Technische Universität Hamburg-Harburg
Institut Softwaresysteme

Diplomarbeit

Konzeptuelle Modellierung durch Domänenexperten

Prozess und Unterstützung durch eine Werkbank

Henner Carl

6.2.2007

Gutachter Prof. Dr. Joachim W. Schmidt
Prof. Dr. Helmut Weberpals

Betreuung Sebastian Bossung

Erklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig angefertigt habe. Die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht.

Die Arbeit wurde weder einer anderen öffentlichen oder privaten Institution vorgelegt noch veröffentlicht.

Hamburg, der 6. Februar 2007 _____

Henner Carl

In dieser Arbeit werden Warenzeichen ohne besondere Kennzeichnung eines evtl. bestehenden Schutzes verwendet. Aus dem Fehlen der Kennzeichnung kann nicht geschlossen werden, dass die Verwendung eines Namens frei ist.

Inhaltsverzeichnis

Abbildungsverzeichnis **vii**

Tabellenverzeichnis **ix**

Abkürzungen **xi**

1 Einleitung **1**

2 Konzeptorientierte Inhaltsverwaltung (CCM) **3**
2.1 Einführung . . . 3
2.2 Assetdefinitionssprache . . . 4
2.2.1 Intensionale Klassendefinition . . . 4
2.2.2 Extensionale Klassendefinition . . . 6
2.3 Realisierung . . . 7
2.3.1 Aufbau eines generierten Systems . . . 7
2.3.2 Realisierung von Offenheit und Dynamik . . . 7
2.3.3 Modellkompiler . . . 8
2.4 Erfahrungen . . . 9

3 Asset Expressions **11**
3.1 Grundlagen . . . 11
3.1.1 Einführung . . . 11
3.1.2 Syntax . . . 11
3.1.3 Grafische Repräsentation . . . 13
3.2 Komponenten und Selektoren . . . 13
3.3 Typisierte Asset Expressions . . . 14
3.3.1 Semantische Typen . . . 14
3.3.2 Typsystem . . . 15
3.3.3 Traits . . . 16
3.3.4 Intensionale Typen: Assetklassen . . . 17
3.4 Anwendungen . . . 18

4 Modellierungsmethodiken **19**
4.1 Begriffsklärung . . . 19
4.1.1 Methodiken, Ansätze und Prozesse . . . 19
4.1.2 Klassifizierung von Modellierungsansätzen . . . 20
4.2 Objektorientierte Analyse . . . 22

4.2.1 Analysemuster . . . 23
4.2.2 Linguistische Analyse . . . 24
4.2.3 Konzeptuelle Kategorien . . . 24
4.3 Anforderungsanalyse . . . 25
4.3.1 Systemvision und Systemkontext . . . 27
4.3.2 Workshops . . . 27
4.3.3 Anwendungsfälle . . . 28
4.3.4 Prototypen . . . 30
4.4 Modellierungsverfahren für Domänenontologien . . . 30
4.4.1 Kollaborative Methode nach Holsapple und Joshi . . . 31
4.4.2 On-To-Knowledge Metaprozess . . . 32
4.5 Konzeptuelle Modellierung . . . 34

5 Prozessentwurf **37**
5.1 Anforderungen an die Prozesse . . . 37
5.1.1 Systemerstellung und Modellierung . . . 37
5.1.2 Prozessvarianten und Prozessframework . . . 37
5.1.3 Zielsetzung für den Modellierungsprozess . . . 38
5.2 CCM Systemerstellungsprozess . . . 39
5.2.1 Prozessschritt: Projektplanung . . . 40
5.2.2 Unterprozess: Infrastrukturprozess . . . 41
5.2.3 Unterprozess: Beispielgetriebene Modellierung . . . 43
5.2.4 Prozessschritt: Inbetriebnahme . . . 45
5.2.5 Beziehung zur Domänenentwicklung . . . 45
5.2.6 Beziehung zum Wissensmanagement . . . 47
5.3 Beispielgetriebene Modellierung . . . 48
5.3.1 Iterativer Prozessverlauf . . . 49
5.3.2 Anforderungsanalyse . . . 51
5.3.3 Beispielgewinnung . . . 54
5.3.4 Schemaerstellung . . . 57
5.3.5 Evaluation . . . 59

6 Die Werkbank **63**
6.1 Prozessunterstützung durch die Werkbank . . . 63
6.1.1 Anforderungsanalyse . . . 63
6.1.2 Beispielgewinnung . . . 64
6.1.3 Schemaerstellung . . . 65
6.1.4 Evaluation . . . 67
6.1.5 Inbetriebnahme . . . 67
6.2 Interaktive Schemainferenz durch Clustering . . . 68
6.2.1 Motivation . . . 68
6.2.2 Ansatz . . . 70
6.2.3 Schritt 1 – Klassifikation . . . 75
6.2.4 Schritt 2 – Optimierung . . . 79
6.2.5 Schritt 3 – Taxonomiebildung . . . 81
6.2.6 Schritt 4 – Visualisierung . . . 82

7 Diskussion **85**
7.1 Diskussion verwandter Ansätze 85
7.1.1 Delphi-Methode nach Holsapple und Joshi 85
7.1.2 Kollaborative Entwicklung konzeptueller Schemata 86
7.1.3 DynamOnt 86
7.2 Bewertung 87
7.3 Ausblick 89
7.3.1 Beispielgetriebene Modellierung 89
7.3.2 Werkbank 89
7.3.3 Kollaborative Modellierung 89
7.3.4 Weiterentwicklung der konzeptorientierten Inhaltsverwaltung . . . 90

Literaturverzeichnis **91**

Abbildungsverzeichnis

2.1 Konzeptorientierte Inhaltsverwaltung: Szenario 4
2.2 Beispiel: Klassendiagramm einer Anwendungsdomäne 5
2.3 Modulkonfiguration zur Personalisierung nach [SBS05] 8
2.4 Abläufe im Modellkompiler aus [Carl06] 9

3.1 Grafische Repräsentation einer Asset Expression 13
3.2 Grafische Asset Expression mit Komponenten 14

4.1 Begrifflichkeiten: Methodologie, Methodik, Prozess 20
4.2 Anwendungsfall (*use case*) 29
4.3 On-To-Knowledge Metaprozess nach [SSSS01] 33

5.1 CCM Systemerstellungsprozess 40
5.2 Artefakte vom Infrastruktur und Modellierungsprozess 42
5.3 Ansatz der beispielgetriebenen Modellierung 44
5.4 Iterative Domänen- und Anwendungsentwicklung nach [KK06] 46
5.5 Phasen der beispielgetriebenen Modellierung 49
5.6 Anforderungen in der BGM 51
5.7 Prozess zur beispielgetriebenen Modellierung 53
5.8 Modellierungsfehler in einer Asset Expression 55
5.9 Analysemuster als Trait formuliert 56
5.10 Ähnlichkeit zwischen Asset Expressions und Mindmaps 57
5.11 Beziehung zwischen Instanzen und Klassen in der BGM 59
5.12 Inkrementelle Modellierung und Prototypen nach [Bossung07] 60

6.1 Werkbankunterstützung der linguistischen Analyse 65
6.2 Navigation durch Cluster von Instanzen 69
6.3 Modell einer vollständigen Asset Expression 72
6.4 Modellierung eines Clustermittelpunktes für Asset Expressions 73
6.5 Metamodell der Assetdefinitionssprache 73
6.6 Distanz zwischen semantischen Typen 76
6.7 Visualisierung der Cluster in der Werkbank 83

Tabellenverzeichnis

3.1 Inhaltsarten (*content kinds*) für Selektoren 14
3.2 Beispiel: semantisch getypte Asset Expression 15

4.1 Ansätze zur Domänenmodellierung 21
4.2 Konzeptuelle Klassenkategorien nach [Larman05] 25

6.1 Kosten der elementaren Operationen für die strukturelle Distanz 78

Abkürzungen

	Begriff	**siehe**	
AE	Asset Expression	Kapitel	3
BGM	beispielgetriebene Modellierung	Abschnitt	5.2.3
CC	Cluster Center	Abschnitt	6.2.2
DE	Domänenexperte	Kapitel	1
DSL	domain specific language	Abschnitt	5.2.5
OOA	objektorientierte Analyse	Abschnitt	4.2

Kapitel 1

Einleitung

Die Entwicklung eines komplexen Informationssystems ist eine Herausforderung – auch wenn die Technologie beherrscht wird. Ein Grund dafür ist, dass Software ein immaterielles Gut ist, dessen Beschreibung sich für fachfremde Personen schwierig gestaltet. Das führt zu dem Dilemma des Softwareengineering: Die Anwender wissen, was für ein System sie benötigen, können es aber nicht beschreiben. Im Gegensatz dazu können Systemanalytiker und Entwickler das System beschreiben, wissen aber nicht, was die Anwender brauchen.

In der Praxis wird dieses Problem gelöst, indem sich Systemanalytiker und Entwickler in die Domäne der Anwender einarbeiten. Dieses Vorgehen hat sich vielfach bewährt und ist wirtschaftlich, weil der Aufwand für die Einarbeitung in Relation zu der gesamten Systementwicklung klein ist.

Konzeptorientierte Inhaltsverwaltung

Die konzeptorientierte Inhaltsverwaltung (*concept-oriented content management*, CCM) verfolgt einen anderen Ansatz als das klassische Softwareengineering: Die Anwender eines Inhaltsverwaltungssystems sollen in der Lage sein, es jederzeit an die sich ändernden kollektiven und individuellen Informationsbedürfnisse anzupassen. Diese Vision wird durch eine Familie generierter Inhaltsverwaltungssysteme realisiert, die auf konzeptuellen Modellen der Anwendungsdomäne basieren. Die konzeptorientierten Inhaltsverwaltungssysteme unterstützen daher Offenheit und Dynamik: Sie sind offen gegenüber Änderungen an den zugrundeliegenden Konzepten und dynamisch, da sie auf diese Änderungen selbstständig reagieren.

Damit die Anwender die Offenheit und Dynamik ausnutzen können, müssen sie in der Lage sein, konzeptuelle Modelle zu erstellen und zu modifizieren. Eine Aufgabe, die im Softwareengineering von Modellierungsexperten durchgeführt wird.

Konzeptuelle Modellierung durch Domänenexperten

Um die Vision der konzeptorientierten Inhaltsverwaltung zu erfüllen, müssen die Anwender selbst der konzeptuellen Modellierung mächtig sein. Daraus folgt das Ziel dieser Arbeit: Es soll Domänenexperten, die gleichzeitig Anwender der konzeptorientierten Inhaltsverwaltung sind, ermöglicht werden, eigenständig konzeptuelle Schemata von hoher Qualität zu erstellen – sie sollen sich für die Systemgenerierung eignen.

Praktische Erfahrungen mit der konzeptorientierten Inhaltsverwaltung legen nahe, die Modellierung auf Grundlage von Beispielen durchzuführen. Die Domänenexperten sollen zuerst Beispieldaten formulieren, von denen ein konzeptuelles Modell abgeleitet wird. Durch die Berücksichtigung mehrerer Instanzen wird es den Domänenexperten erleichtert, von einzelnen Entitäten ihrer Domäne zu abstrahieren. Gleichzeitig belegen die Beispiele die Zweckmäßigkeit des Modells.

Das Resultat dieser Arbeit ist ein Modellierungsprozess und das Konzept einer Werkbank, welche die Domänenexperten bei dieser anspruchsvollen Aufgabe unterstützen. Die Werkbank ist mit einer Entwicklungsumgebung vergleichbar: Sie umfasst verschiedene Anwendungen, die den Domänenexperten die Modellierung so unabhängig wie möglich durchführen lassen.

Überblick

Zunächst werden in Kapitel 2 die konzeptorientierten Inhaltsverwaltungssysteme vorgestellt, die Anlass und Anwendung für diese Arbeit sind. Anschließend folgt in Kapitel 3 eine Beschreibung von *Asset Expressions*, einer Sprache zur semistrukturierten Beschreibung einzelner Entitäten durch mediale Datentypen. Diese Sprache wird für die Modellierung der Beispielinstanzen verwendet.

Für die Bearbeitung der Aufgabe wurde ein methodologischer Ansatz gewählt: Modellierungsmethoden aus verschiedenen Teildisziplinen der angewandten Informatik werden in Hinblick auf bewährte Ansätze und Verfahren untersucht (Kapitel 4). Dabei liegt das Interesse auf solchen Ansätzen, die sich auch für die Modellierung durch Domänenexperten eignen.

Auf Grundlage bewährter Verfahren aus verschiedenen Modellierungsmethoden wird in Kapitel 5 der *Prozess zur beispielgetriebenen Modellierung* entworfen. Dieser Prozess ermöglicht es den Domänenexperten, selbstständig konzeptuelle Modelle hoher Qualität zu erstellen, so dass es ihnen ermöglicht wird, die Offenheit und Dynamik der konzeptorientierten Inhaltsverwaltung auszunutzen.

Um die Domänenexperten bei der Modellierung zu unterstützen, ist eine *Werkbank* vorgesehen, die auf den Modellierungsprozess zugeschnitten ist (Kapitel 6). Die wichtigste Anwendung der Werkbank ist ein Werkzeug, das in einem interaktiven Prozess konzeptuelle Modelle aus den Beispielinstanzen inferiert. Für dieses Kernproblem wird ein neuartiger Algorithmus für die *interaktive Schemainferenz durch Clustering* entworfen, der sich durch Vorteile bei umfangreichen Beispielmengen und ein leichtverständliches Interaktionsschema auszeichnet.

Es folgt in Kapitel 7 eine Zusammenfassung der Ergebnisse, die auch eine Diskussion verwandter Ansätze beinhaltet. Die Arbeit wird mit einem Ausblick auf künftige Entwicklungen abgeschlossen.

Kapitel 2

Konzeptorientierte Inhaltsverwaltung (CCM)

Die Arbeit an der konzeptorientierte Inhaltsverwaltung stellt den Grund und die Anwendung für diese Arbeit dar. Bei dieser Form der Inhaltsverwaltung werden die Anwender in ihren Erkenntnisprozessen dadurch unterstützt, dass sich das System jederzeit an neue Umstände oder persönliche Präferenzen anpassen lässt.

2.1 Einführung

Die Systeme zur konzeptorientierten Inhaltsverwaltung werden auch CCM (*concept-oriented content management*) Systeme genannt. Bei diesen handelt es sich nicht um angepasste generische Systeme, wie sie in der Inhaltsverwaltung (*content management*) üblich sind, sondern um eine Klasse generierter Inhaltsverwaltungssysteme, die auf der Grundlage eines konzeptuellen Modells erzeugt werden. Die folgende Beschreibung der konzeptorientierten Inhaltsverwaltung beruht auf der gleichnamigen Dissertation [Sehring04] und orientiert sich in Umfang und Struktur an einer vorigen Studienarbeit zu diesem Thema [Carl06].

Die Idee, Inhaltsverwaltungssysteme aus einem Modell zu generieren, stammt aus interdisziplinären Projekten wie der Warburg Electronic Library (siehe [Sehring04, SSW02]). Die monotonen und sich wiederholenden Vorgänge bei der Programmierung gängiger Inhaltsverwaltungssysteme legen nahe, dass sich diese Anwendungen gut für eine automatische Generierung eignen. Die konzeptorientierte Inhaltsverwaltung ist daher den Methoden zur modellgetriebenen Softwareentwicklung zuzuordnen, wie auch die zur Zeit vieldiskutierten Verfahren zur MDA (*model driven architecture*) und Domänenentwicklung (*domain engineering*, siehe Abschnitt 5.2.5). Der wesentliche Unterschied zu diesen Verfahren ist, dass diese im Allgemeinen von einem Softwaremodell ausgehen, die CCM Systeme jedoch aus einem Modell der Anwendungsdomäne (*domain model*) erzeugt werden.

Eine weitere Erfahrung aus der WEL ist die Erkenntnis, dass sich die Sicht der Anwender auf ihre Anwendungsdomäne mit zunehmenden Erkenntnisstand ändert, wodurch es notwendig wird, die verwendeten Inhaltsverwaltungssysteme laufend anzupassen. Die wichtigsten Forderungen an die CCM Systeme sind deshalb *Offenheit* und *Dynamik*: Of-

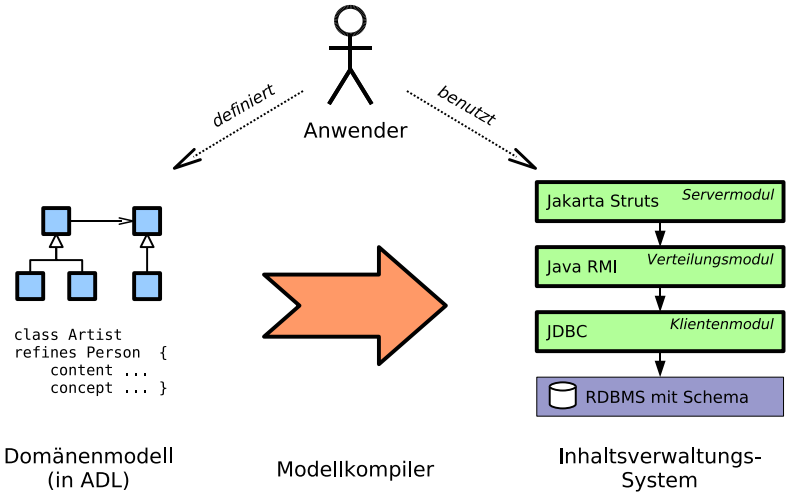

Abbildung 2.1: Konzeptorientierte Inhaltsverwaltung: Szenario

fenheit gegenüber Veränderungen an den ihnen zugrundeliegenden Konzepten (Modellen) und die Dynamik, auf die Konzeptänderungen selbstständig zu reagieren. Dieses steht im Einklang mit den offenen und dynamischen Erkenntnisprozessen, die der Wissenschaftsphilosoph Ernst Cassirer beschreibt [Sehring04, Kap. 1].

Die Vision sind Inhaltsverwaltungssysteme, die von ihren Anwendern jederzeit um neue Konzepte erweitert und personalisiert werden können. Die Anwender selbst optimieren sich ihre Systeme und passen diese an ihre persönlichen Bedürfnisse an. Dies wird in Bild 2.1 veranschaulicht.

2.2 Assetdefinitionssprache

2.2.1 Intensionale Klassendefinition

Die Domänenmodelle für die Systemgenerierung beschreibt man in einer dem Anwender angepassten Sprache, der Assetdefinitionssprache (ADL, *asset definition language*). Assets stellen die zentrale Informationsabstraktion von CCM dar: Sie verweisen auf Entitäten der Anwendungsdomäne durch einen Inhalt (z. B. eine mediale Repräsentation) und eine konzeptionelle Beschreibung der Entität.

Die Struktur von Assets wird durch die Definition von *Assetklassen* festgelegt, vergleichbar mit Objekten und Klassen in der objektorientierten Modellierung. Assetklassen beschreiben Mengen von Assets auf intensionale Weise, d.h. indem die strukturellen Merkmale der Instanzen beschrieben werden. Diese Art der Konzeptdefinition wird neben in den objektorientierten Methoden auch von allen Ontologiesprachen und in der Mathematik verwendet. Die Menge der positiven geraden Zahlen kann z. B. geschrieben werden als:

$$G = \{n \in \mathbb{N} | \exists m \in \mathbb{N} : n = 2 \cdot m\}$$

Als Beispiel soll das konzeptuelle Modell aus Abbildung 2.2 in der ADL formuliert werden. Dabei ist zu beachten, dass in der UML im Gegensatz zur ADL keine Unterschei-

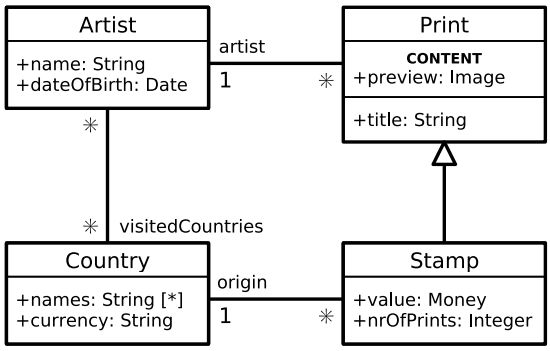

Abbildung 2.2: Beispiel: Klassendiagramm einer Anwendungsdomäne

dung zwischen Inhalt und Konzept vorgenommen wird. Diesem Umstand wird hier durch die Einführung eines gesonderten Abschnittes `content` im Klassendiagramm Rechnung getragen.

```
class Print {
    content preview : java.awt.Image
    concept characteristic title : java.lang.String
            relationship artist : Artist
}
class Stamp refines Print {
    concept characteristic value : junit.samples.money.Money
            characteristic nrOfPrints : java.lang.Integer
            relationship origin : Country
}
```

Jede Assetklasse besitzt je eine Inhalts- und eine Konzeptsektion, die auch leer sein dürfen. Auffällig ist, dass in der Konzeptsektion des Beispiels zwei Arten von Attributen auftreten: Ein Charakteristikum (`characteristic`) ist eine immanente Eigenschaft eines Assets, die untrennbar mit ihm verbunden ist und keine eigene Identität besitzt. Im Gegensatz dazu verweist eine Beziehung (`relationship`) auf ein anderes Asset, welches eine eigenständige Entität beschreibt und deshalb eine eigene Identität besitzt.

Als Typen für die Inhalte und Charakteristika stehen Klassen aus der Implementierungssprache der generierten Systeme zur Verfügung, in diesem Fall von Java. Die Verwendung einer eingebetteten Sprache hat den Vorteil, dass man keiner Beschränkungen von Basistypen unterliegt, wie sie in den meisten generischen Inhaltsverwaltungssystemen vorhanden ist. Im Gegensatz dazu stehen für Beziehungen nur die Assetklassen als Typen zur Verfügung. Darüber hinaus stehen den Anwendern mengenwertige Beziehungstypen zur Verfügung:

```
class Artist {
    concept characteristic name : java.lang.String
            characteristic dateOfBirth : java.lang.String
            relationship visitedCountries : Country*
}
```

```
class Country {
    concept characteristic name : java.lang.String*
            characteristic currency : java.lang.String
}
```

Dieses Beispiel zeigt wie mengenwertige Beziehungen durch einen `*` gekennzeichnet werden. In dieser Arbeit werden auch mengenwertige Inhalte und Charakteristika zugelassen, obwohl diese Verwendung von den ursprünglichen Arbeiten zur konzeptorientierten Inhaltsverwaltung abweicht [Sehring04, Abs. 3.3.1]. Diese Erweiterung der ADL ist jedoch sinnvoll und zugleich trivial zu realisieren: Das mengenwertige Charakteristikum im Beispiel kann z. B. in einen Kollektionstyp der eingebetteten Sprache umgewandelt werden `java.util.Set<String>`.

Die ADL ist noch deutlich umfangreicher, als hier dargestellt wurde. Vor allem wurden Bedingungen (*constraints*) ausgelassen, die den Wertebereich von Attributen durch logische Ausdrücke einschränken. Diese werden jedoch in dieser Arbeit nicht benötigt.

2.2.2 Extensionale Klassendefinition

Neben der statisch intensionalen Klassendefinition, die im vorigen Abschnitt erläutert wurde, ist in der ADL auch eine Möglichkeit zur extensionales Definition von Klassen vorgesehen [Sehring04, Abs. 3.4.2]. Eine extensionale Mengenbeschreibung definiert eine Menge, indem die Elemente aufgezählt werden. Dieses Verfahren ist auch in der Mathematik gebräuchlich:

$$G^{10} = \{2, 4, 6, 8, 10\}$$

In der Assetdefinitionssprache sind zwei extensionale Klassendefinitionsarten vorgesehen. Die statisch extensionale Definition entspricht dem vorhergehenden Beispiel der positiven geraden Zahlen kleiner gleich zehn. Eine Aufzählung sämtlicher Assetinstanzen, die der Klasse angehören:

```
class DayOfWeek monday, tuesday, wednesday, thursday,
                friday, saturday, sunday
```

Da dieses Vorgehen unflexibel ist, gibt es die dynamisch extensionale Definition, durch die eine Klasse durch Aufzählen von typischen Vertretern der Klasse definiert wird. Ein Asset wird dieser Klasse zugeordnet, wenn es hinreichend ähnlich zu den aufgezählten Beispielen ist. Dazu muss ein formales Ähnlichkeitsmaß definiert werden, wie es z. B. durch Verfahren aus der Mustererkennung realisiert werden kann.

```
class Portrait ~ monalisa, kahloportrait, passphoto
```

Der in Abschnitt 5.3 vorgestellte Prozess zur beispielgetriebenen Modellierung greift diesen Ansatz auf, indem Konzepte durch die Formulierung von Beispielen extensional definiert werden. Da nicht sämtliche Instanzen einer Klasse aufgezählt werden können, muss die Menge der Beispiele als ein dynamisch extensionales Klassenmodell interpretiert werden. Die Werkbank, die den Prozess unterstützt, interpretiert dieses dynamisch extensionale Modell und wandelt es in statisch intensionale Assetklassen um.

2.3 Realisierung

2.3.1 Aufbau eines generierten Systems

Bevor die Systemgenerierung erläutert wird, soll ein kurzer Ausblick auf die Struktur der generierten Systeme gegeben werden. Ein CCM System setzt sich aus Modulen zusammen, wobei durch jedes Modul nur ein Teil der Gesamtfunktionen des Systems realisiert wird. Die Modularten werden anhand des Beispielsystems aus Abbildung 2.1 erläutert, weitere Details finden sich in [SBS05].

- Das Servermodul (*server module*) ist für die Präsentation der Daten in einer webbasierten Oberfläche zuständig. In diesem Fall wird das Java basierte Struts Framework [Husted02] verwendet.
- Das Distributionsmodul (*distribution module*) ist für die räumliche Verteilung der Anwendung zuständig. Es verwendet RMI (*Java remote method invocation*), um die Datenhaltung und Präsentation auf verschiedene Rechner zu verteilen.
- Das Klientenmodul (*client module*) dient dem Zugriff auf eine Standardkomponente, hier eine relationale Datenbank. Seine Aufgabe ist es, das Datenmodell der Standardkomponente zu kapseln.
- Die relationale Datenbank ist eine Standardkomponente und kein CCM Modul. In Abbildung 2.1 ist sie deshalb in anderer Schattierung dargestellt, weil sie von dem Kompiler durch ein modellspezifisches Datenbankschema konfiguriert – aber nicht wie die Module generiert – wird.

Die Module kommunizieren über die einheitliche Modulschnittstelle (*module API*) miteinander, dabei handelt es sich um eine Repräsentation des Domänenmodells in Form von Java Schnittstellen. Über die einheitliche Modulschnittstelle ist es möglich, auf die gespeicherten Inhalte zuzugreifen und diese transaktional zu verändern.

2.3.2 Realisierung von Offenheit und Dynamik

Um Offenheit und Dynamik für den Anwender zu realisieren, muss die konzeptorientierte Inhaltsverwaltung auf technischer Ebene zwei Operationen unterstützen: Schemaevolution und Personalisierung. Bei einer Schemaevolution wird ein CCM System auf ein neues Schema umgestellt. Damit die Daten des Bestandssystems weiter verwendet werden können, müssen sie in das neue Schema überführt werden. Dies geschieht durch spezialisierte Module zur Laufzeit des neuen Systems.

Auf gleiche Weise wird auch die Personalisierung, die Spezialisierung des Schemas für einen speziellen Anwenderkreis, realisiert. In diesem Fall müssen die Daten des personalisierten Systems auch in das Bestandssystem zurückgeführt werden, damit diese auch anderen Anwenderkreisen zur Verfügung stehen. Daher kann die Evolution auch als Sonderfall einer Personalisierung angesehen werden. Abbildung 2.3 zeigt eine Modulkonfiguration zur Personalisierung. Dabei wird ein Bestandssystem mit dem Schema M als Basis für ein personalisiertes System mit dem modifizierten Schema M' gezeigt:

- Abbildungsmodule (*mapping modules*) dienen dazu, die Assets des Schemas M in das neue Schema M' zu überführen. In dem Beispiel greift das Modul auf eine

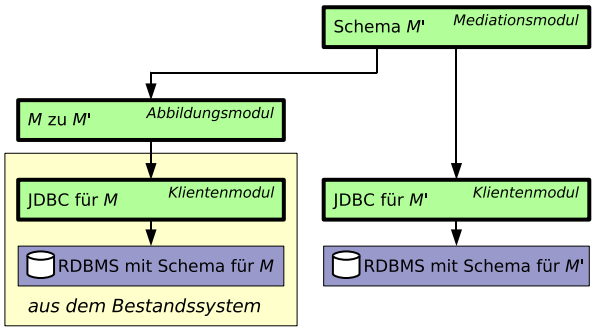

Abbildung 2.3: Modulkonfiguration zur Personalisierung nach [SBS05]

Datenbasis zu, die über die allgemeine Modulschnittstelle des Schemas M verfügt, stellt selbst aber die Modulschnittstelle des Schemas M' bereit.

- Mediationsmodule (*mediation modules*) koordinieren den Zugriff auf die zwei Datenbestände. Im Fall einer Evolution werden lesende Zugriffe auf beiden untergeordneten Modulen durchgeführt, Schreibzugriffe finden allerdings nur auf dem neuen Klientenmodul statt.

2.3.3 Modellkompiler

Der CCM Modellkompiler ist wie die meisten Kompiler zweistufig aufgebaut: Das Frontend liest die Assetdefinitionen, überprüft sie auf Korrektheit und erzeugt aus ihnen ein Zwischenmodell (*intermediate model*). Daraus generiert das Backend den Quellkode eines Inhaltsverwaltungssystems. Die Besonderheit an dem CCM Modellkompiler ist, dass das Backend modular aufgebaut ist und durch *Generatoren* ergänzt wird. Generell sind mehrere Generatoren an der Erzeugung eines Systems beteiligt. Je nach Auswahl der verwendeten Generatoren können verschiedene Arten von Systemen erzeugt werden.

Das Backend ist als Framework realisiert: Der Kompiler bestimmt selbst, in welcher Reihenfolge die einzelnen Generatoren ausgeführt werden. Um seine Aufgabe wahrzunehmen, kann ein Generator neben dem Zwischenmodell noch Informationen von anderen Generatoren benötigen, die durch *Symboltabellen* übermittelt werden. Jeder Generator erzeugt eine Symboltabelle und kann von beliebig vielen anderen abhängen. Anhand der Symboltabellen, die von den Generatoren benötigt werden, kann der Kompiler eine Ausführungsreihenfolge festgelegen. Die eigentliche Aufgabe der Generatoren – die Erzeugung von Quellcode – geschieht unabhängig von dem Austausch der Symboltabellen als Seiteneffekt.

Das Aktivitätsdiagamm in Abbildung 2.4 veranschaulicht die Vorgänge im Modellkompiler durch ein Beispiel mit zwei Generatoren: Ein Generator erzeugt SQL Schemata, dazu benötigt er nur das Zwischenmodell. Der andere generiert JDBC (*Java Database Connectivity*) Code, der von dem Datenbankschema abhängt - dieses wird ihm über die Symboltabelle `schemaSymbolTabelle` zur Verfügung gestellt. Es ist offensichtlich, dass der Schemagenerator zuerst ausgeführt werden muss. Zu dem Diagramm in Abbildung 2.4 sei angemerkt, dass es eine kombinierte Darstellung von Kontroll- und Objektfluss zeigt, wie sie in der UML 2 möglich ist. Aktionen mit mehreren eingehenden Kanten

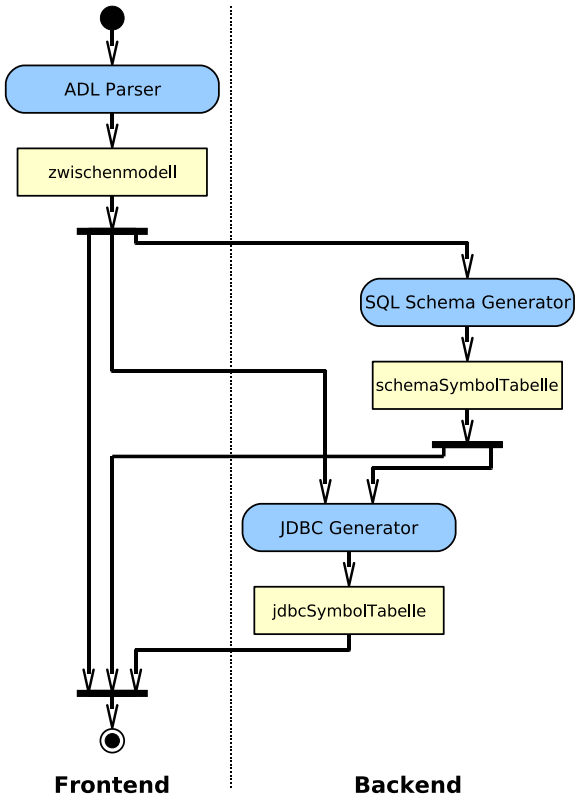

Abbildung 2.4: Abläufe im Modellkompiler aus [Carl06]

vollziehen im Unterschied zu früheren Versionen eine implizite Synchronisation (*join*), siehe [Oestereich06].

Durch die Auswahl der verfügbaren Generatoren, die in einer Konfigurationsdatei aufgezählt werden, kann der Modellkompiler zur Generierung verschiedenartiger Systeme verwendet werden. So können durch Generatoren für Klientenmodule Systeme generiert werden, die ihre Daten in einem relationale Datenbanksystem oder direkt im Dateisystem speichern.

2.4 Erfahrungen

Der CCM Modellkompiler wird bei dem interdisziplinären Forschungsprojekt zur Geschichte der Kunstgeschichte im Nationalsozialismus (GKNS, [GKNS]) eingesetzt. Im Laufe des Projektes hat sich gezeigt, dass die technische Seite der konzeptorientierten Inhaltsverwaltung beherrscht wird. Der Modellkompiler und die Generatoren sind hinreichend ausgereift. Einschränkungen sind lediglich bei den Generatoren für die Abbildungsmodule vorhanden, da eine Abbildung zwischen beliebigen Schemata nicht ohne weiteres automatisiert erfolgen kann [Chayanam06]. Forschungsbedarf besteht darüber hinaus bei Benutzeroberflächen, da diese zur Zeit von projektspezifischen Generatoren erzeugt werden. Ein Ansatz zur Lösung dieses Problems durch allgemein verwendbare Generatoren wurde bereits untersucht [Mofor06].

Während der Realisierung des Inhaltsverwaltungssystems traten allerdings Probleme bei der Modellierung auf. Die Anwender, ihrerseits Experten auf dem Gebiet der Kunstgeschichte, hatten Schwierigkeiten bei dem Erstellen des konzeptuellen Modells. Und dies obwohl sie während der Modellierung durch Modellierungsexperten unterstützt worden sind und eine Delphi-Methode (siehe auch Verfahren von Holsapple und Joshi Abs. 4.4.1) angewendet wurde. Bei der Modellierung wurden folgende Beobachtungen gemacht [Bossung07, Abs. 7.2.1]:

- Änderungen von Klassen des Modells wurden meistens aufgrund einzelner Instanzen von den Domänenexperten vorgeschlagen. Dabei wurde für diese eine Entität der Domäne ein optimales Modell angestrebt. Es wurde selten bedacht, ob diese Modellierung auch für weitere Entitäten sinnvoll ist. Die vorgeschlagenen Klassen waren nicht allgemein verwendbar.

- Die Änderungsvorschläge entstanden auf Grundlage der Meinungen einzelner Domänenexperten. Eine absolute Offenheit und Dynamik hätte für jeden Vorschlag ein personalisiertes System hervorgebracht. Die Abbildung der Daten zwischen sämtlichen Schemata wäre schnell unmöglich geworden. Daraus folgt, dass die Offenheit und Dynamik nicht absolut eingesetzt werden darf, sondern einer Kontrolle bedarf.

Diese praktischen Erfahrungen geben den Anlass zu dieser Arbeit. Es besteht Bedarf an einer geeigneten Modellierungsmethode, welche es den Anwendern bzw. Domänenexperten ermöglicht, konzeptuelle Modelle hoher Qualität zu erstellen. Nur wenn die Domänenexperten selber die Modellierung vornehmen, werden sie in den Genuss der Vorteile der konzeptorientierten Inhaltsverwaltung – der Offenheit und Dynamik – kommen.

Kapitel 3

Asset Expressions

Die Sprache der Asset Expressions ermöglicht es, Entitäten durch mediale Inhalte und konzeptuelle Beschreibungen zu modellieren. Dabei kann jede Entität durch eine individuell angepasste Struktur beschrieben werden. Asset Expressions sind für den Prozess zur beispielgetriebenen Modellierung von besonderer Bedeutung.

3.1 Grundlagen

3.1.1 Einführung

Wie die Assets in der konzeptorientierten Inhaltsverwaltung repräsentieren auch Asset Expressions (AEs) Entitäten der realen Welt, die durch Inhalte und konzeptuelle Beschreibungen dargestellt werden. Asset Expressions werden in der Dissertation *Conceptual Content Modeling* ausführlich beschrieben. Die Ausführungen in diesem Kapitel beschränken sich auf das, was zum Verständnis dieser Arbeit notwendig ist. Für Details wird daher auf die Quelle [Bossung07] verwiesen.

Das Ziel einer Asset Expression, die um einen medialen Inhalt aufgebaut wird, ist es, den Inhalt durch Randinformationen zu erläutern. Inhalte werden generell von Menschen interpretiert. Diese Interpretation ist von dem individuellen Wissen und Erfahrungsschatz des Betrachters abhängig, daher wird derselbe Inhalt von verschiedenen Personen unterschiedlich interpretiert. Darüber hinaus gibt es Inhalte, die ohne Zusatzinformationen überhaupt nicht interpretiert und verstanden werden können.

Damit Inhalte zuverlässig interpretiert werden können, werden sie in den AEs in einen thematischen Kontext gesetzt. Durch einen hinreichenden thematischen Kontext können unterschiedliche Interpretationen vermieden werden. Der Kontext wird in zwei Schritten hergestellt: Zuerst wird der Bedarf an Kontextinformationen festgestellt, indem interpretationsabhängige Aspekte des Inhalts identifiziert werden. Anschließend werden diese Unklarheiten beseitigt, indem der Kontext zu anderen Entitäten hergestellt wird.

