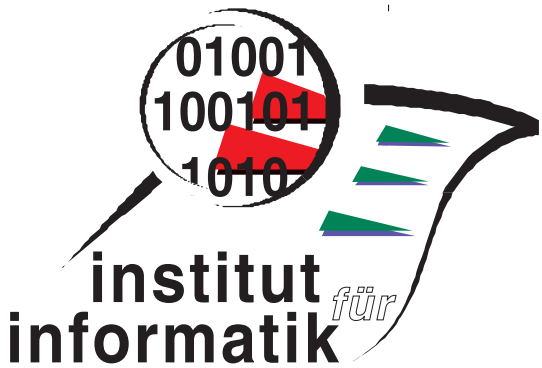
UNIVERSITÄT AUGSBURG

Enterprise Content Integration: Documentation, Implementation and Syndication using Intelligent Metadata (ECIDISI)

Tanja Sieber **Florian Lautenbacher**

Report 2007-17 Dezember 2007



INSTITUT FÜR INFORMATIK
D-86135 AUGSBURG

ECIDISI.

Enterprise Content Integration: Documentation, Implementation and Syndication using Intelligent Metadata.

Verfasser: Tanja Sieber,
University of Miskolc, HU
Florian Lautenbacher,
University of Augsburg, DE

Dieses Projekt wurde von BAYHOST unter der Bewilligungsnummer I 120-01h/ gefördert.

Kurzbeschreibung

Requirements Engineering zielt darauf ab, Anforderungen an ein Produkt so genau wie möglich aus der Sicht der zukünftigen Kunden festzuhalten und genau zu spezifizieren. Dabei beschriebene Anforderungen können in der Regel weder in der Software-Entwicklung noch in sich anschließenden Dokumentationsprozessen aufgrund stattfindender Medien- und Informationsbrüche direkt wiederverwendet werden.
In diesem Bericht wird untersucht, inwieweit solche Methodologien aus den Bereichen Ontologiedesign und linguistische Dokumentmodellierung Ansätze bieten können, interne technische Dokumentationsprozesse aus dem Bereich des Requirements Engineering effektiver zu gestalten und dadurch eine automatisierte Wiederverwendbarkeit von Dokumentationsbausteinen für die Software-Entwicklung und technische Dokumentation zu erhöhen.

Schlüsselbegriffe

Content; Requirements Engineering; linguistische Modellierung; Ontologiedesign; Kommunikationsmodelle; Sprechakttheorie; semantisches Datenmodell;

Inhaltsverzeichnis

1 PROJEKTKONZEPT 7

1.1 EINFÜHRUNG 7
1.2 PROBLEME / HERAUSFORDERUNGEN 8
1.3 ADRESSIERTE PROBLEMFELDER 11
1.4 PROJEKTZIELE 11
1.5 AUFBAU DIESES BERICHTS 12

2 DATEN, METADATEN, CONTENT 13

2.1 SEMANTISCHES DATENMODELL NACH SIEBER/KAMMERER 13
2.2 APORIE DES DATUMS – METADATUM 17
2.3 CHARAKTERISTIKA VON METADATEN 18
2.4 KRITISCHE REFLEXION AUSGEWÄHLTER DEFINITIONEN DES TERMINUS *CONTENT* 19
2.5 TERMINUSDEFINITION FÜR FORSCHUNGSPROJEKT 21

3 CONTENT INTEGRATION: PROBLEME UND HERAUSFORDERUNGEN 24

3.1 PRODUCT DEVELOPMENT CYCLE 24
3.2 REQUIREMENTS ENGINEERING 26
3.3 SOFTWARE REQUIREMENTS ENGINEERING 27
3.4 USE CASES 28
3.5 REUSE IM SOFTWARE ENGINEERING 32
3.6 REUSE IM DOCUMENTATION ENGINEERING 39
3.7 ENTERPRISE CONTENT INTEGRATION 43

4 BEISPIELE FÜR USE-CASES 44

4.1 ZUSAMMENFASSUNG DER ANALYSE 44
4.2 BEISPIEL USE CASE ‚KREDITKARTE SPERREN' 44

5 GRUNDLAGEN DER KOMMUNIKATION 46

5.1 BASISKONZEPTE DER KOMMUNIKATIONSTHEORIE 46
5.2 SEMANTISCHES KOMMUNIKATIONSMODELL 54
5.3 DEFINITION METHODOLOGIEN 60

6 LINGUISTISCHE MODELLE FÜR DOKUMENTATION 61

6.1 LINEARE VS. MODULARE DOKUMENTATIONSERSTELLUNG 61
6.2 AMENT: SINGLE SOURCING 61

6.3 Strukturelle Textkontrolle: Ein linguistisches Informationsmodell für technische Kommunikation (Ley) 62
6.4 DITA (Darwin Information Typing Architecture) 67
6.5 Mumasy: Informationsmodell für den Maschinenbau 68
6.6 Wiegand: funktional-positionale Segmentierung 69
6.7 Schäflein-Armbruster/Muthig: Funktionsdesign® 70
6.8 Information Mapping™ 70
6.9 Textuelle Informationsmodellierung (Lobin) 71

7 METHODOLOGIEN FÜR ONTOLOGIEN 72

7.1 Existierende Ontologie-Methodologien 72
7.2 Parallele Erstellung der Ontologie beim Schreiben der Requirements Spezifikationen 81
7.3 Semantische Annotation von Dokumenten 82

8 EVALUATION DER METHODOLOGIEN 84

8.1 Aufbau der Evaluationsmatrix und des Evaluationsleitfadens 84
8.2 Erfahrungen beim Anwenden der linguistischen Modellierungsmethodologien 87
8.3 Erfahrungen beim Anwenden der Ontologie-Methodologien 94
8.4 Evaluationsmatrix für linguistische Modellierungsmethodologien 99
8.5 Evaluationsmatrix für Ontologie-Methodologien 100
8.6 Zusammenfassung der Evaluation 105

9 RELATED WORK 107

9.1 Requirements Engineering 107
9.2 Semantik-basiertes Requirements Engineering 108
9.3 Use Cases 109

10 ZUSAMMENFASSUNG UND AUSBLICK 110

10.1 Problemreduktion von RE auf eine kommunikative Situation 110
10.2 Kombination linguistische Modellierungsmethodologien / Ontologie-Methodologien als Lösung 110
10.3 Idee eines Framework 111
10.4 Ausblick 111

LITERATUR 112

ANHANG A: EVALUATIONSLEITFADEN 121

ANHANG B: ONTOLOGIE IN OWL ERSTELLT NACH OTKM _____ 125

1 Projektkonzept

1.1 Einführung

Technische Dokumentation beinhaltet auf der einen Seite die *interne Dokumentation* wie Lasten- und Pflichtenhefte und auf der anderen Seite die *externe Dokumentation* wie Bedienungsanleitungen. Die interne Dokumentation richtet sich an die Mitarbeiter des Herstellers, die entlang des Produktlebenszyklus in verschiedenen Abteilungen mit unterschiedlichen Funktionen und Zielstellungen mit der Entstehung und Entwicklung des Produkts zu tun haben. Sowohl die Dokumente, die im Umfeld des *Requirements Engineering* in Form von Anforderungs-Spezifikationen anfallen, als auch jene, die im Umfeld der Software-Entwicklung in Form von technischen Spezifikationen, Analyse- und Designdokumenten, Fachkonzepten etc. entstehen, sind demzufolge zur internen technischen Dokumentation zu zählen. Auf der anderen Seite findet sich die externe Dokumentation, die sich an den Endnutzer richtet, der z.B. mit Betriebsanleitungen, Online-Hilfen, Gebrauchsanweisungen oder Sicherheitshinweisen über die richtige Handhabung des Produkts informiert wird.
Mit der Entwicklung des Internets hielten Buzzwörter wie *Content* und *Content Management* auch in technischen Dokumentationsprozessen ihren Einzug. Im Kontext des Content Managements wird der Terminus *Content* unter Zuhilfenahme der Termini *Daten*, *Wissen* und *Information* benutzt. Dabei bleiben oft die verwendeten Termini undefiniert und es wird versäumt, sie wenigstens hinsichtlich ihres semantischen Gehalts genauer zu beschreiben. Die Termini sind in wissenschaftlichen Beiträgen nicht einheitlich verwendet und so sind Missverständnisse vorprogrammiert, wenn nicht im jeweiligen Diskurs vorab geklärt wird, in welchem Sinn der jeweilige Terminus verwendet wird. Ein Anliegen dieses Projekts ist es deswegen unter anderem, den oben genannten Terminus *Content* genauer zu untersuchen und eine Definition zu erarbeiten. Diese Definition sowie die Überlegungen, die zu der Definition geführt haben, stellen wir in Kapitel 2 vor und beziehen uns dabei auf das semantische Datenmodell von Sieber/Kammerer.
Metadaten sind eine verbreitete Lösung, den Herausforderungen des Wissens-, Dokumenten- und Datenmanagements zu begegnen. Während im Umfeld der externen technischen Dokumentation durch zunehmende Content Management Implementierungen der letzten Jahre Daten, ihre Strukturierung und Standardisierung sowie ihre Beschreibung über Metadaten ins Zentrum des Interesses gerückt sind, sind solche Überlegungen im Bereich der internen technischen Dokumentation nicht geschehen. Metadaten sollten allerdings nicht erst bei der externen Dokumentation oder im laufenden Prozess hinzugefügt werden, sondern bereits im Frühstadium des Produktentwicklungszyklus in die interne Dokumentation einfließen, beispielsweise durch Aufnahme in die bereits sehr früh anzufertigenden Fachkonzepte. Dadurch stünden Metadaten und ermöglichte automatisierte Prozesse während des kompletten Produktentwicklungszyklus zur Verwendung und Erweiterung bereit. Interessant ist dabei, ob und wie diese Metadaten bereits im Rahmen des Requirements Engineering aufgenommen werden können und in welchen Systemen und in welchen Teilen der Entwicklung diese anschließend überall zur Verfügung stehen sollten, damit sie möglichst effektiv genutzt werden können.
Requirements Engineering oder zu deutsch Anforderungsanalyse ist ein Teil des Software- und Systementwicklungsprozesses. Die Anforderungsanalyse hat entscheidenden Einfluss auf den weiteren Prozessverlauf und auf die Qualität und Produktivität des dabei existierenden Systems. Sie sollte aus den Schritten

Anforderungsaufnahme, Anforderungsstrukturierung und Anforderungsbewertung bestehen.
Die Anforderungsanalyse dient dazu, die Anforderungen des Auftraggebers an das zu entwickelnde System zu ermitteln. Diese Anforderungen werden dann in einem oder mehreren Dokumenten (Anforderungskatalog bzw. Lasten-/ Pflichtenheft) schriftlich fixiert. Wie vielfältig die dabei entstehenden Dokumente sein können und wie stark die Abhängigkeiten unter den Dokumenten dabei sind, ist in Abb. 1 skizziert.

