende Suchen, Vergleiche und Analysen von den Inhalten der Musterkataloge in und zwischen Musterkatalogen durchgeführt werden. Zusätzlich wird die Gestaltung der Definition einer Mustersprachen durch die Erweiterungsmöglichkeit des Metamodellprofils für Mustersprachen variable gehalten. Damit aus dem Metamodell für Mustersprachen einfach Musterkataloge abgeleitet werden können, wurde das Metamodell für Mustersprachen als UML-Diagramm festgehalten und entsprechende dem modellgetriebenen Architekturansatz entworfen. Es stellt somit auch eine Grundlage für eine Datenstruktur dar, die eine maschinelle Handhabung und Verwaltung von Mustersprachen und Musterkatalogen ermöglicht.

Plattform für Mustersprachen und Musterkataloge Damit das Metamodell für Mustersprachen maschinell verwendet werden kann, wurde in dieser Arbeit eine Plattform für Mustersprachen und Musterkataloge spezifiziert, entworfen und implementiert. Die Plattform für Mustersprachen und Musterkataloge ermöglicht die Erstellung und Verwaltung von Mustersprachen und Musterkatalogen. Sie verwendet das Metamodell für Mustersprachen als Grundlage für die Datenstruktur der Mustersprachen und Musterkataloge. Das vorgestellte Metamodell für Mustersprachen wurde durch zusätzliche Profilerweiterungen angepasst, damit es den Anforderungen der Plattform für Mustersprachen und Musterkataloge gerecht wird. Die Erweiterung von dem Profil des Metamodells für Mustersprachen bietet unter anderem die Möglichkeit Diskussionen über Musterinhalte zu führen oder Veränderungen der Musterinhalte als neue Musterversionen abzuspeichern. Diese Erweiterung ermöglicht es ebenso Inhalte der Muster durch das Anhängen von Dateien zu erweitern.

Aufbau der Plattform für Musterkataloge Die Plattform für Mustersprachen und Musterkataloge ist in eine Drei-Schichten-Architektur aufgeteilt. Ein Repository, das die Daten der Mustersprachen und Musterkataloge verwaltet, übernimmt die Aufgabe der Datenhaltungsschicht. In der Anwendungsschicht der Drei-Schichten-Architektur werden zwei Dienste zur Verfügung gestellt. Ein Dienst ermöglicht das Laden und Speichern von Daten aus dem Repository sowie die Zusammenstellung, die Aufbereitung und die Konsistenzprüfung der Daten von Mustersprachen und Musterkatalogen. Der zweite Dienst ermöglicht das Suchen nach Musterkatalogen, Mustern, oder deren Musterinhalten. Eine webbasierte Bedienanwendung der Plattform für Mustersprachen und Musterkataloge, die eine Rich Internet Application darstellt, repräsentiert die Präsentationsschicht der Plattform. Mit Hilfe der Bedienanwendung der Plattform für Mustersprachen und Musterkataloge lassen sich die Mustersprachen und Musterkataloge anlegen, pflegen, verwalten und durchstöbern.

5.2 Ausblick

Im Folgenden wird über eine mögliche Fortführung und den Ausblick dieser Arbeit berichtet:

Evaluation des Metamodells für Mustersprachen Das Metamodell für Mustersprachen wird in dieser Arbeit nicht auf die Qualität seiner Verwendbarkeit getestet. Bevor man mit der Entwicklung des Metamodells für Mustersprachen fortfährt, ist es notwendig, die Anforderungen des Metamodells für Mustersprachen zu überprüfen. Damit können Fehler und Erweiterungsbedürfnisse aufgedeckt werden. Die Analyse der Verwendbarkeitsqualität und der Erfüllung der Anforderungen des Metamodells für Mustersprachen kann durch die Verwendung der Plattform für Mustersprachen und Musterkataloge durchgeführt werden. In diesem Zuge kann auch die Plattform für Mustersprachen und Musterkataloge bezüglich der Erfüllung ihrer Anforderungen und der Qualität ihrer Bedienbarkeit überprüft werden.