3.1.2 Syntax

Die Syntax der Asset Expressions (AEs) lehnt sich an den Lambdakalkül an. Die Ausdrücke werden auf gleiche Art aus Literalen (hier: Inhalte), Variablen sowie Operatoren

zur Abstraktion und Applikation aufgebaut. Die Syntax der AEs wird anhand eines einfachen Beispiels erklärt:

ReichstagVictory := (λ*photographer*. )Khaldei

Inhalte: Inhalte entsprechen den Literalen im Lambdakalkül. Im Fall der AEs sind jedoch vor allem mediale Inhalte von Interesse, die als Ganzes im Ausdruck dargestellt werden, wie es das Photo vom Reichstag im Beispiel demonstriert. Grundsätzlich sind Inhalte in allen denkbaren Formen gestattet wie Grafiken, Audioaufnahmen, Filme, Texte und Hypermediadokumente. Das verwendete Medium setzt jedoch Grenzen, wie in diesem Fall das Papier.

Abstraktion: In den AEs werden Abstraktionen in der Form $\lambda var.expr$ dargestellt und dadurch eine neue Variable (*photographer*) eingeführt.

Applikation: Für Applikationen wird die Syntax von Revesz [Revesz88] verwendet. Durch diese Schreibweise $(operator)operand$ wird die Lesbarkeit bei der Darstellung von medialen Inhalten erhöht. Durch die Applikation wird die freie Variable einer vorigen Abstraktion an eine weitere AE gebunden, im Beispiel ist dies eine benannte Referenz Khaldei.

Benannte Ausdrücke: In dem Beispiel wird der Name *ReichstagVictory* durch den Operator := an den dargestellten Ausdruck gebunden.

Benannte Referenzen: Benannte Ausdrücke können über Referenzen in anderen Ausdrücken verwendet werden (Khaldei). Allerdings muss jeder verwendete Name zuvor an einen Ausdruck gebunden werden. Die rekursive Verwendung von Referenzen ist unzulässig.

Listen: Neben dem bisher gezeigten gibt es auch eine Notation für Listen von Ausdrücken, die ihrerseits selbst auch wieder AEs sind: $\{e_1, \ldots, e_n\}$. Durch die Listen können z. B. mehrere Inhalte in einem Ausdruck verwendet werden, oder in einer Applikation mehrere Ausdrücke an eine freie Variable gebunden werden.

Entscheidende Elemente der AEs sind die Abstraktionen und Applikationen, durch die der Kontext hergestellt wird. Eine Abstraktion identifiziert einen Aspekts des Inhalts, der einer näheren Erläuterung bedarf. Aus einer alleinstehenden Abstraktion kann lediglich geschlossen werden, dass zum Verständnis kontextuelle Informationen nötig sind, die in dem Inhalt alleine nicht offensichtlich sind. Durch eine Applikation wird dieser Aspekt erläutert und mit dem Kontext in Beziehung gesetzt.

Die Semantik einer AE ergibt sich aus der Interpretation ihres Inhaltes, die im dargestelltem Kontext stattfindet. Eine solche Interpretation einer AE – sie ergibt die Vorstellung einer realen Entität – bleibt dem Anwender vorbehalten. Symbolische Reduktionsvorgänge, wie sie im Lambdakalkül üblich sind [Hankin94], können nicht angewendet werden.

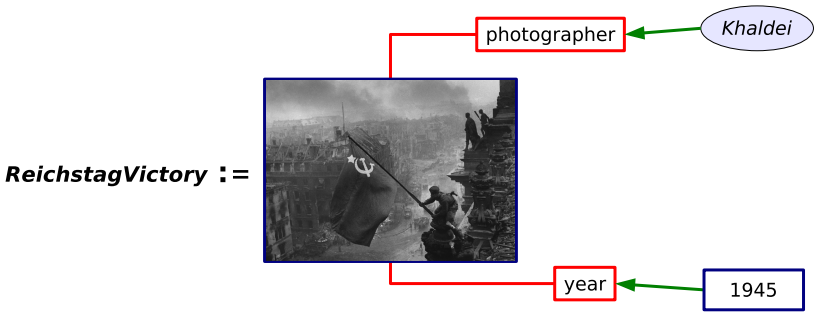

Abbildung 3.1: Grafische Repräsentation einer Asset Expression

3.1.3 Grafische Repräsentation

Neben der bisher vorgestellten Syntax für Asset Expressions, die auf dem Lambdakalkül basiert, ist noch eine grafische Repräsentation für die Ausdrücke vorgesehen. Sie wird vor allem dort verwendet, wo ihre illustrativen Qualitäten erwünscht sind und in Fällen, wo der Formalismus des Lambdakalküls hinderlich ist.

In Abbildung 3.1 wird das zuvor erläuterte Beispiel erweitert. Die Syntax der grafischen Repräsentation wird bei dem Vergleich mit dem vorigen Beispiel offensichtlich: Inhalte und Variablennamen werden in verschieden schattierten Rechtecken, Abstraktionen mit Linien, benannte Referenzen durch Ovale und Applikationen durch Pfeile dargestellt.

Für den Modellierungsprozess in dieser Arbeit wird die grafische Darstellung der Ausdrücke verwendet, da durch sie eine intuitive Modellierung von Beispielen möglich wird. Dies wird deutlich, wenn man bedenkt, dass die Modellierung von den Domänenexperten selbst durchgeführt wird. Es kann davon ausgegangen werden, dass die Domänenexperten mit der formalen Form der AEs überfordert sind, da diese im Allgemeinen über kein Informatik-Fachwissen verfügen.

3.2 Komponenten und Selektoren

Gerade bei medialen Inhalten kann es vorkommen, dass sich eine Erläuterung nur auf einen Teil des Inhaltes bezieht. Möchte man zu einem Bild, das mehrere Menschen zeigt, den Namen einer bestimmten Person hinzufügen, trifft man auf ein Problem. Der Name lässt sich nur dem Bild zuorden, aber keinem einzelnen der dargestellten Menschen. Solche AEs sind nicht eindeutig interpretierbar, obwohl der Versuch unternommen wurde, sie durch einen Kontext zu ergänzen.

Aus diesem Beispiel wird die Notwendigkeit deutlich, dass die Abstraktionen der AEs bestimmten Teilen des Inhaltes zugeordnet werden können. Solche Teile des Inhaltes einer Asset Expression werden als Komponenten (*components*) bezeichnet. Durch sie wird eine eindeutige Interpretation des Ausdruckes durch den Betrachter in vielen Fällen erst ermöglicht. Eine genauere Diskussion dieser Zusammenhänge findet sich in [Bossung07, Abs. 3.2].

Die meisten technischen Datentypen für mediale Inhalte unterstützen keine Identifikation von Teilen. Um keine speziellen Datenformate für sämtliche Inhalte einzuführen,

Kategorie	Inhaltsarten
linear	Text, Audio
Baum	XML
2-dimensional	Bitmap, Vektorgrafik
3-dimensional	Video, CAD-Modell

Tabelle 3.1: Inhaltsarten (*content kinds*) für Selektoren

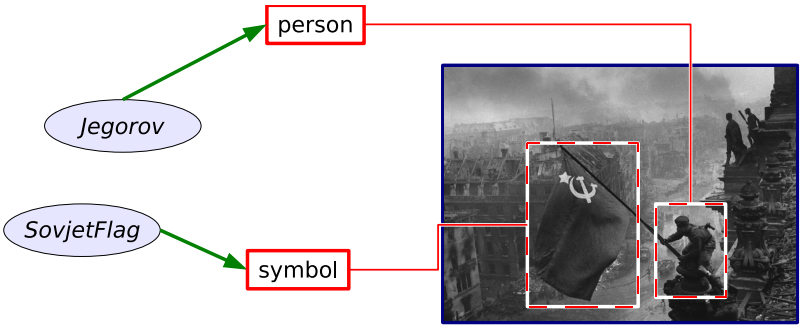

Abbildung 3.2: Grafische Asset Expression mit Komponenten

wird dieser Mangel durch Selektoren (*selectors*) behoben. Durch sie kann ein Teil des Inhaltes adressiert werden, ohne die üblichen Datenformate anzutasten.

Die Selektoren sind abhängig von der Inhaltsart (*content kind*), nicht aber unmittelbar von dem verwendeten Datenformat. Mehrere Datenformate werden zu einer Inhaltsart zusammengefasst, z. B. ist `Bitmap` eine Inhaltsart, welche die Datenformate JPEG, GIF und TIFF umfasst. Eine Übersicht der Inhaltsarten zeigt Abbildung 3.1.

Es existiert eine Syntax für Selektoren und ihre Anwendung in der Lambdasyntax der Asset Expressions [Bossung07, Abs. 3.2.2]. Da sie aufgrund ihrer Komplexität nicht für Domänenexperten geeignet ist, soll hier stattdessen ein Beispiel für ihre grafische Repräsentation gegeben werden, siehe Abbildung 3.2. Es wird deutlich, wie durch die Selektion einer Person im Vordergrund die Interpretierbarkeit des Bildes unterstützt wird. Bei vielen Inhaltsarten ist eine Werkzeugunterstützung bei der Erstellung von Komponenten notwendig. Dies gilt vor allem für komplexe Inhaltsarten wie für CAD-Modelle oder Videosequenzen. Zu diesem Zweck existiert ein grafischer Editor für Asset Expressions, der einen unkomplizierten Umgang mit der grafischen Notation erlaubt.

3.3 Typisierte Asset Expressions

3.3.1 Semantische Typen

Das Typsystem der Asset Expressions basiert auf semantischen Typen (*semantic types*) [Bossung07, Abs. 3.3.1]. Diese Typen unterscheiden sich von gängigen Typsystemen, da sie weder auf die Struktur noch auf die technische Repräsentation der Ausdrücke einen Einfluss haben. AEs werden vom Anwender interpretiert, wobei das Ergebnis dieser Interpretation die Vorstellung von einer Entität der realen Welt ist. Ein wichtiger Aspekt

	Asset Expression	Typ
1.	 : ReiterStatue	ReiterStatue
2.	λ*reiter*: Herrscher.  : ReiterStatue	Herrscher → ReiterStatue
3.	(λ*reiter*: Herrscher.  : ReiterStatue)Napoleon	ReiterStatue

Randbedingung: Napoleon: Herrscher

Tabelle 3.2: Beispiel: semantisch getypte Asset Expression

dieser Typen ist es, dem Anwender bei dieser Interpretation zu helfen, indem sie den Typ der Entität wiederspiegeln.

Aus formaler Sicht handelt es sich bei den semantischen Typen um eine Menge von Konzepten, von denen lediglich ihr Name und Namensraum bekannt sind. Die Typen bilden eine Taxonomie, wobei jeder Typ exakt einen Obertyp besitzt. Ausnahme ist der triviale Typ Any, der als Wurzel der Taxonomie von keinem Typen abgeleitet ist. Die Obertypbeziehung wird durch den Operator :> dargestellt:

Any :> Person

Die semantischen Typen spiegeln zwar Domänenkonzepte wieder, allerdings beeinflussen sie weder die Struktur, noch die technische Repräsentation der durch sie getypten Ausdrücke. Eine AE vom Typ Person kann aus beliebigen Inhalten bestehen und die Struktur des Ausdrucks kann frei gewählt werden. Trotzdem hilft der Typ dem Anwender bei der Interpretation des Ausdrucks, was anhand von Beispielen im nächsten Abschnitt gezeigt wird. Asset Expressions können sich in ihrem semantischem Typ und in ihrer Struktur unterscheiden, diese Dimensionen sind jedoch im Gegensatz zu üblichen Modellierungssprachen orthogonal zueinander.

3.3.2 Typsystem

Das Typsystem der Asset Expressions besitzt die semantischen Typen als Grundtypen. Darüber hinaus gibt es Funktionstypen, wie sie in getypten Lambdakalkülen [Hankin94, Kap. 7] und funktionalen Programmiersprachen (z. B. Haskell [Thompson99]) üblich sind. Der Anwender muss allen Inhalten und Abstraktionsvariablen semantische Typen zuweisen, da diese nicht aus der Inhaltsart abgeleitet werden können. Die Funktionstypen entstehen daraufhin durch Abstraktionen und werden durch Applikationen wieder aufgelöst. Beispiele für getypte Asset Expressions finden sich in Tabelle 3.2.

Entscheidend an dem Typsystem ist eine Typprüfung: An die Abstraktionsvariable *reiter* im Beispiel 2. können nur AEs appliziert werden, die vom Typ Herrscher sind. Das bedeutet, dass die benannte Referenz Napoleon auf einen Ausdruck verweisen muss, der vom Typ Herrscher oder einem seiner Untertypen sein muss.

Im Vergleich mit der Konzeptorientierten Inhaltsverwaltung kann man Arbeit mit getypten AEs als vollkommen offen und dynamisch ansehen [Bossung07, Abs. 4.6.2]. Eine Entität kann jederzeit frei modelliert werden, eine Schemamodifikation wie bei CCM Systemen (siehe 2.3.2) wird nicht benötigt. Semantisch getypte Asset Expressions besitzen gegenüber den ungetypten folgende Vorteile:

- Die Typen erleichtern dem Anwender die Interpretation der Ausdrücke.
- Es wird für den Anwender schwieriger, aus Versehen unsinnige Ausdrücke zu erstellen. Außerdem wird sichergestellt, dass sich jede Applikation auf eine frühere Abstraktion bezieht.
- Für den Fall, dass Namen an neue Ausdrücke gebunden werden, wird sichergestellt, dass sich die Bedeutung anderer Ausdrücke nicht nachhaltig verändert. Das Problem tritt auf, wenn andere Ausdrücke den Namen in ihren benannten Referenzen verwenden. Im Beispiel kann z. B. der Name Napoleon nicht an eine AE gebunden werden, die einen gleichnamigen Kater beschreibt. Es kann davon ausgegangen werden, dass dem Tier nicht der Typ Herrscher zugewiesen wird.

3.3.3 Traits

Asset Expressions erlauben es, den semantischen Typ und die Struktur der Erläuterungen voneinander unabhängig zu gestalten. Diese Freiheit ist wünschenswert, kann jedoch auch zum Hindernis werden, da jede AE von Grund auf eigenständig modelliert werden muss. Wenn die Anwendungsdomäne bereits gut verstanden ist, können Traits (*traits*, englisch für *Eigenschaft*, [Bossung07, Abs. 4.1]) bei der Modellierung helfen. Dabei handelt es sich um Schablonen, die eine vorgefertigte Menge von Abstraktionen definieren. Zur Erläuterung dient der Ausdruck 2. aus Tabelle 3.2, der mittels eines Traits erstellt wird. Zunächst wird ein Trait definiert:

trait *reiterbildnis* **refines** *Any* **of** ReiterStatue **with** λ*reiter*:Herrscher

Ausdrücke, die mit dem Trait *reiterbildnis* erstellt werden, besitzen den semantischen Typ ReiterStatue und die Abstraktion *reiter*. Traits können von anderen Traits Abstraktionen erben, was in diesem Fall nicht geschieht (*Any*). Eine AE wird folgendermaßen aus dem Trait erstellt:

create reiterbildnis  = $\lambda reiter$: Herrscher.  : ReiterStatue

Der entstandene Ausdruck verfügt über den semantischen Typ und die Abstraktionen, die in dem Trait vorgegeben sind. Obwohl Traits für einen semantischen Typ eine Struktur vorgeben, handelt es sich bei ihnen nicht um strukturelle Typen. So kann man einem Ausdruck, der durch einen Trait entstanden ist, weitere Abstraktionen hinzufügen. Auch

können im Beispiel trotz Existenz des Traits Entitäten des Typs ReiterStatue auch ohne Hilfe des Traits formuliert werden.

3.3.4 Intensionale Typen: Assetklassen

Bei der Modellierung konzeptueller Schemata mit Hilfe von Asset Expressions ergibt sich das Problem, wie Asset Expressions in eine äquivalente Schemastruktur umgewandelt werden können. Zu diesem Zweck ist ein zweites Typsystem vorgesehen, in dem AEs Typen in Form von Assetklassen (siehe Abs. 2.2) zugewiesen bekommen.

Dieses Typsystem unterscheidet sich grundlegend von dem vorher beschriebenen, da es durch die intensionalen Konzeptbeschreibungen rein strukturell motiviert ist. Semantische Typen besitzen darin keine Bedeutung. Das Prinzip dieses Typsystems soll anhand eines Beispiels erläutert werden, eine formale Betrachtung findet sich in [Bossung07, Abs. 6.1]. Für folgenden Ausdruck soll eine entsprechende Assetklasse hergeleitet werden:

((λ*zeit*: Jahr.λ*künstler*: Fotograf.  : Siegbildnis)"1945")Khaldei

Die Assetklasse wird nach folgenden Regeln hergeleitet:

Inhalte: Der Inhalt der AE wird zum `content` der Assetklasse, wobei ein Bezeichner erfunden wird.

Der Typ des Inhaltes der Assetklasse wird durch die Hilfsfunktions $map_C()$ aus der Inhaltsart ermittelt.

Abstraktionen und Applikationen: Ein Paar aus Abstraktion und zugehöriger Applikation wird zu einem Attribut der Assetklasse, wobei der Attributname der Assetklasse dem Namen der Abstraktionsvariablen entspricht.

Der Typ des Attributes ergibt sich aus der Hilfsfunktion $map_S()$, durch die ermittelt werden kann, ob die Applikation und Abstraktion auf eine Beziehung oder ein Charakteristikum abgebildet werden muss.

Daraus ergibt sich:

```
class Fotografie {
    content content1 : java.awt.Image      ;; aus map_C()
    concept characteristic zeit : Integer  ;; aus map_S()
            relationship kuenstler : Asset ;; aus map_S()
}
```

Die Hilfsfunktion $map_C()$ für die Inhaltsarten bedarf keiner detaillierten Erklärung. Sie ordnet jeder Inhaltsart der AEs eine Javaklasse zu, die in der entsprechenden Assetklasse verwendet werden kann. Sie ist lediglich eine Abbildung zwischen zwei äquivalenten technisch motivierten Typen. Die Funktion $map_S()$ bedarf einer genaueren Betrachtung. Durch sie wird einigen semantischen Typen T_S der Typ eines Charakteristikums zugeordnet. Das bedeutet, dass Applikationen mit diesen semantischen Typen als strukturlos interpretiert werden.

$$map_S(T_S) = \begin{cases} T_C & \text{wenn eine Abbildungsvorschrift für } T_S \text{ existiert} \\ Asset & \text{sonst} \end{cases}$$

Die Abbildungsvorschriften für $map_S(T_S)$ müssen vorher in Tupelform angegeben werden. In dem oben gegebenen Beispiel wird der Typ Jahr auf `Integer` abgebildet, für den Typ Fotograf ist jedoch keine Abbildungsvorschrift vorhanden, so dass dieses Attribut als Beziehung interpretiert wird. Diese Hilfsfunktion ist notwendig, damit unterschieden werden kann, welches Attribut als Beziehung und welches als Charakteristikum zu interpretieren ist. Allerdings bedeutet dies, dass durch einige semantische Typen die Struktur der AEs eingeschränkt wird.

3.4 Anwendungen

AEs werden in Asset Expressions Systemen (AES, siehe [Bossung07, Abs. 5.1]) verwendet. Diese Systeme unterstützen einzelne und kleine Gruppen von Domänenexperten darin, AEs produktiv einzusetzen. Sie sind nicht dafür geeignet, große Mengen von medialen Entitätsbeschreibungen einer großen Anwendergruppe zur Verfügung zu stellen – dafür sind Inhaltsverwaltungssysteme mit einem festen Schema wie z. B. die CCM Systeme besser geeignet. Denn zum einen erlaubt ein festes Schema sehr effiziente Implementierungstechniken, so dass gute Performanz auch bei vielen simultanen Zugriffen gewährleistet ist. Zum anderen werden die Anwender durch ein festes Schema bei der Dateneingabe dazu gezwungen, sämtliche Attribute der Klasse vollständig zu beschreiben. So kann festgelegt werden, welcher Informationsbedarf für alle Anwender ausreichend ist.

Im Gegensatz zu den CCM Systemen dienen AES zum einen als persönliche mediale Datenbanksysteme, die von Einzelnen oder kleinen Gruppen genutzt werden können. Für diesen Einsatzzweck ist eine Anfragesprache (*Asset Expression Query Language, AEQL*) vorgesehen, mit deren Hilfe sich AEs in dem Datenbestand leicht auffinden lassen [Bossung07, Abs. 4.2].

Die Flexibilität der AEs ist außerdem bei der Erstellung von konzeptuellen Modellen von Nutzen. Es können mit Hilfe der AEs zunächst Entitäten ohne strukturelle Einschränkungen beschrieben werden, um anschließend zu untersuchen, welche Attribute in einem Schema sinnvoll sind. Dieser Ansatz wird in dem Prozess zur beispielgetriebenen Modellierung verfolgt, der in Abschnitt 5.3 beschrieben wird.

Kapitel 4

Modellierungsmethodiken

In vielen Teildisziplinen der Informatik werden Methoden und Prozesse für die Modellierung angewendet, die aufgrund von praktischen Erfahrungen verfeinert wurden. Einige Fachgebiete werden in diesem Kapitel in einem methodologischen Ansatz auf Vorgehensweisen analysiert, die auch bei der konzeptuellen Modellierung durch Domänenexperten angewendet werden können.

4.1 Begriffsklärung

4.1.1 Methodiken, Ansätze und Prozesse

Die Begriffe Methode, Methodik und Methodologie werden häufig ohne nähere Definition synonym eingesetzt. Da diese Begriffe für dieses Kapitel von Bedeutung sind, sollen sie zum besseren Verständnis hier kurz für den Kontext dieser Arbeit definiert werden. Dazu sei angemerkt, dass die deutsche Sprache unter diesen Begriffen eine stärkere Differenzierung erlaubt, als es im angelsächsische Sprachgebrauch üblich ist. Eine konzeptuelles Modell der Beziehungen zwischen den Begriffen zeigt Abbildung 4.1.

Methode: Eine Methode ist eine Vorgehensweise zur planmäßigen Lösung von Problemen. Sie ist die Grundlage für planmäßiges Handeln. (Synonyme: *Verfahren, Vorgehensweise*; Englisch: *method*)

Ansatz: Ein Ansatz ist eine Idee zur Lösung von Problemen. Dabei kann es sich auch um eine Idee für die Planung handeln. (Englisch näherungsweise: *approach*, dieser Begriff ist wenig spezifisch und wird gelegentlich auch synonym zu *method* verwendet.)

Methodik: Unter einer Methodik versteht man die Gesamtheit von Methoden in einer wissenschaftlichen (Teil-)Disziplin. Verwandt ist der Begriff des Methodenbündels, wodurch eine Menge von Methoden beschrieben wird. (Englisch näherungsweise: *methodology*. Dieser Begriff in der englischen Informatikliteratur auch als Synonym zu *method* verwendet, Siehe Diskussion in [GFC04, Abs. 3.0].)

Methodologie: Die Methodologie ist die Wissenschaft der wissenschaftlichen Methoden. Im Fokus der Methodologie steht die Analyse und Modellierung von Metho-

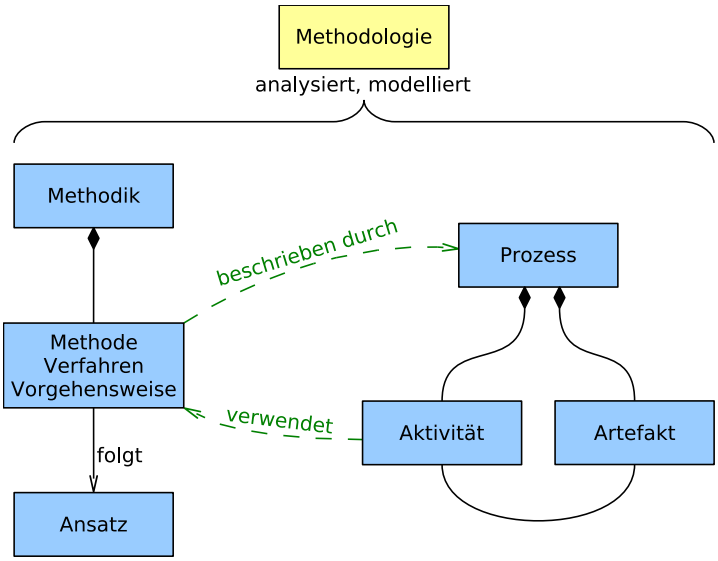

Abbildung 4.1: Begrifflichkeiten: Methodologie, Methodik, Prozess

den, Methodiken sowie den mit ihnen zusammenhängenden Prozessen. (Englisch in grober Näherung: *methodology*, der Begriff wird jedoch häufig anders verwendet.)

Prozess: Ein Arbeitsprozess ist ein Vorgang zwischen Einzelnen und Gruppen, der ein bestimmtes Resultat zum Ziel hat. Prozesse werden meist als eine Abfolge von Prozessschritten und den mit ihnen verknüpften Artefakten beschrieben. (Synonym: *Arbeitsablauf*; Englisch: *process*)

Aktivität: Eine Aktivität ist eine von Personen durchgeführte Tätigkeit im Rahmen eines Prozesses. Dabei kann es sich sowohl um sequenziell abzuarbeitende Prozessschritte handeln, wie auch um kontinuierliche Tätigkeiten, die im Prozesskontext von Bedeutung sind. (Synonym: *Tätigkeit*; Englisch: *activity*)

Artefakt: Unter Artefakten versteht man Ergebnisse und Ausgangswerke von Aktivitäten. Meistens wird der Begriff für die Ergebnisse von Prozessschritten verwendet. (Synonym im engeren Kontext: *Produkt*; Englisch: *artefact, product*)

Die Beziehungen zwischen Methoden und Prozessen sind vielfältig, je nachdem auf welcher Granularitätsebene man die Betrachtung durchführt. Zwei Möglichkeiten sind in Abbildung 4.1 gestrichelt dargestellt. Es ist ungewiss, ob eine präzisere Zuordung vorgenommen werden kann, die sich mit dem allgemeinen Sprachgebrauch der Begriffe deckt. Gemäß diesen Definitionen wird in diesem Kapitel Methodologie betrieben, indem die Methodiken verschiedener Teildisziplinen der Informatik analysiert werden.

4.1.2 Klassifizierung von Modellierungsansätzen

In [HJ02] werden fünf Modellierungsansätze für Domänenontologien beschrieben, die in einem konkreten Prozess auch kombiniert angewendet werden können. Diese Kategorisierung kann ohne weiteres auf Prozesse zur konzeptuellen Modellierung übertragen werden.

Ansatz	**Modellierungsgrundlage**
Inspirational	Individuelle Ansichten über die Domäne
Induktiv	Konkrete Anwendungsfälle und Entitäten aus der Domäne
Deduktiv	Generelle Konzepte und Prinzipien
Synthetisierend	Menge existierender Modelle bzw. Ontologien, von denen jede einen Teil der Domäne beschreibt
Kollaborativ	Mehrere individuelle Ansichten über die Domäne
Transformativ	Konzepte liegen in ungeeigneter Form vor

Tabelle 4.1: Ansätze zur Domänenmodellierung

Domänenontologien und konzeptuelle Modelle dienen demselben Zweck: Der formalisierten Erfassung relevanter Domänenkonzepte und der Beziehungen zwischen ihnen. Hierbei bezieht sich das Wort *relevant* auf die geplante Verwendung der Ontologie bzw. des Modells.

Die Ansätze nach [HJ02] stellen eine sinnvolle und allgemeine Grundlage zur Klassifikation von Modellierungsansätzen dar. Sie sollen im folgenden erläutert werden, wobei das oberste Ziel der Autoren, die Einigung auf eine gemeinsame Ontologie (*ontological commitment*), in dieser Arbeit nicht verfolgt wird. Aufgrund der Personalisierbarkeit der CCM Systeme spielt die Einigung auf ein bestimmtes Modell eine untergeordnete Rolle. Eine Übersicht gibt Tabelle 4.1, wobei der transformative Ansatz eine eigene Ergänzung ist und nicht aus der Quelle [HJ02] stammt.

Inspirationaler Ansatz (*inspirational approach*)**:** Diesem Ansatz folgend wird eine Ontologie aufgrund der Vorstellung, Kreativität und individuellen Ansichten einer einzelnen Person entworfen.

Die Betonung auf individuelle Kreativität bietet die Möglichkeit, zu neuen Lösungen zu gelangen, birgt allerdings auch das Risiko, dass die Ontologien von anderen Anwendern nicht akzeptiert werden.

Induktiver Ansatz (*inductive approach*)**:** Grundlage für die Ontologie sind in diesem Ansatz konkrete Entitäten und Anwendungsfälle aus der Domäne. Ausgehend von diesen wird ein passendes Konzept induziert, dass anhand weiterer Entitäten und Anwendungsfälle auf seine Generalisierbarkeit überprüft und gegebenenfalls angepasst wird.

Die Generalisierbarkeit der Konzepte bleibt allerdings problematisch, weil sich so entstandene Konzepte häufig nicht einmal auf verwandte Domänen übertragen lassen. So können z. B. die Konzepte *Kunde* und *Rechnung* aufgrund des unterschiedlichen Informationsbedarfs nicht in identischer Form bei einer Krankenkasse und Gebäudeversicherung verwendet werden.

Deduktiver Ansatz (*deductive approach*)**:** Im deduktiven Ansatz werden generelle Konzepte dazu verwendet, um spezifische Konzepte zu formulieren. Hierbei bietet sich die Verwendung einer *Upper-Level-Ontologie* an, in der allgemeine Konzepte mit domänenübergreifender Gültigkeit formuliert sind. Eine aus einer Upper-Level-

Ontologie deduzierte Ontologie besteht demnach aus Instanzierungen der allgemeinen Konzepte.

Synthetisierender Ansatz (*synthetic approach*)**:** Eine andere Form der Wiederverwendung von Ontologien bietet der synthetisierende Ansatz. Eine Menge von Ontologien wird zu einer einheitlichen Ontologie synthetisiert, die alle Konzepte der Ursprungsontologien beinhaltet.

Die Stärke dieses Ansatzes liegt in der Wiederverwendung bewährter Ontologien. Es muss bei der Synthese jedoch darauf geachtet werden, Überlappungen zwischen den Ontologien zu identifizieren und zu vereinheitlichen.

Kollaborativer Ansatz (*collaborative approach*)**:** Ein kollaborativer Ansatz zeichnet sich dadurch aus, dass die Entwicklung der Ontologie als kooperative Tätigkeit durchgeführt wird. Dadurch fließen mehrere individuelle Ansichten über die Domäne in die Ontologie ein. Neben der Bereitschaft zur Kooperation der Teilnehmer ist es notwendig, Konflikte, die durch unterschiedliche Ansichten entstehen können, zu klären.

Transformativer Ansatz (nicht in [HJ02] vorhanden)**:** Bei einem transformativen Ansatz ist die zur Modellierung notwendige Information bereits vorhanden, jedoch nicht in der gewünschten Modellform gegeben. Die Konzepte werden aus vorhandenen Dokumenten in eine geeignete Form überführt und gegebenenfalls verfeinert, z. B. in eine Ontologiesprache.

Im Gegensatz zu den vorher genannten Ansätzen wird ein transformatives Verfahren wenig von dem individuellen Verständnis der an der Modellierung beteiligten Personen beeinflusst, da es sich um einen dokumentenorientierten Ansatz handelt.

4.2 Objektorientierte Analyse

Viele in der objektorientierten Analyse (OOA) gängige Methoden lassen sich auch zur konzeptuellen Modellierung verwenden. Obwohl die Methodiken eng verwandt sind, werden konzeptuelle Modelle nur in wenigen Werken zur OOA erwähnt, Ausnahmen sind in [Larman05, Kap. 4] [Fowler04, Kap. 1] zu finden. Generell lässt sich festellen, dass in der OOA das Verhalten von Objekten berücksichtigt wird, was in der konzeptuellen Modellierung nicht der Fall ist. Im *Rational Unified Process* (RUP) findet sich ein Domänenmodell (*domain model*), das einem konzeptuellen Modell entspricht. Es ist ein Ergebnis des Geschäftsmodellierungsprozesses (*Buisiness Modeling Workflow*), dessen Ziele das Verständnis und gegebenenfalls die Modifikation von Geschäftsprozessen sind. Dabei sind vor allem jene Änderungen von Interesse, die durch den Einsatz neuer informationstechnischer Systeme ermöglicht werden, siehe [Kruchten00, Kap. 8].

Darüber hinaus herrscht schon in der Standardliteratur über objektorientierte Analyse und Design (OOAD) kein Konsens darüber, was ein (objektorientiertes) Analysemodell genau beinhaltet, welchem Zweck es dient und wie es sich vom Entwurfsmodell unterscheidet. Übereinstimmung ist darin vorhanden, dass es sich um ein Systemmodell handelt, welches die Funktionalität des Systems in einer von der Implementierungsplattform unabhängigen Weise wiedergibt [CY91, Kap. 0][Jacobson92, 6.5.1]. Booch betont,

dass das Analysemodell im Gegensatz zum Entwurfsmodell wiedergibt, *was* ein System leistet, aber nicht *wie* dieses realisiert wird [Booch94, 6.3]. Eine ähnliche Unterscheidung wird auch im RUP getroffen [Kruchten00, Kap. 10].

Die in der Standardliteratur informell beschriebenen Verfahren zur OOA haben inspirationalen und kollaborativen Charakter. Es werden lediglich eine Reihe von Aktivitäten empfohlen, die in geeigneter Kombination angewendet werden sollen. Obwohl meistens Domänenexperten in das Vorgehen einbezogen werden, hat das individuelle Verständnis des Analysten großen Einfluss auf das Modell. Solche Vorgehensweisen werden z. B. in [Booch94, CY91] erwähnt. Es finden sich unter den Verfahren auch induktivtransformative Ansätze, wobei Anforderungsdokumente wie Anwendungsfälle (*use cases*) als Grundlage für die Analyse verwendet werden [Balzert01, Jacobson92]. Eine andere Ausprägung des transformativen Ansatzes ist die Verwendung von Datenmodellen als Grundlage für die OOA. Diese können aus Altsystemen oder aus Schemasammlungen wie z. B. [Silverstone97] gewonnen werden [Larman05, 9.5][CY91, 3.3.2].

4.2.1 Analysemuster

Analysemuster gleichen den Entwurfsmustern [GHJV94], indem sie gängige Strukturen in Analysemodellen in wiederverwendbarer Weise beschreiben. Kategorisierte Sammlungen von Mustern erleichtern die Verwendung standardisierter und bewährter Lösungen für wiederkehrende Analyseprobleme. Darüber hinaus wird durch die eindeutigen Mustern ein spezialisiertes und wohldefiniertes Vokabular zur Kommunikation zwischen Entwicklern und Analysten geschaffen.

Die Beschreibung der Analysemuster folgt der in [Balzert01, 2.18.2] getroffenen Unterscheidung zwischen allgemeinen und anwendungsspezifischen Mustern, wobei für letztere hier der treffendere Terminus *domänenspezifische Analysemuster* verwendet wird.

Allgemeine Analysemuster bieten Lösungen für Modellierungsprobleme an, die sich über Domänen- und Anwendungsgrenzen hinweg bewährt haben. Daraus folgt, dass die Darstellung solcher Muster allgemein gehalten ist. Sie werden angewendet, indem sie als eine Schablone für die Lösung eines konkreten Analyseproblems verwendet werden. Eine direkte Verwendung eines allgemeinen Analysemusters ist meistens nicht möglich. Beispiele für allgemeine Analysemuster sind [Balzert99]: *Liste*, *Exemplartyp*, *Historie* und *Rolle*. Die Verwendung allgemeiner Analysemuster ist ein deduktiver Ansatz, da generelle Konzepte auf ein spezifisches Problem angewandt werden. Der Einsatz eines Musters kann als eine Instanzierung verstanden werden.

Domänenspezifische Analysemuster unterscheiden sich von den allgemeinen Analysemustern darin, dass sie sich nur im Kontext bestimmter Domänen oder Anwendungen verwenden lassen. Man wendet diese Muster an, indem man sie bei der konzeptuellen Modellierung direkt in das zu erstellende Modell übernimmt, wobei nur geringfügige Anpassungen des Musters üblich sind. Eine Sammlung überwiegend domänenspezifischer Muster findet sich in [Fowler97], z. B. Konto (*Account*), Vertrag (*Contract*) und Steuerberechnung (*Calculating the Tax*).

Im Gegensatz zu den allgemeinen Analysemustern ist die Verwendung konkreter Analysemuster ein synthetisierender Ansatz, da ein anwendungsspezifisches Muster ein partielles Domänenmodell ist.

Der Übergang zwischen den hier unterschiedenen allgemeinen und domänenspezifischen Mustern ist häufig fließend. Die Unterscheidung ist davon abhängig, ob der deduktive oder der synthetisierende Modellierungsansatz überwiegt. Es ist auch möglich, ein bestimmtes Muster sowohl als Allgemeines wie auch als domänenspezifisches einzusetzen: Das domänenspezifische Muster Geld (*money*) dient der Repräsentation eines Geldbetrages unter Berücksichtigung verschiedener Währungen [FRF03, Kap. 18]. Verallgemeinert man dieses Muster, erhält man das Muster Mengenangabe (*quantity*), das auch außerhalb von wirtschaftlichen Domänen die Angabe einer Größe unter Berücksichtigung von Maßeinheiten repräsentiert. [FowlerAP2]

4.2.2 Linguistische Analyse

Varianten der linguistischen Analyse finden sich in den meisten modernen Werken zum Thema objektorientierte Analyse und Entwurf. Der Ansatz dieser Methode ist die Analyse von Anforderungsdokumenten, die das zu entwickelnde System beschreiben, wie z. B. Lastenhefte [Balzert01, 2.18.3] oder Anwendungsfälle [Larman05, 9.5]. Die Texte werden in einem ersten Schritt nach Begriffen (Nomen) untersucht, denen in der Anwendungsdomäne eine nennenswerte Bedeutung zufällt. Diese Begriffe werden im folgenden bewertet, verfeinert und präzisiert, bis eine Menge von relevanten Domänenkonzepten herausgearbeitet ist, die als konzeptuelle Klassen verwendet werden. Die Methodik zur Bewertung und Verfeinerung der Konzepte stützt sich dabei auf Richlinien oder Checklisten. Nachdem die konzeptuellen Klassen gefunden worden sind, wird ein äquivalentes Vorgehen zum Auffinden von Attributen, Assoziationen und Generalisierungsbeziehungen verwendet.