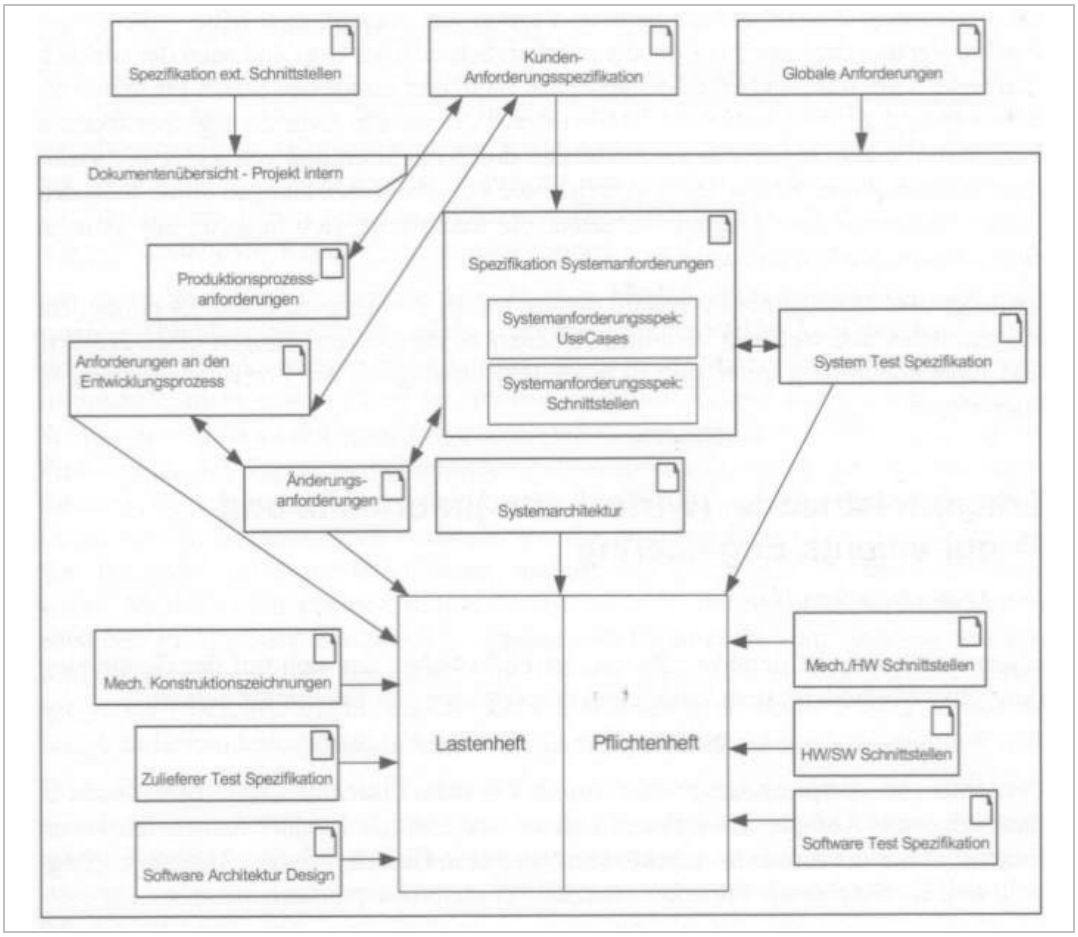


Abb. 1 Dokumentenvielfalt und deren Abhängigkeiten im Requirements Engineering aus [Rupp07]

1.2 Probleme / Herausforderungen

Ein typischer Ablauf (wie in Abb. 2 graphisch dargestellt) läuft bei der Erstellung eines neuen Produkts meist wie folgt: auf Basis der strategischen Ziele, die ein Marketing Manager festgelegt hat, werden diverse Personengruppen, die zukünftig mit dem System/Produkt arbeiten müssen, befragt und diese Anforderungen vom Requirements Engineer gesammelt.

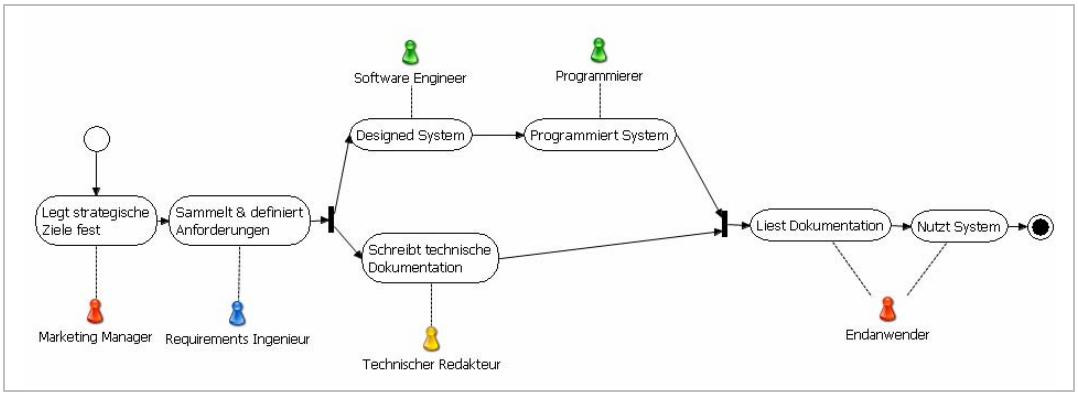


Abb. 2 Grob schematischer Ablauf von den Anforderungen bis zur Endanwendung

Dabei werden ausgewählte Mitarbeiter aus verschiedenen Abteilungen, die mit der zukünftigen Software arbeiten müssen, über Ihre Wünsche und den zukünftigen Prozessablauf interviewed und alle Beschreibungen händisch gesammelt und in einem einzelnen Lastenheft niedergeschrieben. Die befragten Mitarbeiter werden aufgrund bestimmter Eigenschaften zu Anwendergruppen zusammengefasst. Auch die einzelnen Anforderungen der Mitarbeiter sind zukünftig nicht mehr zu erkennen, es werden lediglich deren Gemeinsamkeiten und Unterschiede als Haupt- bzw. Nebenszenarien beschrieben. Von den Mitarbeitern wird derzeit meist unpräzise ein verschiedener Sprachschatz verwendet. Dieser Sprachschatz der unterschiedlichen Mitarbeiter beschreibt Konstrukte und Konzepte des zukünftigen Systems, sowie Abläufe auf verschiedenen Granularitätsstufen (z.B. "das nächste Fenster muss mir erlauben, den Kunden durch Eingabe der ersten drei Buchstaben des Nachnamens zu selektieren, damit die Rechnung geschrieben werden kann" vs. "Rechnungsdateneingabe"). Dies wird vom Requirements Engineer vereinheitlicht, auf dieselbe Granularitätsstufe gebracht und dann ins Lastenheft integriert. Dabei kann es natürlich vorkommen, dass zwei Anforderungen unterschiedlicher Mitarbeiter inkonsistent sind. Dies wird auf hoher Granularitätsstufe meist entdeckt, kann aber gerade bei zahlreichen Anforderungen auch übersehen werden, so dass sich diese Inkonsistenzen im Lastenheft unentdeckt wiederfinden.
Die definierten Anforderungen werden im nächsten Schritt an den Software Engineer weitergegeben, der das System entwirft und seine Ergebnisse dem Programmierer weitergibt, welcher für die Implementierung des Systems verantwortlich ist. Parallel dazu sollten die Anforderungen auch an den technischen Redakteur weitergegeben werden, damit dieser darauf aufbauend die technische Dokumentation erstellen kann. Derzeit findet in der Realität dieser Schritt aber meist nachgelagert statt, so dass ein technischer Redakteur die Spezifikationen des Software Engineers erhält und auf dieser Basis die externe technische Dokumentation erstellt. Von den ursprünglichen Anforderungen erfährt der technische Redakteur meist nichts. Anschließend (nach Tests, Deployment, etc. – diese Schritte wurden hier ausgeblendet) erhält der Endanwender sowohl das Produkt / System als auch die Dokumentation und liest idealerweise zuerst die Dokumentation, bevor er das System/Produkt nutzt oder zieht die Dokumentation bei aktuellen Problemen mit dem System/Produkt zu Rate.

Zusammenfassend lässt sich festhalten, dass sich bei der Verwendung und der Weiterverwendung von content (nicht nur im Bereich der technischen Dokumentation) zahlreiche Herausforderungen entlang des Produktlebenszyklus ergeben:

- Die Entwicklungsprozesse sind in immer komplexere Strukturen eingebunden und das damit verbundene Wissen verteilt sich auf Mitarbeiter in den verschiedensten Abteilungen, die mit unterschiedlichsten Anwendungen arbeiten. Diese Mitarbeiter haben auf das Produkt völlig unterschiedliche Sichtweisen, die jeweilig in die Analyse- und Designphase sowie Entwicklung und Dokumentation entsprechend einfließen.
- In den verschiedenen Abteilungen gibt es unterschiedliche Dokumentationsvarianten. Varianten existieren nicht nur im Zusammenhang mit unterschiedlichen Produkten sondern auch durch verschiedene Zielgruppen (detaillierte Entwicklerdokumentation, 'abgespeckte' Benutzerdokumentation, etc.)
- Die Dokumentation ist oft nicht professionell standardisiert, strukturiert oder so ausgezeichnet, dass sie automatisiert weiterverarbeitet werden können, was damit zusammenhängt, dass sich hierbei auch noch keine der existierenden Modellierungsmethodologien durchsetzen konnte, da die Zusammenhänge von Modellierungsmethodologie und angestrebten gewünschten Ergebnissen nicht hinreichend untersucht und bekannt sind. Ausserdem liegen Teile der Dokumentation in verschiedenen Sprachen vor.
- Ein Dokumentations- und Informationsfluss fehlt – oft verbringen Service-Techniker unnötig Zeit damit, längst bekannte Probleme zu lösen, weil diese nicht entsprechend dokumentiert vorliegen. Die Spezifikationen werden in ausgedruckter Form an die anderen Abteilungen weitergereicht (Medienbruch!) und ein Versionsupdate mit den aktuellen Angaben erreicht die Mitarbeiter der technischen Redaktion zu spät.

Im Rahmen der Software-Entwicklung zeigen sich ähnliche Problemgebiete:

- Probleme und Anwendungsfälle werden meist von Business Managern spezifiziert, die Entwicklung hingegen von Software-Engineers vorgenommen. Beide Seiten haben unterschiedliche Sicht- und Denkweisen, was häufig zu einer Diskrepanz zwischen erwünschtem und erhaltenem Produkt führt.
- Die verwendeten Modellierungssprachen unterscheiden sich stark und lassen sich nur mit größerem Aufwand ineinander übersetzen (bspw. BPMN oder ARIS für Business Manager, auf Seite der Software-Entwickler wird meist UML verwendet)
- Ein Lasten- und Pflichtenheft sammelt die Bedürfnisse des Produktes bzw. Projektes und ist selbst maschinell nicht weiterverwendbar. An dieser Stelle findet meist ein Medienbruch statt, die einmal gewonnenen Erkenntnisse werden wieder von Personen gelesen und die Semantik womöglich neu interpretiert. Ein durchgehender Prozess, der die Semantik erhält, wäre hier wünschenswert.