Weiterentwicklung des Metamodells für Mustersprachen Das Profil des Metamodells für Musterpsrachen enthält und beschreibt die Arten der Strukturelemente im Metamodell. Durch diese Arten und deren Beschreibungen enthalten die Elemente der Struktur im Metamodell für Mustersprachen Bedeutungen, die wiederum Metainformation in Mustern und Musterkatalogen darstellen. Die Metainformationen wie zum Beispiel die Art des Inhaltes von Mustern oder der Beziehungstyp zwischen Mustern sind in dem Profil des Metamodells für Mustersprachen festgelegt.

In dieser Arbeit werden diese Metainformationen verwendet, um den Lesern und Autoren Aufschluss über die Mustersprachen und Muster zu geben. Die Plattform für Mustersprachen und Musterkataloge verwendet die Metainformationen der Muster, um sie zu verwalten und nach ihnen zu suchen. Die Metainformation von Mustersprachen, Mustern und deren Musterkataloge können für viele weitere Hilfsfunktionen, Dienste oder Anwendungen verwendet werden. Diese können Berechnungen ausführen und Ergebnisse liefern, die auf den Beschreibungen der Inhalte von Mustersprachen, Mustern und deren Musterkataloge aufbauen.

Automatische Generierung von Musterlösungen Eine Erweiterungsmöglichkeit des Metamodells für Mustersprachen ist die automatische Generierung von Musterlösungen. Die automatische Generierung von Musterlösungen verwendet die Inhalte und die Metainformationen der Muster und Musterkataloge, um spezielle Lösungen zu generieren, die in den Mustern beschrieben sind. Die generierten Musterlösungen können beliebige Datensätze und Befehle darstellen. Sie können beispielsweise als Konfigurationen, Operationen oder als eigenständige Programme verwendet werden. Die Umsetzung der automatischen Generierung von Musterlösungen benötigt unter anderem die Musterinhalte um Lösungen zu erstellen. Die Mustersprachen und Musterkataloge, die aus dem Metamodell für Mustersprachen aus dieser Arbeit abgeleitet wurden, enthalten Muster, deren Inhalte nicht formal festgehalten sind. Eine Herausforderung wird in der maschinellen Interpretation der Musterinhalte liegen, um die Semantik der Inhalte zu verstehen und darauf basierend Lösungen berechnen zu können.

Quellenverzeichnis

[ACM03] ALUR, Deepak; CRUPI, John ; MALKS, Dan: *Core J2EE Patterns: Best Practices and Design Strategies*. Prentice Hall Professional, 2003

[Apa11] APACHE TOMCAT PROJECT: *Apache Tomcat*. `http://tomcat.apache.org/`. Version: August 2011

[Axi11] AXIS: *WebServices - Axis*. `http://axis.apache.org/axis/`. Version: August 2011

[BCFC06] BOZZON, Alessandro; COMAI, Sara; FRATERNALI, Piero ; CARUGHI, Giovanni T.: Conceptual modeling and code generation for rich internet applications. In: *Proceedings of the 6th international conference on Web engineering*. Palo Alto, California, USA : ACM, 2006 (ICWE '06), S. S. 353–360

[BCHP03] BROWN, Kyle; CRAIG, Gary; HESTER, Greg ; PITT, David: *Enterprise Java programming with IBM WebSphere*. Addison-Wesley Professional, 2003

[Ber94] BERNERS-LEE, T.: *Universal Resource Identifiers in WWW: A Unifying Syntax for the Expression of Names and Addresses of Objects on the Network as used in the World-Wide Web*. `http://tools.ietf.org/html/rfc1630`. Version: Juni 1994

[BHS07] BUSCHMANN, Frank; HENNEY, Kevlin ; SCHMIDT, Douglas C.: *Pattern Oriented Software Architecture Volume 5: On Patterns and Pattern Languages*. Bd. 5. 1. Wiley, 2007

[BJ94] BECK, Kent; JOHNSON, Ralph E.: Patterns Generate Architectures. In: *Proceedings of the 8th European Conference on Object-Oriented Programming*, Springer-Verlag, 1994, S. 139–149

[BM00] BOWER, Andy; MCGLASHAN, Blair: *Twisting the Triad - The evolution of the Dolphin Smaltalk MVP application framework*. `www.object-arts.com/downloads/papers/TwistingTheTriad.PDF`. Version: 2000