Diese Methode geht auf Abbott [Abbott83] zurück, der vorschlug, Programme auf Grundlage präziser umgangssprachlicher Beschreibungen zu entwickeln. Sein Verfahren geht jedoch über die konzeptuelle Modellierung hinaus, da in diesem sowohl ein Datenmodell wie auch programmiersprachliche Algorithmen entwickelt werden. Die in den modernen Varianten des Verfahrens häufig gewählte Ausgangsbasis in Form von Anwendungsfällen hat ihren Ursprung bei Jacobson [Jacobson92], der in seiner Methode das Analysemodell aus den Anwendungsfällen herleitet.

Nach der in Abschnitt 4.1.2 vorgestellten Klassifizierung folgt dieses Verfahren einen induktiven, transformativen und inspirationalen Ansatz. Das Verfahren ist induktiv, weil Anwendungsfälle die Grundlage für die Analyse bilden, und transformativ, weil davon ausgegangen wird, dass das erforderliche Domänenwissen zur Modellierung vollständig in den Anforderungstexten enthalten ist. Der inspirationale Anteil des Verfahrens ergibt sich daraus, dass zur Bewertung der gefundenen Begriffe ein grobes, individuelles Verständnis der Domäne erforderlich ist [Abbott83]. Obwohl die Grundlage Anwendungsfälle sind, fließen Ansichten des Analysten in das konzeptuelle Modell ein.

4.2.3 Konzeptuelle Kategorien

Ein anderes Verfahren zum Identifizieren von konzeptuellen Klassen ist in [Larman05, 9.5] beschrieben. Hierbei nutzt man die Beobachtung, dass sich die meisten konzeptuellen Klassen wenigen Kategorien zuordnen lassen. Die gängigen Kategorien sind in einer Liste (*conceptual class category list*) zusammengefasst, ein Ausschnitt dieser Liste findet sich in

Kategorie	**Bemerkung**
Geschäftliche Transaktionen	kritisch, da entscheidend für das Geschäft
Produkt oder Dienstleistung	stehen in enger Beziehung zu den Transaktionen
Physische Objekte	vor allem relevant, wenn diese kontrolliert oder simuliert werden
Beschreibungen	von Produkten, Objekten, Dienstleistungen

Tabelle 4.2: Konzeptuelle Klassenkategorien nach [Larman05]

Tabelle 4.2. Zur Modellierung werden die Listenelemente in der vorgegebenen Reihenfolge abgearbeitet, indem nach konzeptuelle Klassen der jeweiligen Kategorie gesucht wird. Die Liste ist so geordnet, dass die voraussichtlich wichtigsten Kategorien zuerst bearbeitet werden, wobei in der Quelle geschäftliche Anwendungen im Mittelpunkt stehen.

Dieses Verfahren realisiert einen deduktiven Modellierungsansatz. Die einzelnen konzeptuellen Kategorien formulieren allgemeine, domänenübergreifende Konzepte, die im konzeptuellen Modell individuell ausgeprägt sind.

Das Verfahren bietet sich vor allem als Ergänzung zu anderen Verfahren wie der linguistischen Analyse oder einem kollaborativen Vorgehen an und verleiht diesen durch das deduktive Element Struktur. Neben der Liste der konzeptuellen Kategorien bietet [Larman05] eine entsprechende Liste für Assoziationen (*common association list*) an.

Die Kategorisierung von Klassen ist in den objektorientierten Methoden ein altes Verfahren, sie findet sich z. B. bei [Booch94, 4.2]. Allerdings werden in diesen Fällen keine konzeptuellen Klassen sondern Klassen aus dem Softwareentwurf betrachtet.

4.3 Anforderungsanalyse

Als Anforderungsanalyse versteht man die systematische Erhebung, Dokumentation und Kommunikation von Anforderungen im Softwareengineering [Rupp04]. Dabei handelt es sich um eine Unterdisziplin des Anforderungsengineering (*requirements engineering*), das auch das Anforderungsmanagement (*requirements management*) beinhaltet [Wiegers05][1]. Eine Definition von Anforderungen findet sich in [Balzert01, 2.1]:

> Anforderungen (*requirements*) legen die qualitativen und quantitativen Eigenschaften eines Produkts aus Sicht des Auftraggebers fest.

Anforderungen dienen in einem Softwareentwicklungsprozess als Grundlage für die Systementwicklung, zur Kommunikation mit Auftraggebern und Anwendern, wie auch als Basis für die Vertragsgestaltung. Generell werden Anforderungen auf Softwaresystemebene (*system requirements*) und Geschäftsebene (*business requirements*) unterschieden, wobei letztere die Geschäftsziele wiedergeben, die durch den Einsatz eines neuen Systems unterstützt werden sollen [Wiegers05]. Systemanforderungen werden nach funktionalen und nichtfunktionalen Anforderungen unterschieden. Häufig werden differenziertere Unterscheidungsschemata für Anforderungen empfohlen, wie z. B. das FURPS+ Schema (*functionality, usability, reliability, performance, supportability, ...* [Grady92]).

[1] Aufgrund von Mängeln in der deutschen Übersetzung wird die Anforderungsanalyse in diesem Buch gelegentlich auch als Anforderungsentwicklung bezeichnet.

Die Disziplin der Anforderungsanalyse versucht Widersprüchlichkeiten zu vereinen: Die Auftraggeber und Anwender einer Software sind von sich aus lediglich in der Lage, ungenaue und lückenhafte Anforderungen zu formulieren. Als Grundlage für die Systementwicklung, Vertragsgestaltung und Abnahme sind allerdings exakt formulierte Anforderungen notwendig. Wünschenswerte Qualitäten von Anforderungen werden beispielsweise auch im Standard [IEEE-830] erläutert: Korrektheit, Eindeutigkeit, Vollständigkeit, Konsistenz, Prüfbarkeit, Kategorisierbarkeit, Modifizierbarkeit, Rückverfolgbarkeit. Obwohl die Qualitäten Prüfbarkeit und Eindeutigkeit durch den Einsatz von Formalismen verbessert werden können, sind exakte Spezifikationssprachen wie z. B. RSL [TP91] in der Anforderungsanalyse eine Ausnahme [RP06, 13.5.1]. Der Grund dafür liegt darin, dass die Anforderungen von den Auftraggebern gestellt werden, welche diese im eigenen Interesse auch selbst formulieren sollten [Pieper04]. Daher ist es notwendig, dass die Anforderungsdokumente von den Auftraggebern zumindest verstanden und bestätigt werden. Auch in [SM99] wurde untersucht, welche negativen Folgen die unangemessene Anwendung von Formalismen haben kann. Aus diesem Grund werden natürlichsprachliche Aufzeichnungen möglichst präzise formuliert, z. B. durch Anwendungsfälle (*use cases*) [Cockburn00, Kap. 11] oder linguistische Verfahren [Rupp04].

Darüber hinaus ist die Berücksichtigung weicher Faktoren, wie psychologischer und sozialer Aspekte, in der Praxis häufig wichtiger als formale Beschreibungen. Es ist vielversprechender, implizites Wissen offenzulegen, darunter versteht man Kenntnisse, die als selbstverständlich vorausgesetzt werden. Dazu kann man psychologische Methoden, wie die Neurolinguistische Programmierung (NLP) verwenden [VZ06]. Bei der kollaborativen Anforderungsanalyse müssen gruppendynamische Phänomene berücksichtigt werden, wie Konflikte und Machtverhältnisse [Gottesdiener02, Kap. 9] [Rupp04, 3.4.1]. Diese können schlimmstenfalls dazu führen, dass wichtige und bekannte Anforderungen nicht erfasst werden. Auch kreative Prozesse und Verfahren sind für das Auffinden von Anforderungen essenziell [MG01] [Rupp04, 4.3].

Der Zusammenhang zwischen der Anforderungsanalyse und dieser Arbeit ergibt sich daraus, dass im Softwareengineering Anwender nur über die Anforderungen in den Entwicklungsprozess einbezogen werden. Die Teilnahme an Modellierungsprozessen wie der objektorientierten Analyse bleibt Experten vorbehalten. Lediglich Varianten der konzeptuellen Modellierung werden in der Anforderungsanalyse verwendet, um ein präzises Vokabular für die natürlichsprachlichen Anforderungsdokumente festzulegen. Dabei sollen sprachliche Mehrdeutigkeiten wie Synonyme und Homonyme identifiziert und aufgelöst werden [Rupp04, 3.3.3].

Darüber hinaus ähneln sich gängige Probleme der Anforderungsanalyse und der Modellierung durch Domänenexperten. Bei der Anforderungsanalyse stellt sich häufig das Problem, dass die Auftraggeber und Anwender bestenfalls eine vage Vision des zu entwickelnden Systems formulieren können. Zwei Ursachen werden dafür angeführt: Zum einen sind Anwender nicht in der Lage die von ihnen benötigten Anforderungen zu benennen, zum anderen sind sie nicht in der Lage die Anforderungen in einer präzisen Form zu beschreiben, die als Grundlage für die Entwicklung und Vertragsgestaltung geeignet ist. Diesen Problemen wird durch Methoden für die Erhebung und Spezifikation von Anforderungen entgegengewirkt. Die Erfahrungen aus den bisherigen CCM Projekten [Bossung07, 7.2.1] zeigen, dass bei der Modellierung durch Domänenexperten ähnliche Schwierigkeiten auftreten, so dass eine Übertragung von Methoden der Anforderungs-

analyse vielversprechend ist.

Ähnlich wie bei der objektorientierten Analyse wird die Methodik der Anforderungsanalyse selten durch exakte Prozesse beschrieben. Der starke Einfluss weicher Faktoren erschwert ihre Einhaltung. Es finden sich eine Vielzahl bewährter Vorgehensweisen und Praktiken, die aufgrund der Erfahrung des Analysten ausgewählt und in Kombination eingesetzt werden. Eine Auswahl solcher Verfahren wird im folgenden vorgestellt.

4.3.1 Systemvision und Systemkontext

In der Anforderungsanalyse ist es üblich, eine Produktvision und einen Projektrahmen festzulegen, bevor mit der detaillierten Analyse begonnen wird [Wiegers05, Kap. 5]. Eine Produktvision beschreibt dabei das Fernziel des gewünschen Systems und steht meistens in direktem Bezug zu den Anforderungen auf Geschäftsebene. Da Visionen generell idealisiert und ungenau sind, wird in dem Projektrahmen genauer festgelegt, was in dem Entwicklungsprojekt realisiert werden soll. Eine vollständige Umsetzung der Vision ist im allgemeinen unrealistisch. Wird ein inkrementellen Entwicklungsprozess verwendet, so ist für jede Version des Systems ein eigener Projektrahmen festzulegen.

Ein wichtiger Bestandteil des Projektrahmens ist die Festlegung des Systemkontextes. Darunter versteht man die Beschreibung und Abgrenzung des zu entwickelnden Systems in Bezug zu seiner Umgebung, die aus anderen informationsverarbeitenden, technischen, physischen und sozialen Systemen bestehen kann [MJF03]. Der Systemkontext kann z. B. durch natürlichsprachliche Formulierungen, Negativlisten oder durch Kontextdiagramme beschrieben werden [Rupp04, 5.5.2].

Eine Festlegung der Systemvision und des Systemkontextes ist aus zwei Gründen wichtig: Erstens wird eine Einigung mit dem Auftraggeber getroffen, welche Ziele in dem Entwicklungsprojekt verfolgt werden. Durch die Abgrenzung im Systemkontext wird zudem festgelegt, aus welchen Bestandteile sich das System zusammensetzt, z. B. Software und Hardware. Zweitens wird durch Vision und Kontext ein stabiler Rahmen für die weitere Anforderungsanalyse und Systementwicklung vorgegeben. Die detaillierten Anforderungen können starken Änderungen unterworfen sein. Eine Einigung auf eine Systemvision und einen Systemkontext ist in jedem Softwareentwicklungsprojekt sinnvoll. Dem Systemkontext kommt jedoch gerade im Umfeld komplexer soziotechnischer Systeme besondere Beachtung zu, wie z. B. im Bereich der Flugsicherung [MJF03].

Eine Einordnung in das Klassifizierungsschema aus Abschnitt 4.1.2 ist nur mit Einschränkungen möglich, da es sich bei der Einigung und Festlegung auf Vision und Kontext um keinen Modellierungsvorgang handelt. Die in der Literatur beschriebenen Vorgehen haben durch die Konsensfindung einen kollaborativen Charakter.

4.3.2 Workshops

Workshops sind eines der effektivsten Verfahren der Anforderungsanalyse und können für sämtliche Teilaktivitäten eingesetzt werden, d. h. zur Erhebung, Analyse, Spezifikation und Validierung von Anforderungen. Die Stärke von Workshops liegt darin, dass eine Gruppe aus Teilnehmern verschiedener Hintergründe durch unmittelbare Zusammenarbeit produktiver arbeiten kann, als es verteilten Einzelpersonen möglich ist. Vorbedingungen hierfür sind, dass die Teilnehmer gemeinsame Absichten verfolgen und sich

individuell auf den Workshop vorbereiten. Durch die Kombination von Wissen und Erfahrungen verschiedener Teilnehmer werden Lösungen gefunden, die aus verschiedensten Blickwinkeln gleichermaßen Bestand haben. Daraus folgt, dass zur Durchführung eines erfolgreichen Anforderungs-Workshops Teilnehmer aus allen beiteiligten Interessengruppen benötigt werden, wie z. B. Auftraggeber, Anwender, Architekten und Analysten. Um ein zielgerichtetes und produktives Arbeiten der Gruppe zu unterstützen müssen soziale Faktoren berücksichtigt werden. Dies kann durch Vorbereitung des Workshops sowie durch den Einsatz eines neutralen Moderators geschehen, der durch Gruppenaktivitäten, aktives Zuhören und Mediation das gegenseitige Verständnis und die Bereitschaft zur Zusammenarbeit fördert [Gottesdiener02].

Workshops können zur Erhebung von Anforderungen auf Geschäftsebene und auf Systemebene eingesetzt werden, wobei noch nach Detaillierungsgrad der Systemanforderungen differenziert werden kann, z. B. nach *Charter, Scope, High-Level, Detailed Workshops* [Gottesdiener02]. In dem RESCUE Prozess (*Requirements Engineering with Scenarios for User-Centred Engineering*, [MJF03, RMRD05]) werden sie für zwei Aufgaben eingesetzt: Zum einen dienen sie der Anforderungserhebung, die auch als kreativer Prozess verstanden wird. Erkenntnisse aus der Kreativitätstheorie werden in diesen Workshops angewandt [MG01]. Darüber hinaus dienen sie auch der Synchronisation verschiedener Arbeitsabläufe, indem Entscheidungen für den weiteren Prozessablauf getroffen werden. In diesem Zusammenhang werden die Anforderungen auch validiert, indem gemeinsam mögliche Szenarien durchgespielt werden. Auch in [Gottesdiener02, Kap. 6] wird auf die Bedeutung funktionierender Entscheidungsprozesse in Workshops hingewiesen.

Workshops sind an sich noch kein Modellierungsverfahren, weshalb man sie nicht in die in Abschnitt 4.1.2 vorgestellten Modellierungsansätze einordnen kann. Workshops sind ein Werkzeug zur Förderung kollaborativer Arbeit und lassen sich neben der Anforderungsanalyse auch in Kombination mit anderen Modellierungsverfahren nutzen. In Verbindung mit Anforderungs-Workshops können durch ein Top-Down bzw. Bottom-Up Vorgehen auch deduktive bzw. induktive Elemente in Workshops einfließen. Bei einem Top-Down Ansatz werden allgemeine Anforderungen als Ausgangsbasis für detailliertere Anforderungen verwendet. Dadurch kann ein zielgerichtetes kollaboratives Arbeiten gefördert werden, siehe [Gottesdiener02, Kap. 10].

4.3.3 Anwendungsfälle

Anwendungsfälle (*use cases*) sind in der Anforderungsanalyse die am weitesten verbreiteste Technik zur Dokumentation und Kommunikation von Anforderungen. Die Anforderungen werden in Form von Szenarien beschrieben, die eine mögliche Interaktion eines Akteurs (bzw. Anwenders) mit dem System beschreiben. Ein Akteur ist eine Rolle, die ein Anwender oder ein externes System in Bezug auf das zu entwickelnde System einnimmt. Dabei beschreiben Anwendungsfälle die Funktionen des Systems aus Sicht der Akteure und die Akteure die Umgebung des Systems. Anders ausgedrückt ist ein Anwendungsfall ein Prozess, der ausgeführt wird, um den Bedürfnissen eines oder mehrerer Akteure nachzukommen. Unter einem Szenario versteht man in diesem Zusammenhang eine konkrete Sequenz von Aktionen und Interaktionen mit dem System. Ein Anwendungsfall umfasst mehrere zusammengehörige Erfolgs- und Misserfolgsszenarien, die bei seiner Ausführung stattfinden können. [Larman05, 6.2]

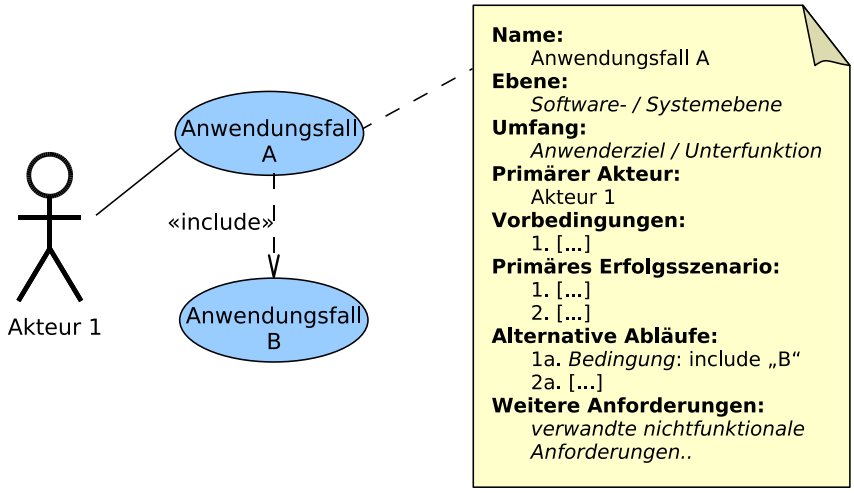

Abbildung 4.2: Anwendungsfall (*use case*)

Die gängigsten Formen, Anwendungsfälle zu dokumentieren, sind UML Anwendungsfalldiagramme, natürlichsprachlicher Text und strukturierter Text. In einer natürlichsprachlichen Formulierung ist dafür zu sorgen, dass die Anforderungen trotz der freien Form hinreichend präzise formuliert werden. Zu diesem Zweck können Methoden der linguistischen Analyse herangezogen werden [Rupp04]. Am gängigsten sind strukturierte Texte, die Aufgrund von Schablonen erstellt werden [Cockburn00], wie z. B. in Abbildung 4.2, rechts. Die Verwendung von Schablonen hat den Vorteil, dass auf bewährte Strukturen zurückgegriffen wird. Außerdem wird eine vollständige Beschreibung gefördert, da durch die Felder der Schablone eine umfassende Struktur vorgegeben wird. Anwendungsfalldiagramme (*use case diagrams*, [Fowler04]) werden häufig eingesetzt, um die Zusammenhänge zwischen Anwendungsfällen und Akteuren zu visualisieren. Ein Beispiel findet sich in Abbildung 4.2, linke Seite. Sie können bei der Identifikation von Anwendungsfällen dienen und um eine Übersicht über die detaillierteren Beschreibungen zu gewinnen.

Die Stärke von Anwendungsfälle liegt darin, dass das System aus Sicht der Anwender (bzw. Akteure) beschreiben wird. Gut formulierte Anwendungsfälle sind daher durch die Beschreibung des Systems als Black-Box und durch ihre Essentialität gekennzeichnet [Larman05, 6.11,6.13][Cockburn00]. Letzteres bedeutet, dass sie unabhängig von Technologien und konkreten Benutzerschnittstellen beschrieben werden. Durch diese Darstellung, die sich an der Sichtweise des Kunden orientiert, wird die Kommunikation zwischen Analysten und Kunden gefördert sowie die Integration von Anwendern in den Entwicklungsprozess erleichtert [RR06, Kap. 6][Larman05, 6.4]. Die Teilnahme von Anwendern an der Entwicklung wird als wichtiger Faktor für den Projekterfolg angesehen [Larman03, Kap. 6]. Darüber hinaus werden sie bei der Anforderungsvalidierung eingesetzt, indem aus Anwendungsfällen Szenarien erzeugt werden, die gemeinsam mit Domänenexperten auf ihre Vollständigkeit und Eindeutigkeit hin überprüft werden. Die Szenarien können auch mit Werkzeugunterstützung generiert und medial aufbereitet werden [MJF03, RMRD05].

4.3.4 Prototypen

In der Softwareentwicklung versteht man unter einem Prototyp eine partielle Realisierung eines Systems, die als Grundlage für die Untersuchung und Bewertung von Systemmerkmalen dient. Prototypen werden im gesamten Softwareentwicklungsprozess eingesetzt und dienen nicht ausschließlich der Anforderungsanalyse.

Prototypen werden anhand ihres Funktionsumfanges, ihrer Weiterverwendung und der Art ihrer Realisierung unterschieden [Rupp04, 3.3.5]. Im Funktionsumfang unterscheidet man zwischen horizontalen und vertikalen Prototypen. In horizontalen Prototypen wird eine große Menge der Funktionen des Zielsystems berücksichtigt, diese aber nicht vollständig umgesetzt, wie es z. B. bei Benutzeroberflächenprototypen üblich ist. In vertikalen Prototypen wird nur eine kleine Teilmenge der geplanten Funktionen realisiert. Allerdings werden diese vollständig umgesetzt, wobei alle Schichten des geplanten Systems realisiert werden. In der Weiterverwendung von Prototypen wird zwischen Wegwerfprototypen und evolutionären Prototypen unterschieden. Da letztere an den Kunden als Teil des fertigen Systems ausgeliefert werden, sind hohe Qualitätsanforderungen an sie zu stellen [Wiegers05, Kap. 13]. Die Art der Realisierung eines Wegwerfprototyps kann – je nach seiner Verwendung – von dem fertigen System abweichen. Es können andere Plattformen und Werkzeuge für seine Realisierung verwendet werden als bei dem Zielsystem. Vor allem wird hier zwischen elektronischen und sogenannten Papierprototypen unterschieden, die meistens aus skizzierten Teilen der Benutzeroberfläche bestehen.

In der Anforderungsanalyse werden Prototypen eingesetzt, um Anforderungen zu spezifizieren und zu validieren. Dadurch wird das Risiko vermindert, durch unvollständige oder falsch verstandene Anforderungen ein falsches System zu entwickeln. In diesem Zusammenhang werden vor allem horizontale Prototypen eingesetzt. Durch vertikale Prototypen kann vor allem die technische Realisierbarkeit von Anforderungen überprüft werden, wie die nichtfunktionalen Anforderungen Performanz und Zuverlässigkeit. Evolutionäre Prototypen können sowohl horizontaler wie auch vertikaler Ausprägung sein und dienen gleichzeitig als Basis für das auszuliefernde System. Bei den agilen Methoden wird meistens vertikalen evolutionären Prototypen der Vorzug gegeben, z. B. bei XP (*Extreme Programming*) durch die Technik ≫Kurze Releasezyklen≪ [WRL05, 2.5]. Details zu den verschiedenen Anwendungsmöglichkeiten von Prototypen findet sich in [Wiegers05, Kap. 13].

Der Einsatz von Prototypen in der Anforderungsanalyse hat immer einen kommunikativen Aspekt und steht daher in Zusammenhang mit den kollaborativen Modellierungsverfahren. Dabei können sie auch dazu dienen, die Auftraggeber und Anwender zur Mitarbeit zu bewegen, indem sie Neugier und Spieltrieb wecken [Rupp04, 3.3.5].

4.4 Modellierungsverfahren für Domänenontologien

Im Zusammenhang mit der Forschung an ontologiebasierten Informationssystemen wurde eine Vielzahl von Methoden und Prozessen zur Entwicklung von Ontologien vorgeschlagen. Eine Übersicht über einige Verfahren findet sich in [GFC04]. Die Verfahren unterscheiden sich wesentlich in Bezug auf ihren Einsatzbereich: Einige wurden in Hinblick auf die Vision des Semantischen Web [BHL01] entwickelt, andere zielen auf die Entwicklung betrieblicher Informationssysteme ab. In dem gleichen Zusammenhang decken einige

Verfahren einen vollständigen Softwareentwicklungsprozess ab, während viele sich auf die Erstellung einer Ontologie beschränken.

Der Prozess der Formalisierung, der in dieser Arbeit von besonderem Interesse ist, wird nur in wenigen Methoden erläutert [GFC04, Tabelle 3.12]. Die empfohlene Vorgehensweise basiert meistens auf einer Variante der linguistischen Analyse. So wird in der Methode von Grüninger und Fox [GFC04, 3.3.3] das Verfahren auf Prädikatenlogik erster Ordung als Ontologiesprache angepasst. In dieser Methode kommen sogenannte *Kompetenzfragen* (*competency questions*) zum Einsatz: Das sind natürlichsprachlich formulierte Fragen, die mit Hilfe der Ontologie beantwortet werden sollen. Sie dienen als eine Anforderungsspezifikation für die Ontologie und als Ausgangspunkt für die linguistische Analyse. Darüber hinaus werden die Kompetenzfragen formalisiert und dienen der Herleitung von Vollständigkeitsaxiomen.

Zwei wichtige Probleme bei der Erstellung von Ontologien sind die komplizierte Wiederverwendung bestehender Ontologien und die Komplexität der Ontologiesprachen, die für Domänenexperten unverständlich sind. In dem Prozessframework *Methontology* [GFC04, 3.3.5] kommt daher der Synthese und Integration verschiedener Ontologien besondere Bedeutung zu. Das Problem der komplexen Ontologiesprachen wird durch Einführung von Zwischenrepräsentationen (*intermediate representations*) gelöst. Das Domänenwissen wird in Form von Tabellen und informellen grafischen Repräsentationen beschrieben. Dies geschieht zusammen mit Domänenexperten während einer Aktivität, die Konzeptualisierung (*conceptualization*) genannt wird. Darauf folgt die Formalisierung (*formalization*), die Übertragung des konzeptualisierten Domänenwissens in die gewählte Ontologiesprache.

Im Folgenden werden zwei Modellierungsprozesse detaillierterer beschrieben, die im Rahmen dieser Arbeit besonders interessant sind.

4.4.1 Kollaborative Methode nach Holsapple und Joshi

Ein grundlegender Ansatz zur kollaborativen Ontologieerstellung wird in [HJ02] beschrieben. Die Autoren schlagen vor, eine Variante der Delphi-Methode auf die Modellierungsaufgabe anzuwenden. Die Delphi-Methode [LT75] ist ein Verfahren, das Gruppenarbeit auch in verteilten Umgebungen ermöglicht. Grundlegendes Prinzip ist es, die Kommunikation unter den Teilnehmern so zu kontrollieren, dass ein objektiver und produktiver Umgang mit dem Thema gefördert wird. Dazu werden die Meinungen aller Teilnehmer zusammengefasst und anonymisiert der Gruppe präsentiert, um einen unvoreingenommenen Gruppenlernprozess auszulösen. Die Delphi-Methode wird z. B. erfolgreich auf Schätzverfahren angewendet, indem die Schätzungen mehrerer Experten zu einem Konsens geführt werden.

Ziel des Ontologieentwurfes bei [HJ02] ist die Einigung aller Teilnehmer auf eine Ontologie, wobei Wert auf unterschiedliche Wissenshintergründe der Teilnehmer im Fachgebiet gelegt wird. Dazu wird eine Ausgangsontologie an alle Teilnehmer des Prozesses verteilt, die über einen Fragebogen die Ursprungsontologie bewerten und Änderungsvorschläge unterbreiten. Die Fragebögen beinhalten sowohl frei formulierbare Fragen wie auch Bewertungen nach einer Likert-Skala, durch die Antworten auf qualitative Fragen (z. B. von 1≡*stimme vollkommen zu* bis 9≡*lehne vollkommen ab*) bewertet werden. Eine Zusammenfassung der Kommentare und Änderungsvorschläge wird angefertigt und auf dieser

Grundlage eine zweite, verbesserte Ontologie erstellt, die zusammen mit der Zusammenfassung und einem weiteren Fragebogen an die Teilnehmer verteilt wird. Dieser Prozess wird wiederholt, bis die Ontologie den Bedürfnissen der Teilnehmer entspricht.

Bei diesem Verfahren überwiegt die Anwendung des kollaborativen Modellierungsansatzes. Allerdings muss noch eine weitere Methode angewendet werden, da zur Anwendung der Delphi-Methode eine Ursprungsontologie benötigt wird. Die Wahl einer geeigneten Ursprungsontologie kann große Auswirkungen auf die Effizienz des Verfahrens haben. Die empfohlenen Phasen des Prozesses werden im folgenden erläutert:

1. **Vorbereitung** (*preparation*): Zunächst werden die Entwurfskriterien, Bewertungsstandards und Randbedingungen (Grenzen) der Ontologie festgelegt. Als Entwurfskriterien werden hier generelle Ansprüche an die Ontologie verstanden, wie z. B. Vollständigkeit und Korrektheit.

2. **Verankerung** (*anchoring*): Eine initiale Ontologie wird erstellt, die als Ausgangspunkt für die iterative Verbesserung dient. Es wird hierfür eine sehr umfassende und allgemeine Ontologie empfohlen, wie sie durch einen synthetisierenden Ansatz gewonnen werden kann.

3. **Iterative Verbesserung** (*iterative improvement*): Die Delphi-Methode wird angewendet, indem systematisch die Meinungen der einzelnen Teilnehmer erfasst, klassifiziert und kategorisiert werden. Die quantitativ relevanten Meinungen werden zusammengefasst und zur Erstellung einer verbesserten Ontologie verwendet. Solange kein Konsens erreicht ist, wird auf Grundlage der Zusammenfassung und der verbesserten Ontologie eine weitere Iteration eingeleitet.

4. **Anwendung** (*application*): Die durch die Delphi-Methode gewonnene Ontologie stellt in der Anwendung ihre Qualität unter Beweis.

4.4.2 On-To-Knowledge Metaprozess

Ziel des On-To-Knowledge Projektes [SSSS01][GFC04, 3.3.7] ist es, das Wissensmanagement innerhalb von Unternehmen zu verbessern. Dazu werden in dem Projekt ein Prozess und ein Metaprozess zum Wissensmanagement (*knowledge management*) angewendet, die durch verschiedene Werkzeuge unterstützt werden. Der Wissensprozess (*knowledge process*) beschreibt, wie in einem Unternehmen auf wirtschaftliche Weise ontologiegestütztes Wissensmanagement eingesetzt werden kann. Parallel wird ein Wissens-Metaprozess (*knowledge metaprocess*) angewendet, um die Einführung und Wartung von Wissensmanagementsystemen zu unterstützen.

Der Wissens-Metaprozess, dargestellt in Abbildung 4.3, beschreibt die Entwicklung einer Ontologie als Teil einer Anwendungsentwicklung für Wissensmanagement Applikationen (CommonKADS [SAA99]). Die entstehenden Domänenontologien sind spezifisch auf die geplante Anwendung zugeschnitten. Im Metaprozess werden bewährte Verfahren des Softwareengineering und der Ontologieentwicklung kombiniert angewendet, wie z. B. Anwendungsfälle (siehe Abschnitt 4.3.3), die Festlegung des Systemkontextes (4.3.1) und informelle Kompetenzfragen. Die Kompetenzfragen dienen als Anforderung im Sinne der Spezifikation und Evaluierung von Systemfunktionen und dienen damit auch als Abnahmekriterium. Darüber hinaus wird Wert auf die Integration von Domänenexperten in

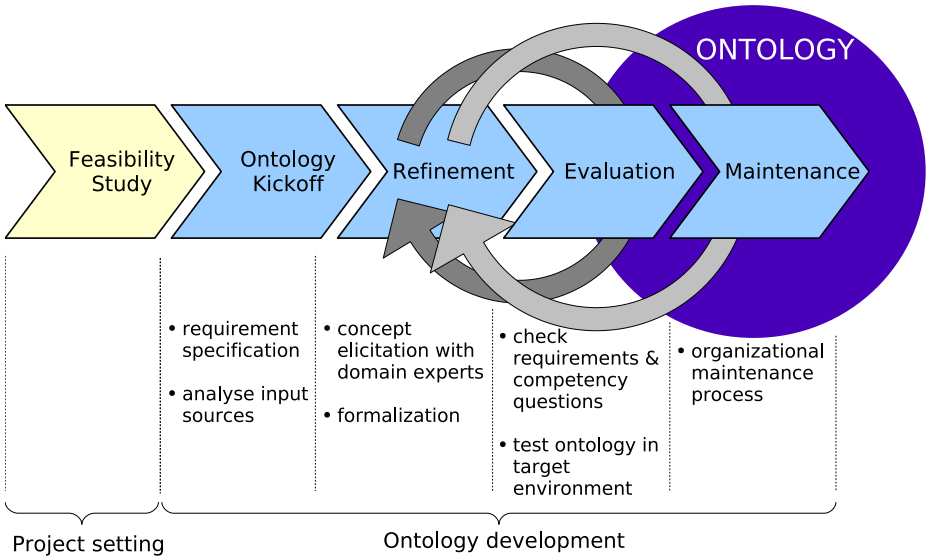

Abbildung 4.3: On-To-Knowledge Metaprozess nach [SSSS01]

den Prozess gelegt. Eine Beschreibung des Prozesses folgt, wobei zu beachten ist, dass die Phasen 2 und 3 iterativ angewendet werden:

1. **Durchführbarkeitsstudie** (*feasibility study*)**:** Ziel dieser Phase ist es, die technische und wirtschaftliche Durchführbarkeit der ontologiegestützten Anwendung zu gewährleisten. Dabei sollen mögliche Probleme und Potentiale aufgedeckt werden. Diese Phase ist nicht Teil der Ontologieentwicklung sondern des Softwareentwicklungsprozesses. Sie stellt allerdings den Bezug zwischen der Software- und der Ontologieentwicklung her.

2. **Vorarbeit** (*ontology kickoff*)**:** Anschließend werden Anforderungen an die Ontologie und Richtlinien erarbeitet sowie wiederverwendbare Ontologien, Wissensquellen, Akteure und Anwendungsfälle identifiziert. Letztere werden nur oberflächlich beschrieben und durch Kompetenzfragen ergänzt.

3. **Verfeinerung** (*refinement*)**:** Zunächst werden aus den Anforderungen heraus die wichtigsten Domänenkonzepte identifiziert und informell zusammengestellt. Anschließend wird ein iterativer Prozess eingeleitet, durch den die gesammelten Konzepte schrittweise in eine ausgereifte und anwendungsspezifische Ontologie überführt werden. Auch die Integration von wiederverwendbaren Ontologien findet in diesem Schritt statt. Bei dem iterativen Vorgehen wird zwischen zwei Tätigkeiten unterschieden:

   - Die Erhebung von Domänenwissen geschieht gemeinsam mit Domänenexperten, zuvor identifizierte Konzepte dienen dabei als Orientierung.
   - Die Formalisierung, die Übertragung der Domänenkonzepte in die gewünschte Ontologiesprache, wie z. B. Beschreibungslogik [BCM03]

4. **Evaluation** (*evaluation*)**:** Durch die Evaluation soll sichergestellt werden, dass die Ontologie und ihre Anwendungsumgebung die an sie gestellten Anforderungen er-

füllen. Dazu wird zum einen überprüft, ob die zuvor formulierten Kompetenzfragen durch die Ontologie beantwortet werden können. Zum anderen wird die Ontologie in ihrer Anwendungsumgebung getestet. Aus diesen Tests werden durch Einbeziehung von den Anwendern und Domänenexperten Anregungen für die weitere Verfeinerung der Ontologie gewonnen. Dieses Vorgehen gleicht dem Einsatz von evolutionären Prototypen in der Anforderungsananlyse, siehe Abschnitt 4.3.4.

5. Pflege (*maintenance*)**:** Da veränderliche Umstände der Realität in der Ontologie berücksichtigt werden müssen, ist die Pflege der Ontologie notwendiger Bestandteil der Systempflege. Das Verändern der Ontologie muss kontrolliert geschehen und ist demnach vorwiegend ein organisatorischer Prozess. Es wird empfohlen, Änderungen zu sammeln und nach gründlicher Evaluierung die Anwendung auf eine neue Version der Ontologie umzustellen.

Der Prozess hat durch die Einbeziehung von Domänenexperten einen kollaborativen Charakter, wobei synthetisierende und deduktive Elemente in Ansätzen ebenfalls vorhanden sind. Allerdings muss angemerkt werden, dass das kollaborative Element bei weitem nicht so stark ausgeprägt ist wie z. B. bei dem Verfahren von Holsapple und Joshi (siehe Abschnitt 4.4.1) und die Kollaborationsmechanismen zudem nur oberflächlich beschrieben sind [SSSS01]. Die meisten Aktivitäten im Prozess können nur von Ontologieexperten durchgeführt werden.

Interessant ist der Prozess für diese Arbeit vor allem deshalb, weil er zeigt, wie Domänenmodelle als Teil einer Anwendungsentwicklung erstellt werden können. Viele bewährte Verfahren aus anderen Disziplinen wie die Festlegung eines Systemkontextes und Anwendungsfälle werden mit Verfahren aus der Ontologieentwicklung wie den Kompetenzfragen kombiniert angewendet. Darüber hinaus wird auch auf eine umfassende Werkzeugunterstützung des Prozesses Wert gelegt, Details sind in [SSSS01] zu finden.