Diese Situationen machen es sehr schwierig, vorhandenes Wissen voll auszuschöpfen und dort entsprechend aufbereitet zu bündeln, wo es jeweils gebraucht wird.
Der Schlüssel zu einer befriedigenden Lösung liegt in einer ‚intelligenten' Modellierung und Modularisierung, die eine modulbasierte Weiterverarbeitung erlaubt. Diese Modellierung und Modularisierung zieht sich durch sämtliche Bereiche der Datenerfassung: von der ‚Verdatung' der Produktidee bis hin zu einem sauberen ‚verdateten' Rückfluss des Erfahrungswissens des Service-Technikers, so dass eine differenzierte Betrachtung sämtlicher betroffener Bereiche erstrebenswert ist.

1.3 Adressierte Problemfelder

Das ECIDISI-Projekt adressiert oben aufgeführte Probleme und Herausforderungen mit dem Fokus auf die Untersuchung möglicher (semantischer) Anreicherungen von internen Dokumentationen, die sehr früh im Produktlebenszyklus erstellt werden und den dadurch entstehenden Möglichkeiten der automatisierten Weiterverarbeitung von Dokumentationsmodulen innerhalb des Produktlebenszyklus für sich anschließende Dokumentationszwecke und für automatisch generierten Quellcode. Dementsprechend liegen die Schwerpunkte des Projekts auf den Forschungszweigen

- Requirements Engineering
  In welcher Art und Weise und nach welchen Standards werden die hier anfallenden Dokumentationen verfasst und wie können diese so erstellt werden, dass sie in den folgenden Arbeitsschritten wieder verwendet werden können?
- Document/Content Engineering
  Eine angestrebte Untersuchung und der Vergleich von Modellierungsmethodologien, die ein unternehmensweites Content-Management überhaupt unterstützen, soll hier die zentrale Grundlage unserer Forschungsarbeit bieten. Inwieweit es möglich ist, über eine entsprechende Modellierungsmethodologie und semantische Anreicherungen hier dem gewünschten semi-automatisierten Weiterverarbeitungsprozess nachzukommen, ist hier die Leitfrage, der das aktuelle Forschungsprojekt nachgehen möchte.
- Semantic Software Engineering
  Den Schwerpunkt bilden hier Fragestellungen, wie über eine entsprechend fundierte Modellierungsmethodologie eine derartige semantische Anreicherung erfolgen kann, die gewährleistet, dass die verwendete ‚Verdatungsstrategie' des Wissens für nachfolgende Prozesse am effektivsten und nachhaltigsten wirken kann. Des Weiteren muss hier technologisch gewährleistet sein, dass diese semantische Anreicherung der formalen Gestalt ist, dass sie innerhalb des Produktentwicklungszyklus nicht nur maschinen-lesbar, sondern auch maschinen-verständlich ist.

1.4 Projektziele

Das Ziel von ECIDISI ist es, anhand eines aus reellen Beispielen abgeleiteten unternehmerischen Anwendungsfalls die Forschungsbereiche aus zwei unterschiedlichen Sichten zu analysieren und zu vertiefen: zum einen aus der Sicht eines Entwicklungs-Engineers in einer Software-Entwicklungs-Abteilung und zum anderen aus der Sicht eines technischen Redakteurs der Dokumentationsabteilung. Anhand dieser Sichten sollen die konkret auftretenden Probleme herausgearbeitet werden, die im Unternehmen zum Alltag gehören und unterschiedliche vorhandene Modellierungsmethodologien systematisch untersucht und getestet werden. Davon ausgehend kann in einem Folgeprojekt eine Methodologie entwickelt werden, die es erlaubt, die unterschiedlichen Sichten und Ziele abzubilden und die entsprechend modellierte Dokumentation effektiv in den weitergehenden Prozessen auszuschöpfen.
Die dazu notwendigen Teilziele sind:

(1) Terminologische Abgrenzung: Daten, Metadaten, Content

- Aufbauend auf der Grundlagenanalyse von Sieber/Kammerer [SieKam06] wird die Terminologiearbeit für Metadaten und Content fortgeführt, um eine saubere Begriffswelt zu haben, auf der im Forschungsprojekt aufgebaut werden kann.

(2) Analyse und Definition : « Enterprise Content Integration »

- Für die beiden Sichten externer technischer Dokumentationsprozess und Software Engineering Prozess erfolgt eine Einordnung, was im speziellen Fall unter content integration verstanden wird und wo aktuell die Probleme sind. Das Ergebnis wird eine IST/SOLL-Zustandsbeschreibung sein, die klar herausstellt, wo aktuell Defizite vorhanden sind.

(3) Systematische Analyse und Evaluierung ausgesuchter vorhandener Modellierungsmethodologien

- Einarbeitung und Erarbeitung einer Übersicht vorhandener ausgesuchter Modellierungsansätze aus dem Software Engineering Bereich und ausgewählten Bereichen der technischen Dokumentation. Hierbei werden gängige in den unterschiedlichen Bereichen verwendete Modellierungsmethodologien sowie Methodologien untersucht werden.
- Entwickeln eines Referenzmodells, anhand dessen eine Evaluationsmatrix aufgestellt wird, um ausgesuchte Methodologien der linguistischen Modellierung und Ontologie-Designs testen und beurteilen zu können.
- Leitfragen hierzu: Welche ModellierungsMethodologien können auf interne technische Dokumentationen übertragen werden? Was wird dabei genau modelliert und welcher Grad der relevanten gewünschten semantischen Anreicherung wird dadurch abgedeckt, was fehlt? Wie effektiv lässt sich der angestrebte nachfolgende Automatisierungsprozess bzgl. möglicher Wiederverwendung und Personalisierung auf Grundlage welcher Methodologie gestalten?

1.5 Aufbau dieses Berichts

Der weitere Bericht gliedert sich wie folgt: in Kapitel 2 beschreiben wir auf der Basis des Datenmodells von Sieber/Kammerer die Herausforderungen und Probleme, die mit Daten und Metadaten verbunden sind sowie erarbeiten eine Definition des Begriffs Content. In Kapitel 3 werden die Probleme und Herausforderungen von Content speziell im Requirements Engineering, Software Engineering und Documentation Engineering beschrieben. Einen kurzen Überblick über ausgewählte Use Cases (die aufgrund von Vereinbarungen mit den Praxispartnern nicht veröffentlicht wurden) gibt Kapitel 4. Im folgenden Kapitel erfolgt die Definition eines semantischen Kommunikationsmodells, welches auf existierenden Kommunikationsmodellen bzw. den Grundlagen der Kommunikationstheorie aufsetzt, um dieses als Referenzmodell in den nachfolgenden Kapiteln zu nutzen. Im Kapitel 6 und 7 werden verschiedene linguistische Modellierungsmethodologien und Methodologien für die Erstellung von Ontologien vorgestellt. Diese werden im Kapitel 8 unter Nutzung des Referenzmodells evaluiert und Erfahrungen bei Anwendung dieser Methodologien beschrieben. Kapitel 9 gibt einen Überblick über ähnliche Arbeiten, bevor in Kapitel 10 die Herausforderungen nochmals zusammengefasst werden, schematisch skizziert wird, wie die Kombination von linguistischen Modellierungsmethodologien und Ontologien aussehen könnte, die Idee eines Frameworks erläutert wird und mit einem Ausblick auf mögliche zukünftige Arbeiten der Projektbericht endet.

2 Daten, Metadaten, Content

Auf Grundlage der Reflexion kritisch geprüfter Definitionen aus der Literatur, entwickeln wir unsere Definition des Terminus *Content*. Wir nehmen als Grundlage hierfür das semantische Datenmodell nach Sieber/Kammerer an, das ausführlich in [SieKam07] hergeleitet wird.

2.1 Semantisches Datenmodell nach Sieber/Kammerer

Das semantische Datenmodell sieht sich als Brückenschlag zwischen Semiotik[1] und der Informatik und vereint dadurch Sichtweisen und Erkenntnisse, die aus diesen unterschiedlichen wissenschaftlichen Gebieten zum Wesen und Gebrauch des Datenbegriffs beitragen. Es bildet damit eine Grundlage für die wissenschaftliche Auseinandersetzung auf der Ebene semantischer Technologien.

Semiotische Einheiten nach Peirce

Das Datenmodell setzt dabei auf dem Zeichenbegriff im Sinne von Peirce auf, nach dem Zeichen auf drei Ebenen vorkommen können. Demnach unterscheidet Peirce zwischen folgenden semiotischen Einheiten:

- Indizes – Zeichenobjekte haben einen kausalen Zusammenhang zu ihrer Bedeutung. In Abb. 3 besteht eben dieser kausale Zusammenhang zwischen dem, was ein wahrnehmendes Subjekt wahrnimmt und der anzunehmenden vorausgegangenen Begebenheit, dass hier eine Person entlanggelaufen ist, deren Füße die Abdrücke in den Sand gemacht haben.

Abb. 3 Index-Beispiel im Sinne von Peirce

- Ikonen – Zeichenobjekte verfügen über eine natürliche Ähnlichkeit zu ihrer Bedeutung, z.B. ein Bild einer Person (siehe Abb. 4)

Abb. 4 Ikon-Beispiel im Sinne von Peirce

- Symbole – Zeichenobjekte sind willkürlich mit ihrer Bedeutung verknüpft. Diese Verknüpfung ist durch Konvention hergestellt. Hierzu zählen z.B. sprachliche Vereinbarungen oder Straßenverkehrsschilder (siehe Abb. 5).

Abb. 5 Symbol-Beispiel im Sinne von Peirce

[1] Semiotik ist die Lehre von den Zeichen, Zeichensystemen und Zeichenprozessen

Definitionen des semantischen Datenmodells

Auf Grundlage dieses Verständnisses für Zeichen im Sinne von Peirce, beinhaltet das semantische Datenmodell nach Sieber/Kammerer folgende grundlegenden Definitionen:

Def. 1 *Dateninstanz* (engl.: *data instance*)

Eine Dateninstanz d_i ist das konkrete Vorkommen einer semiotischen Einheit. Gleichzeitig gilt: Jede semiotische Einheit ist eine Dateninstanz.

Ann. 1-1: Somit kann im Sinne von Peirce eine Dateninstanz auf allen drei Ebenen vorkommen: als Ikon, Index oder Symbol.

Def. 2 *Datenrepräsentationssystem* (engl.: *data representation system*)

Ein Datenrepräsentationssystem d_{rs} ist eine Menge von Prädikaten p.

Ann. 2-1: Unter der Annahme, dass eine vernetzte Struktur von Konzepten und deren Relationen zugrundegelegt werden kann, ist ein Prädikat p als eine logische Funktion einer Dateninstanz d_i und eines Konzepts zu verstehen, für die nach Sieber/Kovács gilt:

> "A *predicate* $p(d_i,c)$ is a logical function of a data instance d_i and a concept c with:
>
> $P: D_I \times C \rightarrow \{TRUE, FALSE\}$
>
> $p \in P$
>
> $p(d_i,c)=\{TRUE, if \exists (f \in M_{cm}, f(d_i)=c) OR (f \in M_{ci}, f(c)=d_i)\}$ " (SieKov07)

Def. 3 *Datenrepräsentant (*engl.: *data representative)*

Ein Datenrepräsentant d_r ist die Abstraktionsmenge über alle Dateninstanzen eines bestimmten beliebig gewählten Datenrepräsentationssystems d_{rs}.