[BMR+98] BUSCHMANN, Frank; MEUNIER, Regine; ROHNERT, Hans; SOMMERLAD, Peter ; STAL, Michael: *Pattern-orientierte Software-Architektur: Ein Pattern-System*. Addison-Wesley, 1998

[CAA77] CHRISTOPHER ALEXANDER, Murray Silverstein Max Jacobson Ingrid Fiksdahl-King Sara I. Sara Ishikawa; ANGEL, Shlomo: *A PATTERN LANGUAGE: TOWNS BUILDINGS CONSTRUCTION*. Oxford University Press, 1977

[CC08] COOPER, Robert; COLLINS, Charles: *GWT in Practice*. Manning Publications, 2008

[CCMS01] CHRISTENSEN, Erik; CURBERA, Francisco; MEREDITH, Greg ; SANJIVA, Weerawarana: *Web Services Description Language (WSDL)*. http://www.w3.org/TR/wsdl. Version: März 2001

[DC99] DORIGO, Marco; CARO, Gianni D.: Ant Colony Optimization: A New Meta-Heuristic. In: *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on* (1999)

[dwo] A Concept of a Web Application Blending Thin and Fat Client Architectures. In: *Dependability of Computer Systems, International Conference on*. Los Alamitos, CA, USA : IEEE Computer Society

[Dwy08] DWYER, Jeff: *Pro Web 2.0 Application Development with GWT*. 1. Apress, 2008

[Ecl11] ECLIPSE FOUNDATION, INC.: *Eclipse - The Eclipse Foundation Open Source Community Website*. http://www.eclipse.org/. Version: August 2011

[Etz64] ETZIONI, Amitai: *Modern Organizations*. 1. Prentice Hall, 1964

[Fow02] FOWLER, Martin: *Patterns of Enterprise Application Architecture*. Addison-Wesley Longman, Amsterdam, 2002

[Fow06] FOWLER, Martin: *GUI Architectures*. http://www.martinfowler.com/eaaDev/uiArchs.html. Version: Juli 2006

[Fre09] FREIBERGER, Jörg: Was für ein Tier. In: *database pro* (2009), Juni, S. S. 76–81

[GHJV95] GAMMA, Erich; HELM, Richard; JOHNSON, Ralph ; VLISSIDES, John: *Design Patterns*. Addison-Wesley, 1995

[Gig91] GIGCH, John P. V.: *System design modeling and metamodeling*. Springer, 1991

[Goo97] GOOS, Gerhard: *Vorlesungen Uber Informatik: Band 3: Berechenbarkeit, Formale Sprachen, Spezifikationen*. Springer, 1997

[Goo11] GOOGLE WEB TOOLKIT: *Google Web Toolkit - Google Code*. http://code.google.com/webtoolkit/. Version: August 2011

[Gow10] GOWDAR, Girish: *Java Servlet Technology*. http://www.oracle.com/technetwork/java/javaee/servlet/index.html. Version: September 2010

[HW04] HOHPE, Gregor; WOOLF, Bobby: *Enterprise integration patterns: Designing, building, and deploying messaging solutions.* Addison-Wesley, 2004

[Jet11] JETTY WEBSERVER: *jetty - Jetty WebServer*. http://jetty.codehaus.org/jetty/. Version: August 2011

[Jos08] JOSUTTIS, Nicolai: *SOA in der Praxis: System-Design für verteilte Geschäftsprozesse*. 1. Dpunkt Verlag, 2008

[Laa03] LAAKSO, Sari A.: *User Interface Design Patterns*. http://www.cs.helsinki.fi/u/salaakso/patterns/. Version: September 2003

[Lam07] LAMMI, Janne: *Developing a UI Design Pattern Library - A Case Study at eCraft*, HELSINKI UNIVERSITY OF TECHNOLOGY, Department of Computer Science and Engineering und Laboratory of Software Business and Engineering, Mastersthesis, September 2007. http://www.scribd.com/doc/29180579/Developing-a-UI-Design-Pattern-Library-A-Case-Study-at-eCraft/-Master-s-Thesis

[LL10] LUDEWIG, Jochen; LICHTER, Horst: *Software Engineering: Grundlagen, Menschen, Prozesse, Techniken*. 2. überarb. u. akt. Aufl. dpunkt Verlag, 2010