4.5 Konzeptuelle Modellierung

Auch in der Literatur zur Konzeptuellen Modellierung findet sich eine Modellierungsmethodik. In [BBJW97, Kap. 6] wird die Modellierung als kollaborative Tätigkeit zwischen Domänenexperten und Modellierungsexperten verstanden, die in einem Workshop (*modelling seminar*) durchgeführt werden soll. Für den Workshop werden folgende Tätigkeiten empfohlen:

Schulung (*education*)**:** Die Teilnehmer, vor allem die Domänenexperten, in Modellierungsverfahren schulen.

Ziel und Rahmen festlegen (*checking focus and demarcations*)**:** Festlegung was modelliert werden soll - und was nicht.

Konzepte und Instanzen sammeln (*listing concepts and objects*)**:** Kann mit Hilfe der linguistischen Analyse durchgeführt werden.

Modellierung (*modelling*)**:** Die im vorigen Schritt identifizierten Konzepte werden miteinander in Beziehung gesetzt und um Attribute ergänzt. Analysemuster (*prototypical object structures* in [BBJW97]) werden als Hilfe eingesetzt.

Die in [BBJW97] erwähnten Aktivitäten werden in anderen Abschnitten dieses Kapitels behandelt: z. B. Analysemuster in 4.2.1, linguistische Analyse in 4.2.2, Rahmenfestlegung in 4.3.1 und Workshops in Abschnitt 4.3.2. Alle diese Verfahren werden in der Literatur zur objektorientierten Analyse und Anforderungsanalyse wesentlich umfassender behandelt, weshalb hier auf eine Wiederholung verzichtet wird.

Da die konzeptuelle Modellierung ihren Ursprung in der Datenbankmodellierung hat, liegt in der Literatur der Schwerpunkt meistens auf theoretischen Aspekten. Auch Technologien zur Anwendung konzeptueller Modelle werden häufig diskutiert. Den Formalisierungsprozessen kommt eine untergeordnete Bedeutung zu, sie werden häufig im Rahmen von Softwareentwicklungsprozessen beschrieben. So wird z. B. in [BMS84, Kap 2:5.4] auf die „Information Systems Design Methodologies“ Konferenz verwiesen. In [Brandt83] findet sich eine Übersicht über die Zusammenhänge zwischen verschiedenen Entwicklungsprozessen und ihr Verhältnis zu Datenmodellen, die teilweise auch in der Konzeptuellen Modellierung verwendet werden.

Kapitel 5

Prozessentwurf

Die Vorgehensweise bei der konzeptuellen Modellierung durch Domänenexperten wird durch den Prozess zur beispielgetriebenen Modellierung beschrieben. Dieser Modellierungsprozess ist Teil eines übergeordneten Prozesses zur Erstellung von konzeptorientierten Inhaltsverwaltungssystemen.

5.1 Anforderungen an die Prozesse

5.1.1 Systemerstellung und Modellierung

Da die Modellierung durch die Domänenexperten (DEn) Teil der Entwicklung eines konzeptorientierten Inhaltsverwaltungssystems ist, werden in diesem Kapitel zwei Prozesse beschrieben: Zunächst wird ein Systemerstellungsprozess für CCM Systeme skizziert. Anschließend wird eine Teilaktivität dieses Prozesses im Detail erläutert, nämlich der Prozess zur beispielgetriebenen Modellierung, durch den es Domänenexperten ermöglicht wird, selbstständig konzeptuelle Modelle zu formulieren.

Im Zentrum dieser Arbeit steht der Modellierungsprozess. Die Beschreibung des Systemerstellungsprozesses ist jedoch notwendig, um zu verstehen, wie der Modellierungsprozess mit den anderen Aktivitäten der Entwicklung von CCM Systemen in Beziehung steht. Der Kontext für die Modellierung wird zum einen durch einen Projektrahmen festgelegt, in dem die Einführung eines konzeptorientierten Inhaltsverwaltungssystems beschlossen wird. Zum anderen werden, bedingt durch den momentanen Entwicklungsstand der CCM Generatoren, projektspezifische Generatoren und Generatorenkonfigurationen benötigt. Diese haben Einfluß auf den in Abschnitt 5.3 beschriebenen Modellierungsprozess.

5.1.2 Prozessvarianten und Prozessframework

Konzeptorientierte Inhaltsverwaltungssysteme können in unterschiedlichen technischen und organisationalen Umgebungen eingesetzt werden. Mit dem Einsatzzweck ändern sich auch die Umstände für den Systemerstellungs- und den Modellierungsprozess. Diesen Variabilitäten muss durch angepasste Prozesse Rechnung getragen werden. Im Folgenden sollen einige Einflussfaktoren aus dem Umfeld der CCM Systementwicklung erläutert

werden. Darüber hinaus werden Beispiele gegeben, inwieweit durch diese Faktoren auch der Systemerstellungs- und der Modellierungsprozess beeinflusst werden.

Anwenderkreis: Grundsätzlich handelt es sich bei CCM Systemen um kollaborative Informationssysteme. Die Anzahl der Anwender und ihr fachlicher Hintergrund beeinflussen die Systemerstellung und Modellierung. Soll das System von Anwendern unterschiedlicher Wissensgebiete genutzt werden, ist es sinnvoll, Vertreter sämtlicher Fachrichtungen in die Modellierung einzubeziehen. Der andere Extremfall ist die individuelle Personalisierung, in diesem Fall wird das System nur von einem einzigen Anwender verwendet.

Bestehende CCM Systeme: Die Besonderheit der konzeptorientierten Inhaltsverwaltungssysteme, die Möglichkeit zur Offenheit und Dynamik (siehe Abschnitt 2.1), beeinflusst alle Aktivitäten der Systemerstellung, auch die Modellierung. Im Vergleich zu der Erstellung eines neuen CCM Systems ist der Aufwand für die Realisierung projektspezifischer Generatoren und Konfigurationen bei einer Evolution oder Personalisierung als gering einzustufen. Auch die Modellierung gestaltet sich einfacher, da bereits auf ein Schema zurückgegriffen werden kann. Allerdings muss sichergestellt werden, dass der Austausch von Informationen zwischen dem Ausgangssystem und dem System mit personalisiertem Schema möglich ist.

Erfahrung der Anwender und Domänenexperten: Da die konzeptuelle Modellierung von den DEn durchgeführt wird, ist deren bisherige Modellierungserfahrung eine entscheidende Variable. Es ist offensichtlich, dass DEn in der Modellierungsmethode und den verwendeten Werkzeugen geschult werden müssen, sofern diese Aufgaben für sie neu sind. Der Umfang der nötigen Schulung und Beratung hängt zudem davon ab, ob die DEn ein neues CCM System modellieren oder ob ein bestehendes personalisiert oder angepasst werden soll. In dem letzterem Fall können die DEn von Erfahrungen mit dem Bestandssystem profitieren.

Technologie: Die Wahl der Technologien hat Einfluss auf den Systemerstellungsprozess. Vor allem die Wahl zwischen einer webbasierten oder einer grafischen Benutzeroberfläche beeinflusst den Aufwand zur Inbetriebnahme bzw. Installation des CCM Systems.

Aufgrund dieser vielen variablen Faktoren reicht ein einzelner Prozess für die Modellierung und die Systemerstellung nicht aus. Stattdessen wird im Folgenden ein Prozessframework zur konzeptuellen Modellierung und eines zur Entwicklung von CCM Systemen entworfen. Die konkrete Instanziierung der Prozesse geschieht ähnlich wie beim Rational Unified Process (RUP, [Kruchten00]): Durch das Framework wird ein umfassender Prozess beschrieben, durch den alle Eventualitäten abgedeckt werden. Im Einzelfall ist zu entscheiden, ob bestimmte Aktivitäten und Artefakte wirklich sinnvoll sind. Alle optionalen Prozessschritte, deren Nutzen im Verhältnis zu ihrem Aufwand vorraussichtlich gering ist, sollten ausgelassen werden.

5.1.3 Zielsetzung für den Modellierungsprozess

Das Ziel dieser Arbeit ist es, dass Domänenexperten eigenständig konzeptuelle Modelle von hoher Qualität erstellen können. Vor allem für die konzeptorientierte Inhaltsverwal-

tung ist dies erstrebenswert, da die Anwender selbst die CCM Systeme ihren Bedürfnissen anpassen sollen. Erfahrungen mit CCM Systemen haben jedoch gezeigt, dass es Anwendern ohne naturwissenschaftlichen Hintergrund schwerfällt, klassenbasierte Schemasprachen wie die Assetdefinitionssprache sinnvoll einzusetzen (siehe Abschnitt 2.4). Nur wenn es den Anwendern möglich ist, selbstständig konzeptuelle Modelle zu erstellen, können diese die Vorteile der CCM Systeme ausnutzen, nämlich die Möglichkeit zur Offenheit und Dynamik. Vor diesem Hintergrund ergeben sich zwei Ziele für den Modellierungsprozess:

1 – Eigenständige Modellierung durch Domänenexperten: Wichtigstes Ziel des Modellierungsprozesses ist es, dass die Modellierung von den DEn selbst durchgeführt wird. Dabei soll, soweit es möglich ist, auf die Unterstützung von Modellierungsexperten verzichtet werden. Dieses steht im Gegensatz zu allen in Kapitel 4 beschriebenen Modellierungsmethoden, bei denen die DEn lediglich als eine Informationsquelle betrachtet werden.

2 – Umfassende Werkzeugunterstützung: Damit die DEn die Modellierung durchführen können, benötigen sie Unterstützung. Diese erhalten sie am besten in Form einer Werkbank, um nicht von Modellierungsexperten abhängig zu sein. Bei dem Prozessentwurf soll darauf geachtet werden, dass sich alle Aktivitäten und Artefakte durch ein einfach bedienbares Werkzeug unterstützen lassen.

Die Ziele stellen ein idealisiertes Bild dar. Es ist unwahrscheinlich, dass es für jeden potenziellen Anwender möglich ist, ein neues CCM System zu erstellen, da die Modellierung für die meisten Benutzer keine alltägliche Aufgabe ist.

5.2 CCM Systemerstellungsprozess

In diesem Abschnitt wird ein Prozess zur Erstellung und Entwicklung von konzeptorientierten Inhaltsverwaltungssystemen präsentiert. Die Beschreibung deckt sich weitgehend mit dem bislang informellen Vorgehen, das bei der Entwicklung bestehender CCM Systeme angewendet wurde. [GKNS, BSH06] Da der Modellierungsprozess in dieser Arbeit besonders wichtig ist, wird dieser als einziger Bestandteil des Systemerstellungsprozesses detailliert beschrieben. Die anderen Tätigkeiten werden lediglich kurz dargestellt, um das Umfeld und die Abhängigkeiten der Modellierung festzuhalten.

Die Systemerstellung wird hier im Kontext einer Organisation, wie z. B. einem Unternehmen oder einem Institut, beschrieben. Diese Annahme ist für die Anwendung der meisten Inhaltsverwaltungssysteme erfüllt. Die bestehenden CCM Systeme werden jedoch von heterogenen Nutzergruppen verwendet, die keiner gemeinsamen Organisation angehören, da sie in überregionalen und interdisziplinären Forschungsprojekten eingesetzt werden. Eine Systemerstellung ist auch in einem solchen Rahmen möglich, allerdings wird durch diesen Umstand die Planung erschwert.

In Abbildung 5.1 werden die Prozessschritte und Unterprozesse des CCM Systemerstellungsprozesses in zeitlicher Abfolge dargestellt. Lediglich der Modellierungsprozess und die Inbetriebnahme sind für alle CCM Projekte verbindlich. Auf eine Projektplanung oder den Infrastrukturprozess kann verzichtet werden, wenn es sich bei dem Projekt

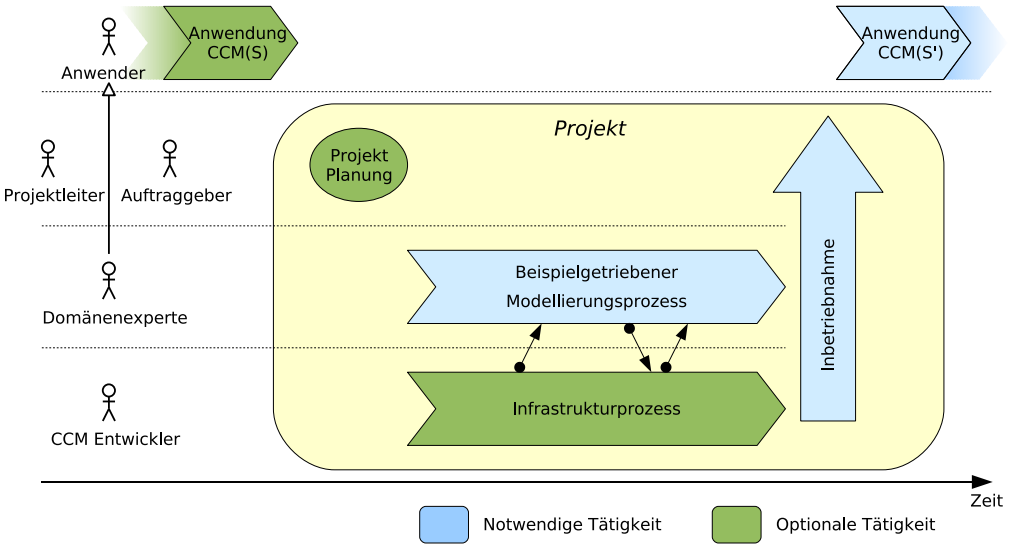

Abbildung 5.1: CCM Systemerstellungsprozess

um die Evolution oder Personalisierung eines bestehenden Systems handelt. Die Abbildung ist vereinfacht und zeigt aus Gründen der Übersicht nur die wichtigsten Akteure.

5.2.1 Prozessschritt: Projektplanung

Aktivitäten

Während der Projektplanung wird der Einsatz eines konzeptorientierten Inhaltsverwaltungssystems beschlossen. Daher ist die Durchführung einer Machbarkeitsstudie entscheidend, in der die technische und wirtschaftliche Durchführbarkeit der Systementwicklung und des Systemeinsatzes geprüft werden. Bevor der (wirtschaftliche) Nutzen des Systems festgestellt werden kann, muss eine Vision für den Systemeinsatz entwickelt werden. Durch die Vision kann der erwartete Nutzen des Systems den veranschlagten Kosten gegenübergestellt werden. Systemvisionen und Projektrahmen wurden bereits in Abschnitt 4.3.1 erläutert.

Die Projektplanung kann im Rahmen eines Prozesses zum organisationalen Wissensmanagement durchgeführt werden. In diesem Fall werden verschiedene Maßnahmen auf Chancen und Risiken überprüft, durch die der Umgang mit Informationen innerhalb der Organisation verbessert werden kann. Die Eignung von CCM Systemen zum Wissensmanagement wird in Abschnitt 5.2.6 diskutiert.

Artefakte

Neben der Entscheidung für den Einsatz eines CCM Systems sind zwei Artefakte der Planungsphase für den weiteren Prozess besonders wichtig: ein Projektrahmen und ein Projektplan. In dem Projektrahmen werden die Budgets und die Zielvorgaben spezifiziert. Die Zielvorgaben für das Projekt greifen einen Teil der Systemvision heraus, der innerhalb des Projektes mit dem verfügbaren Budget realisiert werden soll.

Der Projektplan besteht aus einer Zeitplanung, einem Kostenplan und einem Personalplan. Ein gesonderter Personalplan ist sinnvoll, da eine sorgfältige Auswahl der DEn für die Modellierung eines kollaborativen Informationssystems entscheidend ist. Darüber hinaus gehören die DEn der Organisation des Auftraggebers an, der sie für die Systemmodellierung von ihrer alltäglichen Arbeit freistellen muss.

Akteure

Die beiden entscheidenden Akteure für die Projektplanung sind der Auftraggeber selbst und der Projektleiter. Der Auftraggeber besitzt die Autorität, den Einsatz des Systems in der Organisation zu beschließen und die finanziellen und personellen Mittel zur Entwicklung bereitzustellen. Ihm steht der Projektleiter zur Seite, der die technische Machbarkeit prüft und den finanziellen Umfang des Projektes ermittelt. Es ist auch möglich, dass die Rolle des Projektleiters durch mehrere Personen übernommen wird, z. B. einen CCM Spezialisten und einem organisationsinternen Projektleiter. Dies kann von Vorteil sein, wenn die Rekrutierung der DEn kompliziert ist.

Darüber hinaus können viele verschiedene Personen an der Machbarkeitsstudie beteiligt sein, wenn diese in einem größeren Rahmen durchgeführt werden muss, z. B. Unternehmensberater, Ethnographen und andere.

5.2.2 Unterprozess: Infrastrukturprozess

Aktivitäten

Die Aufgabe des Infrastrukturprozesses ist es, alle Bestandteile eines konzeptorientierten Inhaltsverwaltungssystems bereitzustellen, abgesehen vom konzeptuellen Modell, das im Modellierungsprozess entwickelt wird. Als *Infrastruktur* wird in diesem Zusammenhang nicht nur die technische Infrastruktur wie Computer, Systemsoftware und Netzwerke bezeichnet. Auch der Modellkompiler, die Generatoren und die benötigten Standardkomponenten sind aus Sicht der Anwender und Domänenexperten eine Form von Infrastruktur. Vor allem wenn auf kein bestehendes CCM System zurückgegriffen werden kann, ist es notwendig, die Funktionsfähigkeit der neuen Infrastruktur durch Probeläufe sicherzustellen.

Innerhalb des Infrastrukturprozesses wird Hardware bereitgestellt, der Modellkompiler konfiguriert und wenn nötig projektspezifische Generatoren entwickelt. Allerdings werden diese aus ökonomischen Gründen erst dann erstellt, wenn ein vorläufiges Schema und zuverlässige Anforderungen aus dem Modellierungsprozess vorliegen. Darüber können auch bestimmte Schemata innerhalb des Infrastrukturprozesses entwickelt werden, sofern das für die DEn zu aufwendig ist. Ein Beispiel ist die konzeptuelle Modellierung von Benutzeroberflächen [Mofor06]. Da hierfür Fachwissen im Oberflächenentwurf notwendig ist, bietet sich die Modellierung durch einen CCM Entwickler unter Beteiligung von DEn an. Der Aufwand für die Schulung der DEn im Oberflächenentwurf kann unverhältnismäßig hoch sein.

Da die Bereitstellung der Infrastruktur im wesentlichen aus der Entwicklung und Anpassung von Software besteht, können hierfür gängige Methoden und Prozesse des Softwareengineerings angewendet werden, wie z. B. Extreme Programming [WRL05] oder der Rational Unified Process [Kruchten00]. Besonders ähnlich ist der Infrastrukturprozess

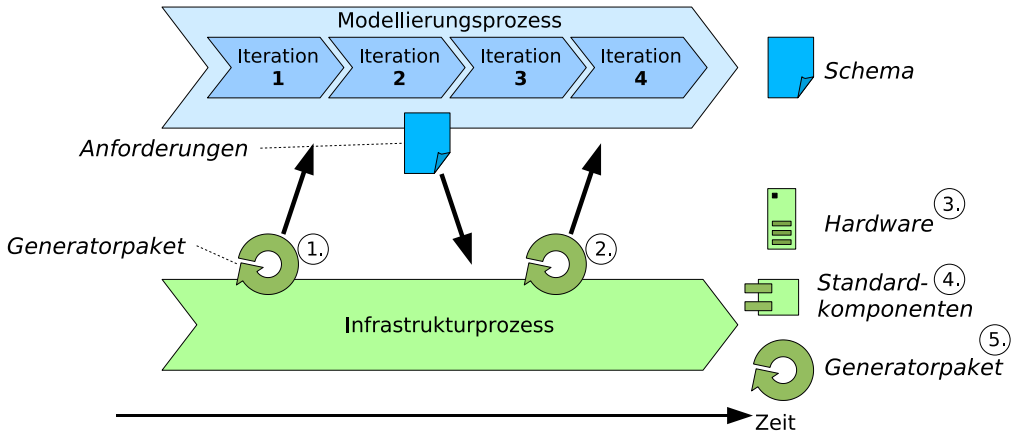

Abbildung 5.2: Artefakte vom Infrastruktur und Modellierungsprozess

der Domänenentwicklung, einer Teildisziplin der modellgetriebenen Softwareentwicklung (siehe Abschnitt 5.2.5).

Artefakte

Um die Artefakte zu erläutern, soll hier ein weiterer Begriff eingeführt werden: Unter einem *Generatorpaket* wird in dieser Arbeit die einsatzfähige Gesamtheit von Generatoren, zugehöriger Standardkomponenten und einer entsprechenden Konfiguration für den CCM Modellkompiler verstanden. Die Artefakte und Abhängigkeiten zwischen Infrastruktur- und Modellierungsprozess veranschaulicht Abbildung 5.2. Es folgt eine Diskussion der im Infrastrukturprozess erstellten Artefakte:

1. Zuerst wird ein *Generatorpaket für Prototypen* erstellt. Bei der Erstellung eines neuen CCM Systems sind darin keine projektspezifischen Generatoren enthalten. Durch dieses Generatorpaket soll kein Produktivsystem erzeugt werden, sondern Prototypen, mit denen im Modellierungsprozess das Schema evaluiert werden kann. Wird auf ein bestehendes System zurückgegriffen, ist es wünschenswert, durch die Prototypen auf die bestehenden Datenbestände zugreifen zu können.

2. Bei dem momentanen Entwicklungsstand des CCM Modellkompilers sind *projektspezifische Generatoren* nötig, die dem Modellierungsprozess in Form eines erweiterten Generatorpaketes zur Verfügung gestellt werden.

3. Die *technische Infrastruktur* für das Produktivsystem umfasst die nötige Hardware und Systemsoftware.

4. Viele der generierten Module benötigen *Standardkomponenten*, wie z. B. eine Laufzeitumgebung, Datenbanksysteme oder Middleware.

5. Das *Generatorpaket für das Produktivsystem* enthält auch Generatoren, die sich nicht für die Generierung von Prototypen eignen. Zum Beispiel benötigen bestimmte Klientenmodule (*client modules* [SBS05]) ein Datenbanksystem, desses Lizenzkosten und Administrationsaufwand einen Einsatz im Modellierungsprozess verbieten.

Darüber hinaus ist der Infrastrukturprozess von Anforderungen abhängig, um die Eigenheiten des Projektes in den Generatorpaketen berücksichtigen zu können. Es sei angemerkt, dass diese Betrachtung und die Abbildung 5.2 vereinfacht sind. Unter Umständen greifen die Entwickler für Probeläufe auf das aktuelle Schema des Modellierungsprozess zurück. Dieses ist vor allem für projektspezifische Generatoren für Mediations- und Abbildungsmodule (*mediation modules, mapping modules*, siehe Abs. 2.3.2) sinnvoll, die besondere Anforderungen an die Schemata stellen.

Akteure

Da der Infrastrukturprozess im Wesentlichen eine Softwareentwicklung ist, finden sich hier alle Rollen des Softwareengineerings wieder, wie z. B. Projektleiter, Architekten, Entwickler und Tester. Spezialisten für die Anforderungsanalyse werden allerdings nicht benötigt, da die Anforderungen an projektspezifische Generatoren im Modellierungsprozess ermittelt werden.

Durch die vollständige Systemgenerierung sind hier Spezialisten mit besonderen Kenntnissen gefragt. Für die üblichen Projekte sollten wenige Entwickler und ein Projektleiter ausreichend sein, da die meisten Anforderungen durch generische Generatoren befriedigt werden können. Die Projektleitung kann auch vom Projektleiter des Modellierungsprozesses übernommen werden.

5.2.3 Unterprozess: Beispielgetriebene Modellierung

Die Erstellung eines CCM Systems kann mit verschiedenen Modellierungsmethoden durchgeführt werden. In der Vergangenheit wurde eine objektorientierte Modellierungsmethode angewandt, auch in Kombination mit einer Delphi-Methode ähnlich dem Verfahren von Holsapple und Joshi (siehe Abschnitt 4.4.1). Hier wird der im Unterkapitel 5.3 beschriebene Prozess zur beispielgetriebenen Modellierung (BGM) kurz erläutert, der sich zur Modellierung konzeptorientierter Inhaltsverwaltungssysteme besonders eignet.

Aktivitäten

Der hier beschriebene inkrementelle Modellierungsprozess unterscheidet sich wesentlich von traditionellen Modellierungsansätzen wie z. B. denen der Objektorientierung: Die Modellierung geschieht bei der BGM auf extensionaler Ebene (siehe auch Abs. 2.2.2). Das bedeutet, dass hier Konzepte in Form von Beispielmengen definiert werden, aus denen eine intensionale Konzeptbeschreibung abgeleitet wird, wie sie in der ADL verwendet wird (siehe Abs. 2.2.1). Dies wird in Abbildung 5.3 veranschaulicht. Eine direkte Manipulation des intensionales Modells durch die DEn ist auch gestattet, soll jedoch die Ausnahme bleiben. Eine Diskussion, warum ein solches Modellierungsverfahren adäquat für die Modellierung durch DEn ist, findet sich in Abschnitt 5.3.4.

Die DEn benötigen Unterstützung bei der Modellierung, da diese Tätigkeit nicht zu ihren üblichen Aufgaben gehört. Neben einer Schulung in Methode und Werkzeug der BGM übernehmen Modellierungsexperten eine beratende Funktion. Darüber hinaus wird eine Projektleitung für die Modellierung eingesetzt, die z. B. Workshops organisiert und den Zeit- und Personalplan überwacht.

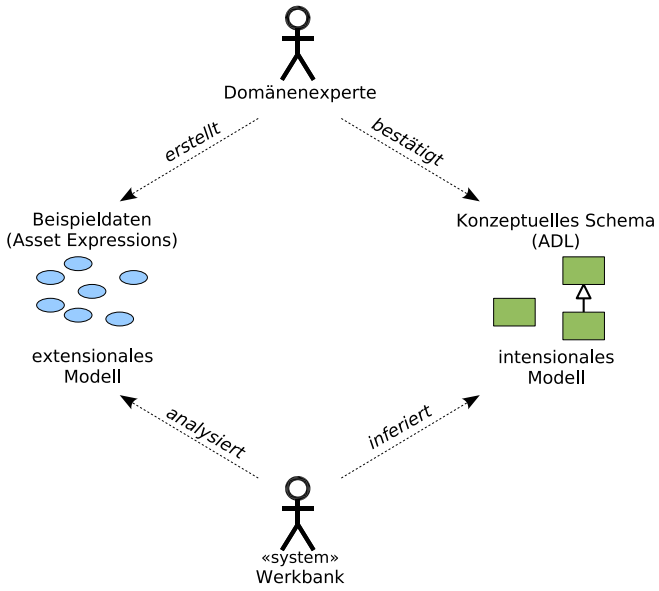

Abbildung 5.3: Ansatz der beispielgetriebenen Modellierung

Artefakte

Das Ergebnis des Modellierungsprozesses ist ein *Schema*, das die Anforderungen an das CCM System aus Expertensicht erfüllt. Im Gegensatz zu intensionalen Modellierungsverfahren werden in der BGM *Beispieldaten* erzeugt, die das Schema durch reale Beispiele stützen. Gerade bei großen Beispielmengen ist es sinnvoll, diese in das Produktivsystem zu übernehmen, da sie ein weiteres wertvolles Resultat des Prozesses sind.

Es wurde bereits angedeutet, dass die *Anforderungen zur Entwicklung projektspezifischer Generatoren* nur in dem Modellierungsprozess ermittelt werden können. Diese Anforderungen sind häufig eng mit dem Schema verbunden, das iterativ entwickelt wird. Vorläufige Versionen sind eine wichtige Informationsquelle für den Infrastrukturprozess.

Akteure

Die wichtigsten Akteure dieses Prozesses sind die *Domänenexperten* (DEn), da sie die Modellierung maßgeblich durchführen. Domänenexperten sind Anwender des geplanten Systems, oder auch des Bestandssystems, sofern eines existiert. Sie zeichnen sich durch ein unter den Anwendern herrausragendes Domänenwissen aus. Wenn das System kollaborativ genutzt werden soll, müssen sie außerdem fähig sein, konstruktiv in einem Team zu arbeiten, da die Modellierung in diesem Fall von mehreren DEn durchgeführt wird. Die Auswahl dieser Akteure unter den Anwendern ist entscheidend für die Qualität des Schemas und den Nutzen des Systems.

Im Weiteren ist ein *Modellierungsexperte* beteiligt, der bei Bedarf die DEn schult und berät. Dieser hat idealerweise einen ähnlichen fachlichen Hintergrund, was das gegenseitige Verständnis fördert. Allerdings sind Modellierungserfahrung mit intensionalen und extensionalen Ansätzen sowie umfassende soziale Kompetenzen für diese Aufgabe essenziell.

Darüber hinaus ist ein *Projektleiter* beteiligt, der planende und überwachende Aufgaben übernimmt, wie die Organisation von Workshops, die Personalplanung und die Überwachung von Zeit- und Kostenplan. Darüber hinaus muss er die Zusammenarbeit mit dem Infrastrukturprozess koordinieren. Sowohl die Übernahme von Generatorpaketen in den Modellierungsprozess wie auch das Weiterleiten von projektspezifischen Anforderungen an den Infrastrukturprozess müssen geplant werden.

5.2.4 Prozessschritt: Inbetriebnahme

Aktivitäten

Der Zweck der Inbetriebnahme ist es, aus dem Schema ein CCM System zu generieren und dieses auf der im Infrastrukturprozess bereitgestellten Hardware zu installieren. Im einfachsten Fall muss lediglich die endgültige Systemgenerierung angestoßen werden und das Ergebnis installiert werden. Dazu gehören Tätigkeiten wie das Übertragen von Schemata auf das Datenbankmanagementsystem. In besonderen Fällen können abschließende Testläufe des Systems sinnvoll sein, z. B. wenn das CCM System für sensible Daten verwendet wird.

Artefakt

Das einzige Ergebnis dieses Prozesschrittes ist ein einsatzbereites konzeptorientiertes Inhaltsverwaltungssystem. Bei dem in dieser Arbeit entwickelten Prozess zur beispielgetriebenen Modellierung ist zu beachten, dass die während der Modellierung entstandenen Beispieldaten ebenfalls in das Produktivsystem übernommen werden können.

Akteure

Die Inbetriebnahme kann von einem CCM Entwickler gemeinsam mit dem Administrator des Produktivsystems durchgeführt werden. Da die Infrastruktur zu diesem Zeitpunkt ausreichend getestet wurde und die Systemgenerierung automatisiert abläuft, sind keine arbeitsintensiven Tätigkeiten durchzuführen. Allerdings ist es sinnvoll, auf CCM Entwickler zurückgreifen zu können, um eventuell auftretende Probleme schnell zu lösen.

5.2.5 Beziehung zur Domänenentwicklung

Die Domänenentwicklung (*domain engineering*) ist eine Teildisziplin der modellgetriebenen Softwareentwicklung, die durch generative Techniken realisiert wird. In der deutschen Literatur findet sich auch die unglückliche Bezeichnung »Domänenarchitektur-Entwicklung« [SV05], obwohl es sich hierbei um die Entwicklung einer DSL (*domain specific language*), einer Plattform und den zur Generierung verwendeten Transformationen handelt. Die (Domänen-) Architektur ist nur eines der Artefakte, die während der Prozesse erstellt werden.

Die Ähnlichkeit zwischen der CCM Systementwicklung und der modellgetriebenen Softwareentwicklung (*model driven software development, MDSD*) besteht darin, dass in beiden Verfahren Familien verwandter Systeme durch die Verwendung generativer Techniken erstellt werden. Leider wird der Begriff *Domäne* in der Domänenentwicklung nicht genau definiert [SV05, Abs. 1.1] und sehr allgemein verwendet. Mehr als eine beliebige

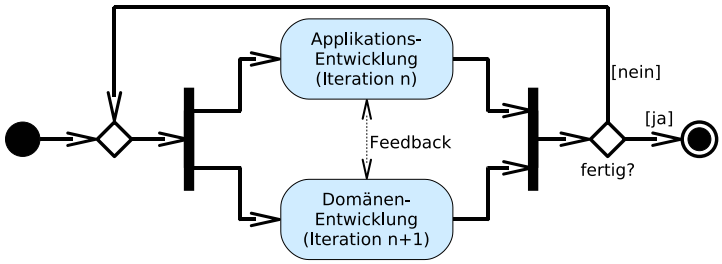

Abbildung 5.4: Iterative Domänen- und Anwendungsentwicklung nach [KK06]

Einschränkung des Anwendungsbereiches der entstehenden Systemfamilien kann aus dem Begriff nicht abgeleitet werden. Im CCM Umfeld wird mit Domäne dagegen der Wissens- und Kompetenzbereich der Anwender bezeichnet.

Prozesse zur modellgetriebenen und generativen Softwareentwicklung

Die Prozesse zur modellgetriebenen Softwareentwicklung, die in [SV05, Kap. 11] und [KK06, Kap. 8] vorgestellt werden, unterscheiden zwischen zwei miteinander verwobenen Entwicklungsprozessen: der Domänenentwicklung und der Anwendungsentwicklung. Die Domänenentwicklung erstellt die DSL, die Plattform und die Transformationen für die Systemgenerierung. Die Anwendungsentwicklung setzt die Ergebnisse der Domänenarchitektur ein, um konkrete Anwendungen zu realisieren. Dabei findet auch Feedback von der Anwendungsentwicklung zur Domänenentwicklung statt, das die Domänenentwicklung vorantreibt. Diese Prozesse werden meist iterativ ausgeführt, wie es in Abbildung 5.4 veranschaulicht wird.

Domänenentwicklung als Teil des CCM Systemerstellungsprozess

Die Beziehung zwischen der CCM Systementwicklung und der modellgetriebenen Softwareentwicklung kann folgendermaßen zusammengefasst werden: Die Domänenentwicklung entspricht dem Infrastrukturprozess und die Anwendungsentwicklung dem (beispielgetriebenen) Modellierungsprozess. Vor allem für den Infrastrukturprozess können aus der Literatur zur Domänenentwicklung [SV05, KK06] Anregungen bezogen werden. Deshalb wird der Infrastrukturprozess in dieser Arbeit nicht weiter erläutert.

Allerdings wird in [KK06, 9.1.3] auf psychologische Aspekte der Arbeitsteilung zwischen Anwendungs- und Domänenentwicklung hingewiesen, die auch bei der CCM Entwicklung von Bedeutung sind. Es ist entscheidend, dass Domänenexperten und CCM Entwickler bei der Systemerstellung gemeinsam die Systementwicklung vornehmen. Potentiellem Mistrauen, Schuldzuweisungen und Missverständnissen muss durch eine angemessene Kommunikation zwischen den Akteuren begegnet werden.

In dieser Betrachtung wird ein CCM System mit seinen Evolutions- und Personalisierungsstufen als Systemfamilie im Sinne der MDSD betrachtet. Auf projektübergreifender Ebene kann man auch die Entwicklung des Modellkompilers und der Modellierungssprachen (Assetdefinitionssprache, Asset Expression Language) selbst als Domänenentwicklung betrachten.

5.2.6 Beziehung zum Wissensmanagement

Informationssysteme im Allgemeinen und Inhaltsverwaltungssysteme im Speziellen werden häufig als Werkzeuge zum Wissensmanagement eingesetzt. Eine einheitliche Definition des Begriffs Wissensmanagement ist nicht trivial [Wegner02, 2.1]. Für diese Arbeit ist folgende Definition ausreichend [WMWiPe]:

> Wissensmanagement (englisch: knowledge management) bezeichnet eine Richtung der Managementlehre, die darauf abzielt, in Organisationen Wissen einzusetzen und zu entwickeln, um die Unternehmensziele bestmöglich zu erreichen.

Im Folgenden sollen die Beziehungen zwischen der konzeptorientierten Inhaltsverwaltung und dem Wissensmanagement geklärt werden.

Beispiel: CommonKADS und On-To-Knowledge

Das Methodenbündel CommonKADS, meist als Methode bezeichnet, beinhaltet zwei Methoden: Ursprünglich ist KADS (*Knowledge Acquisition Documentation System*, Akronym wurde mittlerweile verworfen) eine Methode zur Entwicklung wissensbasierter Systeme. In diesem Sinne wurde sie auch als Bestandteil des On-To-Knowledge Wissens-Metaprozesses aufgenommen, siehe Abschnitt 4.4.2 und [SSSS01]. Darüber hinaus beinhaltet CommonKADS eine Methode zum Wissensmanagement (*Knowledge Management Framework*, siehe [Studer02]).

Im Gegensatz zu On-To-Knowledge ist die Einbettung des CCM Systemerstellungsprozesses in die Anwendungsentwicklung nach CommonKADS nicht sinnvoll. Die Methode legt den Schwerpunkt auf wissensbasierte Systeme, insbesonders auf die Inferenz neuen Wissens. Im Gegensatz dazu werden bei CCM Systemen Inferenztechniken im Allgemeinen nicht verwendet. Allerdings ist es denkbar und sinnvoll, dass CCM Systeme im Rahmen eines Wissensmanagmentprozesses, wie z. B. dem von CommonKADS, eingesetzt werden.

CCM Systeme und Wissensmanagement

CommonKADS versteht Wissensmanagement als eine Metaaktivität (auf Wissensmanagementebene, *Knowledge Management Level*), welche die eigentliche Wissensarbeit (auf Objektebene, *Knowledge Object Level*) analysiert, plant und modelliert [Studer02]. Die Entscheidung zur Einführung von Systemen und ihre Modellierung geschieht dabei auf der Managementebene. Dieses Verständnis vom Wissensmanagement als Metaaktivität überwiegt in der Literatur, obwohl es selten explizit formuliert wird.