Def. 4 *Datum* (engl.: *data item*)

Ein Datum d ist die Abstraktionsmenge über alle Datenrepräsentanten einer bestimmten beliebig gewählten Bedeutung.

Ann. 4-1: Def. 4 impliziert, dass Daten nach diesem Verständnis unbedingt eine Bedeutung zugesprochen werden muss, ansonsten wäre keine Mengenbildung möglich.

Ann. 4-2: Eine Dateninstanz ist das konkrete Vorkommen eines Datums und – a fortiori – eines Datenrepräsentanten in einem bestimmten Datenrepräsentationssystem.

Das semantische Datenmodell unterscheidet abschließend zwischen sog. atomaren und komplexen Daten mit den folgenden Definitionen:

Def. 5 *atomares Datum* (Pl.: *atomare Daten*; eng. Sg.: *atomic data item*, eng. Pl.: *atomic data items*)

Ein atomares Datum kann nicht in weitere Daten zerlegt werden.

Anmerkung 5-1: Da Daten immer eine Bedeutung haben, folgt aus Def. 5, dass ein atomares Datum nicht in weitere bedeutungstragende Bestandteile zerlegt werden kann.

Def. 6 *komplexes Datum* (Pl.: *komplexe Daten*; engl. Sg.: *complex data item*, engl. Pl.: *complexc data item*)

Ein komplexes Datum ist ein Datum, das in weitere Daten zerlegt werden kann.

Visualisiertes Semantisches Datenmodell

Veranschaulicht werden die unterschiedlichen Abstraktionsstufen in folgender Abbildung (siehe Abb. 6):

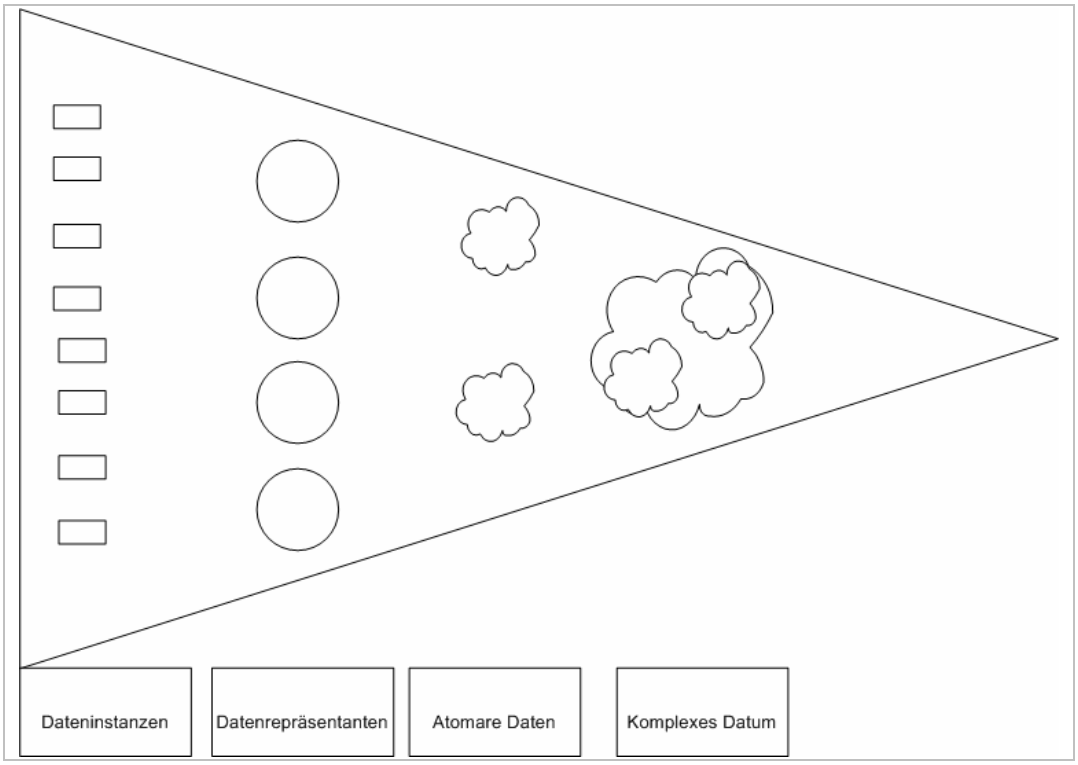


Abb. 6 Semantisches Datenmodell nach Sieber/Kammerer

Die einzige Ebene, auf der Daten extra-personal wahrgenommen werden können, ist nach dem semantischen Datenmodell die Ebene der Dateninstanzen.

Signal-verarbeitung über Agenten

Nach [SieKov07] ist ein Agent als eine Einheit in der Welt zu betrachten, die über Sensoren verfügt, die Dateninstanzen wahrnehmen können und über Aktuatoren, die für die Außenwelt wahrnehmbare Dateninstanzen produzieren können (siehe Abb. 7).

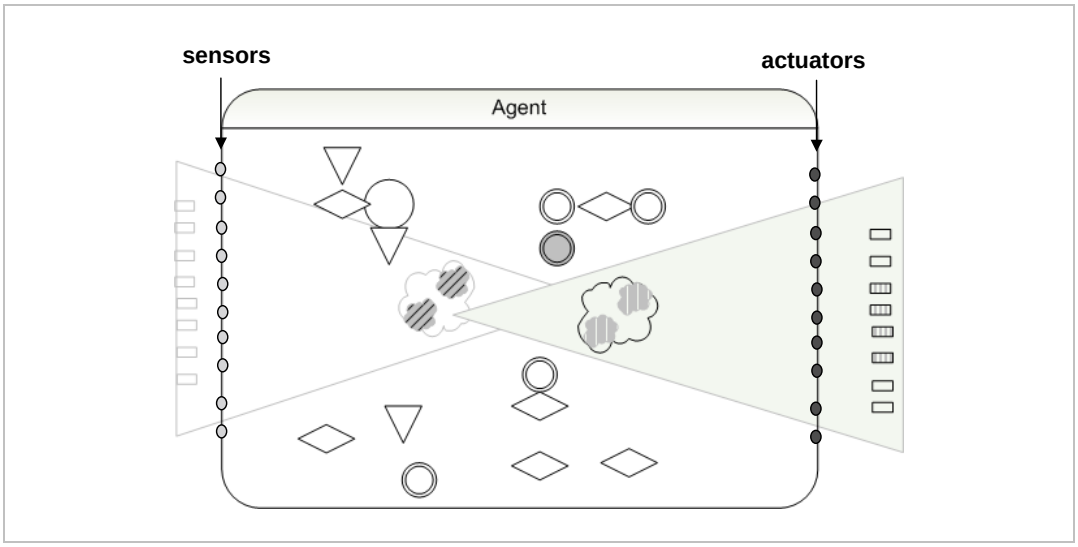


Abb. 7 Agent im Signal-Verarbeitungs-Modell nach Sieber/Kovács

Auf dieser Grundlage kann man sich eine klassische Kommunikationssituation zwischen zwei Agenten als Sender-Empfänger-Modell folgendermaßen vorstellen:

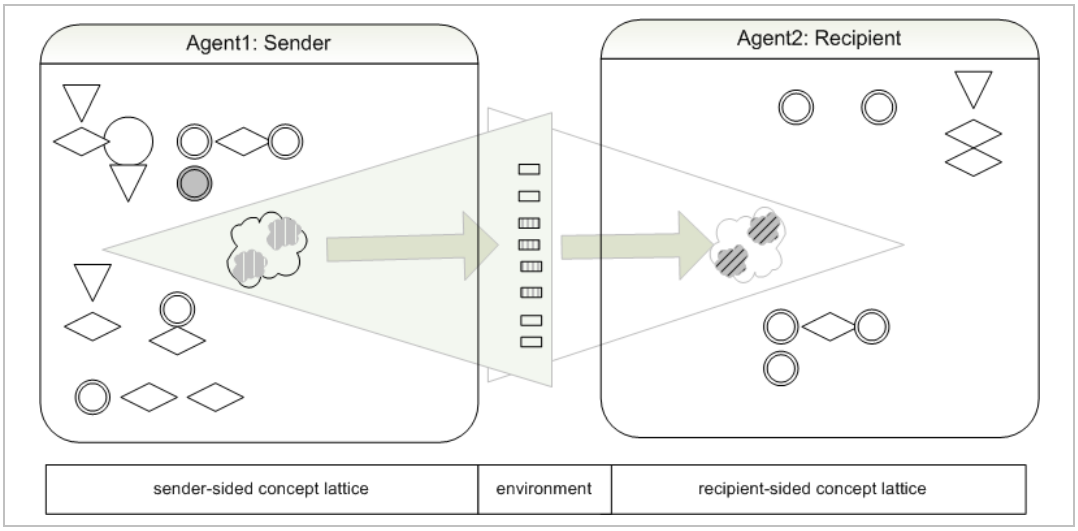


Abb. 8 Kommunikation zwischen zwei Agenten im Signal-Verarbeitungs-Modell nach Sieber/Kovács

Nach [SieKam07] treten dabei Probleme „zum einen durch die Tatsache auf, dass senderseitig eine Art vernetzte Wissensstruktur vorhanden ist, die sich wesentlich von der auf der Empfängerseite unterscheiden kann und wird. Ein Sender kann dementsprechend nur durch Annahmen und Vermutungen an das empfängerseitig vorhandene Wissensnetz anknüpfen und auf dieser Grundlage festlegen, welcher Wissensbaustein wahrscheinlich angebracht ist. Zum anderen ist empfängerseitig nicht garantiert, dass augrund der gewählten Verdatung und der daraus resultierenden Dateninstanzen auch tatsächlich das Datum abstrahiert werden kann, welches senderseitig konkretisiert wurde.“

2.2 Aporie des Datums – Metadatum

Unter Aporie versteht man die Einsicht in das eigene Nichtwissen und die Unlösbarkeit eines Problems.
Diese Aporie des Datums führt zu der Erkenntnis, dass Daten nicht wirklich greifbar sind, sondern nur die Dateninstanzen. Grund für die Aporie ist, dass Daten, wie wir sie verstehen, letztlich auf den Menschen ausgerichtet sind und deswegen der Mensch als Benutzer oder Erzeuger von Dateninstanzen selbst Teil der Betrachtungen sein kann. Dadurch enthalten die verwendeten Begriffe eine auf menschliches Bewusstsein ausgerichtete Komponente der Semantik. „Dies hat zur Folge, dass Daten – obwohl nur *intra*personal vorgestellt – dennoch als *extra*personal gedacht werden müssen [...], was hier die *Aporie des Datums* genannt werden soll.“ [SieKam07]

Herausforderung an technische Dokumente

Da die alltägliche Herausforderung entlang eines Produktlebenszyklus im Bereich der technischen Dokumentation darin liegt, Mitarbeitern in anderen Abteilungen (im Fall interner technischer Dokumente) oder zukünftigen Produktkunden (im Fall externer technischer Dokumente) fachliches Wissen zu vermitteln, muss die Motivation aus der Sicht des “sender-seitigen Agenten“ sein, eine möglichst eindeutige Abstraktion aus den publizierten Dateninstanzen für den zukünftigen “Empfänger-Agenten“ zu ermöglichen, so dass die ursprünglich zugrundeliegende Semantik des konkretisierten Datums erhalten bleibt.
Der Sprechakttheorie von Austin [Austin62] folgend kann das Äußern eines Satzes in folgende Akte aufgeteilt werden:

- lokutionärer Akt, der sich wiederum aufteilt in
  - - einen phonetischen Akt (Äußern gewisser Laute und Lautkombinationen, die zum Lautsystem einer bestimmten Sprache gehören),
  - - einen phatischen Akt (Bestandteile des Satzes werden gemäß gewisser grammatischer Prinzipien einer bestimmten Sprache und unter Verwendung der Wörter dieser Sprache konstruiert) und
  - - einen rhetischen Akt (determiniert, worüber gesprochen wird (Referenz) und was darüber ausgesagt wird (Prädikation)),
- illokutionärer Akt (die Intention des Sprechers) und
- perlokutionärer Akt (Effekt, der beim Hörer erzielt wird).