[LMW05] LEACOCK, Matt; MALONE, Erin ; WHEELER, Chanel: *Implementing a Pattern Library in the Real World: A Yahoo! Case Study*. http://www.leacock.com/patterns/leacock_malone_wheeler.pdf. Version: Februar 2005

[LVH11] LAMMI, Janne; VARJOKALLIO, Matti ; HOCKSELL, Johannes: *UI Design Pattern Library | Patternry*. http://patternry.com/patterns/. Version: 2011

[Mah08] MAHEMOFF, Michael: *Ajax Patterns*. http://ajaxpatterns.org/wiki/index.php?title=Main_Page. Version: August 2008

[McL02] MCLAUGHLIN, Brett: *Building Java Enterprise Applications: Architecture*. O'Reilly Media, Inc., 2002

[MD96] MESZAROS, Gerard; DOBLE, Jim: MetaPatterns: A Pattern Language for Pattern Writing. In: *In 3rd Pattern Languages of Programming conference*, 1996, S. S. 4–6

[MM03] MILLER, J.; MUKERJI, J.: MDA Guide Version 1.0.1 / Object Management Group (OMG). 2003 (omg/03-06-01). – Forschungsbericht

[MO99] MARTIN, James; ODELL, James J.: *Objektorientierte Modellierung mit UML. Das Fundament.* Prentice Hall, 1999

[MyS11] MYSQL: *MySQL :: Die populärste Open-Source-Datenbank der Welt*. http://www.mysql.de/. Version: August 2011

[New72] NEWELL, Allen: *Human Problem Solving*. 4. Auflage. Longman Higher Education, 1972

[Obj10] OBJECT MANAGEMENT GROUP: *UML 2.3 Superstructure*. http://www.omg.org/spec/UML/2.3/Superstructure/PDF. Version: Mai 2010

[Ora11] ORACLE: *Oracle Technology Network for Java Developers*. http://www.oracle.com/technetwork/java/index.html. Version: August 2011

[PHBO10] PAUWELS, Stefan L.; HÜBSCHER, Christian; BARGAS-AVILA, Javier A. ; OPWIS, Klaus: Building an interaction design pattern language: A case study. In: *Computers in Human Behavior* 26 (2010), Mai, S. S. 452–463. – ACM ID: 1750044

[Php11] PHPMYADMIN DEVEL TEAM: *phpMyAdmin*. http://www.phpmyadmin.net/home_page/. Version: August 2011

[Pot96] POTEL, Mike: *MVP: Model-View-Presenter The Taligent Programming Model for C and Java*. Taligent Inc., 1996

[Ram10] RAMSDALE, Chris: *Large scale application development and MVP*. http://code.google.com/webtoolkit/articles/mvp-architecture.html. Version: März 2010

[RJB99] RUMBAUGH, James; JACOBSON, Ivar ; BOOCH, Grady: *The Unified Modeling Language Reference Manual*. Addison Wesley Professional, 1999

[SBB08] SMEETS, Bram; BONESS, Uri ; BANKRAS, Roald: *Beginning Google Web Toolkit: From Novice to Professional*. Apress, 2008

[Sch01] SCHÖNING, Uwe: *Theoretische Informatik - kurzgefasst*. Spektrum Akademischer Verlag GmbH, 2001

[Sco06] SCOTT, Bill: *Yahoo! Design Pattern Library Released*. http://www.yuiblog.com/blog/2006/02/13/yahoo_patterns_released/. Version: Februar 2006

[SE07] SUMATHI, S.; ESAKKIRAJAN, S.: *Fundamentals of Relational Database Management Systems*. Springer, 2007

[Sim07] SIMMONS, Andrea: Rich Internet Applications 101: A Primer for Marketing Agencies & Multimedia Developers, Integration New Media, 2007

[Spo06] SPOOL, Jared M.: *The Elements of a Design Pattern*. http://www.uie.com/articles/elements_of_a_design_pattern/. Version: Januar 2006

[Tar09] TARKOMA, Sasu: *Mobile middleware: architecture, patterns and practice*. John Wiley and Sons, 2009