Dieses Vorgehen steht im Gegensatz zu einer offenen und dynamischen Inhaltsverwaltung, dem besonderen Kennzeichen der CCM Systeme. Es ist zwar sinnvoll, den Einsatz dieser Systeme auf (Wissens-) Managementebene zu beschließen, aber die Modellierung ist bei diesen Systemen Bestandteil der Objektebene. Die Anwender erstellen und optimieren sich ihr Werkzeug nach eigenen Bedürfnissen. Dies bedeutet zunächst einen Kontrollverlust auf der Managementebene, dem allerdings folgende Vorteile der CCM Systeme gegenüberstehen:

Entwicklung des richtigen Systems: Bei der klassischen Anwendungsentwicklung, wie sie in CommonKADS angenommen wird, ist es möglich, dass durch eine fehlgeleitete Anforderungsanalyse ein System entwickelt wird, das die Bedürfnisse der Anwender nicht angemessen berücksichtigt.

Im Fall der konzeptorientierten Inhaltsverwaltung sind solche Probleme unwahrscheinlich, da die DEn die Modellierung selbst vornehmen. Da diese aus den Reihen der Anwender rekrutiert werden, ist die Missachtung von Anwenderbedürfnissen nicht zu erwarten. Dies gilt vor allem für den Fall einer Evolution oder Personalisierung, da hier auf praktische Erfahrungen mit dem bestehenden System zurückgegriffen werden kann.

Akzeptanz der Anwender: Selbst effektiv einsetzbare Systeme werden unter Umständen nicht von den Anwendern akzeptiert. Das kann daran liegen, dass die Anwender über bestimmte Aspekte ihrer Arbeit die Kontrolle behalten möchten, obwohl eine Automatisierung die Effizienz der Organisation steigern würde. Dies wird am Beispiel der automatisierten Terminplanung in [SM99] erläutert. Darüber hinaus besteht bei Inhaltsverwaltungssystemen häufig die Befürchtung der Anwender, dass sie sich durch Offenlegung ihres Fachwissens austauschbar machen.

Solche Akzeptanzprobleme sind im Fall der Modellierung durch DEn nicht zu erwarten. Darüber hinaus können sie eine missionarische Funktion bei der Einführung des neuen Systems haben. Sofern die Modellierung eine positive Erfahrung ist, kann sich dieser Effekt sogar von selbst einstellen.

Evolution und Personalisierung: Neben den Vorteilen der Offenheit und Dynamik für die Anwender ist die Möglichkeit einer automatisierten Evolution und Personalisierung auch aus wirtschaftlicher Sicht interessant. Manuell realisierte Systemevolutionen sind in der Regel mit hohem technischen Aufwand verbunden. Dieses wirtschaftliche Risiko ist bei den CCM Systemen stark reduziert.

5.3 Beispielgetriebene Modellierung

Im Folgenden wird der induktive Modellierungsprozess dieser Arbeit beschrieben. Er folgt weitgehend den Ideen und Phasen des Genex Framework (*generator of excellence*) [Shneiderman00]. Grundlagen für das Framework sind die folgenden Annahmen: Neues Wissen baut auf vorhandenem Wissen auf, ausgefeilte Werkzeuge können Kreativität unterstützen, Verfeinerung ist ein sozialer Prozess, schöpferische Arbeit wird erst durch ihre Verbreitung abgeschlossen. Daraus folgen vier Phasen für das Prozessframework:

Sammeln (*collect*) – von voriger Arbeit lernen

Austausch (*relate*) – Austausch mit Gleichen und Beratung mit Mentoren

Erschaffen (*create*) – erforschen und bewerten möglicher Lösungen

Beitragen (*donate*) – verbreiten der Ergebnisse

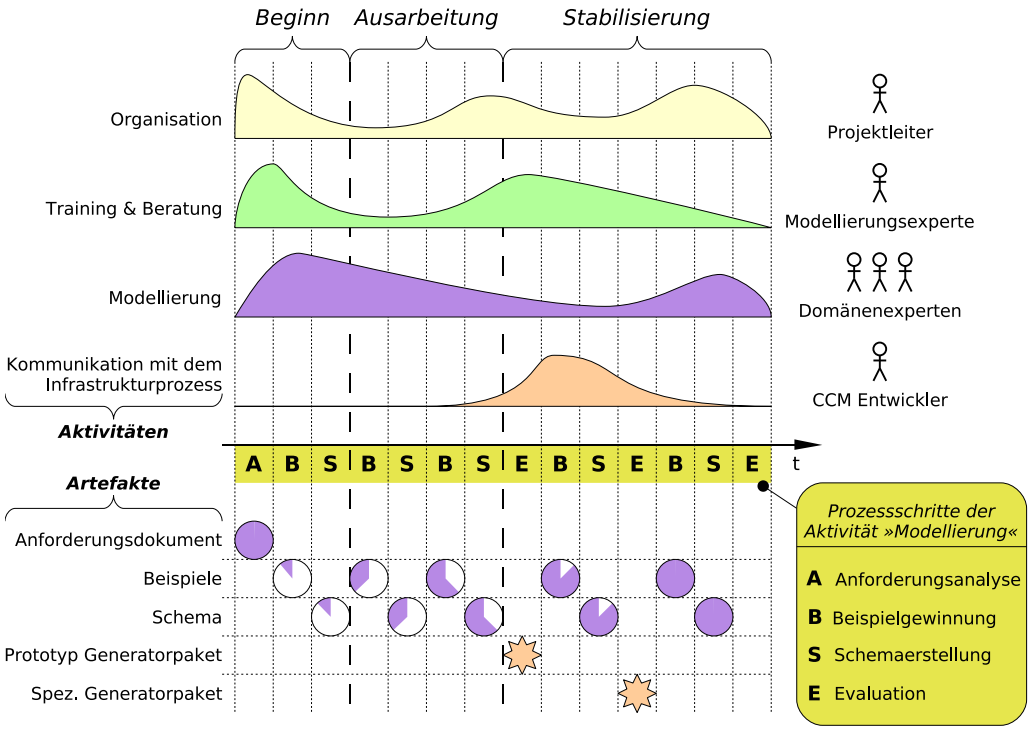

Abbildung 5.5: Phasen der beispielgetriebenen Modellierung

5.3.1 Iterativer Prozessverlauf

Wie bereits erwähnt wurde, wird bei der beispielgetriebenen Modellierung (BGM) iterativ und inkrementell vorgegangen. Neben der eigentlichen Modellierung durch die Domänenexperten (DEn) sind Hilfsaktivitäten vorgesehen, welche die Modellierung unterstützen und die Systemerstellung erleichtern. Eine Übersicht über die Aktivitäten, Akteure und wichtigsten Artefakte der beispielgetriebenen Modellierung zeigt Abbildung 5.5.

Die Aktivität der Modellierung wird in Prozesschritte eingeteilt (Anforderungsanalyse, Beispielgewinnung, Schemaerstellung, Evaluation), die in den folgenden Abschnitten des Kapitels beschrieben werden. Im unteren Teil der Abbildung 5.5 sind die wichtigsten Artefakte des Modellierungsprozesses mit ihrem Fortschritt als Tortendiagramm dargestellt. Sie sind unter den Prozessschritten der Modellierung angeordnet, in denen sie erstellt werden. Die Generatorpakete werden im Modellierungsprozess zur Evaluation benötigt, da zu diesem Zweck Prototypen des CCM Systems generiert werden. Die Generatorpakete werden vom Infrastrukturprozess bereitgestellt.

Aktivitäten

Die einzige essentielle Aktivität im Modellierungsprozess ist die Modellierung selbst. Alle anderen Hilfsaktivitäten können im Sinne des Prozessframeworks ausgelassen werden, sofern sie nicht als notwendig erachtet werden. So sind bei Personalisierungen für einen kleinen Anwenderkreis die Hilfsaktivitäten unangemessen aufwendig, vor allem wenn diese von in der Systemanwendung und Modellierung erfahrenen DEn durchgeführt werden. Folgende parallele Aktivitäten werden in dem Modellierungsprozess durchgeführt:

Modellierung: In der Modellierungsaktivität, die von den DEn eigenverantwortlich durchgeführt wird, werden drei Artefakte erstellt: ein Anforderungsdokument, eine Menge von Instanzen und ein konzeptuelles Schema. Das Anforderungsdokument beschreibt, was für ein System benötigt wird. Die Instanzen dienen als Beispieldaten für das Schema und das endgültige System. Diese Aktivität stellt den Großteil der Arbeit in der BGM dar, die Kurven in Abbildung 5.5 stellen lediglich die relative Intensität jeder Aktivität über die Zeit dar, setzen aber nicht den Gesamtaufwand der Aktivitäten in ein Verhältnis.

Training & Beratung: Die DEn werden durch Modellierungsexperten in ihrem Vorgehen unterstützt. Dies ist nötig, da die meisten DEn über keine Modellierungserfahrung verfügen. Gerade zu Begin des Prozesses sind Schulungen und Workshops wichtig, um die DEn auf ihre neue Aufgabe vorzubereiten.

Organisation: Zur Unterstützung der Modellierung wird eine Projektleitung benötigt. Schulungen und Workshops müssen organisiert, der Zeit- und Budgetplan muss eingehalten werden. Gerade bei neuen Systemen muss verstärkt auf die Bereitstellung der Infrastruktur geachtet werden, da wahrscheinlich spezielle Generatoren erforderlich sind. Der Projektleiter kann aus Reihen der Organisation oder der CCM Entwickler stammen, je nachdem, ob die Kommunikation mit dem Infrastrukturprozess oder die Organisation der DEn schwieriger ist.

Kommunikation mit dem Infrastrukturprozess: Im Gegensatz zu den anderen Aktivitäten, hat diese wenig mit der eigentlichen Modellierung zu tun. Gerade bei neuen Systemen ist es nötig, detaillierte Anforderung für projektspezifische Eigenheiten der Infrastruktur zu erhalten. Diese Anforderungen stützen sich am besten auf ein bestehendes Schema, um Missverständnissen vorzubeugen. Darüber hinaus ist es notwendig, dass die Generatorpakete in der Werkbank installiert werden. Dies wird von einem CCM Entwickler vorbereitet, da hierzu ein umfassendes technisches Verständnis nötig ist.

Phasen

1. Während der **Beginn** Phase der BGM werden zunächst die DEn rekrutiert und anschließend im Modellierungsprozess und den verwendeten Werkzeugen geschult. Außerdem werden die Anforderungen an das System formuliert und die DEn beginnen die Modellierung gemeinsam mit dem Modellierungsexperten. Hierfür sind Workshops (siehe Abschnitt 4.3.2) besonders geeignet. Das Ziel dieser Phase liegt darin, ein Projektteam aufzubauen, soziale Kontakte zu etablieren und eine gemeinsame Vorstellung für das System zu entwickeln.

2. In der **Ausarbeitung** geschieht der Großteil der Modellierung. Ausgehend von den Anforderungen und der gemeinsamen Vorstellung des Systems arbeiten die DEn so eigenständig wie möglich an der Modellierung, wobei der Modellierungsexperte sie berät. Dazu modellieren sie Beispielinstanzen und ein vorläufiges Schema. Die Ausarbeitung erstreckt sich über einen längeren Zeitraum als die erste Phase. Es ist sinnvoll, dass die DEn die Modellierung parallel zu ihrer täglichen Arbeit

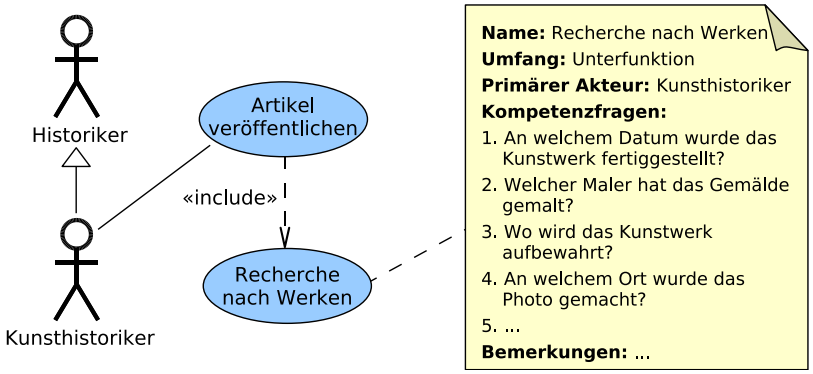

Abbildung 5.6: Anforderungen in der BGM

durchführen, da hieraus sinnvolle Ideen für das System zu erwarten sind. Auch kreative Arbeit über die formulierten Anforderungen hinaus ist ausdrücklich erwünscht.

3. Nachdem der Großteil der Modellierung stattgefunden hat, wird während der **Stabilisierung** das entstandene Schema kritisch auf seine Qualität hin untersucht. Dazu werden aus dem Schema Prototypen des CCM Systems generiert und überprüft, ob die Anforderungen und Erwartungen erfüllt sind. Vereinfachungen und letzte Ergänzungen an dem Schema werden vorgenommen. Darüber hinaus werden die Anforderungen an projektspezifische Generatorpakete erhoben. Durch die Wechselbeziehungen mit dem Infrastrukturprozess wird diese Phase die meiste Zeit in Anspruch nehmen.

5.3.2 Anforderungsanalyse

Ziel

Das Ziel der Anforderungsanalyse ist es, eine Menge von Anforderungen auf Softwaresystemebene zu formulieren. Dabei wird nicht die vollständige Spezifikation des Systems angestrebt, sondern eine Sammlung der wichtigsten Anforderungen, damit die modellierenden DEn eine gemeinsame Vorstellung des fertigen Systems entwickeln. Die Anforderungen sollen daher notwendig, aber keinesfalls hinreichend sein.

Als Form für die Anforderungen wird hier eine Kombination aus Anwendungsfällen und informellen Kompetenzfragen vorgeschlagen. Eine Untersuchung der beteiligten Akteure sollte bereits vor der Anforderungsanalyse stattfinden. In Abbildung 5.6 findet sich ein Beispiel für die Anforderungen in der BGM.

Akteure Es soll eine Liste von Akteuren erstellt werden, die das System benutzen werden. Idealerweise wurde diese Liste bereits im Vorfeld erstellt und Repräsentanten aller Anwendergruppen sind in den Modellierungsprozess einbezogen. Sollten weitere Akteure während der Anforderungsanalyse erkannt werden, so ist ihre Teilnahme wünschenswert.

Anwendungsfälle Es werden hier ausschließlich Anwendungsfälle auf *Geschäftsebene* betrachtet. Es sollen Akteure mit Tätigkeiten aus ihrem Arbeitsalltag dargestellt werden, bei deren Durchführung das System ihnen dienen soll. Die Tätigkeiten

werden nur knapp beschrieben. Ein UML Anwendungsfalldiagramm und eine kurze, allgemeine Beschreibung der Tätigkeit sollte ausreichen, wobei Kompetenzfragen für die Details verwendet werden.

Kompetenzfragen Statt der Beschreibung von Anwendungsfällen durch Szenarien sollen hier informelle Kompetenzfragen eingesetzt werden. Sie sollen Auskunft darüber geben, welche Informationen die Anwender bei ihrer Arbeit benötigen.

Vorgehen

Ausgehend von der Vision und dem Projektrahmen, die im CCM Systemerstellungsprozess geschaffen wurden, sollen die Anforderungen an das neue System entwickelt werden. Wenn ein Bestandssystem existiert, wird es als Ausgangspunkt verwendet. In diesem Fall werden nur Modifikationen des Bestandssystems betrachtet. Bei Personalisierungen für einen kleinen Anwenderkreis kann auf diese Phase verzichtet werden, da bereits eine genaue Vorstellung über notwendige Anpassungen vorhanden sein dürfte.

Die Abfolge der Prozessschritte und die Beziehungen zwischen ihnen ist in Abbildung 5.7 als UML Aktivitätsdiagramm dargestellt. Neben den Abhängigkeiten zwischen den als Aktivitäten dargestellten Prozessschritten, werden die von den einzelnen Schritten benötigten und produzierten Artefakte gezeigt, wie auch die Beziehungen zwischen den Modellierungsprozess und dem Infrastrukturprozess.

Als Vorgehen bietet sich hier ein Workshop an, in dem mögliche Anforderungen z. B. durch den Einsatz von Kreativitätstechniken gesammelt werden. Diese werden anschließend kondensiert und auf die essentiellen Anforderungen reduziert, so dass eine kollektiv geteilte Vorstellung des Systems entsteht. Es bietet sich hier an, dass sowohl der Projektleiter wie auch der Modellierungsexperte den DEn bei diesem Prozessschritt intensiv zur Seite stehen. Das Vorgehen in diesem Schritt ist sowohl inspirational wie auch kollaborativ.

Begründung

Die Beschreibung der Systemanforderungen geschieht hier in Form von Anwendungsfällen (siehe Abs. 4.3.3), die für Wissensmanagementsysteme angepasst wurden. Eine Beschreibung der Anwendungsfälle durch Szenarien oder Systeminteraktionen ist hier nicht sinnvoll, da hier der Wissens- oder Informationsbedarf für eine bestimmte Anwendertätigkeit im Vordergrund steht und nicht der zeitliche Ablauf. Der Wissensbedarf der Anwendungsfälle lässt sich in Form von informellen Kompetenzfragen gut ausdrücken. Die durch Kompetenzfragen ergänzten Anwendungsfälle stellen die funktionalen Anforderungen an das System dar, vor allem jene, die sich im konzeptuellen Schema manifestieren.

Eine andere Besonderheit der Anforderungsanalyse ist hier, dass durch die Anforderungen eine kollektiv geteilte Vorstellung des Systems entwickelt wird, die lediglich die wichtigsten notwendigen Anforderungen an das System enthält. Es ist bei der BGM nicht nötig, eine vollständige Systemspezifikation in Form von Anforderungen zu entwickeln, da die Domänenexperten die Modellierung selbst durchführen. Beim klassischen Softwareengineering ist dies notwendig, da die Anforderungen der einzige Kommunikationskanal zwischen den Auftraggebern und den Entwicklern ist. In der BGM dienen die Anforderungen als ein Anfangspunkt für die Modellierung. Die DEn erhalten so eine Vorstellung,

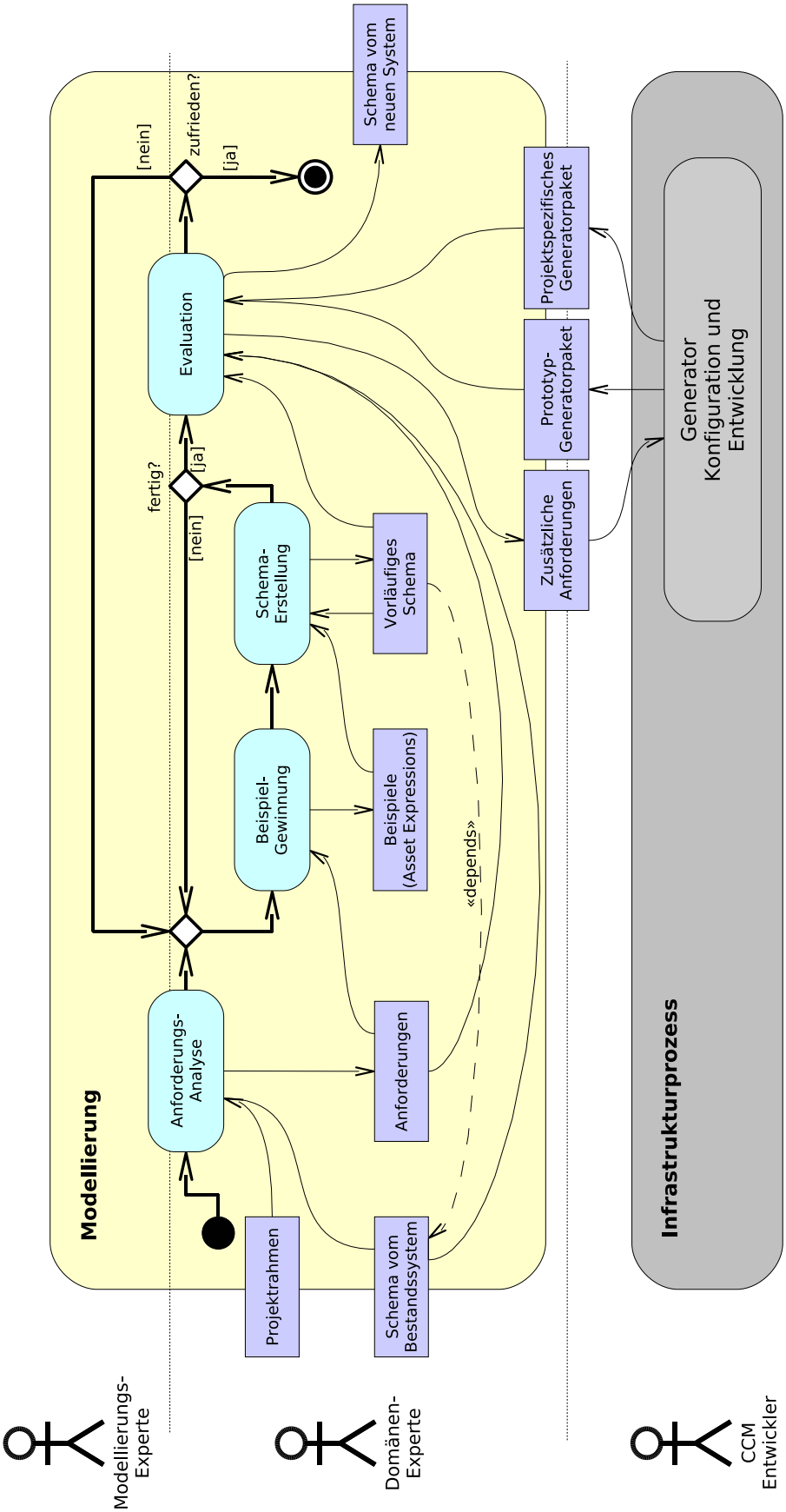

Abbildung 5.7: Prozess zur beispielgetriebenen Modellierung

was sie modellieren müssen. Diese Anforderungen sollen nicht als hinreichend angesehen werden, da sonst kreatives Arbeiten in den anderen Prozessschritten eingeschränkt wird. Vor allem in der Evaluation entstehen weitere Anregungen für die Modellierung.

5.3.3 Beispielgewinnung

Ziel

Das Ergebnis der Beispielgewinnung ist eine Menge von Beispielinstanzen, die ein extensionales Modell der Anwendungsdomäne darstellen, siehe Abschnitt 2.2.2. Das bedeutet, dass die Beispiele hinreichend ausführlich modelliert werden, so dass sich in ihnen alle für die Systemgenerierung relevanten Domänenkonzepte wiederfinden. Aufgrund des iterativen Prozessverlaufes wird zunächst nur ein partielles extensionales Domänenmodell erstellt.

Das extensionale Domänenmodell soll in Form von semantisch getypten Asset Expressions (AEs, siehe Abschnitt 3.3.1) vorliegen. Daraus folgt, dass die Beispielinstanzen durch eine Menge semantischer Typen genauer beschrieben werden. Neben den AEs muss auch ein solches semantisches Typsystem erstellt werden.

Vorgehen

Die Beispielgewinnung soll als ein kreativer Prozess verstanden werden, in dem relevantes Domänenwissen in Form von Beispielinstanzen gesammelt wird. Die Anforderungen sind zwar Ausgangspunkt für diesen Prozessschritt (siehe Abbildung 5.7), allerdings dienen sie vor allem dazu, den Modellierungsprozess zielgerichtet beginnen zu können, indem gleich zu Anfang mit relevantem Domänenwisssen begonnen wird.

Am anspruchvollsten für die Domänenexperten ist die Modellierung des semantischen Typsystems, da diese taxonomische Sammlung von Domänenkonzepten der stärkste Formalismus in der BGM ist, mit denen sie in Kontakt kommen. Vor allem in den ersten Iterationen, wenn das Typsystem noch große Lücken aufweist, können linguistische Methoden (siehe Abschnitt 4.2.2) verwendet werden.

Dabei kann eine Liste mit konzeptuellen Kategorien (Abschnitt 4.2.3) den DEn als Hilfe dienen. Darüber hinaus ist bei der Modellierung des Typsystems eine Unterstützung durch den Modellierungsexperten wünschenswert, da diese Methoden generell eine gewisse Erfahrung erfordern, um sprachliche Mehrdeutigkeiten wie Synonyme und Homonyme zu identifizieren. Unter Umständen kann die Unterstützung auch in Form eines Werkzeuges bereitgestellt werden, wie es in Abschnitt 6.1.2 vorgeschlagen wird.

Für die Formulierung der AEs wird deren grafische Repräsentation (Abschnitt 3.1.3) verwendet. Zum einen werden die DEn so nicht mit den Formalismus des Lambda-Kalküls belastet, zum anderen ermöglicht die grafische Repräsentation einen sehr intuitiven und kreativen Umgang mit AEs. Es gibt folgende Möglichkeiten, an Beispielinstanzen für die Modellierung zu gelangen:

Anlegen und Modifizieren von AEs: Domänenexperten können die AEs am besten mit Hilfe eines grafischen Editors anlegen. Durch die Modifikation bestehender AEs können die Instanzen im Laufe der Modellierung weiter verfeinert werden. Es ist wichtig, dass die DEn jederzeit Beispielinstanzen anlegen können. Gerade während ihrer täglichen Arbeit, bei der sie das System in Zukunft unterstützen soll, sind

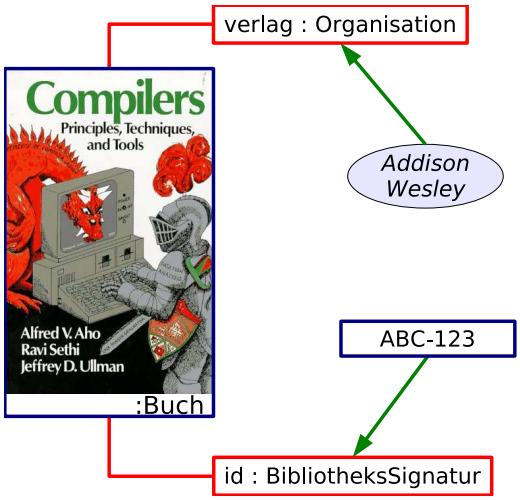

Abbildung 5.8: Modellierungsfehler in einer Asset Expression

kreative Anregungen für die Modellierung zu erwarten. Diese langwierige Arbeit kann durch die Verwendung von Traits (siehe Abschnitt 3.3.3) erleichtert werden.

Importieren von Instanzen aus CCM Systemen: Gerade bei der Evolution oder Personalisierung eines Bestandssystems können Instanzen aus diesem importiert werden. Diese Instanzen werden inhaltlich und strukturell personalisiert. Auch wenn es kein Bestandsssytem gibt, können importierte Instanzen aus anderen CCM Systemen die Modellierung vereinfachen.

Importieren von AEs aus persönlichen Beständen: Grafische Editoren für AEs eignen sich nicht nur zur Modellierung von CCM Systemen, sondern können auch als eine Art multimediale private Datenbank verwendet werden siehe auch Abschnitt 3.4. Individuelle Notizen, die potentielle Anwender in einem solchen System gespeichert haben, können als Grundlage für die Modellierung eines kollaborativ genutzten Systems herangezogen werden.

Das Anlegen von AEs kann durch einen Katalog von Analysemustern unterstützt werden, entsprechend ihrer Verwendung in der objektorientierten Analyse, siehe Abschnitt 4.2.1. Lösungen für Modellierungsprobleme können so wiederverwendet und Fehler vermieden werden. Allerdings bieten die objektorientierten Analysemuster nur Lösungen für eine Modellierung auf intensionaler Ebene und können daher nicht direkt verwendet werden. Abbildung 5.8 zeigt das Beispiel einer fehlerhaften Modellierung eines Buchexemplars aus einer Bibliothek. Der Modellierungsfehler besteht darin, dass keine Trennung zwischen der Ausgabe eines Buches, die in vielen Exemplaren käuflich ist, und dem einzelnen Buchexemplar gemacht wird, welches ausgeliehen werden kann. Dieser Modellierungsfehler kann durch das Analysemuster Exemplartyp [Fowler97] behoben werden. In Abbildung 5.9 wird dieses in Form eines Traits (siehe Abschnitt 3.3.3) grafisch dargestellt. Die zugehörigen semantischen Typen sind den Typen der Abstraktionen zu entnehmen, sie müssen in der Typhierarchie der AEs berücksichtigt werden. Es ist zu prüfen, ob es möglich ist, weitere Muster auf gleiche Weise zu formulieren.

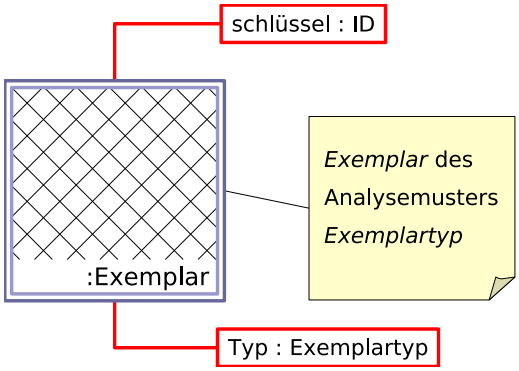

Abbildung 5.9: Analysemuster als Trait formuliert

Das Vorgehen in diesem Prozessschritt ist inspirational, da individuelle kreative Einflüsse in dieser Phase entscheidend sind. Darüber hinaus empfiehlt sich für die Beispielgewinnung ein kollaboratives Vorgehen, sofern das entstehende CCM System kollaborativ genutzt werden soll. Werden Beispiele aus verschiedenen Quellen importiert, ergibt sich zudem ein synthetisierendes Element. Deduktive Ansätze sind in der Verwendung der konzeptuellen Kategorien und den Analysemustern vorhanden, den Hilfsmitteln dieses Prozessschrittes.

Begründung

Während der Beispielgewinnung sollen Instanzen aller Domänenkonzepte erfasst werden, die in dem endgültigen CCM System benötigt werden. Aus diesem Grund ist dieser Schritt als ein kreativer Prozess angelegt. Sollten in dieser kreativen Phase einzelne Instanzen modelliert werden, die unwichtig oder missverständlich sind, darf dies keine unmittelbaren Auswirkungen auf das Ergebnis haben. Daher wird durch die nachfolgende Schemaerstellung und Evaluation die Qualität des konzeptuellen Modells sichergestellt. Das für diesen Prozessschritt vorgeschlagene Vorgehen zielt darauf ab, die Beispielgewinnung für die DEn zu erleichtern, so dass diese möglichst intuitiv und kreativ arbeiten können.

Obwohl ein intuitives Arbeiten gewünscht wird, sollen semantisch sinnvolle Applikationen auf die Abstraktionen angewendet werden. Um dies zu gewährleisten werden die semantisch getypten AEs gewählt, obwohl dies einen größeren Aufwand für die DEn bedeutet. Für den Entwurf der semantischen Typhierarchie können linguistische Methoden eingesetzt werden, die sich schon in der Anforderungsanalyse und in der objektorientierten Analyse bewährt haben (siehe Abschnitte 4.3 und 4.2.2). Durch diese Methoden ist es möglich, DEn über natürlichsprachliche Formulierungen in die Modellierung einzubeziehen. Daher ist ein solches Verfahren – gegebenenfalls unter Anleitung eines Modellierungsexperten – auch für den Entwurf der semantischen Typen geeignet.

Die Wahl der grafischen Darstellung für Asset Expressions hat mehrere Hintergründe, abgesehen davon, dass den DEn der Umgang mit dem syntaktischen Formalismus des Lambda-Kalküls erspart bleibt. Die grafische Repräsentation der AEs ähneln Mindmaps [BB05], einer visuellen Notation der populären Psychologie, die häufig für kreative und assoziative Arbeit verwendet wird, Abbildung 5.10 veranschaulicht die Ähnlichkeit. Aus

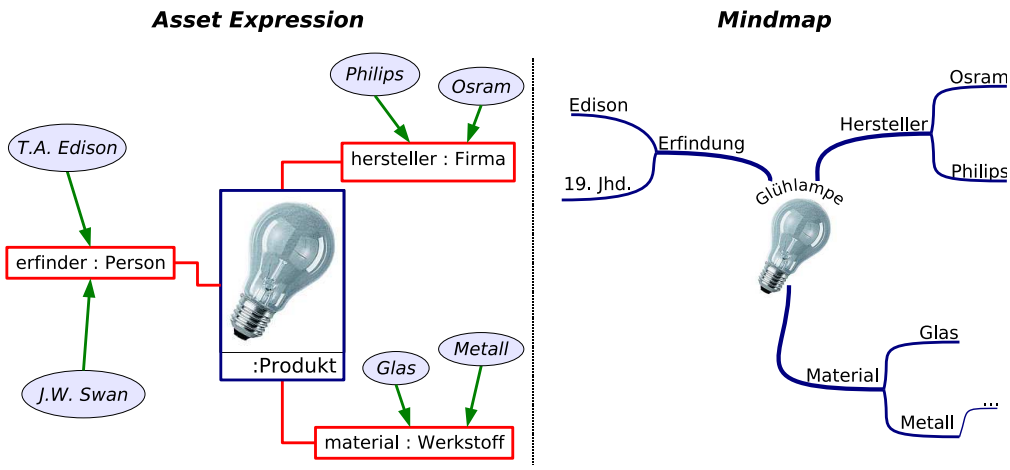

Abbildung 5.10: Ähnlichkeit zwischen Asset Expressions und Mindmaps

dieser Ähnlichkeit kann abgeleitet werden, dass sich auch grafische AEs für kreatives und intuitives Arbeiten eignen. In diesem Sinne erfüllen die grafischen AEs die Funktion einer Zwischenrepräsentation, wie sie in der Ontologieentwicklung verwendet wird. Durch den Einsatz einer Sprache, die auch für DEn verständlich ist, werden diese an der Ontologieentwicklung beteiligt (siehe Abschnitt 4.4). Die Analysemuster und konzeptuellen Kategorien werden in dem Prozess als bewährte Hilfmittel der objektorientierten Analyse wieder aufgegriffen.

5.3.4 Schemaerstellung

Ziel

In diesem Schritt wird ein – zunächst partielles – konzeptuelles Schema der Domäne in Form der Assetdefinitionssprache (ADL) (siehe Abschnitt 2.2) erstellt. Dieses wird in den Iterationen vervollständigt und verfeinert, bis ein Schema vorliegt, welches zur Generierung des neuen CCM Systems geeignet ist.

Vorgehen

Während dieses Prozessschrittes wird aus dem extensionalem Domänenmodell in Form von AEs ein intensionales hergeleitet (Abbildung 5.7). Das entstehende konzeptuelle Schema wird in den Iterationen von einem vorläufigen zu einem endgültiges Schema verfeinert. Sollte ein Bestandssystem vorliegen, so wird dessen Schema als erstes vorläufiges Schema übernommen. Die Schemainferenz auf Grundlage der Beispielinstanzen kann nicht von den DEn bewerkstelligt werden, so dass hier eine Werkzeugunterstützung erforderlich ist. Allerdings müssen die DEn an der Inferenz beteiligt werden, da nur sie beurteilen können, ob ein Schema aus Anwendersicht sinnvoll ist. Darüber hinaus ist es auch denkbar, dass die DEn auf intensionaler Ebene Klassenattribute vorschlagen. Um zu beweisen, dass diese Attribute sinnvoll sind, müssen die DEn auch Beispielinstanzen modellieren, welche die Attribute bestätigen. Nur dann werden sie in das engültige Schema übernommen.

Aufgrund der Flexibilität der AEs kann für jede einzelne Entität eine optimale strukturelle und technische Repräsentation gewählt werden. Diese Flexibilität ist jedoch im CCM System nicht mehr gewünscht: Die Klassen des Schemas sollen innerhalb der Domäne allgemein verwendbar und für mehrere Assets gültig sein. Deshalb sind für jede Assetklasse mehrere AEs als Beispiele erforderlich. Die Herleitung einer Assetklasse darf nur in Ausnahmefällen aufgrund eines einzelnen Beispiels geschehen.

Im Gegensatz zur Beispielgewinnung ist die Schemaerstellung aus Sicht der DEn ein analytischer Vorgang. Die Menge der Beispielinstanzen wird kategorisiert, wobei sich einzelne Exemplare als ungeeignet für die Modellierung herausstellen können. Die Herleitung eines Schemas aufgrund von Beispielen ist ein induktiver Vorgang, da hier von einzelnen Fällen auf das Allgemeine geschlossen wird, d.h. von den Beispielen auf das Schema. Da die DEn in die Schemainferenz einbezogen werden müssen, bietet es sich an, die Schemaerstellung kollaborativ durchzuführen, um die Ansichten aller Experten mit einzubeziehen.

Begründung

Der Hintergrund für die Forderung nach einer Werkzeugunterstützung folgt daraus, dass die Schemainferenz ein aufwendiger und anspruchsvoller Vorgang ist – zumindest für einen Menschen. Da dieser Schritt in den Iterationen der BGM laufend wiederholt wird, ist der Aufwand für diesen Vorgang ohne Werkzeug unangemessen hoch. Die DEn müssen zudem lernen, wie sie den Inferenzvorgang am besten beeinflussen können. Ein Werkzeug kann bei diesem Vorgang geduldiger mit den DEn interagieren, als es von einem Modellierungsexperten verlangt werden kann.

In der Schemaerstellung liegt zudem der Grund für die BGM insgesamt. Die Erfahrungen mit den bisherigen CCM Systemen haben gezeigt, dass Assetklassen (bzw. Domänenkonzepte im allgemeinen) häufig aufgrund einzelner Entitäten entworfen werden, siehe Abschnitt 2.4. Daher wird der Vorgang, die Beschreibungen einzelner Entitäten zu Konzepten zu abstrahieren, nicht den DEn überlassen. Die Beziehung zwischen den Beispielinstanzen und den konzeptuellen Klassen in der BGM wird in Abbildung 5.11 veranschaulicht: Zum einen werden die Klassen aus den Instanzen hergeleitet, um die Modellierung für die Domänenexperten zu erleichtern. Dies geschieht entsprechend der Ansicht von Albert Einstein: *example isn't another way to teach, it is the only way to teach* [Sendall02]. Darüber hinaus werden die Instanzen als Belege für die sinnvolle Modellierung der Klassen verwendet. Aus diesem Grund ist es sinnvoll, für jede Klasse mehrere Beispiele zu verlangen. So wird die Generalisierbarkeit der Klassen sichergestellt.