Überträgt man diese Sprechakttheorie auf den Anwendungsfall technische Dokumentation, kann man von einer Schreibakttheorie sprechen, die sich aufteilt in die bei Austin erwähnten Teilakte, wobei der phonetische Akt ersetzt wird durch einen scribalen Akt und das Verwenden gewisser Zeichen und Zeichenkombinationen bezeichnet, die zum Zeichensystem einer bestimmten Sprache gehören.
In dem Zusammenhang ist semantisch als das zu verstehen, was bei Austin in der Sprechakttheorie und in der oben eingeführten Schreibakttheorie als rhetischer Akt bezeichnet wird.
Nach [SieKam07] geht es also „letztlich beim Umgang mit Daten darum, die Bedeutung des Datums, das verwendete Datenrepräsentationssystem und gegebenenfalls die syntaktischen Kompositionsregeln – wenn es sich um komplexe Dateninstanzen handelt – explizit anzugeben, um Missverständnissen vorzubeugen. Da diese Angaben ihrerseits (evtl. komplexe) Dateninstanzen sind, besteht hier prinzipiell wieder dasselbe Problem, wie bei denjenigen Dateninstanzen, die sie beschreiben sollen: sie können ebenfalls missverstanden werden. Um dem vorzubeugen, können diese prinzipiell selbst wieder annotiert werden, usw.“ Was letztendlich dabei passiert, ist, dass Daten auf eine

Objektebene gestellt werden und sogenannte Metadaten auf einer Metaebene dazu benutzt werden, die darunter liegenden Daten näher zu beschreiben.

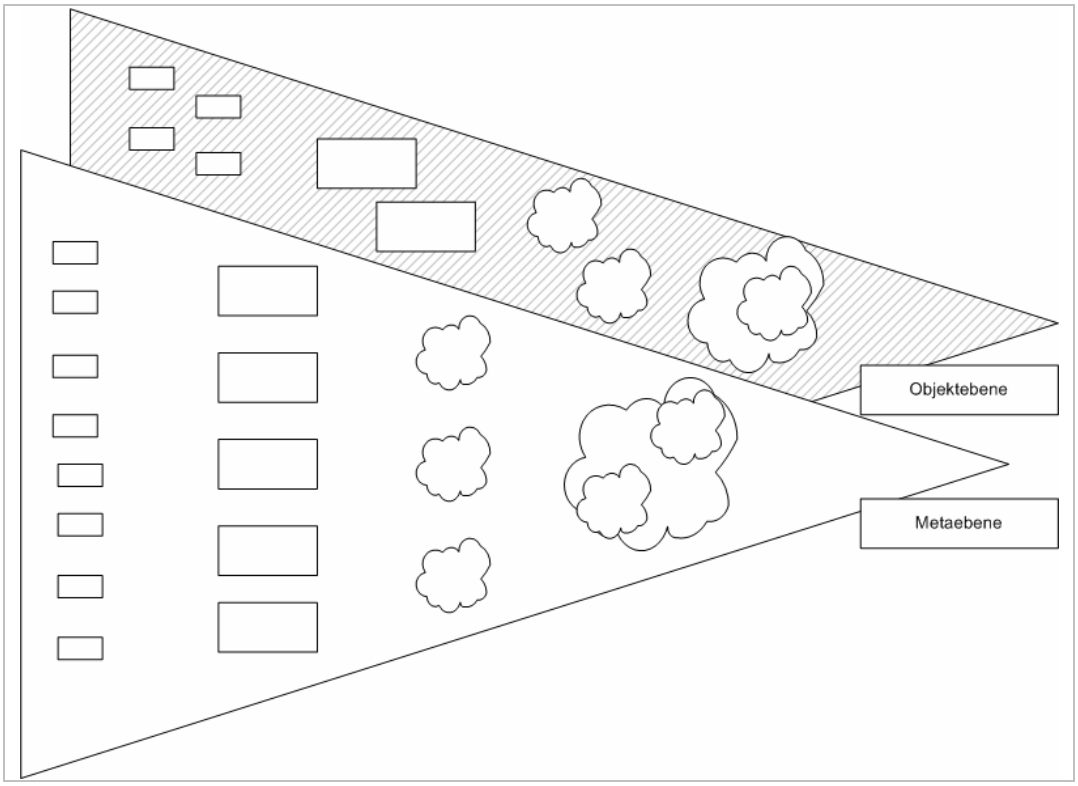


Abb. 9 Metadaten – „Daten über Daten“ im Semantischen Datenmodell nach Sieber/Kammerer

2.3 Charakteristika von Metadaten

Wichtig im Zusammenhang von Metadaten sind folgende Feststellungen (vgl. [SieKam06], [SieKam07] und [SieKov06]):

Metadaten sind Daten

Metadaten sind Daten und kommen dementsprechend nach dem definierten semantischen Datenmodell auf drei Ebenen vor:

- Auf der Formebene konkret als Dateninstanzen,
- auf der Repräsentationsebene als abstrahierte Datenrepräsentanten und
- auf der Bedeutungsebene als abstrahierte Daten.

Metadaten sind relativ

Metadaten sind Daten über/zu anderen Daten. Metadaten gehören damit der Metaebene an, und die Daten, über die etwas ausgesagt wird, gehören der Objektebene an. Diese Metaebene kann aber ihrerseits auch als Objektebene betrachtet werden, und dadurch können Metadaten zu Metadaten entstehen.

Metadaten beschreiben Daten

Hinsichtlich der Funktion von Metadaten kann man die folgenden Arten unterscheiden:

- syntaktische Metadaten: Metadaten zur Struktur der Daten
- semantische Metadaten: Metadaten zur Bedeutung der Daten
- pragmatische Metadaten: Metadaten zur Verwendung und Prozessierung der Daten.

Beim Einsatz von Metadaten treten dementsprechend dieselben Herausforderungen auf wie bei Daten: es liegen ‚außen' wiederum konkrete Dateninstanzen vor, die es gilt, in geeigneter Art und Weise so zu gestalten, dass die Bedeutung des zugrundeliegenden Metadatums und damit die Eigenschaft des Datums, das dadurch beschrieben wird, abgeleitet werden kann. Gegenstand dieses Berichts ist es, unterschiedliche Repräsentationsformen von Metadaten und Methodologien der (Meta-) Datenmodellierung zu analysieren.

2.4 Kritische Reflexion ausgewählter Definitionen des Terminus *Content*

Im Folgenden werden einige ausgewählte Definitionen des Terminus *Content* (engl.: *content*) vorgestellt und kritisch beleuchtet. Grundlage für unsere Betrachtungen ist dabei der Datenbegriff, wie er in 2.1 im semantischen Datenmodell nach Sieber/Kammerer definiert ist.

> Zitat Z1: [MaaSta03, S.41]
> *„Traditionell wird Content Management häufig als prozessuales Framework für die systematische und strukturierte Erzeugung (Generierung), Verwaltung (Organisation und Aufbereitung), Zur-Verfügung-Stellung (Präsentation, Publikation und Distribution) und Schaffung von Nutzungs- und Verarbeitungsmöglichkeiten (Nutzung und Wiederverwendung) von elektronischem Inhalt unabhängig ob im Internet, Intranet oder unternehmensweiten Systemen definiert."*

Nach Z1 darf man Content als elektronischen „Inhalt" verstehen, der zudem durch den Fokus auf web-basierte Content Management Systeme auf Internet, Intranet und unternehmensweite Systeme eingeschränkt ist. Demzufolge könnte nichts als Content bezeichnet werden, was nicht über einen Anschluß an ein derartiges System verfügbar gemacht ist. Es stellt sich die Frage, was dann dieser elektronische Inhalt zuvor ist? Elektronische Inhalte, die beispielsweise in einem WORD-Dokument gehalten sind, aber weder im Internet, noch in einem Intranet noch in einem unternehmensweiten System verfügbar sind, können im Sinne von Z1 nicht als Content bezeichnet werden. Hier findet durch die Auflistung der geforderten Medienkanäle eine Einschränkung des Terminus Content statt, die einer sinngebenden Verwendung des Begriffs sicherlich entgegenstehen.

> Zitat Z2: [tekom05, S.23]
> *„Der Begriff "Content Management" ist in vielen Fällen nicht eindeutig fassbar. Als Content bezeichnet man im Prinzip alles, was an inhaltlicher Information in elektronischen Systemen in strukturierter und unstrukturierter Form vorgehalten wird:*
>
> *Strukturierter Content sind Daten, die in einem standardisierten Layout z.B. aus datenbankgestützten Systemen bereitgestellt werden (z.B. formatierte Datensätze aus einer Datenbank, XML-Daten, strukturierte DTP-Formate).*
>
> *Schwach strukturierter Content sind Informationen und Dokumente, die zum Teil Layout und Meta-Daten mit sich tragen, jedoch nicht standardisiert sind (teilweise Textverarbeitungsdateien).*
>
> *Unstrukturierter Content besteht aus beliebigen Informationsobjekten, deren Inhalt nicht direkt erschlossen werden kann und bei denen Inhalt, Layout und Metadaten nicht getrennt sind."*

An Z2 ist äußerst interessant, dass die Termini *inhaltliche Informationen*, *Informationsobjekte*, *Informationen*, *Dokumente* und *Daten* offensichtlich gleichgesetzt werden. Die Verfasser der tekom-Content Management Studie räumen selbst ein, dass der Terminus *Content Management* nicht eindeutig fassbar ist. Für den Terminus *Content* wird zwar der Versuch einer Begriffsdefinition unternommen, im selben Moment aber auch schon eingeschränkt durch das vage Andeuten „im Prinzip alles, was ...". Dieser Versuch der Terminusdefinition hinterlässt zahlreiche Fragen, die sich durch die Gleichstellung der verwendeten Termini *Daten*, *Informationen*, *Informationsobjekte* und *Dokumente* ergeben, da hier nicht eindeutig geklärt wird, was wiederum unter den einzelnen Termini verstanden wird. Dem semantischen Datenmodell und dem gemeinen Verständnis dieser Termini folgend, kann man dem in Z2 skizzierten Verständnis von Content nicht zustimmen.