[Tho11] THOMAS, Erl: *SOA Patterns - Service Data Replication*. http://www.soapatterns.org/service_data_replication.php. Version: Juli 2011

[Tid99] TIDWELL, Jenifer: *Common Ground: A Pattern Language for Human-Computer Interface Design*. http://www.mit.edu/~jtidwell/interaction_patterns.html. Version: 1999

[Van11] VAN WELIE, Martijn: *Interaction Design Pattern Library - Welie.com*. http://welie.com/patterns/. Version: 2011

[vv03] VAN WELIE, Martijn; VAN DER VEER, Gerrit C.: Pattern Language in Interation Design: Structure and Organization. In: *Proceedings of Interact* (2003), September, S. S. 527–534

[WCL+05] WEERAWARANA, Sanjiva; CURBERA, Francisco; LEYMANN, Frank; STOREY, Tony ; FERGUSON, Donald F.: *Web services platform architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging, and more*. Prentice Hall PTR, 2005

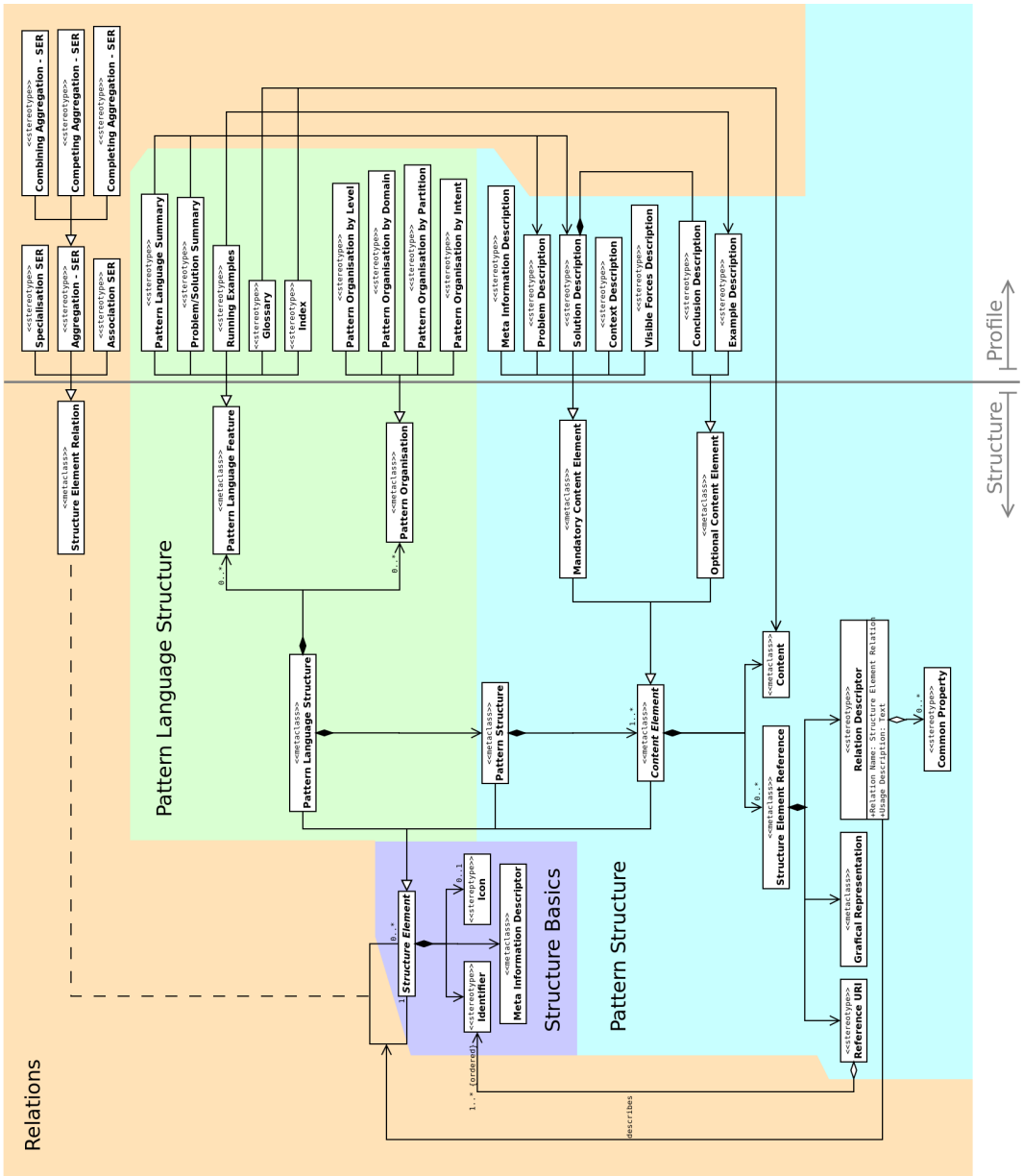
[Yah11] YAHOO! INC. (2006): *Yahoo! Design Pattern Library*. Version: August 2011. http://developer.yahoo.com/ypatterns/