Sollte der Fall auftreten, dass eine Klasse nur durch eine einzelne Beispielinstanz gestützt wird, so liegt einer der folgenden Fälle vor:

1. Es ist die bislang einzige Instanz einer wichtigen Klasse. In diesem Fall müssen weitere Instanzen angelegt werden.

2. Es ist eine unwichtige Instanz, die nicht im Schema berücksichtigt werden muss. Sie wird aus der Beispielmenge entfernt.

3. Die Instanz ist unvollständig oder missverständlich modelliert. Sie muss gelöscht oder korrigiert werden.

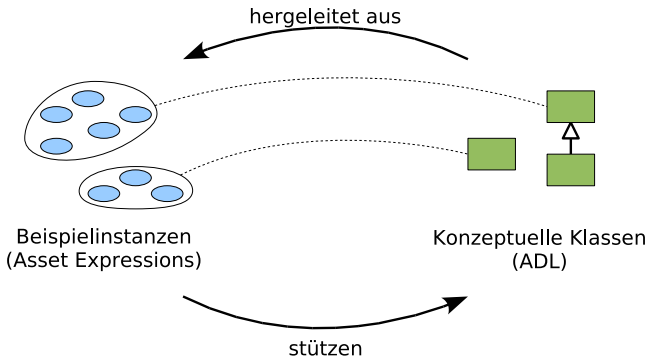

Abbildung 5.11: Beziehung zwischen Instanzen und Klassen in der BGM

4. Es handelt sich bei dieser Instanz um das einzige Exemplar seiner Klasse (Singleton). Die DE müssen entscheiden, ob diese einzelne Entität von so herausragender Bedeutung für die Domäne ist, dass sie in das Schema aufgenommen werden muss.

5.3.5 Evaluation

Ziel

Das Ziel der Evaluation ist die Feststellung, ob das in den bisherigen Iterationen modellierte Schema sich für die Generierung des CCM Systems eignet. Dazu müssen folgende Fragen geklärt werden:

1. Sind die formulierten Anforderungen an das System erfüllt?

2. Entspricht ein auf Basis des Schemas generiertes System den bislang nicht explizit formulierten Anforderungen der DEn?

3. Ist die Übernahme von Informationen aus dem Bestandssystem möglich? Diese Frage ist lediglich bei einer Evolution oder Personalisierung zu klären.

Neben der Klärung dieser Fragen gewinnen die DEn in diesem Schritt Erkenntnisse darüber, wie das Modell verfeinert werden kann. Dieses Wissen wird nicht explizit formuliert, da zu erwarten ist, dass die DEn diese Erfahrungen selbstständig in der folgenden Iteration berücksichtigen werden.

Ein anderes Ergebnis der Evaluation sind Anforderungen für den Infrastrukturprozess, die in den Generatorpaketen berücksichtigt müssen. Dazu gehören auch die Anforderungen an – eventuell notwendige – projektspezifische Generatoren.

Vorgehen

Die Evaluation wird am effektivsten durchgeführt, indem aus dem vorläufigen Schema CCM Systeme generiert werden. Diese werden dann im Sinne von Prototypen für die Evaluierung verwendet. Von erfahrenen DEn können die Kompetenzfragen auch direkt mit dem Schema abgeglichen werden. Dies bietet sich allerdings nur in frühen Iterationen an, sofern noch keine hinreichenden Generatorpakete zur Verfügung stehen.

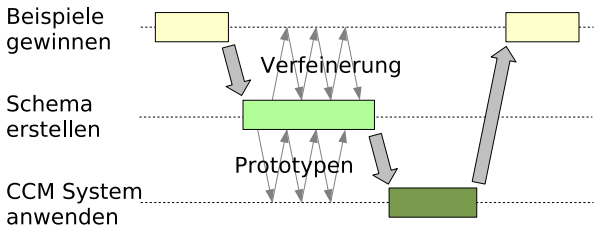

Abbildung 5.12: Inkrementelle Modellierung und Prototypen nach [Bossung07]

Wie aus Abbildung 5.7 ersichtlich wird, ist die Evaluation durch die prototypische Systemgenerierung von vielen Artefakten des Modellierungsprozesses und einigen des Infrastrukturprozesses abhängig. Dafür können die durch die Prototypen gewonnen Erkenntnisse in den folgenden Iterationen der Modellierung genutzt werden, siehe Abbildung 5.12. Aufgrund des Prototyps können die Fragen 1 und 2 leicht von den DEn geklärt werden. Es ist unter Umständen sinnvoll, auch Anwender, die nicht an der Modellierung teilnehmen, mit einzubeziehen. Gerade gegen Ende des Prozesses empfiehlt es sich, die Evaluation in Form eines Workshops durchzuführen, um sich der Mitarbeit und der Expertise aller DEn sicher zu sein.

Eine Werkzeugunterstützung ist für die Evaluation sehr wichtig: Zum einen erfordert die Systemgenerierung mit dem CCM Modellkompiler ein Maß an technischem Wissen, das nicht bei den DEn erwartet werden kann. Zum anderen ist die Klärung von Frage 3 nur von einem CCM Entwickler möglich. Ob eine Transformation der Informationen in das neue Schema möglich ist, hängt von den Generatoren für die Abbildungsmodule (*mapping modules*, siehe Abschnitt 2.3.2) ab, welche die Evolution und Personalisierung technisch ermöglichen. Die Ausgaben dieser Generatoren sind für die weitere Modellierung wichtig und müssen daher den DEn verständlich gemacht werden.

Die Erhebungen von Anforderungen für den Infrastrukturprozess geschieht als klassische Anforderungsanalyse und wird von dem CCM Entwickler durchgeführt. Allerdings liegt zu diesem Zeitpunkt bereits ein konzeptuelles Modell vor, wodurch sich die Analyse verhältnismäßig einfach gestaltet. Da diese spezielle Anforderungsanalyse nicht im engeren Sinne zu der Modellierung gehört, ist die Beteiligung des CCM Entwicklers an diesem Prozessschritt in Abbildung 5.7 nicht dargestellt.

Begründung

In diesem Prozessschritt werden vor allem Verfahren aus der Anforderungsanalyse verwendet. Besonders wichtig sind hier die Vorteile von Prototypen, obwohl ihre Verwendung den Modellierungsprozess komplizierter gestaltet. Sie werden hier vor allem im Sinne von horizontalen Prototypen eingesetzt. Folgende positive Auswirkungen auf die Modellierung werden durch ihren Einsatz erwartet, vergleiche auch Abschnitt 4.3.4:

- Es werden zuverlässige Erkenntnisse über die Anwendbarkeit des Systems und des konzeptuellen Modells gewonnen. Die DEn können das Ergebnis ihrer Arbeit so unmittelbar bewerten, dies entspricht der Validierung von Anforderungen.

- Die DEn und eventuell andere Anwender werden zur Mitarbeit motiviert, weil so das Ergebnis ihrer Arbeit so unmittelbar sichtbar ist.
- Die Kommunikation zwischen den DEn wird gefördert. Ein Prototyp kann als Verständigungsgrundlage für die DEn dienen und die Diskussion anregen.

Darüber hinaus ist auch die Verwendung von Anforderungen als Abnahmekriterium in der Anforderungsanalyse üblich. In diesem Sinne werden hier die zuvor formulierten Anforderungen eingesetzt.

Kapitel 6

Die Werkbank

Damit die Domänenexperten die Modellierung möglichst selbstständig durchführen können, benötigen sie neben dem Prozess die Unterstützung durch ein Werkzeug, der Werkbank zur beispielgetriebene Modellierung.

6.1 Prozessunterstützung durch die Werkbank

Im Folgenden wird eine Werkbank für die beispielgetriebene Modellierung (BGM) aus Sicht der Anwender beschrieben. Die Werkbank wird überwiegend von den Domänenexperten verwendet, allerdings auch von dem Modellierungsexperten und dem CCM Entwickler, der aus den Beispielinstanzen und den vorläufigen Schemata Informationen für den Infrastrukturprozess ableiten kann. Werden keine anderen Akteure erwähnt, konzentrieren sich die folgenden Ausführungen auf die Sicht der modellierenden Domänenexperten (DEn).

6.1.1 Anforderungsanalyse

Es ist heute Stand der Technik, Werkzeuge zur Anforderungsanalyse im Softwareengineering zu verwenden und diese z. B. in Entwicklungsumgebungen zu integrieren. Ähnliche Funktionen können auch in die Werkbank integriert werden.

- Durch die Werkbank kann eine **Akteurliste** verwaltet werden, in der die Akteure aufgezählt werden und ihre Motivation mit dem System zu arbeiten beschrieben wird. Dazu kann jeder Gruppe von Akteuren ein Domänenexperte zugeordnet werden, der die Interessen dieser Nutzergruppe in der Modellierung vertritt.

- Mit einer Funktion zur **Anwendungsfallerfassung** können Anwendungsfälle in vorgegebener Struktur beschrieben werden. Dazu werden der Name, die Akteure und eine Kurzbeschreibung des Anwendungsfalles festgehalten, ähnlich wie bei üblichen Vorlagen [Cockburn00]. Der Wissensbedarf wird allerdings in Form von informellen Kompetenzfragen ausgedrückt (siehe Abschnitt 4.4).

- Durch Integration einer Funktion zur **Anforderungskommunikation** ist es möglich, dass verteilte Gruppen von Anwendern in ihrer Werkbank der gleiche Satz an

Anforderungen zugänglich ist. Hierzu sind Synchronisationsmaßnahmen erforderlich.

- Neben der Erfassung bietet die **Verwaltung des Anwendungsfallstatus** während der folgenden Modellierung die Möglichkeit, einen Überblick über die Modellierung zu gewinnen, indem Anwendungsfälle oder einzelne Kompetenzfragen abgehakt werden können, wenn sie bereits im Modell berücksichtigt sind.

Genau wie der Prozessschritt der Anforderungsanalyse in der BGM optional ist, ist auch eine Unterstützung dieses Schrittes in der Werkbank nicht zwingend notwendig, zumal die meisten der Funktionen gut durch übliche Bürosoftware unterstützt werden können. Diese können z. B. durch Vorlagen (*templates*) an die Aufgabe angepasst werden.

6.1.2 Beispielgewinnung

Da die Beispielgewinnung ein kreativer Vorgang ist (vergleiche Abschnitt 5.3.3), muss sie für die Anwender möglichst einfach und intuitiv gestaltet werden. Dabei kommt der Werkbank eine entscheidende Bedeutung zu. Folgende Hilfen sind für die Beispielgewinnung wünschenswert:

Grafischer Asset Expression Editor: Da sich die grafische Repräsentation der AEs für ein intuitives Arbeiten eignet (Abs. 5.3.3), ist es wünschenswert, wenn die DEn beim formulieren der AEs adäquat unterstützt werden. Dabei ist großer Wert auf eine leichte Bedienbarkeit zu legen, damit die DEn bei ihrer kreativen Arbeit nicht durch technische Belange gestört werden.

Darüber hinaus ist eine Funktion zu einer kollaborativen Bearbeitung der AEs notwendig, da die Verfeinerung der Instanzen über die Iterationen auch ein sozialer Prozess [Shneiderman00] ist.

Assistent für die linguistische Analyse: Das Erstellen der semantischen Typhierarchie stellt für die DEn die größte Herausforderung dar, da es sich hierbei um stärksten Formalismus handelt, den sie bewältigen müssen. Für das Entwerfen eines neuen semantischen Typsystems wurde die linguistische Analyse der Anforderungen, vor allem der Kompetenzfragen, empfohlen. Dieser Prozess kann durch die Werkbank unterstützt werden, indem relevante Domänenkonzepte in den Anforderungen markiert werden. Dies wird in Abbildung 6.1 mit den Beispielanforderungen aus Abbildung 5.6 demonstriert.

Die so entstandene Konzeptliste muss in eine taxonomische Ordnung gebracht werden. Die DEn können eine solche Ordnung selbst vornehmen, oder durch einen Assistenten gestützt werden, der die Einordnung durch Fragen ermittelt, wie z. B.: »Ist jede denkbare *Ente* auch ein *Säugetier*?« Für diesen Vorgang eignet sich eine erweiterte semantische Typhierarchie, die Synonyme berücksichtigt und Homonyme als Typbezeichner verbietet. In Abbildung 6.1 wird das Interrogativpronomen »Wo« wie ein Synonym behandelt. Der Typ »Person« kann aus einer konzeptuellen Kategorie stammen, oder manuell hinzugefügt sein. Für diesen Vorgang können auch Informationen aus elektronischen Wörterbüchern verwendet werden.

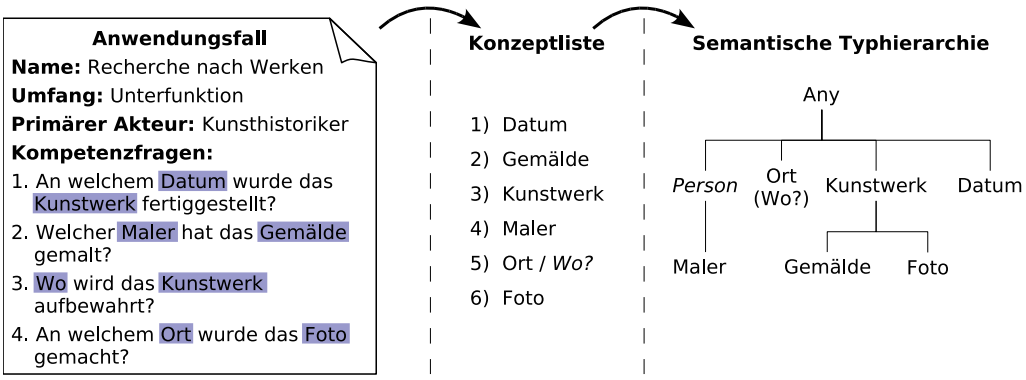

Abbildung 6.1: Werkbankunterstützung der linguistischen Analyse

Importieren von Beispielen aus CCM Systemen: Gerade bei Evolutionen und Personalisierungen ist es notwendig, vorhandene Instanzen zu importieren und nach den Bedürfnissen zu modifizieren. Die Importfunktion muss es erlauben, eine Auswahl der zu importierenden Instanzen des CCM Systems zu treffen. Außerdem müssen den Assets semantische Typen zugewiesen werden, die vor dem Importieren zusammen mit Abbildungsvorschriften für die Hilfsfunktion $map_S^{-1}()$ erstellt werden müssen. Diese Funktion ordnet Assetklassen semantische Typen für die Konvertierung von Assets zu AEs zu [Bossung07, Abs. 6.2.1], vergleiche auch $map_S()$ in Abschnitt 3.3.4.

Importieren von Asset Expressions: Da sich AEs nicht nur zur Modellierung von CCM Systemen eignen, können weitere AE Bestände verfügbar sein. Diese können für die Modellierung wiederverwendet werden, wobei ein abweichendes semantisches Typsystem der Quelle berücksichtigt werden muss.

Bei den vorgeschlagenen Funktionen für die Werkbank ist der grafische AE Editor am wichtigsten. Ein Umgang mit medialen Daten ist ohne Werkzeugunterstützung nicht praktikabel. Ein solches Werkzeug wird bereits im Zusammenhang mit einer Dissertation entwickelt [Bossung07]. Daneben ist das Importieren von Assets aus CCM Systemen die wichtigste Funktion für die Werkbank. Diese kann jedoch zur Zeit nur mit hohem technischen Aufwand realisiert werden, da ein reflektiver Zugriff auf die allgemeine Modulschnittstelle (siehe Abschnitt 2.3) noch nicht möglich ist.

6.1.3 Schemaerstellung

Die Schemaerstellung ist ein aufwendiger Vorgang, der von den Domänenexperten nicht ohne Werkbankunterstützung durchgeführt werden kann. Voraussetzung dafür ist allerdings, dass die Beispielinstanzen bereits in elektronisch erfasster Form vorliegen. Assetklassen können auf zwei Arten entstehen: durch Inferenz aus den Beispielen oder durch direkte Modifikation von Assetklassen. Letzteres ist vor allem für erfahrenere DEn interessant.

Interaktive Schemainferenz: Aus den Beispielinstanzen muss ein konzeptuelles Schema abgeleitet werden. Dies geschieht unter Berücksichtigung des vorläufigen Sche-

mas bzw. des Schemas des Altsystems, sofern eines vorliegt. Die Inferenz kann nicht vollständig automatisch erfolgen, wie dies bei typischen strukturellen Schemainferenzmethoden der Fall ist. Bei diesen Verfahren wird zunächst ein triviales gemeinsames Schema hergeleitet, das aufgrund eines formalen Optimierungskriteriums vereinfacht und generalisiert wird [WS03].

Die Notwendigkeit einer interaktiven Schemainferenz, die der Domänenexperte beeinflussen kann, ergibt sich aus folgenden Beispielen:

- Nur der DE kann entscheiden, ob zwei Instanzen der selben Klasse zuzuordnen sind. Durch die strukturelle Freiheit bei der Modellierung von AEs (siehe Abschnitt 3.3.1) können konzeptuell gleiche Entitäten unterschiedliche Strukturen besitzen.
- Auch der umgekehrte Fall ist denkbar, dass unterschiedliche Entitäten zufällig durch eine ähnliche Struktur beschrieben werden. Das gilt z. B. für triviale modellierte Entitäten, wenn deren Inhalte in einer AE lediglich durch einen Namen ergänzt werden. Dieser Fall kann nur von dem DE erkannt werden, der dem System Hinweise zur Lösung des Problems gibt.
- Zwei Instanzen können bis auf einige Attributnamen gleich sein. Auch hier kann nicht aufgrund der Struktur entschieden werden, ob sie der gleichen Klasse zugeordnet werden sollen.
- Nicht jede einzelne Instanz muss im Schema berücksichtigt werden. Einzelne Instanzen können strukturell einzigartig sein. Sofern diese nicht von herausragender Bedeutung für die Domäne sind, müssen sie nicht in das Schema aufgenommen werden. Dadurch wird erreicht, dass jede Klasse mehrere Instanzen besitzt.
- Die Entscheidung, ob der *Content* einer AE im Sinne von Assetklassen als `content` oder `concept` zu verwendet ist, kann nicht pauschal getroffen werden. So können z. B. Grafiken entweder einen Inhalt (Photo eines Gemäldes) oder eine konzeptuelle Beschreibung darstellen (eine Entwurfsskizze) [Sehring04, Abs. 3.3.1].

Schemamodifikation: Auch die Möglichkeit, unmittelbare Änderungen im vorläufigen Schema vorzunehmen, sollte vorhanden sein. Die DEn können, z. B. nach einer Evaluation, fehlende Attribute im Schema hinzufügen. Allerdings müssen diese Änderungen am Schema anschließend durch Beispiele belegt werden, vergleiche Abbildung 5.11. Auch die Namen von Klassen und Attributen können manuell geändert werden.

Vor allem die Realisierung der interaktiven Schemainferenz ist anspruchsvoll. Die DEn sind auf eine einfache und möglichst intuitiv bedienbare Werkbank angewiesen. Sie müssen jedoch in den Inferenzvorgang einbezogen werden, der sehr formaler Natur ist. Damit dieser Vorgang für die DEn einfach gestaltet werden kann, sind aufwendige Benutzeroberflächen mit Assistenten und anderen Hilfsfunktionen notwendig. Auch die Schulung der DEn muss adäquat berücksichtigt werden.

6.1.4 Evaluation

In der Evaluation muss geklärt werden, ob ein System auf Basis des vorläufigen Schemas den expliziten und impliziten Anforderungen der DEn genügt. Dies erfordert die Generierung von Prototypen, die durch die Werkbank unterstützt werden kann. Die Überprüfung, ob sich Bestandsdaten in einem personalisierten oder evolutionären System weiterverwenden lassen, kann nur durch die verwendeten Generatoren für Abbildungsmodule geklärt werden.

Integration des Modellkompilers: Der Modellkompiler ist zum Zeitpunkt dieser Arbeit zu kompliziert, als dass die DEn ihn zur Generierung von Prototypen verwenden könnten. Aus diesem Grund dient die Werkbank als vereinfachte Oberfläche zur Systemgenerierung, die durch ein einfaches Kommando gestartet wird. Die in Abschnitt 5.2.2 eingeführten Generatorpakete können als Distributionsform für Modellkompilerkonfigurationen realisiert werden. Auch die automatisierte Verteilung der Generatorpakete über das Internet ist denkbar. Die DEn werden so nicht mit den Aufgaben der Installation und Inbetriebnahme des Modellkompilers belastet.

Überprüfung der Abbildbarkeit zwischen Schemata: Nur durch einen Probelauf des Generators für Abbildungsmodule mit dem alten und dem neuen Schema kann sichergestellt werden, dass die Übernahme bestehender Daten möglich ist. Bei der Generierung eines Prototypen können die Meldungen dieses Generators abgefangen werden. Gibt es bei der automatischen Abbildung zwischen den Schemata Probleme, sind die Meldungen in einer für die DEn verständlicher Form auszugeben.

Anforderungsverwaltung: Sollte die Anforderungsanalyse in der Werkbank unterstützt werden, können ihre Ergebnisse in der Evaluation weiterverwendet werden. Wenn die DEn die Anforderungen am Prototypen überprüfen, können die Anwendungsfälle bzw. die Kompetenzfragen in der Werkbank als bearbeitet gekennzeichnet werden.

Sollte die Abbildbarkeit zwischen den Schemata nicht gegeben sein, gibt es zwei Möglichkeiten: Entweder passen die DEn das Schema an, so dass die Abbildung möglich wird. Alternativ können sie Hinweise geben, wie die Abbildung zwischen den Schemata durchgeführt werden soll. Die Abbildung zwischen Daten verschiedener XML-Schemata kann durch ein interaktives Werkzeug unterstützt werden [BSGAK04]. Ob ein solcher Vorgang so einfach gestaltet werden kann, dass er für Domänenexperten leicht ausführbar ist, muss noch untersucht werden.

6.1.5 Inbetriebnahme

Obwohl die Inbetriebnahme kein Teil des Modellierungsprozesses ist (Abbildung 5.1), kann sie durch die Werkbank im Sinne der Anwender unterstützt werden. Sollen die Beispielinstanzen aus dem Modellierungsprozess – und eventuell aus anderen Beständen (siehe Abschnitt 3.4) – in das CCM System übernommen werden, so müssen die AEs in Assets konvertiert werden, siehe Abbildung 5.12.

Die Zuordnung von AEs zu Assetklassen kann nicht ausschließlich aufgrund des Schemas geschehen. Da die Assetklassen lediglich die Struktur wiedergeben, können semantisch unterschiedliche AEs mit gleicher Struktur nicht sinnvoll den Assetklassen zugewiesen werden. Um solche Unterscheidungen treffen zu können, müssen die Assetklassen

auch mit semantischen Typen assoziiert werden, ähnlich wie beim Importieren von Instanzen aus CCM Systemen. Darüber hinaus kann es vorkommen, dass einzelne AEs nicht sämtliche Attribute des Assets definieren. In diesem Fall müssen Standardwerte für die Attribute definiert werden.

6.2 Interaktive Schemainferenz durch Clustering

6.2.1 Motivation

Gängige Methoden zur Schemainferenz sind für die Werkbank zur konzeptuellen Modellierung ungeeignet, siehe Abschnitt 6.1.3. Es wird ein interaktives Verfahren zur Schemainferenz benötigt, da nur der DE entscheiden kann, ob ein bestimmtes Schema die Domäne gut modelliert. Der DE wird durch das interaktive Verfahren ein Bestandteil des Inferenzalgorithmus. Zwei Ansätze zur interaktiven Schemainferenz werden im folgenden kurz beschrieben: Eine strukturelle Inferenzmethode und die Schemainferenz durch Clustering.

Interaktive strukturelle Schemainferenz

Bei der interaktiven strukturellen Schemainferenz werden bewährte Verfahren zur Schemainferenz, wie sie z. B. für XML Schemata üblich sind [WS03] in Verbindung mit Benutzerinteraktionen angewendet. In diesem Verfahren werden zunächst für alle Asset Expressions entsprechende Assetklassen hergeleitet [Bossung07, Abs. 7.2]. Diese Menge von Klassen wird durch Anwendung elementarer Operationen auf Schemaebene vereinfacht und generalisiert. Beispiele für elementare Operationen sind z. B. [Bernstein03]: *match* (berechnet eine Abbildung zwischen zwei Klassen) und *merge* (zwei Klassen werden zu einer zusammengefasst, wobei zwei gleiche Attribute zu einem verschmelzen).

Einige Operationen können unmittelbar ausgeführt werden, wie beispielsweise das Zusammenfassen zweier exakt gleicher Klassen. Viele Vereinfachungen können aber nur in Absprache mit dem DEn durchgeführt werden, z. B. wenn die Attributnamen zweier Klassen geringfügig voneinander abweichen. (Für weitere Beispiele siehe Abschnitt 6.1.3) Dies hat zur Folge, dass die Schemainferenz für den Anwender trotz Werkzeugunterstützung ein aufwendiger Vorgang ist, vor allem, da sie in jeder Iteration der BGM durchgeführt wird.

Um dem DEn die gleichen Fragen nicht mehrmals zu stellen, kann eine Wissensbasis aufgebaut werden, in der frühere Entscheidungen des Anwenders festgehalten werden. Die Wissensbasis kann bei folgenden Iterationen des Modellierungsprozesses herangezogen werden. Das Problem bei diesem Bedienungsmodell ist, dass es für den Anwender nicht ersichtlich ist, wie die im Hintergrund aufgebaute Wissensbasis die Schemainferenz beeinflusst. Das gilt vor allem für den wahrscheinlichen Fall, dass sich die Ansichten des DEn im Verlauf der Modellierung ändern. Wenn der DE zuvor zwei Instanzen der gleichen Klasse zugeordnet hat, die er in einer späteren Iteration als unterschiedlich ansieht, kann die Wissensbasis zu unerwünschten Ergebnissen führen.

Diese Form der interaktiven Schemainferenz beruht zwar auf bewährten Algorithmen, allerdings führt das Interaktionsmodell zu Schwierigkeiten. In diesem Fall kann das Interaktionsmodell als Beantwortung von Nachfragen beschrieben werden. Die Folge ist ein Mangel an Transparenz für den DE, der nicht abschätzen kann durch welche vorigen

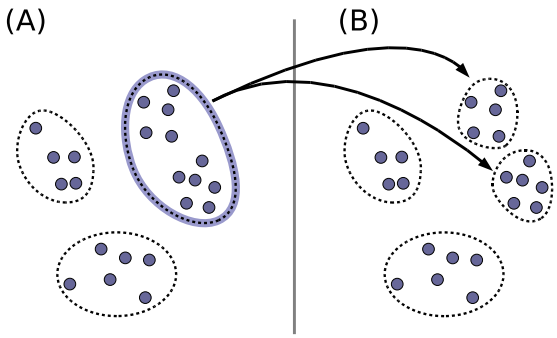

Abbildung 6.2: Navigation durch Cluster von Instanzen

Entscheidungen der Inferenzprozess weiter beeinflusst wird.

Schemainferenz durch Clustering

Für die Werkbank zur beispielgetriebenen Modellierung wird daher ein anderer Ansatz für die Schemainferenz vorgeschlagen, der sich von den üblichen Ansätzen unterscheidet. Die Motivation für diesen Ansatz ist das Bestreben, die Bedienung der Werkbank möglichst einfach zu gestalten. Die DEn sollen den Inferenzprozess durch ein geeignetes und einfaches Interaktionsmodell steuern: durch die Navigation durch Instanzmengen.

Die Instanzen (AEs) werden mit Hilfe der Clusteranalyse (*data clustering*, [JD88]) in Mengen von ähnlichen Elementen aufgeteilt, wobei für jede Assetklasse ein eigener Cluster entsteht. Der DE beeinflusst dabei die Aufteilung der Instanzen, indem er z. B. die Werkbank auf ungewünschte Zuordnungen hinweist. Abbildung 6.2 (A) veranschaulicht eine Aufteilung von Instanzen auf drei Cluster in einer zweidimensionalen Darstellung. Der Anwender identifiziert eine Menge, die nach seiner Meinung zu unterschiedliche Instanzen enthält. Daraufhin reagiert die Werkbank mit der Einführung eines neuen Clusters, siehe Abbildung 6.2 (B). Aus den Instanzen dieses neuen Clusters wird eine neue Assetklasse hergeleitet.

Der Schemainferenz durch Clustering liegen zwei Gedanken zugrunde: Zum Einen beeinflusst der DE die Clusteranalyse so, dass die Instanzmengen eine sinnvolle Klassenzuordnung ergeben. Anschließend wird aus den Instanzen jedes Clusters eine Assetklasse inferiert. Diese Inferenz kann ohne Eingriffe des DE erfolgen, da die Zuordnung der Instanzen zu den Clustern bereits im Vorfeld geklärt wurde.

Der entscheidende Vorteil der Schemainferenz durch Clustering ist das zugrundeliegende Interaktionsmodell – das der Navigation. Gerade wenn der Schemainferenz viele Beispielinstanzen zugrundeliegen, ist ein solcher Ansatz von Vorteil, weil dann im interaktiven strukturellen Verfahren sehr viele Nachfragen beim Anwender nötig sind. Darüber hinaus ist das Navigieren durch Datenbestände ein verbreitetes Prinzip und wird bei vielen Webapplikation angewendet. Daher sollte ein solches Interaktionsmodell auch für die DEn leicht verständlich sein. Nachteil dieses Ansatzes ist seine Neuartigkeit und die damit verbundenen Risiken. Zu diesem Zeitpunkt sind keine anderen Quellen bekannt, in denen Methoden des statistischen Lernens für eine Schemainferenz verwendet werden.

6.2.2 Ansatz

Clusteranalyse (data clustering)

Die Clusteranalyse (*data clustering*) ist ein Werkzeug der multivarianten statistischen Datenanalyse. Das Ziel des Verfahrens ist es, ähnliche Elemente s in einem sogenannten *Cluster* zusammenzufassen. Ausgangspunkt ist die Gesamtmenge aller Elemente $\mathbf{S} = \{s_1, s_2, \dots\}$ und ein Distanzmaß $d(s_i, s_j)$ $\mathbf{S} \times \mathbf{S} \to \mathbb{R}^+$, welches die Unterschiedlichkeit zwischen den einzelnen Elementen quantifiziert. Je nach verwendetem Algorithmus werden weitere Ausgangsdaten benötigt, wie z. B. die Anzahl der Cluster oder eine Qualitätsschranke. Daraufhin berechnet der Algorithmus eine möglichst optimale Zuordung der Elemente zu verschiedenen Clustern. Dabei wird ein vom Algorithmus abhängiges Qualitätsmaß für Cluster optimiert (*objective function*, siehe [JTLB04]). Die Clusteranalyse wird den Verfahren des unüberwachten (automatischen) statistischen Lernens [HTF01] zugeordnet. Im Gegensatz dazu gehören Verfahren zur Mustererkennung (*pattern recognition*) [Schurmann96] den überwachten Verfahren an.

In der Literatur wird am häufigsten der Sonderfall betrachtet, dass der Datenraum $\mathbf{D} \supset \mathbf{S}$ ein endlichdimensionaler Vektorraum ist ($\mathbf{D} = \mathbb{R}^n$). Viele Algorithmen lassen sich allerdings auch auf andere Datenräume wie z. B. metrische Räume anwenden.

Es existieren zahlreiche Algorithmen für die Clusteranalyse. Diese unterscheiden sich z. B. nach der Struktur des Datenraums, den benötigten Eingabeparametern, dem Qualitätsmaß für die Cluster und anderen Eigenschaften. Viele Algorithmen basieren auf Heuristiken, da das Grundproblem NP-hart ist [DPV06, Abs. 9.2.2]. Die Vielzahl der Algorithmen soll hier nicht diskutiert werden, dazu sei auf die umfangreiche Literatur verwiesen [HTF01, JD88, Zschunke03] und auf einen empirischen Vergleich von Algorithmen [JTLB04]. Stattdessen soll hier als Beispiel der einfache Algorithmus *k-means* kurz vorgestellt werden, der Grundlage für den in dieser Arbeit entwickelten Algorithmus ist. In diesem Algorithmus wird die Anzahl der Cluster k fest vorgegeben.

1. Inital die Mittelpunkte $c_1, \dots, c_k$ auswählen, auch eine zufällige Wahl ist möglich.
2. Zuordnung jedes Elements $s \in \mathbf{S}$ zu dem Mittelpunkt c_i, der ihm am nächsten ist, auch *nearest neighbour classification* genannt [HTF01, Kap. 13]. Dazu wird das Distanzmaß $d(s_i, s_j)$ verwendet.
3. Neuberechnung der Clustermittelpunkte c_i als Mittelwerte der ihnen zugeordneten Elemente (z. B. ein arithmetisches Mittel im $\mathbb{R}^n$).
4. Wiederhole die Schritte 2 und 3 bis sich die Zuordnung der Elemente zu den Clustern nicht mehr ändert.

Es kann vorkommen, dass der Algorithmus bei bestimmten Anfangswerten für die Clustermittelpunkte nicht terminiert. Daher ist es ratsam, ein zweites Abbruchkriterium zu verwenden.

Inferenzalgorithmus

Der im Folgenden beschriebene Inferenzalgorithmus besitzt die gleiche Struktur wie der zuvor beschriebene k-means Clusteralgorithmus. Dieser besteht im Wesentlichen aus zwei

Phasen, die iterativ durchgeführt werden, der *Klassifikation* und der *Optimierung*. In der *Klassifikation* wird jedes Element $s \in \mathbf{S}$ einem Cluster $\mathbf{C_i} \subseteq \mathbf{S}$ zugeordnet, der durch einen Clustermittelpunkt c_i repräsentiert wird. Dazu wird ein Distanzmaß verwendet, das die Unterschiedlichkeit zwischen einem Clustermittelpunkt c_i und einem Element s_j quantifiziert ($d(s_j, c_i)$). Anschließend wird durch die *Optimierung* jeder Clustermittelpunkt c_i so angepasst, dass er im Zentrum aller ihm zugewiesener Elemente $s \in \mathbf{C_i}$ liegt.

Die Anwendung eines solchen Algorithmus auf das Schemainferenzproblem gelingt, wenn man die Beispielinstanzen in Form von AEs als Elemente versteht und als Clustermittelpunkte eine Struktur wählt, die einer Assetklasse entspricht. Im Gegensatz zu k-means und vielen anderen Clusteralgorithmen, bei denen die Clustermittelpunkte und die Elemente aus der gleichen Grundmenge (üblicherweise dem $\mathbb{R}^n$) stammen, ist hier die Menge der Beispielinstanzen $\mathbf{B} \subset \mathbf{AE}$ von den Clustermittelpunkten $c \in \mathbf{CC}$ (*cluster center*) verschieden.

Neben der Anwendung des Clustering-Algorithmus muss dieser so erweitert werden, dass eine Schemainferenz durchgeführt wird. Die Clustermittelpunkte müssen daher äquivalent zu Assetklassen sein, so werden in der Optimierung mit den Clustermittelpunkten auch die Assetklassen angepasst. Allerdings geben die Clustering-Algorithmen keine Hinweise, wie eine taxonomische Beziehung zwischen den Clustermittelpunkten aufgebaut werden kann. In der Clusteranalyse werden die Cluster als disjunkt angesehen.

Damit ein Clustering-Algorithmus mit der gleichen Struktur wie k-means auf die Beispielinstanzen $e \in \mathbf{B}$ und Clustermittelpunkte angewendet werden kann und dabei eine Schemainferenz ermöglicht, sind folgende Probleme zu lösen:

Repräsentation der Clustermittelpunkte Für die Clustermittelpunkte $c \in \mathbf{CC}$ ist eine geeignete Repräsentationsform zu finden. Ein Clustermittelpunkt muss eine Art Mittelwert von mehreren AEs repräsentieren und sich gleichzeitig in eine äquivalente Assetklasse übersetzen lassen.

Distanzmaß Darauf aufbauend ist ein geeignetes Distanzmaß zwischen einer Beispielinstanz $e \in \mathbf{B}$ und einem Clustermittelpunkt c zu entwickeln. Dieses Distanzmaß gibt eine strukturelle Ähnlichkeit wieder, so dass den Instanzen Clustern von gleicher Struktur zugeordnet werden. Außerdem sollte das Distanzmaß eine semantische Ähnlichkeit widerspiegeln, so dass die Instanzen eines Clusters ähnliche Entitäten repräsentieren.

Optimierungsalgorithmus Aus einem Cluster muss ein neuer Clustermittelpunkt berechnet werden. Es wurde bereits erwähnt, dass die Mittelpunkte c eine zu Assetklassen äquivalente Struktur besitzen. Der Optimierungsalgorithmus beinhaltet demnach eine einfache automatische Schemainferenz, die aus den AEs eines Clusters gemeinsame Strukturen herleitet.

Taxonomiebildung Der Clustering Algorithmus muss um eine Möglichkeit erweitert werden, Taxonomien von Clustermittelpunkten und Clustern aufzubauen. Zu diesem Zweck wird nach der Optimierung eine weitere Phase in dem Inferenzalgorithmus vorgesehen.

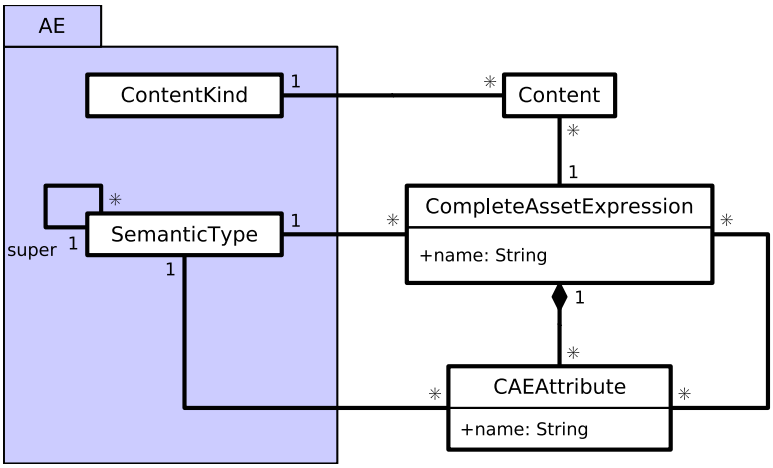

Abbildung 6.3: Modell einer vollständigen Asset Expression

Visualisierung Damit die Domänenexperten den Inferenzprozess einfach steuern können, müssen die Assetklassen und Cluster übersichtlich präsentiert werden. Nur so können die DE einfach durch eine große Menge der Beispielinstanzen navigieren.