> Zitat Z3: [Bauman03, S.80]
> *„Content setzt sich aus der Substanz, dem Inhalt im engeren Sinne, sowie einem assoziierten Metadatensatz zusammen. Metadaten sind Informationen über den Inhalt wie beispielsweise Name des Erstellers, Titel, Erstellungsdatum, eine Beschreibung des Inhalts mit Hilfe von Schlagwörtern usf. Unter Asset wird hier die Kombination aus Inhalt, also Content, und den dazugehörigen Publikationsrechten, unabhängig vom Medium, verstanden. Erst wenn ausser dem eigentlichen Inhalt auch die Nutzungsrechte vorliegen, wird aus dem Content ein wertbehaftetes Asset."*

Nach Z3 ist Content als Zusammensetzung einer Substanz und einem damit assoziierten Metadatensatz zu verstehen. Im zweiten Satz heisst es als Erläuterung zum Terminus *Metadaten*, dass es sich dabei um Informationen handelt. Dem muss man dem Verständnis von Daten nach dem semantischen Datenmodell von Sieber/Kammerer eindeutig widersprechen: Metadaten sind Daten und keine Informationen. Bei den Ausführungen zum Terminus *Asset* wird dann Content gleichgesetzt zum Inhalt und die eingangs erwähnten Metadaten fallen weg. Nach Z3 kann dementsprechend keine eindeutige Aussage und Abgrenzung des Terminus *Content* gezogen werden.

> Zitat Z4: [Gersdo03, S.64]
> *„Ins Deutsche übersetzt hat der englische Begriff Content in etwa die Bedeutung von Gehalt, Inhalt oder Aussage. In diesem Beitrag wird unter Content jeglicher in digitalisierter Form vorliegender, modularisierter Inhalt verstanden, der einem Rezipienten aufgund seiner Immaterialität auf einem beliebigen Informationsträger üblicherweise als Teil eines komplexen Dokumentes präsentiert werden kann. Content repräsentiert in der Regel schwach bzw. unstrukturierte Informationen und kann somit von den stark strukturierten Daten abgegrenzt werden. Allerdings können Daten und Content bei der Publikation [...] integriert werden "*

Gersdorf definiert Content als Inhalt, der digitalisiert vorliegt. Im Gegensatz zu Z1, wo der Terminus Content an eingeschränkte Medienkanäle gekoppelt waren, kann man mit dieser Aussage konform gehen und hier ist sicherlich etwas, was bezeichnend für eine Annäherung an den Terminus Content sein wird. Was bei Gersdorf allerdings zu einer Einschränkung führt und nicht eindeutig abgehandelt wird, ist die Einschränkung, dass es sich um modularisierten Inhalt handelt. Dies wirft vor allem im weiteren Verlauf der zitierten Aussage Fragen auf, da dann wiederum von schwach bzw. unstrukturierten Informationen die Rede ist und Content daraufhin von stark strukturierten Daten abgegrenzt wird.

Zitat Z5: [Anding04, S.23]
„Content ist eine durch (maßgeblich durch menschliche Intelligenz getriebene) redaktionelle MIttel angereicherte, individuell schützbare und zweckorientierte Abbildung impliziter Informationen.“

Nach Anding ist Content als eine Abbildung impliziter Information zu verstehen, wobei implizite Information im semiotischen Sinn so verstanden wird:

Zitat Z6: [Anding04, S.22]
„beschreibt [...] eine zugrunde liegende Semantik bzw. Bedeutung, welche unabhängig von jeglicher Abbildung ist und in verschiedener Form abgebildet werden kann.“

Dabei werden Daten als auf die syntaktische Ebene reduzierte Informationen angesehen, was im Widerspruch zum semantischen Datenmodell von Sieber/Kammerer steht, bei dem Daten nicht nur eine syntaktische, sondern auch eine semantische Ebene zugeordnet wird.
Anding legt den Fokus desweiteren in seiner terminologischen Analyse auf die Herleitung einer Terminusdefinition, die sich mit dem Ziel seiner Forschungsarbeit vereinbaren lässt, in der es um die Entwicklung von Geschäftsmodellen geht, die im Bereich Online Content Syndication erfolgversprechend eingesetzt werden können. Dieser Fokus ist ein anderer als der in dieser Forschungsarbeit wichtige, da hierbei Content, dessen Integration und potentielle automatisierte Weiterverwendung das Untersuchungsgebiet darstellen. Die von Anding in Z5 erfolgten Einschränkungen der redaktionellen Anreicherung und individuellen Schützbarkeit sind im Kontext unserer Forschungsarbeit nicht relevant.
Die zweckorientierte Abbildung als Kriterium dafür zu nehmen, ob etwas als Content bezeichnet werden kann, erscheint auf den ersten Blick fraglich. Woran erkennt ein wahrnehmendes Subjekt, ob einer Abbildung impliziter Information oder in Worten des semantischen Datenmodells ausgedrückt, ob bei der Konkretisierung eines Datums eine Zweckorientierung zugrunde lag? Welche Qualität muss eine solche Zweckorientierung haben? Versteht man aber unter Zweckorientierung im Bereich der technischen Dokumentation den Entscheidungsvorgang, welcher intra-personal abläuft bei der Auswahl der Daten, die einem potentiellen Empfänger in Form von Dateninstanzen zur Verfügung gestellt werden müssen, geht es beim Umgang mit Content sicherlich um ein zweckorientiertes Konkretisieren von Daten und zweckorientiertes Weiterverwenden bereits vorhandenen Contents. Wenn wir ein Subjekt als Produzent von Dateninstanzen betrachten und uns dabei nochmals Austin’s Sprechakttheorie vor Augen führen ist es sogar so, dass diese Zweckorientierung automatisch stattfindet, was aber wiederum heisst, dass es keinen Sinn macht, dieses Kriterium in irgendeiner Art und Weise als Alleinstellungsmerkmal von Content zu definieren.

2.5 Terminusdefinition für Forschungsprojekt

Def. 7 *Content*

Content ist als ein Sammelbegriff zu verstehen, der digitalisiert vorliegende Dateninstanzen eines bestimmten beliebigen Datums (i.S.v. Sieber/Kammerer) auf einer abstrakten Ebene inhaltlich zusammenfasst.

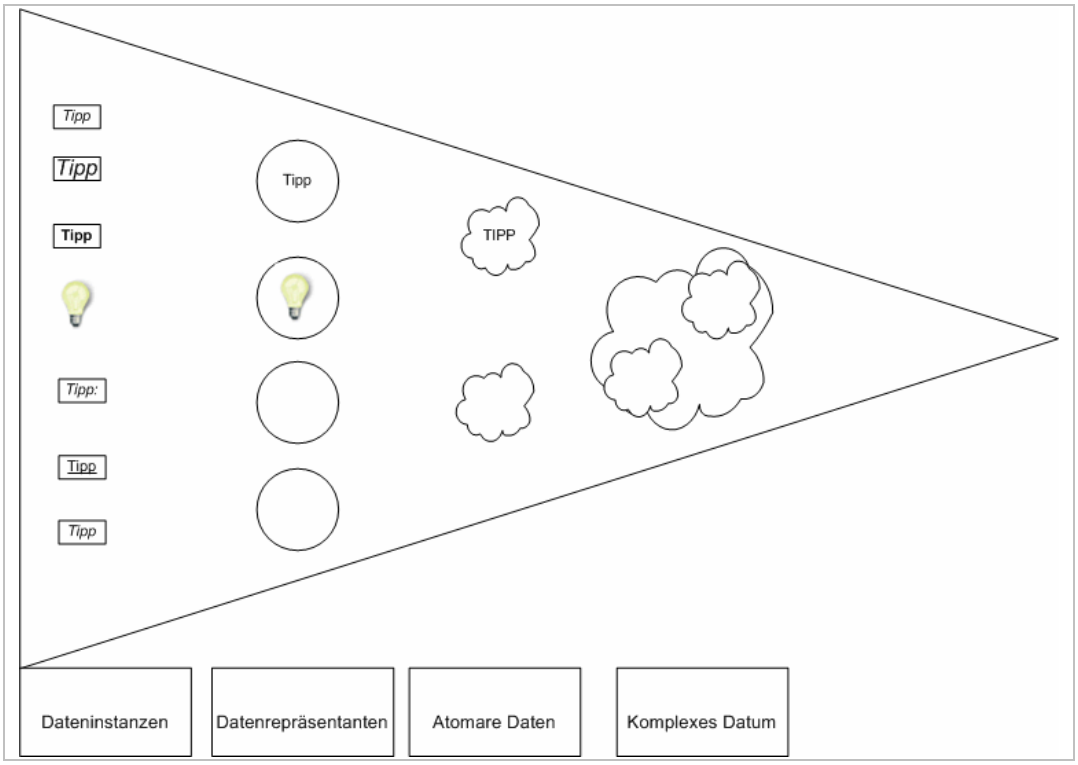


Abb. 10 Beispiel für unterschiedliche vorliegende Dateninstanzen des Datums ‚TIPP'

In Abb. 10 ist beispielsweise ein Fall skizziert, wie er in einer Softwaredokumentation auftreten kann: das Datum ‚TIPP' kann über eines oder verschiedene Datenrepräsentationssysteme beliebig häufig konkretisiert werden. Es entstehen auf diese Art und Weise verschiedene Dateninstanzen, die extra-personal vorliegen. Mit dem Begriff ‚Content' wird auf einer davon abstrahierten, intra-personalen Ebene die Vorstellung verknüpft, dass sich diese unterschiedlichen Dateninstanzen inhaltlich auf eine Quelle zurückverfolgen lassen. Content ist nach unserem Verständnis dementsprechend etwas, was intra-personal existiert und Dateninstanzen (die extra-personal vorliegen) ideell zusammenführt und von einer äußeren Darstellung abstrahiert. Damit verknüpft ist die Erwartung, dass bei Kenntnis dessen, was der Content ist, eine beliebige Dateninstanz in jede andere potentielle Dateninstanz umgewandelt werden kann, ohne ‚inhaltlich' Verluste zu erzeugen. Content beruht im Gegensatz zum Datum nicht auf einer semantischen, sondern auf einer rein inhaltlichen Abstraktion über Dateninstanzen.

In der folgenden Abb. 11 können die Dateninstanzen *1*, ***1*** und **1** inhaltlich zu einem Content zusammengefasst werden. Abstrahiert man über die Bedeutung kann man sämtliche vorliegende Dateninstanzen (*1*,***1***, 51, etc.) zum Datum ‚ALTER' abstrahieren.