Abbildungsverzeichnis

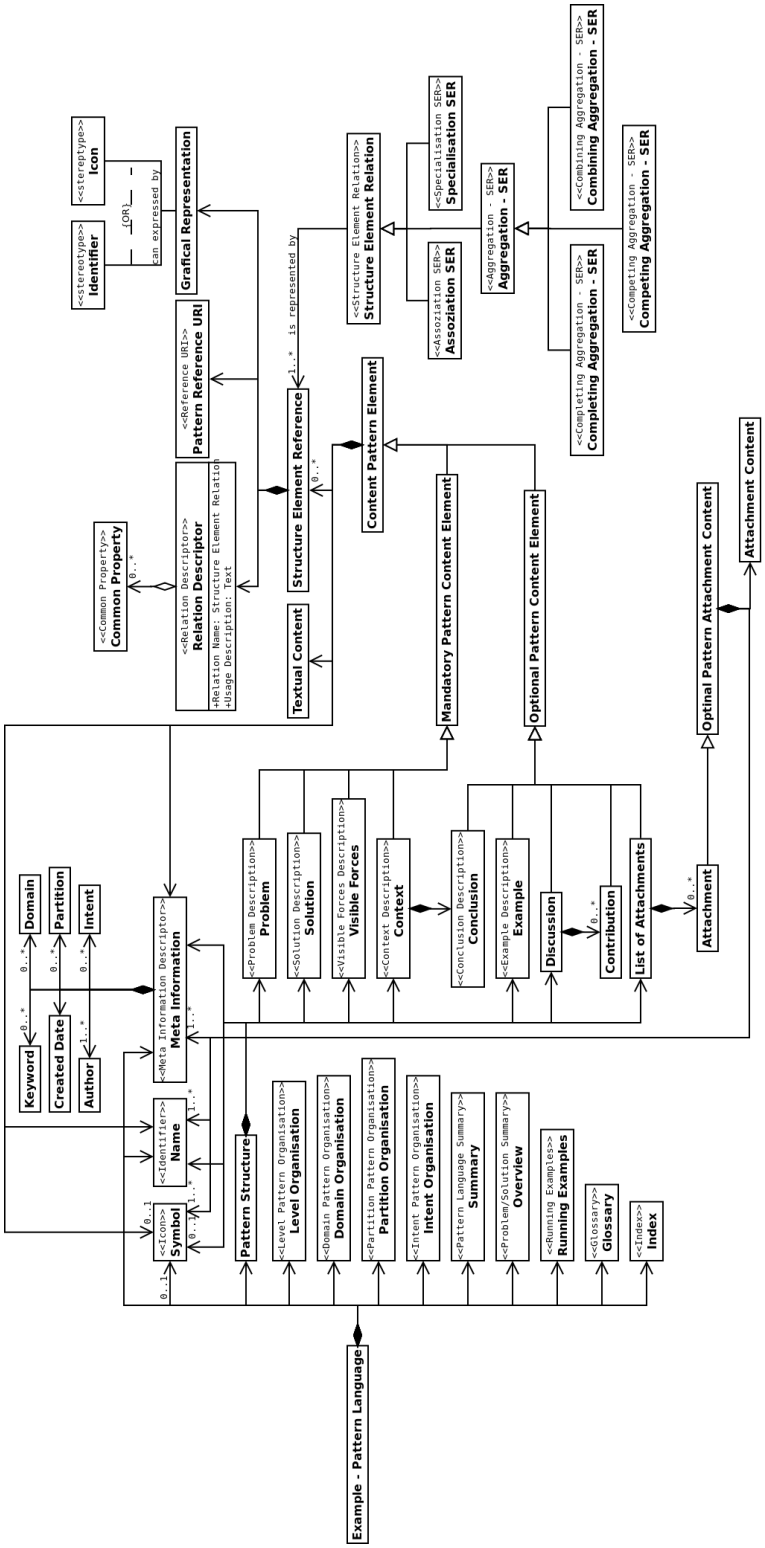
Algorithmenverzeichnis

Anhang

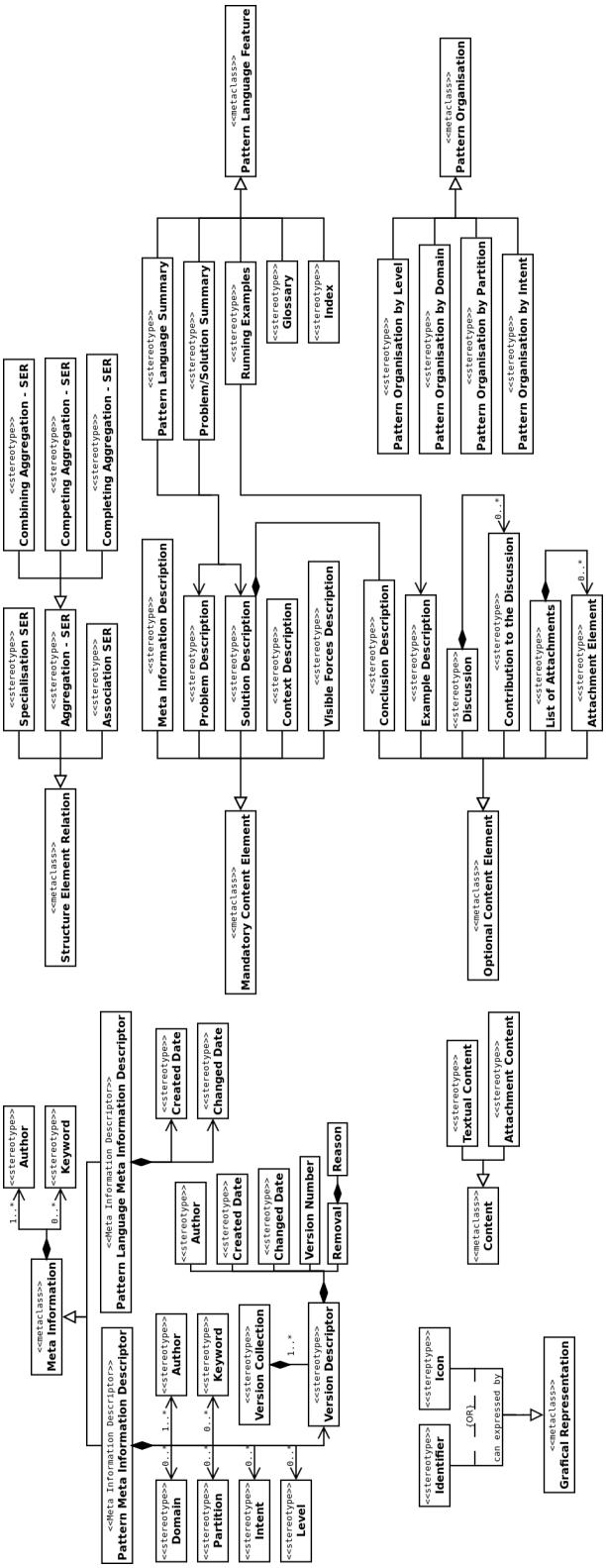
Anhangsverzeichnis



Anhang A: Metamodell für Mustersprachen



Anhang B: Beispiel einer Mustersprache



Anhang C: Profilerweiterungen des Metamodellprofils für Mustersprachen

Erklärung

Hiermit versichere ich, diese Arbeit selbständig verfasst und nur die angegebenen Quellen benutzt zu haben.

_____

(Philipp Grimm)