Im Folgenden werden diese Probleme nacheinander gelöst und damit im Detail beschrieben, wie ein solcher Schemainferenzalgorithmus funktioniert. Dafür werden zunächst die Datenstrukturen des Algorithmus, zu denen auch die Clustermittelpunkte gehören, und sein Ablauf erläutert.

Datenstrukturen des Algorithmus

Die Beispielinstanzen liegen in Form von Asset Expressions ohne offene Abstraktionen vor. Deren syntaktische Struktur ist nicht von Interesse: Ob in einer Applikation ein Ausdruck unmittelbar oder in Form einer benannten Referenz verwendet wird ist für die Schemainferenz unbedeutend. Die *Complete Asset Expressions* (CAEs) werden vielmehr als Mengen von benannten Attributen interpretiert. Abbildung 6.3 zeigt das vereinfachte Modell der CAEs, das in dem Algorithmus zum Einsatz kommt.

Abbildung 6.4 veranschaulicht die Modellierung der Clustermittelpunkte $c \in \mathbf{CC}$, die einem objektorientierten Typsystem mit einfacher Vererbung entsprechen. Ein Clustermittelpunkt (*Cluster Center*, CC) wird durch die Klasse `AEClusterCenter` repräsentiert, die eine Menge von benannten Attributen aggregiert, ähnlich wie die CAEs. Durch die Attribute, die sich leicht in eine äquivalente Assetklasse übersetzen lassen (vergleiche Abbildung 6.5), werden die strukturellen Gemeinsamkeiten der Elemente im Cluster zusammengefasst. Darüber hinaus steht jeder `AEClusterCenter` mit einer kleinen Anzahl von semantischen Typen in Beziehung. Dies ist eine möglichst kleine Menge von semantischen Obertypen für alle CAEs des Clusters.

Die Cluster Center stehen in einer taxonomischen Ordnung, wobei jeder CC mit genau einem direkt übergeordneten CC in Beziehung steht. Der triviale Cluster Center als Ursprung der Taxonomie ist `AnyAEClusterCenter`, der über keine Attribute verfügt. Die anderen CC besitzen neben einer Kardinalität einen Typ. Ein Typ ist entweder ein CC oder ein unstrukturierter Inhaltstyp.

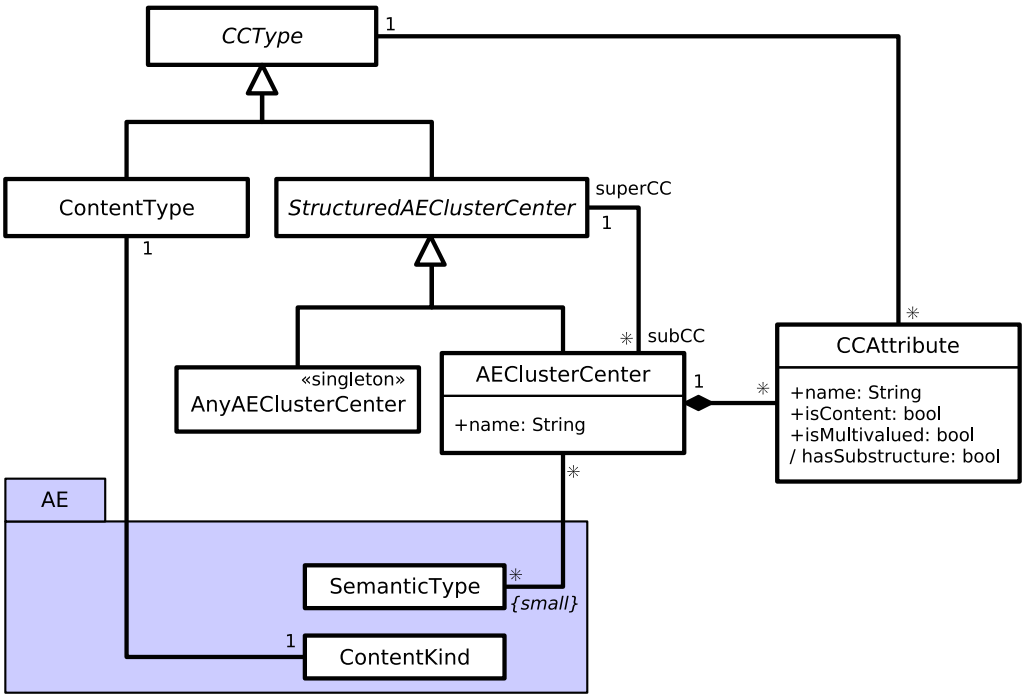

Abbildung 6.4: Modellierung eines Clustermittelpunktes für Asset Expressions

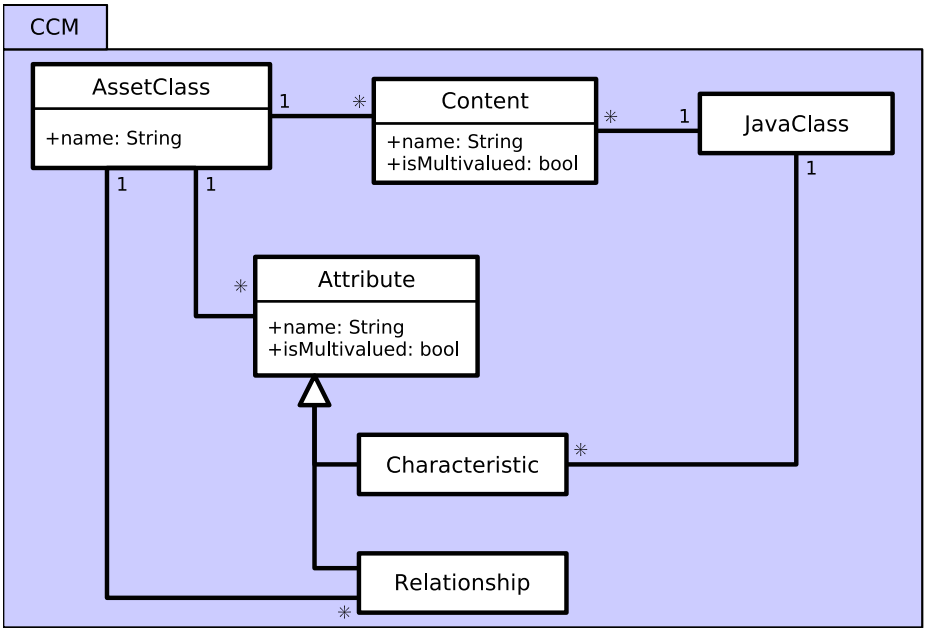

Abbildung 6.5: Metamodell der Assetdefinitionssprache

Die gezeigte Modellierung der CC erfüllt die zuvor formulierten Anforderungen: Sie sind ohne Schwierigkeiten in Assetklassen zu überführen und spiegeln die gemeinsamen Eigenschaften einer Menge von AEs wieder. Dies gilt sowohl für die strukturellen Gemeinsamkeiten (die Attribute) wie auch für deren semantische Typen.

In den AEs kann jede Entität durch eine individuelle Struktur beschrieben werden. Es besteht daher eine Unabhängigkeit zwischen der Bedeutung einer modellierten Instanz, d.h. der repräsentierten Entität, und der Struktur, mit der sie beschrieben wird. Diese Unabhängigkeit ist in Assetklassen nicht gegeben, die überwiegend strukturell motiviert sind. Trotzdem werden vollkommen unterschiedliche Entitäten nicht als Instanzen der gleichen Klasse modelliert, wie z. B. eine Person und ein Himmelskörper, die nur durch ihren Namen beschrieben werden. Durch die CC wird eine Verbindung zwischen der Struktur und der Semantik von Entitätsbeschreibungen eingeführt. Eine direkte Abbildung zwischen semantischen Typen und Assetklassen entsteht allerdings nicht.

Ablauf

Der Algorithmus benötigt als Ausgangsparameter die Menge der Beispielinstanzen $\mathbf{B} \subset \mathbf{CAE}$ und eine Menge von Cluster Center $\mathbf{M} \subset \mathbf{CC}$ (*Modell*), die als Ausgangspunkt für die Optimierung verwendet werden.

Das Verfahren zielt darauf ab, Cluster zu bilden, die den Ansprüchen des DEn gerecht werden. Formale Optimierungskriterien, wie sie bei gängigen Schemainferenzalgorithmen eingesetzt werden, werden nicht verwendet. Stattdesses steuert der DE den Prozess durch Eingriffe zwischen den einzelnen Läufen des Algorithmus. Das Vorwissen, das durch das Modell $\mathbf{M}$ in den Algorithmus eingeht, muss berücksichtigt werden, da es bereits die Wünsche des Anwenders oder das Schema eines Bestandssystems beinhaltet. Das hier angewendete Verfahren ist daher dem halbüberwachtem statistischen Lernen [Pedrycz05, Kap. 5] zuzuordnen und nicht dem unüberwachten Lernen, wie die Clusteranalyse.

Schritt 1 – Klassifikation: Zuordnung aller Instanzen $e \in \mathbf{B}$ zu einem Cluster $\mathbf{C_i}$ mit dem nächstgelegenen Cluster Center $c_i \in \mathbf{M}$.

Schritt 2 – Optimierung: Neuberechnung aller Cluster Center c_i auf Grundlage der ihnen zugeordneten Instanzen $\mathbf{C_i}$.

Wiederholung der Schritte 1. und 2. solange sich die Cluster Center c_i ändern.

Schritt 3 – Taxonomiebildung: Optimieren der taxonomischen Ordnung der Cluster Center, sofern dies vom DEn gewünscht wird. Wurde eine Veränderung in der Hierarchie der CC durchgeführt, werden die Schritte 1.-3. wiederholt.

Schritt 4 – Visualisierung: Darstellen der Cluster Center und Clusterzuordnung, so dass der DEn den Status der Schemainferenz begutachten kann.

Genau wie beim k-means Algorithmus muss ein sekundäres Abbruchkriterium in Schritt 3. realisiert werden. Die Möglichkeit einer Oszillation der Cluster kann auch hier nicht ausgeschlossen werden.

6.2.3 Schritt 1 – Klassifikation

Die Aufgabe der Klassifikation ist es, alle Beispielinstanzen $e \in \mathbf{B}$ dem Cluster $\mathbf{C_i} \subseteq \mathbf{M}$ mit dem nächsten Cluster Center $c_i \in \mathbf{CC}$ zuzuordnen. Die Menge der Beispiele $\mathbf{B}$ und die Menge der Cluster Center $\mathbf{M}$ sind dabei vorgegeben. Dazu wird das Distanzmaß $d_{iCC}(e, c)$ verwendet, das den Abstand zwischen einer Instanz und einem CC gibt. Die Abkürzung iCC bedeutet, dass dieses Distanzmaß nur zwischen einer Instanz un einem Cluster Center definiert ist, die aus unterschiedlichen Grundmengen stammen. Dies unterscheidet $d_{iCC}(e, c)$ von dem Distanzmaß $d(\cdot, \cdot)$ der üblichen Clusteranalyse, das die Distanz zwischen zwei Elementen aus der gleichen Grundmenge angibt.

Das Distanzmaß muss zwei Aspekte wiedergeben: die strukturelle Distanz und die semantische Distanz. Anders ausgedrückt sollen eine Instanz und ein Cluster Center mit einer geringen Distanz zwei Eigenschaften teilen:

- Sie sollen semantisch ähnlich sein. Das bedeutet, dass die Instanz eine Entität repräsentiert, die gleich oder ähnlich zu den anderen Entitäten des Clusters ist.
- Sie sind strukturell ähnlich. Das heißt, dass sich die Instanz mit der gleichen Struktur beschreiben lässt wie die anderen Elemente des Clusters. Daraus folgt, dass sie sich gut durch eine gemeinsame Assetklasse beschreiben lassen.

Aus diesem Grund wird die Distanz zwischen einer Instanz und einem Cluster Center als gewichtete Summe zweier Distanzmaße gewählt: Der semantischen Distanz d_{sem} und der strukturellen Distanz d_{st}:

$$d_{iCC}(e, c) := \alpha \cdot d_{sem}(e, c) + (1 - \alpha) \cdot d_{st}(e, c) \qquad , \alpha \in [0, 1]$$

Statt einer gewichteten Summe erscheinen auch Verfahren des *multi-view clustering* als vielversprechend, weil in diesen mehrere Distanzmaße zwischen den Elementen berücksichtigt werden. Aufgrund der Neuartigkeit des hier vorgestellten Verfahrens bleibt die Untersuchung, ob sich diese Verfahren auch für die interaktive Schemainferenz einsetzen lassen, für nachfolgende Arbeiten offen.

Im folgenden werden zunächst die Distanzmaße definiert und anschließend der Klassifikationsalgorithmus erläutert, da hier kein gewöhnlicher Nächster-Nachbar Klassifikator verwendet werden kann.

Semantische Distanz

Die semantische Distanz $d_{sem}(e, c)$ gibt die Ungleichheit zwischen einer Entität (durch e repräsentiert) und einer Menge von Entitäten (repräsentiert durch die Ausdrücke in $\mathbf{C}$) wieder. Eine allgemeine Quantifizierung der Ungleichheit zwischen Entitäten ist sehr schwierig, weshalb hier eine Heuristik auf Grundlage der semantischen Typen verwendet wird.

Zu diesem Zweck wird ein heuristisches Distanzmaß zwischen zwei semantischen Typen eingeführt $d^T_{sem}(T_1, T_2)$, das ausschließlich auf der Taxonomie der semantischen Typen beruht. Abbildung 6.6 veranschaulicht die Typdistanz an einem Beispiel. Die Distanz zwischen den beiden Typen *Buch* und *Bild* wird ermittelt als die Summe der Kantengewichte des kürzesten Weges im Vererbungsbaum zwischen den beiden Typen. Es ist offensichtlich, dass dieser über den speziellsten gemeinsamen Obertyp *Werk* führt. Die

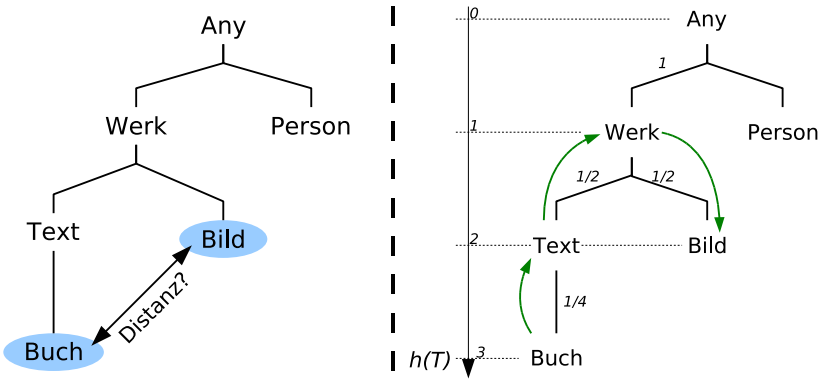

Abbildung 6.6: Distanz zwischen semantischen Typen

Kante zwischen zwei Typen wird mit einem Faktor gewichtet, der sich mit zunehmender Entfernung von der Wurzel *Any* exponentiell verkleinert. Die semantischen Typen sind in Namensräume eingeteilt, die für verschiedene Anwendungsdomänen verwendet werden. Für jeden Übergang zwischen zwei Domänen wird ein Betrag δ veranschlagt.

Dieser Heuristik für die semantische Typdistanz liegen folgende Überlegungen zugrunde: Je tiefer ein Typ im Baum liegt, desto spezifischer ist das Domänenkonzept, das er repräsentiert. Aus diesem Grund sind zwei Typen, die von der Wurzel abgeleitet sind (*Werk* und *Person*) als unterschiedlicher anzusehen, als Typen, die sich auf einer tieferen Ebene treffen (z. B. zwei Arten von Büchern: *Roman* und *Gedichtband*).

In einer rekursiven Formulierung ergibt sich die Berechnungsvorschrift für die semantische Typdistanz $d^T_{sem}(T_1, T_2)$ wie folgt, wobei $h(T)$ die Entfernung eines Typen T von *Any* bezeichnet.

$$d^T_{sem}(T_1,T_2) = \begin{cases} \frac{1}{2^{h(T_1)}}, & \text{wenn } T_1 \text{ direkter Obertyp von } T_2, \text{ sie in der gleichen Domäne liegen} \\ \frac{1}{2^{h(T_1)}} + \delta, & \text{wenn } T_1 \text{ direkter Obertyp von } T_2 \text{ und sie in verschiedenen Domänen liegen} \\ d^T_{sem}(T_1,T_m) + d^T_{sem}(T_m,T_2), & \text{wenn } T_1 \text{ direkter Obertyp von } T_m \text{ und } T_m \text{ Obertyp von } T_2 \text{ ist} \\ d^T_{sem}(T_s,T_1) + d^T_{sem}(T_s,T_2), & \text{wenn } T_s \text{ der speziellste gemeinsame Obertyp von } T_1 \text{ und } T_2 \text{ ist} \end{cases}$$

Die semantische Distanz zwischen Instanz und Cluster Center $d_{sem}(e, c)$ wird berechnet, indem die minimale Typdistanz zwischen dem semantischen Typ von e und jedem semantischen Typ von c gewählt wird. Dabei werden die Distanzen mit einem Faktor $\varsigma < 1$ begünstigt, bei denen der Typ der Instanz Untertyp des Clustertyps ist. Schließlich sollen die semantischen Typen des CC Obertypen für alle Elemente des Clusters darstellen. Für die folgende Berechnungsvorschrift wird zunächst die Hilfsfunktion $d'_{sem}()$ eingeführt. Die Funktion $type_{sem}(e)$ ermittelt den semantischen Typ einer Asset Expression e, entsprechend ermittelt $types_{sem}(c)$ die Menge der semantischen Typen, die mit dem Cluster Center c assoziert werden.

$$d'_{sem}(T_e,T_c) = \begin{cases} \varsigma \cdot d^T_{sem}(T_e,T_c), & \text{wenn } T_c \text{ Obertyp von } T_e \text{ ist} \\ d^T_{sem}(T_e,T_c), & \text{sonst} \end{cases} \quad ,0 \leq \varsigma < 1$$

$$d_{sem}(e,c) = min\Big\{ d'_{sem}(T_e, T_c) \quad | \quad T_e = type_{sem}(e), T_c \in types_{sem}(c) \Big\}$$

Ausgefeiltere Modelle für die semantische Distanz sind möglich. In diesem Fall wurde nur der semantische Typ des Ausdruckes e selbst berücksichtigt, aber auch die semantischen Typen der Attribute von e können in die Betrachtung mit einbezogen werden, z. B. durch Kontraduktion und Abduktion [Bossung07, Abs. 7.1] Darüber hinaus ist es grundsätzlich möglich, die Inhalte in die Distanzberechnung mit einzubeziehen: Ausdrücke mit ähnlichen Inhalten repräsentieren wahrscheinlich ähnliche Entitäten. Ein solcher Ansatz wird in [RHV03] vorgestellt.

Strukturelle Distanz

Die strukturelle Distanz soll wiedergeben, wie gut eine Instanz mit einem Cluster Center in struktureller Hinsicht übereinstimmt. Die strukturellen Unterschiede zwischen Instanz und CC, die beide die Form von benannten Attribute besitzen, wird in Form einer Editierdistanz berechnet. Ausgangspunkt ist eine Instanz, die in der Struktur eines CC vorliegt. Diese Struktur wird durch Bearbeitungsschritte so umgeformt, dass sie mit der Struktur des CC identisch ist, wobei für jede elementare Bearbeitungsoperation Kosten veranschlagt werden. Die Distanz ergibt sich aus der minimalen Summe der elementaren Operationen, die zur Umwandlung der Instanz erforderlich sind. Mit einem ähnlichen Verfahren kann auch überprüft werden, welche XML Dokumente auf dem gleichen Schema basieren [NJ02].

Die Berechnung der strukturellen Distanz $d_{st}(e,c)$ zwischen der Instanz e und dem Cluster Center c geschieht demnach wie folgt:

1. Die Instanz e in eine Menge von `CCAttriutes` $\mathbf{A_e}$ konvertieren. (siehe auch Abbildung 6.4)

2. Konvertieren des Cluster Center c in die Attributmenge $\mathbf{A_c}$.

3. Berechnung der minimalen Editierdistanz, um die Attributmenge $\mathbf{A_e}$ in die des CCs $\mathbf{A_c}$ zu überführen. Diese Funktion wird mir $d^A_{st}(\mathbf{A_e}, \mathbf{A_c})$ bezeichnet.

Tabelle 6.1 gibt qualitativ die Kosten der einzelnen elementaren Bearbeitungsoperationen wieder. Es fällt auf, dass die Kosten asymetrisch sind, z. B. sind die Kosten für das Hinzufügen eines Attributes gering, die für das Entfernen hoch. Dem liegt zugrunde, dass die Instanzen eines Clusters möglichst alle Attribute ihres CC besitzen sollten. Besitzt eine bestimmte AE Attribute, die nicht in der entsprechenden Assetklasse auftreten, spricht das nicht gegen die Klassenzugehörigkeit: Jede AE kann mit einer individuellen, der repräsentierten Entität angepassten Struktur beschrieben werden. Ähnliche Überlegungen führen zu den anderen Kosten.

Die eigentliche Berechnung der minimalen Kosten kann man mit rekursiven Algorithmen wie einer vollständigen Suche durchführen. Dabei können aus Effizienzgründen Techniken der Memoisation (*memoization* [CLRS01, Kap. 15] verwendet werden.

Bei den Operationen in Tabelle 6.1, die sich auf den Attributtyp beziehen, ist zu beachten, dass hier ein Attributtyp einem Cluster Center entspricht. Um eine AE e in

Operation	**Kosten**
Attribut hinzufügen	niedrig
Attribut entfernen	sehr hoch
Attributname ändern	niedrig
Attributtyp aufweiten	mittel
Attributtyp einengen	sehr niedrig
Mengenwertigkeit setzen	niedrig bis mittel
Mengenwertigkeit entfernen	sehr niedrig

Tabelle 6.1: Kosten der elementaren Operationen für die strukturelle Distanz

eine äquivalente Attributmenge $\mathbf{A_e}$ umzuwandeln, wird – entsprechend der Konvertierung von AEs in Assets – eine Hilfsfunktion benötigt. Diese entspricht $map_S(T_S)$ (siehe Abschnitt 3.3.4), allerdings wird durch sie ein semantischer Typ auf eine Inhaltsart T_{CK} (*content kind*) abgebildet. Genau wie bei $map_S()$ legt der DE Abbildungsvorschriften für die semantischen Typen fest, die bei der Schemainferenz als Basistypen für Attribute interpretiert werden sollen.

$$map_{SCC}(T_S) = \begin{cases} T_{CK}, & \text{wenn eine Abbildungsvorschrift für } T_S \\ & \text{existiert} \\ AnyAEClusterCenter, & \text{sonst} \end{cases}$$

Die Konvertierung eines Attributes einer AE zu einem Attribut $a \in \mathbf{A_e}$, welches durch CCs typisiert ist, geschieht folgendermaßen:

- Besitzt der semantische Typ der an das Attribut applizierten Asset Expression e_a eine Abbildungsvorschrift für map_{SSC}, so wird ein entsprechender *ContentType* als Typ für a verwendet (siehe auch Abbildung 6.4).
- Ist noch keine Clusterzuordung bekannt, so wird dem Attribut der Typ *AnyAEClusterCenter* zugeordnet.
- Ist bereits eine Clusterzuordung bekannt, so wird dem Attribut als Typ der Cluster Center zugewiesen, dem e_a zugeordnet wurde.

Zuordnungsalgorithmus

Die Zuordnung der Instanzen aus $e \in \mathbf{B}$ zu einem Cluster $\mathbf{C_i}$ wird durch einen Nächster-Nachbar Klassifikator durchgeführt. Es ist nicht möglich den üblichen Algorithmus zu verwenden, da die Konvertierung einer Instanz in eine Menge von Attributen $\mathbf{A}$ – wie sie für die Berechnung der strukturellen Distanz nötig ist – von einer bestehenden Clusterzuordnung abhängt. Aus diesem Grund wird ein iterativer Zuordnungsalgorithmus verwendet, der diesen Umstand berücksichtigt:

1. Es wird zu Anfang davon ausgegangen, dass alle substrukturierten Attribute einer AE auf den Typ *AnyAEClusterCenter* abgebildet werden. Dies entspricht der Definition von $map_{SSC}(T)$.

2. Es wird eine Klassifikation durchgeführt, in der jede Instanz e dem Cluster $\mathbf{C_i}$ mit der geringsten Distanz $d_{iCC}(e, c_i)$ zugeordnet wird. Das entspricht der üblichen Nächster-Nachbar Klassifikation.

   Dabei wird bei der Berechnung der strukturellen Distanz immer die akuelle Clusterzuordnung beachtet. Das bedeutet, dass sowohl $d_{st}(e, c)$ wie auch $d_{iCC}(e, c)$ von der in der vorigen Iteration vorgenommenen Clusterzuordnung $\mathbf{Z}$ abhängig sind.

$$\mathbf{Z} = \{(c_1, \mathbf{C_1}), \ldots, (c_m, \mathbf{C_m})\}, \quad c_i \in \mathbf{M}$$

3. Schritt 2. wird solange wiederholt, bis sich die Zuordnung der Instanzen zu den Clustern $\mathbf{C_i}$ nicht mehr ändert.

Weil eine Oszillation der Cluster nicht ausgeschlossen werden kann, ist in Schitt 3. ein sekundäres Abbruchkriterium zu verwenden.

6.2.4 Schritt 2 – Optimierung

In der Optimierung werden die Cluster Center semantisch und strukturell angepasst, so dass die Distanz zwischen den CC und den Elementen des Clusters minimiert wird. Das bedeutet, dass für einen Cluster $\mathbf{C}$ mit Mittelpunkt c der Mittelpunkt so angepasst wird, dass $\sum_{e \in \mathbf{C}} d_{iCC}(e, c)$ minimiert wird. Die Optimierung der dem CC zugeordneten semantischen Typen und die Optimierung der Struktur wird in den folgenden Abschnitten beschrieben.

Semantische Optimierung der Cluster Center

Durch die semantische Optimierung soll jedem Cluster Center c eine Menge an semantischen Typen zugeordnet werden, so dass für jeden semantischen Typ, der im Cluster vorkommt $\{type_{sem}(e) | e \in \mathbf{C}\}$, ein Obertyp vorhanden ist. Eine solche Menge, die aus möglichst wenigen Elementen besteht, kann als ein semantischer Mittelpunkt verstanden werden. Die Typen müssen allerdings so gewählt werden, dass die kumulierte semantische Distanz des CC zu den Elementen des Clusters minimal wird.

Diese minimale Obertypmenge wird durch eine Heuristik ermittelt, der an hierarchisch-agglomerative Algorithmen zur Clusteranalyse [JD88, Abs. 3.2] angelehnt ist. Für die Berechnung werden Mengen von semantischen Typen als Tripel dargestellt, wobei jeder Tripel eine Menge von AEs des Clusters repräsentiert, die einen gemeinsamen semantischen Obertyp besitzen:

$$s_i^{sem} = (T_i, n_i, d_i^{\Sigma}) \quad \text{mit} \quad \begin{cases} T_i & \text{gemeinsamer Obertyp der Typmenge} \\ n_i & \text{Anzahl der AEs, die der Typmenge zugeordnet werden} \\ d_i^{\Sigma} & \text{kumulierte semantische Distanz zw. } T_i \text{ und den AEs} \end{cases}$$

Eine möglichst kleine Menge an Obertypen wird nach folgendem Algorithmus ermittelt:

1. Aufbau einer initalen Tripelmenge aus den Elementen des Clusters. Kommen m unterschiedliche semantische Typen in den AEs des Clusters vor, so werden m Tripel $\mathbf{S} = \{s_1, \ldots, s_m\}$ erstellt. Dabei sind auch die Elemente der untergeordneten

Cluster mit einzubeziehen, d.h. alle AEs, die entweder c_i oder einem der von c_i abgeleiteten CCs zugeordnet sind.

Dabei ist T_i der semantische Typ, n_i die Anzahl der AEs mit Typ T_i und d_i^{Σ} wird auf 0 festgelegt.

2. Zusammenfassen der beiden Tripel s_i^{sem} und s_j^{sem}, mit der geringsten semantischen Typdistanz $d_{sem}^{T}(T_i, T_j)$. Die beiden Tripel werden aus der Menge aller Tripel $\mathbf{S}$ entfernt und durch einen neuen Tripel mit einem gemeinsamen Obertyp ersetzt.

   Sei T_{sup} der speziellste gemeinsame Obertyp von T_i und T_j, so ergibt sich der neue Tripel zu:

$$s' = \Big(T_{sup}, \Big(n_i + n_j\Big), \Big(n_i \cdot d_{sem}^{T}(T_{sup}, T_i) + n_j \cdot d_{sem}^{T}(T_{sup}, T_j) + d_i^{\Sigma} + d_j^{\Sigma}\Big)\Big)$$

3. Der Schritt 2. wird solange wiederholt, bis ein Minimum für das Optimierungskriteriums $f(\mathbf{S})$ gefunden ist.

$$f(\mathbf{S}) = g(|\mathbf{S}|) + \sum_{i=1}^{|\mathbf{S}|} d_i^{\Sigma} \qquad mit \qquad g(l) = l^2$$

Die semantischen Typen, die direkt in der Tripelmenge $\mathbf{S}$ mit dem kleinsten Optimierungskriterium $f(\mathbf{S})$ enthalten sind, ergeben die gewünschte minimale Menge an semantischen Obertypen. Der Term $g(|\mathbf{S}|)$ in dem Kriterium dient dazu, die Menge klein zu halten.

Strukturelle Optimierung der Cluster Center

Die Optimierung der Struktur des Cluster Center c geschieht, indem seine Attribute $a \in \mathbf{A_c}$ so angepasst werden, dass die strukturelle Distanz innerhalb des Clusters $\mathbf{C}$ minimiert wird. Dabei muss beachtet werden, dass c auch Attribute seines übergeordnetem Cluster c_{sup} erbt, die nicht modifiziert werden können.

In dem beschriebenen Algorithmus wird der Cluster Center $\mathbf{C}$ nicht vollständig berechnet, sondern lediglich der bisherige CC so modifiziert, dass er dem wahren strukturellem Mittelpunkt näher kommt. Da der Inferenzalgorithmus iterativ ausgeführt wird, ist dies jedoch kein Hindernis.

Die Menge aller Attribute von c, einschließlich der geerbten, wird im Folgenden mit dem Formelzeichen $\mathbf{A_c^*}$ gekennzeichnet. Die Elemente, die c sowie allen seiner abgeleiteten CCs zugeordnet worden sind, bilden die Menge $\mathbf{C}^*$. Als Maß für die Güte des Cluster Centers wird die kumulative strukturelle Gesamtdistanz innerhalb des Clusters verwendet:

$$d_{st}^{\Sigma}(\mathbf{A}) := \sum_{c \in \mathbf{C}^*} d_{st}^{d}(\mathbf{A_c}, \mathbf{A})$$

Der Algorithmus zur strukturellen Optimierung besitzt folgende Schritte:

1. Zunächst wird $\mathbf{A}$ von einem Attribut befreit, das nicht mehr durch die Menge der Beispiele gestützt wird: Es wird nacheinander jedes Attribut $a \in \mathbf{A_c}$ entfernt und überprüft, ob die kumulative Gesamtdistanz dadurch kleiner wird. Existieren mehrere solcher Attribute, so wird letztendlich das Attribut a_i aus $\mathbf{A}$ entfernt, für das $d_{st}^{\Sigma}(\mathbf{A_c})$ am kleinsten wird.

2. Nun werden die Typen und Kardinalitäten der Attribute optimiert. Für jedes Attribut $a_i \in \mathbf{A}$ werden folgende Schritte zur Typoptimierung durchgeführt:

   (a) Das Attribut a_i wird einmal als mengenwertig und einmal als einzelwertig angenommen. a_i' bekommt die Kardinalität zugewiesen, mit der

   $$d_{st}^{\Sigma}\Big(\big(\mathbf{A_c} \setminus \{a_i\}\big) \cup \{a_i'\}\Big)$$

   am kleinsten wird.

   (b) Ein optimierter Typ des Attributes kann gefunden werden, indem man a_i verschiedene Typen zuweist und den Typ wählt, für den die kumulative Gesamtdistanz am kleinsten wird. Bei der Optimierung werden folgende Typen berücksichtigt: der von a_i selbst, der Obertyp von a_i und alle direkt von a_i abgeleiteten Typen.

   Durch dieses Vorgehen wird der Typ um eine Stufe in der Taxonomie der Cluster Center optimiert. Durch wiederholte Ausführung dieses Schrittes wird der optimale Typ gefunden.

   Der Vorgang kann mit jedem Attribut $a_i \in \mathbf{A}$ durchgeführt werden, der keinen *ContentType* (siehe Abbildung 6.4) besitzt, da diese primitiven Typen in keiner taxonomischen Ordnung stehen (entsprechend den Charakteristika der ADL).

3. Anschließend wird ein Attribut gesucht, dass in vielen Instanzen vorhanden ist, aber noch nicht im Cluster Center berücksichtigt wurde. Nach einem solchen Attribut wird in der Menge aller Instanzen aus $\mathbf{C}$ gesucht. Die Menge aller vorkommenden Attribute ist:

   $$\mathbf{A^C} := \bigcup_{c \in \mathbf{C}} \mathbf{A_c}$$

   Man ermittelt ein solches Attribut, indem $\mathbf{A}$ nacheinander um jedes Attribut $a_i \in \mathbf{A^C}$ ergänzt wird. Dabei wird a_i zusätzlich der Typoptimierungsprozedur aus Schritt 2. unterzogen. Der CC wird letztendlich um das Attribut a_i ergänzt, durch das sich das kleinste $d_{st}^{\Sigma}(\mathbf{A} \cup \{a_i\})$ ergibt.

Es sei darauf hingewiesen, dass im Schritt 3 des Algorithmus ein neues Attribut a_i nur in den c unmittelbar zugeordneten Instanzen $\mathbf{C}$ gesucht wird. Für die Berechnung der kumulativen Distanz und die Typoptimierung von a_i werden jedoch auch die Instanzen der untergeordneten Cluster $\mathbf{C}^*$ mit einbezogen. Der Grund für dieses Vorgehen ist folgendes Szenario: Wenn man die Suche nach neuen Attributen auf $\mathbf{C}^*$ vornimmt, ist es wahrscheinlich, dass der CC c_{sup} eines spärlich besetzten Clusters $\mathbf{C_{sup}}$ strukturell zu einem untergeordnetem CC mit vielen Elementen entarten kann. Ein solches Verhalten ist nicht wünschenswert, da auch spärlich besetzte Klassen gewünscht sein können.

Der Schritt 3 des Algorithmus ist mit einem relativ hohem Rechenaufwand verbunden. Dieses Problem kann wahrscheinlich durch Heuristiken gelöst werden, die eine Vorauswahl unter den zur Verfügung stehenden Attributen treffen.

6.2.5 Schritt 3 – Taxonomiebildung

In der üblichen Clusteranalyse werden keine Taxonomien von Cluster Centern berücksichtigt. Der Aufbau einer solchen Taxonomie ist für die Schemainferenz notwendig – sie

entspricht der Vererbungshierarchie der äquivalenten Assetklassen. Aus diesem Grund werden die CCs hier auch als Typen bezeichnet.

Es besteht zwischen dem Cluster Center c_{sub} und dem Obertyp c_{sup} die Beziehung, dass sämtliche Attribute des Obertyps an den Untertyp vererbt werden. Aufgrund dieser strikten strukturellen Beziehung zwischen einem Ober- und einem Untertyp ist die Bildung der Clustertaxonomie im Wesentlichen strukturell motiviert. Ob ein Cluster Center c_i strukturell eine Obermenge von c_j darstellt, kann einfach mit einem asymmetrischen Distanzmaß überprüft werden, wie es schon in Abschnitt 6.2.3 mit $d^A_{st}(\mathbf{A_i}, \mathbf{A_j})$ definiert wurde.

Die Entscheidung, ob eine Vererbungsbeziehung zwischen c_i und c_j tatsächlich sinnvoll ist, kann genau wie bei der interaktiven strukturellen Schemainferenz nur vom DE getroffen werden. Aus diesem Grund werden von der Werkbank Vorschläge gemacht, welche Klassen in eine Vererbungsbeziehung gesetzt werden können. Diese werden von dem Domänenexperten angenommen oder abgelehnt.

Die Aufhebung einer taxonomischen Beziehung geschieht auf ähnliche Weise: Die Werkbank bereitet dem DE Vorschläge, potentiell ungeeignete Vererbungsbeziehungen aufzulösen. Es deutet auf eine ungewünschte Vererbungsbeziehung zwischen c_{sup} und c_{sub} hin, wenn der Cluster $\mathbf{C_{sub}}$ spärlich belegt ist und seine Elemente eine große strukturelle Distanz zum Cluster Center besitzen. In diesem Fall kann auf eine ungeeignete Struktur von c_{sub} geschlossen werden. Durch den Inferenzalgorithmus werden solche Strukturen im allgemeinen optimiert, sofern diese nicht von einem Obertyp geerbt werden.

Bei dieser Form der Taxonomiebildung wird keine Wissensbasis aufgebaut. Das hat zur Folge, dass der DE bei mehrfacher Durchführung dieses Schrittes die gleichen Fragen beantworten muss. Der Aufwand hierfür ist trotzdem als gering zu bewerten, da eine Veränderung der taxonomischen Ordnung lediglich nach umfassenden Veränderungen am Modell und damit auch an der Beispielmenge nötig ist. Deshalb ist die Taxonomiebildung nur selten durchzuführen.