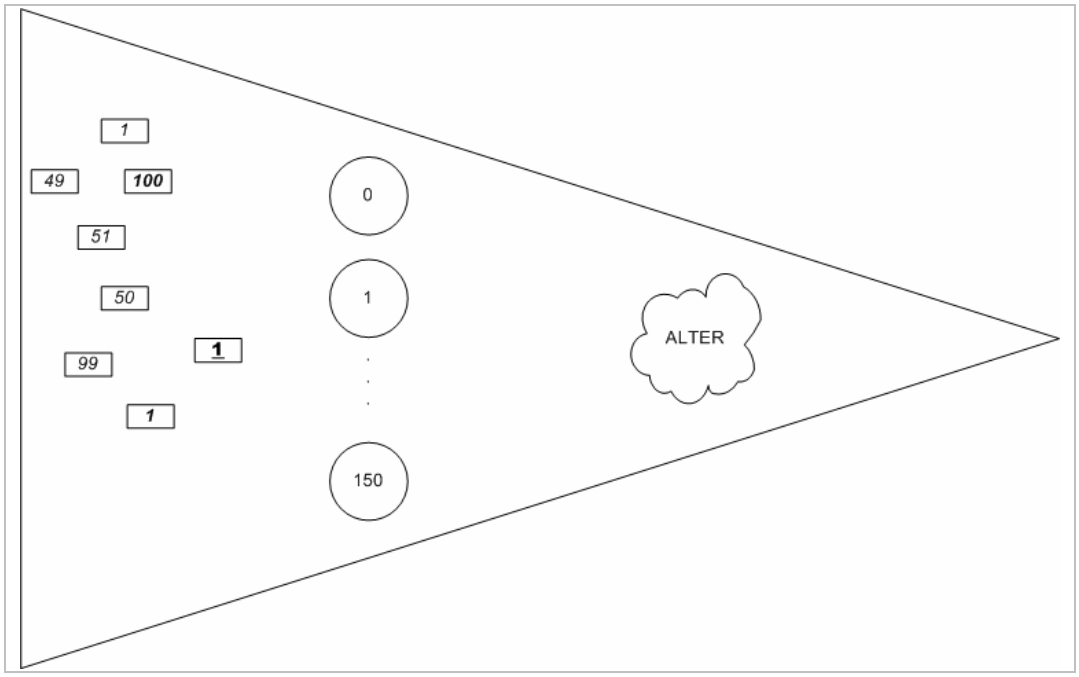


Abb. 11 Beispiel für unterschiedliche vorliegende Dateninstanzen des Datums ‚ALTER'

Für den Content ist es jedoch relevant, dass nur Dateninstanzen inhaltlich zusammengefasst werden können, die zu einem bestimmten beliebigen Datum gehören (vgl. Definition). In Abb. 11 und Abb. 12 sind zwei unterschiedliche Daten mit ähnlichen Dateninstanzen konkretisiert. Es entspricht nicht unserem Verständnis von Content, wenn nun über alle Dateninstanzen von den Daten ‚ALTER' und ‚HAUSNUMMER' hinweg inhaltlich eine Contentzugehörigkeit getroffen werden würde.

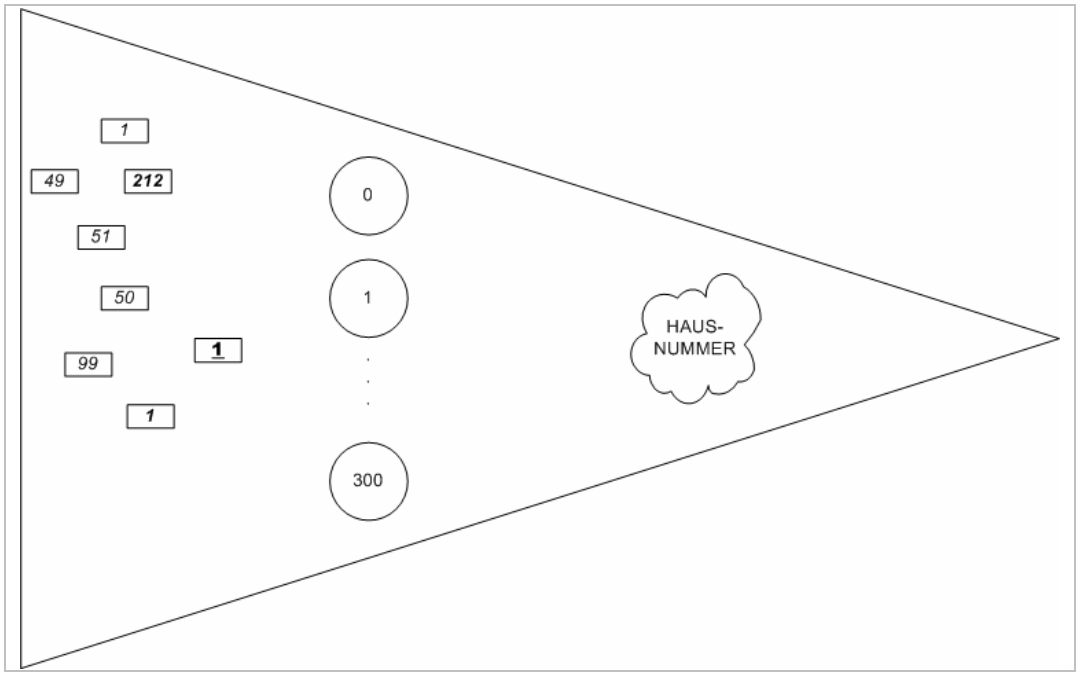


Abb. 12 Beispiel für unterschiedliche vorliegende Dateninstanzen des Datums ‚HAUSNUMMER'

3 Content Integration: Probleme und Herausforderungen

3.1 Product Development Cycle

Produkt-lebenszyklus/ Produkt-entwicklungs-zyklus

Im betriebswirtschaftlichen Kontext wird unter Produktlebenszyklus die Zeitdauer zwischen der Markteinführung eines Produkts und seiner Entfernung aus dem Markt verstanden: Ein Produkt "lebt", solange es einen wirtschaftlichen Umsatz auf dem Markt erzielt.
Für die laufende Forschungsarbeit interessanter ist jedoch der Bereich der Produktentwicklung, unter der wir den Vorgang bezeichnen wollen, den eine Ware von der Idee bis zum verkaufsfähigen Erzeugnis nimmt. Die Phase der Produktentwicklung wird in unterschiedlichen Unternehmen mit verschiedenen Begriffen benannt, so heisst bei SAP die Phase *Product Innovation Lifecycle* (PIL), bei Bosch ist die Rede vom *Product Development Process* (PDP).
Die Phase umfasst – gemäß der Komplexität der Realisierung eines serienreifes Produkts aus einer umzusetzenden Idee heraus – diverse Kernprozesse, deren Prozess-Verantwortliche und -Treiber im gesamten Unternehmen (und bei Outsourcing Prozessen auch ausserhalb des Unternehmens) verteilt vorzufinden sind.
Zu diesen Prozessen gehören beispielsweise Marktforschungsprozesse, Planungsprozesse (strategische, technologische, etc.), Entwicklungs- und Testprozesse.
Diese Kernprozesse lassen sich in weitere Unterprozesse aufteilen
an deren Umsetzung unterschiedliche Abteilungen des Unternehmens, externe Dienstleister oder andere Unternehmen beteiligt sind. Bei globalen Unternehmen kommen zu dieser Personenvielfalt auch diverse Sprachen hinzu, die bei der Kommunikation innerhalb der Prozesse und an den Schnittstellen zu berücksichtigen sind.

Quality Gates

Übergänge zwischen zwei Phasen sind dabei oft durch so genannte „Quality Gates“ begleitet. Quality Gates werden als interne Kontrollstellen eines standardisierten Produkt Entwicklungs Lebenszyklus definiert, die eine Übereinstimmung mit definierten Prozessstandards gewährleisten und aufzeigen sollen.
Die Motivation zur Einführung und Einhaltung der Quality Gates liegt in folgenden Gründen:

- Transparenz des Produkt Entwicklungs Lebenszyklus-Status und der Produktreife,
- Herstellen vernünftiger Kontrollpunkte entlang des Produkt Entwicklungs Lebenszyklus – oft gefordert “as lean as possible and as comprehensive as necessary”,
- Ermöglichen einer Synchronisation über Programme/ Projekte/ Komponenten/ Szenarios und beteiligte Parteien hinweg und
- Sichern der Übereinstimmung von Entscheidungen.

Deliverables

Die erfolgte Arbeit in den Phasen bzw. Prozessen und Teilprozessen spiegelt sich zumindest theoretisch in dokumentierter Form in sogenannten *Deliverables* wieder, die an den unterschiedlichen definierten Quality Gates vorliegen müssen. Diese liegen je nach Gate und je nach Prozess in sehr unterschiedlicher Granularität vor, im Rahmen der Entwicklungsphase einer Software geht es dabei beispielsweise um Benutzeroberflächen-Texte oder eine Szenario-Beschreibung.
Hierbei zu erstellenden Dokumente und Daten fokussieren unterschiedliche Zielgruppen und repräsentieren –abhängig von den beteiligten Personen, die an der Ausarbeitung der Deliverables arbeiten – unterschiedliche Sichtweisen auf das zukünftige Produkt.

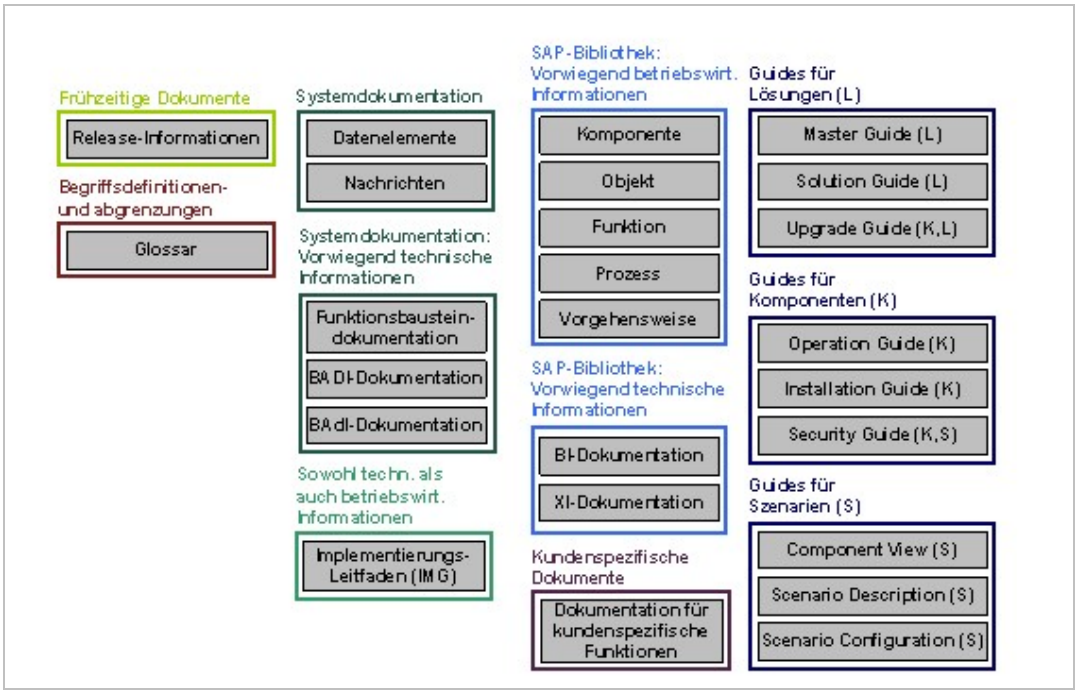


Abb. 13 Dokumenttypen – Übersicht (aus [SAPSch07])

Ein Deliverable eines Teilprozesses wird dabei im Ganzen als Input für einen anderen Teilprozess verwendet. Die Verwendung ist dabei durch die oft unterschiedliche Formatierung und Zielsetzung der Deliverables und die verschiedenartige Granularität nicht der Art, dass eine echte 1:1 Wiederverwendung stattfinden kann, sondern es handelt sich dabei oft lediglich um das ‚zur Verfügung stellen' von Dokumenten, die dann je nach Prozess und Toolunterstützung unterschiedlich weiter verarbeitet werden. Ein- und dieselben Wissens-Inhalte finden sich dementsprechend je nach Zielgruppe und je nach Prozess, in dem sie verdatet werden, redundant an mehreren Stellen in verschiedenen Dokumenten wieder, ohne dass sie einheitlich aus einer Quelle (optimalerweise der originären Wissens-Quelle) gespeist werden, sondern bei der Weiterverarbeitung verändert (und damit auch möglicherweise verfälscht) werden.