6.2.6 Schritt 4 – Visualisierung

Der Domänenexperte beeinflusst den Schemainferenzprozess, indem er durch die Cluster navigiert und so Schwächen im Modell entdeckt. Er benötigt dafür eine entsprechende Visualisierung der Cluster und der Cluster Center, so dass die Zusammenhänge übersichtlich dargestellt werden.

Abbildung 6.7 skizziert eine mögliche Benutzeroberfläche für diesen Zweck. Die Taxonomie der Cluster – und damit auch die der Assetklassen – kann gut in einer Baumstruktur dargestellt werden. Die AEs der Cluster sind in einer zweidimensionalen Visualisierung als Punkteschwarm dargestellt. Durch Anwahl einer bestimmten Instanz wird der grafische Editor für Asset Expressions geöffnet, so dass der DE die angewählte AE betrachten und verändern kann. Instanzen, die im Kontext ihrer Klasse unpassend modelliert wurden, können durch ihre Position im Randbereich des Clusters identifiziert und – sofern dies gewünscht wird – auch bearbeitet werden. Zu der Abbildung 6.7 sei angemerkt, dass in der AE auf der rechten Seite semantische Typen verwendet werden, die Taxonomie auf der linken Seite jedoch aus Assetklassen besteht.

Statt der Darstellung als Punkteschwarm sind auch einfachere Visualisierungen denkbar. Zum Beispiel kann man die zu einem Cluster gehörenden Instanzen in einer Liste

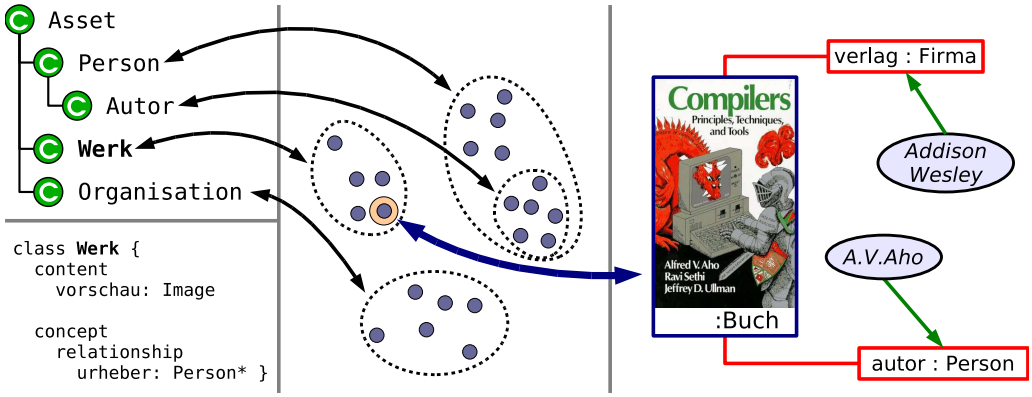

Abbildung 6.7: Visualisierung der Cluster in der Werkbank

darstellen. Wenn zu dem Namen der AE auch ihre Distanz zum Cluster Center aufgeführt wird, können strukturell unterschiedliche Instanzen auch in einer Listendarstellung einfach gefunden werden.

Für die Darstellung der Instanzen als Punkteschwarm können Techniken des multidimensionalen Skalierens (*multi-dimensional scaling*, MDS [JD88, Abs. 2.7]) angewendet werden. Durch sie ist es möglich, eine endlichdimensionale Darstellung von Objekten zu erzeugen, von denen nur die paarweisen Distanzmaße bekannt sind. Dabei wird eine Darstellung gesucht, so dass die relativen Abstände im endlichdimensionalem Raum möglichst gut mit den gegebenen Distanzen übereinstimmen. Die Verfahren des MDS gehen allerdings von einem symmetrischen Distanzmaß aus, das bedeutet $d(x, y) = d(y, x)$. Die in der Schemainferenz verwendete Distanz $d_{iCC}(e, c)$ ist asymetrisch und auch nur zwischen einer Instanz e und einem Cluster Center c definiert und nicht zwischen zwei Intanzen. Für die Anwendung dieser Verfahren wird also ein symmetrisches Distanzmaß zwischen zwei AEs benötigt. Dazu sind lediglich die Parameter zur Berechnung der strukturellen Distanz entsprechend anzupassen.

Dieses Verfahren hat allerdings den Nachteil, dass für die Visualisierung ein anderes Distanzmaß als für die Klassifikation verwendet wird. Es bleibt offen, ob ein Algorithmus für das MDS entworfen werden kann, der die CC und die asymmetrischen Distanzmaße entsprechend berücksichtigt.

Kapitel 7

Diskussion

Dieses Kapitel fasst die Ergebnisse dieser Arbeit zusammen und erläutert sie im Kontext verwandter Ansätze. Abschließend wird ein Ausblick auf mögliche Weiterentwicklungen vorgestellt.

7.1 Diskussion verwandter Ansätze

In dieser Arbeit wurde untersucht, wie die Domänenexperten (DEn) möglichst eigenständig konzeptuelle Modelle ihrer Domäne erstellen können. Dieses Problem wurde bisher nur in sehr wenigen Arbeiten untersucht, so dass hier als verwandte Ansätze auch solche betrachtet werden, welche die Rolle der Domänenexperten in der Modellierung besonders hervorheben.

Dieses Problem wurde deshalb so selten untersucht, weil die Domänenmodellierung meistens im Kontext einer Systementwicklung durchgeführt wird. Die Modellierung ist nur ein Arbeitsschritt in der Entwicklung und wird nicht als besonders aufwendig eingestuft, deshalb wird sie durch Modellierungsexperten durchgeführt. Durch die konzeptorientierte Inhaltsverwaltung verändern sich die Randbedingungen erheblich: Eine klassische Systementwicklung ist nicht notwendig, da die CCM Systeme vollständig generiert werden. Die konzeptuelle Modellierung wird damit zum aufwendigsten Vorgang in der Systementwicklung. Darüber hinaus wird auch nach der Entwicklung eines CCM Systems modelliert, nämlich bei jeder Evolution oder Personalisierung des Systems.

7.1.1 Delphi-Methode nach Holsapple und Joshi

Die kollaborative Methode zur Entwicklung von Domänenontologien nach Holsapple und Joshi [HJ02] wurde bereits in Abschnitt 4.4.1 beschrieben. In diesem iterativen Prozess werden viele DEn um Bewertungen und Verbesserungsvorschläge für eine Ontologie gebeten. Diese Vorschläge und Bewertungen werden von Modellierungsexperten in einer verbesserten Ontologie berücksichtigt, die in folgenden Iterationen weiter verfeinert wird.

Die Stärke dieses Ansatzes ist das kollaborative Vorgehen, durch das die Expertise von mehreren DEn in das Ergebnis einfließt. Allerdings ist die Durchführung der Delphi-Methode personal- und zeitintensiv. Die Domänenexperten sind zudem stark von den Modellierungsexperten abhängig, die alle Vorschläge in die Ontologie einarbeiten müssen.

In der Quelle [HJ02] bleiben einige Fragen ungeklärt, wie z. B., zu welchem Zweck die Ontologien erstellt wurden und ob sie sich in einem praktisch eingesetzten Informationssystem bewährt haben. Außerdem ist unklar, in welcher Form die Ontologien den Domänenexperten vorgelegt wurden. Die Erfahrungen mit der konzeptorientierten Inhaltsverwaltung zeigen, wie wichtig eine angemessene Repräsentation der Domänenkonzepte ist, siehe Abschnitt 2.4.

7.1.2 Kollaborative Entwicklung konzeptueller Schemata

In dem Artikel *Collaborative Conceptual Schema Design* [RR98] wird ein anderer Ansatz beschrieben, Domänenexperten in die Modellierung einzubeziehen. Zu diesem Zweck verwenden die Autoren ein Modellierungswerkzeug, das es mehreren DEn ermöglicht, simultan am selben konzeptuellen Modell zu arbeiten. Die Modellierung geschieht auf intensionaler Ebene in einer grafischen Sprache, die der ER Notation (*Entity-Relationship*, [Chen76]) ähnlich ist. Durch das Werkzeug steht den Anwendern eine gemeinsame virtuelle Modellierungsfläche zur Verfügung. Darüber hinaus sind Mechanismen zur Konfliktlösung, Kommunikation und Hilfsfunktionen für die Modellierung vorgesehen.

Der Vorteil dieses Ansatzes liegt in der kollaborativen Domänenmodellierung, die ohne eine zeit- und personalintensive Delphi-Methode durchgeführt wird. Bei der Modellierung arbeiten Domänen- und Modellierungsexperten gemeinsam und unterstützen sich gegenseitig. Außerdem hat sich das Werkzeug in empirischen Versuchen bewährt, die ausschließlich mit Domänenexperten durchgeführt wurden. Dabei wurde gezeigt, dass die Funktionen zur Kollaboration entscheidend für eine effektive Modellierung sind.

Ein Nachteil dieses Ansatzes ist, dass er sich im wesentlichen auf das Werkzeug beschränkt. Die verwendete Modellierungsmethode wird in der Quelle nicht diskutiert. So bleibt unklar, wie die Domänenexperten vor den Experimenten geschult wurden – sie hatten zuvor keine Modellierungskenntnisse. Auch bei diesem Ansatz bleibt offen, ob und wofür die erstellten Schemata verwendet wurden.

7.1.3 DynamOnt

Das *DynamOnt* Projekt zielt darauf ab, Ontologien maßgeblich durch Domänenexperten entwickeln zu lassen [GWG06]. Die Modelle sollen für die Inferenz neuen Wissens geeignet sein und strengen Formalismen gehorchen. Das wird erreicht, indem für die Domänenexperten vereinfachte Sichten auf die Ontologie erstellt werden, so dass sie mit den Formalismen nicht direkt in Kontakt kommen. Diese Ziele sollen durch eine Modellierungsmethode und unterstützende Werkzeuge erreicht werden.

Das Projekt ist in einem frühen Stadium, so dass die Anforderungen an die Modellierungsmethode und den Prozess bereits formuliert sind, konkrete Lösungsansätze aber noch nicht vorliegen. Die zentralen Anforderungen an Prozess und Werkzeug sind:

- Modellierungshilfen für Domänenexperten
- kollaborative Modellierung
- Unterstützung von verknüpften Ontologien (Upper-Level Ontologie, verwandte Domänenontologien)
- Unterstützung für alle Phasen im Lebenszyklus der Anwendung

7.2 Bewertung

Das Ziel dieser Arbeit ist es, Domänenexperten die konzeptuelle Modellierung zu ermöglichen. Zu diesem Zweck wurde ein Prozess entwickelt, der durch eine Werkbank unterstützt wird. Durch die Werkbank können die Domänenexperten eigenständig modellieren, wobei eine vollkommene Unabhängigkeit von Modellierungsexperten nicht unter allen Umständen möglich ist. Die Annahme, Menschen könnten komplexe Vorgänge wie die Domänenmodellierung ohne Einarbeitung oder Schulung durchführen ist unrealistisch. Unter günstigen Umständen können die Domänenexperten den Vorgang ohne fremde Hilfe durchführen, z. B. wenn eine Modifikation des Modells durch Domänenexperten durchgeführt wird, die über Erfahrungen in Prozess und Werkbank verfügen.

Ob die Domänenexperten tatsächlich eigenständig konzeptuelle Modelle erstellen können muss noch empirisch belegt werden. Eine umfassende Realisierung der Werkbank und die Bewährung von Prozess und Werkbank in der Praxis stehen noch aus.

Im Vergleich zu den verwandten Ansätzen ist in dieser Arbeit sowohl die Methode wie auch ein entsprechendes Werkzeug berücksichtigt. Die bisherigen Ansätze, die DEn stärker in die Modellierung einzubeziehen, lassen sich in zwei Gruppen einteilen: Die eine Gruppe legt den Schwerpunkt auf den Modellierungsprozess, der eine besondere Arbeitsteilung zwischen Modellierungsexperten und Domänenexperten festlegt, so dass letztere einen größeren Einfluss auf die Modellierung bekommen (wie bei Holsapple und Joshi Abs. 7.1.1). Die zweite Gruppe unterstützt die DEn durch ein Werkzeug – Modellierungsmethoden und Prozesse sind sekundär, sofern sie überhaupt diskutiert werden. Dieser Gruppe ist der Ansatz *Collaborative Conceptual Schema Design* zuzuordnen (siehe Abschnitt 7.3.3). Der Ansatz von DynamOnt berücksichtigt zwar sowohl die Modellierungsmethode wie auch eine Werkzeugunterstützung, allerdings sind zur Zeit noch keine konkreten Ansätze veröffentlicht, wie die Probleme der Modellierung durch Domänenexperten gelöst werden können (siehe Abschnitt 7.1.3).

Die praktischen Erfolge der Ansätze von Holsapple & Joshi und dem dem *Collaborative Conceptual Schema Design* lassen sich nicht auf alle Gruppen von Domänenexperten übertragen: Die Versuche wurden stets mit Naturwissenschaftlern durchgeführt. Für die Probanden war die Modellierung zwar neu, allerdings sind Klassifikationssysteme in den meisten Naturwissenschaften gebräuchlich, so dass eine Modellierung auf extensionaler Ebene gewohnten Denkmustern entspricht. Die Erfahrungen mit den CCM Systemen haben gezeigt, dass z. B. Geisteswissenschaftler mit Modellierungsaufgaben auf Konzeptebene große Probleme haben. In dieser Arbeit wird der Standpunkt vertreten, dass moderne Informationssysteme nicht den Naturwissenschaften vorbehalten sein dürfen.

Im den folgenden Abschnitten werden die einzelnen Ergebnisse dieser Arbeit einzeln diskutiert: Der Prozess zur beispielgetriebenen Modellierung, die Werkbank und der Algorithmus zur interaktiven Schemainferenz durch Clustering.

Modellierungsprozess

Der Prozess zur *beispielgetriebenen Modellierung* vereint bewährte Ansätze und Praktiken vieler Modellierungsmethoden. Die entscheidenden Schritte des Prozesses, die Beispielgewinnung und die Schemaerstellung, können durch eine angemessene Werkzeugunterstützung so einfach gestaltet werden, dass sie von den Domänenexperten ohne

fachfremde Hilfe durchgeführt werden können. Die Domänenexperten können bei Bedarf auf die Unterstützung von Modellierungsexperten zurückgreifen, dies ist nötig, wenn die Domänenexperten über keine Erfahrung mit dem Prozess verfügen.

Aus diesem Grund ist der Prozess als Prozessframework ausgelegt und wird bei einer konkreten Anwendung an die Modellierungsaufgabe angepasst. Umfassende Projekte mit vielen unerfahrenen Domänenexperten benötigen organisatorische und beratende Hilfsaktivitäten. Bei geringfügigen Modifikationen eines Modells kann auf diese Aktivitäten verzichtet werden.

Es soll betont werden, dass die Domänenexperten durch den Prozess keineswegs zu Modellierungsexperten ausgebildet werden. Der Prozess setzt weder Modellierungskenntnisse (z. B. in der objektorientierten Analyse) voraus, noch werden diese bei der Modellierung erworben. Die beispielgetriebene Modellierungsmethode ist einzigartig, indem sie der gedanklichen Welt der Domänenexperten entgegenkommt und mit einem Minimum an Formalismen auskommt.

Werkbank

Es wurde gezeigt, wie die Domänenexperten bei der Durchführung des Modellierungsprozesses so durch eine Werkbank unterstützt werden können, dass sie die Modellierung ohne personelle Unterstützung durchführen können. Eine Einarbeitung in Prozess und Werkbank wird allerdings vorrausgesetzt.

Die softwaretechnische Realisierung der Werkbank und ihrer einzelnen Anwendungen steht noch aus, wobei die Anforderungen an die Bedienbarkeit sehr hoch sind. Die für den Modellierungsprozess wichtigsten Anwendungen der Werkbank sind in dem Stadium von Prototypen implementiert: der grafische Editor für Asset Expressions für die Beispielgewinnung und das Werkzeug zur interaktiven Schemainferenz. Es wurde somit gezeigt, dass die beschriebene Werkbank realisierbar ist.

Schemainferenzalgorithmus

Aus den hohen Anforderungen an die Bedienbarkeit der Werkbank ergibt sich ein Problem für die Schemainferenz: Die Schemainferenz muss als interaktiver Prozess gestaltet werden, der für die Domänenexperten möglichst einfach zu bedienen ist. Weil die bekannten Ansätze zur Schemainferenz vor diesem Hintergrund ungeeignet erscheinen, wird eine neuartiger Algorithmus beschrieben. Die *Schemainferenz durch Clustering* setzt Methoden des statistischen Lernens für die Schemainferenz ein.

Der Vorteil dieses Algorithmus ist ein einfaches, den Domänenexperten angemessenes, Interaktionsmodell, das auch bei umfangreichen Beispielmengen anwendbar ist. In diesem Zusammenhang sind auch effektive Visualisierungen möglich, so dass die Steuerung des Inferenzvorgangs weiter erleichtert wird. Allerdings wird dieser komplexe Algorithmus noch nicht so gut beherrscht, wie es für einen praktischen Einsatz wünschenswert ist.

7.3 Ausblick

7.3.1 Beispielgetriebene Modellierung

Obwohl der in dieser Arbeit entwickelte Modellierungsprozess auf bewährten Ansätzen aufbaut ist es wahrscheinlich, dass in der Praxis Schwächen offenbart werden. Außerdem wurden viele Details noch nicht untersucht, z. B. wie man Domänenexperten am besten für die Modellierung mit der Werkbank schult. Es ist abzusehen, dass der Prozess durch praktische Erfahrungen komplettiert und optimiert werden kann.

Es ist auch möglich, den Prozess auf theoretischer Ebene durch Methoden zum Prozess-Reengineering zu verbessern. Zum Beispiel kann die Methode *Map-driven Modular Method Re-engineering* (MMMR) [RR01, RMRD05] angewendet werden, um die beispielgetriebene Modellierung zu verbessern.

7.3.2 Werkbank

Wie bei aktuellen integrierten Entwicklungsumgebungen (IDEs) integriert die Werkbank verschiedenen Anwendungen, die noch nicht vollständig implementiert sind. Die softwaretechnische Realisierung solcher Systeme ist ein aufwendiges Unterfangen, vor allem weil hier die einfache Bedienung der Werkbank im Vordergrund steht.

Der Algorithmus zur interaktiven Schemainferenz ist aufgrund seiner Neuartigkeit noch nicht für den praktischen Einsatz geeignet. Er hängt von einer Reihe von Parametern ab, für die auf empirischen Weg geeignete Werte ermittelt werden müssen. Zudem sind bei großen Datenbeständen Effiziensprobleme zu erwarten, die allerdings durch Techniken wie *Caching* behebbar sind. Zudem eignen sich die Algorithmen zur Schemainferenz durch Clustering gut für die Parallelverarbeitung. Wie auch bei dem Modellierungsprozess, kann davon ausgegangen werden, dass sich Schwächen erst im praktischen Einsatz offenbaren, so dass die weitere Entwicklung noch nicht abzusehen ist.

7.3.3 Kollaborative Modellierung

Viele der zitierten Ansätze zur Schema- und Ontologieentlicklung betonen den positiven Einfluss kollaborativer Ansätze. Es ist naheliegend, dass gemeinschaftlich genutzte Informationssysteme, die aus dem Fachwissen eines einzelnen Domänenexperten entstanden sind, auf Akzeptanzprobleme stoßen. Aus diesem Grund muss auch die Modellierung kollaborativ durchgeführt werden.

Eine kollaborative Modellierung kann in Prozess und Werkbank leicht realisiert werden, indem man mehrere Domänenexperten an der Beispielgewinnung beteiligt. Ähnlich wie bei dem Ansatz *Collaborative Conceptual Schema Design* (siehe Abs. 7.1.2) können die Domänenexperten gleichzeitig mit einem gemeinsam genutzten Bestand von AEs arbeiten [Bossung07, Abs. 5.4.1]. Für diesen Ansatz sind Mechanismen zur Konfliktlösung unverzichtbar. Da Asset Expression Systeme auch als persönliche mediale Datenbanken verwendet werden, ist es naheliegend, bestehende persönliche AE-Bestände in den Modellierungsprozess zu übernehmen.

7.3.4 Weiterentwicklung der konzeptorientierten Inhaltsverwaltung

Anpassung der Assetsysteme an Asset Expressions

Bei der Beschreibung der Werkbank wird deutlich, dass sich Asset Expressions und Assets nicht direkt ineinander überführen lassen, eine Konvertierung ist nur unter Voraussetzungen möglich, siehe [Bossung07, Abs. 6.2]. Ein Grund dafür sind die semantischen Typen, die den CCM Systemen unbekannt sind. Stattdessen werden in vielen CCM Systemen Schlagworte verwendet, die einen ähnlichen Zweck wie die semantischen Typen erfüllen. Es ist daher naheliegend, die semantischen Typen auch in Assetsystemen einzuführen. Die Typ-Taxonomie kann zum Beispiel zu einer Hierarchie von Schlagworten erweitert werden, in der jedes Schlagwort einem semantischen Typ zugeordnet werden kann.

Darüber hinaus bieten Asset Expressions durch Selektoren die Möglichkeit, Teile eines medialen Inhaltes zu beschreiben. Im Gegensatz dazu beziehen sich Charakteristika und Beziehungen der Assets immer auf die gesamte Instanz. Selektoren können auch in Assetsystemen eingesetzt werden [Bossung07, Abs. 6.2.2]. Durch die aktuellen Entwicklungen bei Webbrowsern ist es sogar denkbar, Selektoren und Komponenten in einer webbasierten Benutzerschnittstelle zu realisieren.

Individuelle Offenheit und Dynamik durch AE Systeme

Die Erfahrungen mit den CCM Systemen haben gezeigt, dass eine Offenheit und Dynamik auf individueller Ebene zu Problemen führt. Durch zu häufige und undurchdachte Personalisierungen können die Daten der personalisierten Systemen nicht mehr automatisch auf das Schema des Bestandsystems abgebildet werden. Im schlimmsten Fall isolieren sich die personalisierten Systeme vollständig.

Für Offenheit und Dynamik auf individueller Ebene sind Asset Expression Systeme besser geeignet: Die Domänenexperten können ihre Entitäten frei modellieren, ohne an ein Schema gebunden zu sein. Es liegt daher nahe CCM und AE Systeme kombiniert anzuwenden [Bossung07, Abs. 7.2]: Ein CCM System wird für umfangreiche, gemeinschaftlich genutzte Datenbestände verwendet. Wenn ein Anwender einzelne Assets personalisieren möchte, importiert er sie in sein AE System. So entstehen im Laufe der Zeit unabhängige Bestände von personalisierten Assets in Form von AEs. Die persönlichen Bestände können z. B. mit Methoden der Clusteranalyse nach Kanidaten für neue Assetklassen durchsucht werden. Wenn mehrere Anwender Assets einer Klasse mit einer bestimmten Struktur personalisieren, deutet dies auf eine Möglichkeit zur Schemaevolution hin, wie z. B. die Einführung einer neuen Unterklasse.

Für die parallele Anwendung von AE und CCM Systemen ist auch eine gemeinsame Benutzeroberfläche denkbar, die dem Anwender auch bei Arbeit mit dem CCM System seine personalisierten Instanzen zeigt. So können die schemabasierten CCM Systeme mit den AE Systemen zu einem Werkzeug kombiniert werden.

Literaturverzeichnis

[Abbott83] Abbott, Russel J.: Program Design by Informal English Descriptions. Communications of the ACM, 26(11)

[Balzert99] Balzert, Heide: Lehrbuch der Objektmodellierung - Analyse und Entwurf. Spektrum Akademischer Verlag, 1999.

[Balzert01] Balzert, Helmut: Lehrbuch der Softwaretechnik. Spektrum Akademischer Verlag, 2001

[BB05] Buzan, Tony, Barry Buzan: Das Mind-Map Buch. Übersetzung der 5. aktualisierten Auflage. mvg-Verlag, 2005

[BBJW97] Boman, Magnus, Janis A. Bubenko Jr., Paul Johannesson, Benkt Wangler: Conceptual Modelling. Prentice Hall, 1997

[BCM03] Baader, Franz, Diego Calvanese, Deborah McGuinness: The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press, 2003

[Bernstein03] Bernstein, Phillip A.: Applying Model Management to Classical Meta Data Problems. Proceedings of the CIDR Conference, 2003

[BHL01] Berners-Lee, Tim, James Hendler, Ora Lassila: The Semantic Web. Scientific American, May 2001

[BMS84] Brodie, Michael L., John Mylopoulos, Joachim W. Schmidt: On Conceptual Modelling, Perspective form Artificial Intelligence, Databases and Programming Languages. Springer, 1984

[Bossung07] Bossung, Sebastian: Conceptual Content Modeling: Abstractions, Applications and System Implementations. *Bislang unveröffentlichte Dissertation*, Arbeitsbereich Softwaresysteme, Technische Universität Hamburg-Harburg, Deutschland, Veröffentlichung vorraussichtlich 2007

[Booch94] Booch, Grady: Object-Oriented Analysis and Design with Applications - 2nd ed. Benjamin/Cummings Publishing Company, 1994

[Brandt83] Brandt, Ingunn: A Comparative Study of Information Systems Design Methodologies. Information Systems Design Methodologies: A Feature Analysis. Proceedings of the IFIP WG 8.1 Conference, 1983

[BSGAK04] Bossung, Sebastian, Hermann Stoeckle, John Grundy, Robert Amor, John Hosking: Automated Data Mapping Specification via Schema Heuristics and User Interaction. Proceedings of the IEEE/ACM International Conference on Automated Software Engineering, 2004

[BSH06] Sebastian Bossung, Hans-Werner Sehring, Patrick Hupe and Joachim W. Schmidt. Open and Dynamic Schema Evolution in Content-intensive Web Applications. Proceedings of the Second International Conference on Web Information Systems and Technologies, pp. 109-116, April 2006

[Carl06] Carl, Henner: Generierung plattformunabhängiger Inhaltsverwaltungssysteme am Beispiel von Zope3. Studienarbeit, Arbeitsbereich Softwaresysteme, Technische Universität Hamburg-Harburg, Deutschland, 2006.

[Chayanam06] Chayanam, Venkata: Generation of Mapping Code for Schemata of Conceptual Content Management Systems. Master Thesis, Arbeitsbereich Softwaresysteme, Technische Universität Hamburg-Harburg, Deutschland, 2006.

[Chen76] Chen, Peter P.: The Entity-Relationship Model – Toward a Unified View of Data. ACM Transactions on Database Systems, 1(1): 9-36, 1976

[Cockburn00] Cockburn, Alistair: Writing Effective Use Cases. Addison-Wesley Professional, 2000

[CLRS01] Cormen, Thomas H., Charles E. Leiserson, Ronald L. Rivest, Clifford Stein: Introduction to Algorithms, Second Edition. MIT Press, 2001

[CY91] Coad, Peter, Edward Yourdon: Object-Oriented Analysis. Prentice Hall, 1991

[DPV06] Dasgupta, Sanjoy, Christos Papadimitriou, Umesh Vazirani: Algorithms. McGraw Hill, 2006, Vorabfassung erhältlich unter (5.2.2007):
`http://www.cse.ucsd.edu/users/dasgupta/mcgrawhill/`

[FowlerAP2] Fowler, Martin: Analysis Patterns 2 – Work in Progess. Homepage von Martin Fowler, erreichbar über (5.2.2007):
`http://www.martinfowler.com/ap2/`

[Fowler97] Fowler, Martin: Analysis Patterns - Reusable Object Models. Addison-Wesley, 1997.

[Fowler04] Fowler, Martin: UML Distilled, 3rd Edition: A Brief Guide to the Standard Object Modeling Language. Addison-Wesley, 2004

[FRF03] Fowler, Martin, David Rice und Matthew Foemmel: Patterns of Enterprise Application Architecture. Addison Wesley, 2002

[GFC04] Gómez-Pérez, Asunción, Mariano Fernández-López, Oscar Corcho: Ontological Engineering: with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web. Springer, 2004

[GHJV94] Gamma, Erich, Richard Helm, Ralph Johnson und John Vlissides: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, 1994

[GKNS] Geschichte der Kunstgeschichte im Nationalsozialismus. System erreichbar unter (5.2.2007):
`http://www.welib.de/gkns/index.html`

[Gottesdiener02] Gottesdiener, Ellen: Requirements by Collaboration: Workshops for Defining Needs. Addison-Wesley, 2002

[Grady92] Grady, Robert B.: Practical Software Metrics for Project Management and Process Improvement. Prentice Hall, 1992

[GWG06] Gruber, Andreas, Rupert Westenthaler, Eva Gahleitner: Supporting Domain Experts in creating formal knowledge models (ontologies). Projektreport, 2006, erhältlich über (5.2.2007):
`http://www.salzburgresearch.at/research/gfx/...`
`.../gruber_westenthaler2006_domain_experts_ontologies.pdf`

[IEEE-830] IEEE: Guide to Software Requirements Specification, ANSI/IEEE Std 830-1984.

[Hankin94] Hankin, Chris: Lambda calculi : A guide for computer scientists. Clarendon Press, 1994

[HJ02] Holsapple, Clyde W., K.D. Joshi: A Collaborative Approach to Ontology Design. Communications of the ACM, 45(2):42-47, 2002

[HTF01] Hastie, Trevor, Robert Tibshirani, Jerome Friedman: The Elements of Statistical Learning: Data Mining, Inference and Prediction. Springer, 2001

[Husted02] Husted, Ted: Struts in Action A Practical Guide to the Leading Java Web Framework. Manning Publications, 2002

[Jacobson92] Jacobson, Ivar: Object-Oriented Software Engineering: A Use Case Driven Approach. ACM press, 1992

[JD88] Jain, Anil K., Richard C. Dubes: Algorithms for Clustering Data. Prentice Hall, 1988

[JTLB04] Jain, Anil K., Alexander Topchy, Martin H.C. Law, Joachim M. Buhmann: Landscape of Clustering Algorithms. Proceedings of the IAPR, International Conference on Pattern Recognition, 2004

[KK06] Klar, Michael, Susanne Klar: Einfach Generieren: Generative Programmierung verständlich und praxisnah. Hanser Verlag, 2006

[Kruchten00] Kruchten, Philippe: The Rational Unified Process, An Introduction - 2nd edition. Addison-Wesley, 2000

[Larman03] Larman, Craig: Agile and Interative Development: A Manager's Guide. Addison Wesley, 2003

[Larman05] Larman, Craig: Applying UML and patterns, an introduction to object-oriented analysis and design and iterative development, 3rd edition. Prentice Hall, 2005

[LT75] Lindstone, Harold A., Murray Turoff: The Delphi Method: Techniques and Applications. Addison-Wesley, 1975 Ausgabe von 2002 erhältlich über (5.2.2007): `http://www.is.njit.edu/pubs/delphibook/`

[MG01] Maiden, Neil, Alexis Gizikis: Where Do Requirements Come From? IEEE Software, September/October 2001

[MJF03] Maiden, Neil, Sara Jones, Mary Flynn: Innovative Requirements Engineering Applied to ATM[1]. Proceedings Joint Eurocontrol/FAA ATM'2003 Conference, 2003

[Mofor06] Mofor, Gerald: Modeling of User Interfaces for Conceptual Content Management Systems. Master Thesis, Arbeitsbereich Softwaresysteme, Technische Universität Hamburg-Harburg, Deutschland, 2006

[NJ02] Nierman, Andrew, H.V. Jagadish: Evaluating Structural Similarity in XML Documents. Proceedings of 5th International Workshop on the Web and Databases, 2002

[Oestereich06] Oestereich, Bernd: Analyse und Design mit UML 2.1, 8. aktualisierte Auflage. Oldenbourg Verlag, 2006

[Pedrycz05] Pedrycz, Witold: Knowledge-Based Clustering: From Data to Information Granules. John Wiley & Sons, 2005

[Pieper04] Pieper, Riko: Anforderungsanalyse aus Sicht des Auftraggebers. Aufsatz in [Rupp04], Kapitel 1.4

[Revesz88] Revesz, György: Lambda-Calculus: Combinators, and Functional Programming. Number 4 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1988

[RHV03] Roddick, John F., Kathleen Hornsby, Denise de Vries: A Unifying Semantic Distance Model for Determining the Similarity of Attribute Values. Proceedings of the the Twenty-Sixth Australasian Computer Science Conference, 2003

[RMRD05] Ralyté, Jolita, Neil Maiden, Colette Rolland, Rébecca Deneckère: Applying Modular Method Engineering to Validate and Extend the RESCUE Requirements Process. Proceedings of the International Conference of Conceptual Modeling (ER'05), 2005

[RP06] Rechenberg, Peter, Gustav Pomberger (Herausgeber): Informatik Handbuch, 4. Auflage. Hanser Verlag, 2006

[RR98] Ram, Sudha, V. Ramesh: Collaborative Conceptual Schema Diesign: A Process Model and Prototype System. ACM Transactions on Information Systems, 16(4):347-371, 1998

[RR01] Ralyté, Jolita, Colette Rolland: An Approach for Method Reengineering. Proceedings of the International Conference of Conceptual Modeling (ER'01), 2001

[RR06] Robertson, Suzanne, James Robertson: Mastering the Requirements Process, Second Edition. Addison Wesley, 2006

[1] Abkürzung für ≫air traffic management≪

[Rupp04] Rupp, Chris: Requirements-Engineering und -Management: Professionelle, iterative Anforderungsanalyse für die Praxis, 3. Auflage. Carl Hanser Verlag, 2004

[SAA99] Schreiber, Guus, Hans Akkermans, Anjo Anjewierden: Knowledge Engineering and Management: The CommonKADS Methodology. MIT Press, 1999

[SBS05] Sehring, Hans-Werner, Sebastian Bossung, Joachim W. Schmidt: Active learning by personalization, lessons learnt from lessons in conceptual content management. Proceedings of the 1st International Conference on Web Information Systems and Technologies, 2005

[Schurmann96] Schürmann, Jürgen: Pattern Classification: A Unified View of Statistical and Neural Approaches. John Wiley & Sons, 1996

[Sehring04] Sehring, Hans-Werner: Konzeptorientierte Inhaltsverwaltung - Modell, Systemarchitektur und Prototypen. Dissertation, Arbeitsbereich Softwaresysteme, Technische Universität Hamburg-Harburg, Deutschland, 2004

[Sendall02] Sendall, Shane: Gauging the Quality of Examples for Teaching Design Patterns. In: Workshop on „Killer Examples“ for Design Patterns and Objects First. OOPSLA 2002, Conference on Object-Oriented Programming Systems, Languages and Applications, 2002

[Shneiderman00] Shneiderman, Ben: Creating Creativity: User Interfaces for Supporting Innovation. ACM Transactions on Computer-Human Interaction, 7(1):114-138, 2000

[Silverstone97] Silverstone, Len: The Data Model Resource Book, Vol. 1&2. John Wiley & Sons, 1997

[SM99] Shipman III, Frank M., Catherine C. Marshall: Formality Considered Harmful: Experiences Emerging Themes, and Directions on the Use of Formal Representations in Interactive Systems. Computer Supported Cooperative Work Vol. 8, 4-1999

[SSSS01] Staab, Steffen, Rudi Studer, Hans-Peter Schnurr, York Sure: Knowledge Processes and Ontologies. IEEE Intelligent Systems, Januar/Februar 2001

[SSW02] Schmidt, J.W., H.-W. Sehring und M. Warnke: Der Bildindex zur Politischen Ikonographie in der Warburg Electronic Library – Einsichten eines interdisziplinären Projektes. In: Pompe, Hedwig und Leander Scholz (Herausgeber): Archivprozesse. Die Kommunikation der Aufbewahrung, Seiten 238-268. Dumont, 2002

[Studer02] Studer, Rudi: Wissensmanagement. Vorlesung des Instituts für Angewandte Informatik und Formale Beschreibungsverfahren der Universität Karlsruhe. Abgehalten im Sommersemester 2002, Unterlagen erhältlich über (5.2.2007):
`http://www.aifb.uni-karlsruhe.de/Lehrangebot/Sommer2002/...`
`.../Wissensmanagement/`

[SV05] Stahl, Thomas, Markus Völter: Modellgetriebene Softwareentwicklung: Techniken, Engineering, Management. dpunkt.verlag GmbH, 2005

[Thompson99] Thompson, Simon: Haskell: The Craft of Functional Programming, Second Edition. Addison-Wesley, 1999

[TP91] Tse, T.H., L. Pong: An Examination of Requirements Specification Languages. The Computer Journal, vol. 34, no. 2, 1991

[VZ06] Vigenschow, Uwe, Guido Zockoll: Wieso, weshalb, warum? Erfolgsfaktor Mensch, Teil3: Fragetechniken in der Anforderungsanalyse von Softwareprojekten erfolgreich einsetzen. Javamagazin 08/2006, Software & Support Verlag GmbH, 2006

[Wegner02] Wegner, Holm: Analyse und objektorientierter Entwurf eines integrierten Portalsystems für das Wissensmanagement. Dissertation, Arbeitsbereich Softwaresysteme, Technische Universität Hamburg-Harburg, Deutschland, 2002.

[Wiegers05] Wiegers, Karl E.: Software-Requirements, Deutsche Ausgabe der Second Edition. Microsoft Press Deutschland 2005.

[WMWiPe] *unbekannte Autoren:* Wissensmanagement, aus Wikipedia, der freien Enzyklopädie. erreichbar über (5.2.2007):
`http://de.wikipedia.org/wiki/Wissensmanagement`

[WRL05] Wolf, Henning, Stefan Roock, Martin Lippert: eXtreme Programming: Eine Einführung mit Empfehlungen und Erfahrungen aus der Praxis, 2. überarbeitete und erweiterte Auflage. dpunkt.verlag GmbH, 2005

[WS03] Wong, Raymond K., Jason Sankey: On Structural Inference for XML Data. Technical report, University of New South Wales, 2003

[Zschunke03] Zschunke, Matthias: Clustering algorithms. Seminarunterlagen zu *Algorithmen zum Wirkstoffdesign*, Universität Tübingen, 2003, erhältlich unter (5.2.2007):
`http://www-ra.informatik.uni-tuebingen.de/lehre/ws03/sem_wirkstoff.html`