Wir möchten im Folgenden für einen noch zu konstruierenden Anwendungsfall analysieren,

- ob eine Wiederverwendung
- welchen Contents und
- in welcher Form

zum einen für das sich anschließende Software Engineering und zum anderen für das weiter hinten gelagerte Document Engineering sinnvoll ist.
Diese Wiederverwendung sehen wir dabei aber nicht in einem bloßen Übermitteln von geforderten Deliverables, sondern möchten es eingebettet in einen Rahmen verstehen, in dem wir nicht nur betrachten, ob Content wie eine Art Dokumentationsbaustein wieder-verwendbar ist, sondern ob evtl. das Wissen, das zur Ausarbeitung eines Deliverables geführt hat, nicht besser in einer anderen Form verdatet werden sollte, um für nachgelagerte Prozesse zugänglich und wieder-verwendbar zu sein.

3.2 Requirements Engineering

Requirements Engineering

Anforderungsanalyse (oder im Englischen Requirements Engineering) beschäftigt sich mit der Analyse des Ist-Zustandes bei einem existierenden System bzw. der Analyse der Kundenwünsche als Soll-Zustand für ein neu zu erstellendes System.
Gemäß IEEE ist eine Anforderung wie folgt definiert:

> Zitat Z7: [IEEE610]
> *„Eine dokumentierte Darstellung einer Bedingung oder Fähigkeit, die entweder von einem Benutzer zur Lösung eines Problems oder zur Erreichung eines Ziels benötigt wird oder die ein System oder System-Teile erfüllen oder besitzen muss, um einen Vertrag, eine Norm, eine Spezifikation oder andere, formell vorgegebene Dokumente zu erfüllen.“.*

Oftmals wird eine Anforderung auch folgendermaßen definiert:

> Zitat Z8: [Rupp07]
> *„Eine Anforderung ist eine Aussage über eine Eigenschaft oder Leistung eines Produktes, eines Prozesses oder der am Prozess beteiligten Personen“.*

Anforderungsarten

Es existieren dabei verschiedene Anforderungsarten:

- funktionale Anforderungen,
- technische Anforderungen,
- Anforderungen an die Benutzerschnittstelle sowie
- Qualitätsanforderungen.

In [NeStZd02] werden drei verschiedene Arten von Anforderungen aufgelistet:

- Kundenanforderungen: Beschreiben was der Kunde wünscht und welche Kundenprobleme gelöst werden sollen.
- Produktanforderungen fokussieren eher auf das Produkt (bzw. die Software Komponente), welches erstellt werden soll.
- Projektanforderungen bestimmen die technische Lösung für eine bestimmte Anforderung und beschreiben wie die Anforderungen des Kunden in eine zufriedenstellende technische Lösung umgewandelt werden kann.

Produktanforderungen sind dabei die Brücke zwischen Kunden- und Projektanforderungen und hinterfragen, wann der Kunde ein bestimmtes Feature im Produkt erhalten soll.

Probleme

Typische Probleme in der Anforderungsanalyse sind unklare Zielvorstellungen, hohe Komplexität, Sprachbarrieren, veränderliche Anforderungen, schlechte Qualität, unnötige Merkmale sowie ungenaue Planung [Rupp07]. Wie [BrEtAl07] genauer darstellen, gelten ausserdem folgende Herausforderungen als besonders kritisch:

- es gibt keine eindeutige Definition, was eine „Best Practice“ ist und es fehlen akzeptierte Referenzmodelle
- Anforderungen sind nur mangelhaft dokumentiert, zu lösungsorientiert, unvollständig, inkonsistent, nicht implementierbar oder nicht skalierbar strukturiert. Verfügbare Werkzeuge sind ineffektiv, bieten nur sehr generische Konzepte, fordern hohen Administrationsaufwand und bieten schlechte Visualisierungskonzepte an
- Zwischen Produktmanagement, Forschung und Entwicklung, Marketing und Vertrieb gibt es oft keine klar abgestimmten Vorgehensweisen und kein einheitliches Kommunikationsmedium.

- Häufige und späte Änderungen der Anforderungen sind meist nicht vermeidbar und werden insbesondere bei Prozessen mit sequenziellen Vorgehensweisen nicht ausreichend beherrscht.

Dokumente

Derzeit werden häufig folgende wesentlichen Dokumente im Rahmen der Anforderungsanalyse erstellt:

- eine Liste von wesentlichen Funktionalitäten des zu entwickelnden Produkts oder Systems,
- eine Liste der wesentlichen Produktattribute,
- Anwendungsfälle,
- ein konzeptuelles Modell sowie
- ein Glossar (Data Dictionary), welches die verwendeten wesentlichen Begriffe beschreibt.

Die Produkt- oder Systemfunktionalität beschreibt dabei eine Liste von Eigenschaften, welche ein Produkt oder System leisten soll. Die Elemente dieser Liste werden dabei kategorisiert, priorisiert und nach sinnvollen Kriterien gruppiert. Kategorien können entweder evident (notwendig und sichtbar), versteckt oder optional sein. Attribute bezeichnen Charakteristiken des Produkts oder Systems, die keine Funktionalität darstellen, also beispielsweise Größe, Gewicht, Effizienz, Antwortzeiten, Fehlertoleranz, Bedienungskomfort, etc. und werden in gewünschte und verlangte Attribute unterschieden. Die Anwendungsfälle werden oft auch Use Cases genannt und im nachfolgenden Abschnitt genauer beschrieben. Zuerst wird noch auf den Spezialbereich des Software Requirement Engineering eingegangen.

3.3 Software Requirements Engineering

Software Requirements Specification

Die Software Requirement Specification (SRS) ist ein vom IEEE (Institute of Electrical and Electronic Engineers) erstmals unter (ANSI/IEEE Std 830-1984) veröffentlichter Standard zur Spezifikation von Software. Das IEEE hat die Spezifikation mehrmals überarbeitet und die momentan neueste Version ist Std 830-1998.
Gemäß dieser IEEE-Norm ist ein Anforderungsdokument grundsätzlich in 2 Bereiche aufgeteilt:

- C-Requirement (Customer-Requirement)
- D-Requirement (Development-Requirement)

Unter C-Requirement sind die Anforderungen aus Sicht des Kunden und/oder des End-Anwenders zu erfassen. Unter D-Requirement versteht man die Entwicklungs-Anforderungen. Dies ist die Sicht des Entwicklers, der im Gegensatz zum Kunden technische Aspekte in den Vordergrund stellt.

Eine SRS enthält gemäß IEEE Standard mindestens drei Hauptkapitel. Die vorgeschlagene Gliederung sollte zwar in den Kernpunkten eingehalten werden. In der Praxis wird diese jedoch häufig im Detail modifiziert. Eine exemplarische Gliederung kann wie folgt aussehen:

- Name des Softwareprodukts
- Name des Herstellers
- Versionsdatum des Dokuments und / oder der Software

1. Einleitung
    - Zweck (des Dokuments)
    - Ziel (des Softwareprodukts)

- Verweise auf sonstige Ressourcen oder Quellen
- Erläuterungen zu Begriffen und / oder Abkürzungen
- Übersicht (Wie ist das Dokument aufgebaut?)

2. Allgemeine Beschreibung (des Softwareprodukts)
   - Produkt Perspektive (zu anderen Softwareprodukten)
   - Produktfunktionen (eine Zusammenfassung und Übersicht)
   - Benutzermerkmale (Informationen zu erwarteten Nutzern, z.B. Bildung, Erfahrung, Sachkenntnis)
   - Einschränkungen (für den Entwickler)
   - Annahmen und Abhängigkeiten (nicht Realisierbares und auf spätere Versionen verschobene Eigenschaften)
3. Spezifizierte Anforderungen (im Gegensatz zu 2.)
   - funktionale Anforderungen (Stark abhängig von der Art des Softwareprodukts)
   - nicht-funktionale Anforderungen
   - externe Schnittstellen
   - Design Constraints
   - Anforderungen an Performance
   - Qualitätsanforderungen
   - Sonstige Anforderungen

3.4 Use Cases

Im Folgenden beschreiben wir die ‚Entwicklung und Darstellung von Use Cases' als typischen Anwendungsfall für content aus dem Bereich der internen technischen Dokumentation im Rahmen eines Software Entwicklungsprozesses.

Was ist ein Use Case?

Use Cases sind eine Dokumentationstechnik von Softwareanforderungen, die ursprünglich – zu dieser Zeit unabhängig von der Unified Modeling Language UML – von Ivar Jacobson entwickelt wurde. Als Mitentwickler von UML beeinflusste er später hierbei die Einbeziehung von Use Cases als eine Art der bei UML graphisch zu modellierenden Diagrammtypen. Bevor wir UML und seine Diagrammtypen berücksichtigen, möchten wir hier an dieser Stelle jedoch zuerst die Technik der so genannten textuellen Use Cases betrachten.

> Zitat Z9: [Cockbu01]
> *„Actors are basically users of the system. They are actually user types or categories. Actors are external entities (people or other systems) who interact with the system to achieve a desired goal. Therefore: A use case is a collection of possible sequences of interactions between the system under discussion and its Users (or Actors), relating to a particular goal. The collection of Use Cases should define all system be-havior relevant to the actors to assure them that their goals will be carried out properly. Any system be-havior that is irrelevant to the actors should not be included in the use cases."*

Use Cases sind nach Zitat Z9: das, was passiert, wenn ein Aktor mit dem System interagiert. Ein Aktor benutzt das System, um ein gewünschtes Ziel zu erreichen. Unter Aufnahme aller Möglichkeiten, wie das System benutzt wird ("cases of use" oder Use Cases) werden alle Ziele bzw. Anforderungen an das System entsprechend akkumuliert.

Zitat Z10: [Martin03]
„A use case is a description of the behavior of a system. That description is written from the point of view of a user who has just told the system to do something particular. A use case captures the visible sequence of events that a system goes through in response to a single user stimulus. A visible event is an event that the user can see. Use cases do not describe hidden behavior at all. They don't discuss the hidden mechanisms of the system. They only describe those things that a user can see. "

Ein Use Case wird nach Zitat Z10: aus der Sicht des Aktors beschrieben und umfasst alle sichtbaren Sequenzen der Ereignisse, durch die das System aufgrund einer bestimmten Aktor-Stimulanz geht.
Anwendungsfälle oder Use Cases sind also eine Beschreibung, wie ein oder mehrere Anwender ein System benutzen, um eine Aufgabe zu erledigen, die ein definiertes Ergebnis (das Ziel) hat. Der wesentliche Punkt dabei ist die Anwendersicht. Use Cases werden dabei auf verschiedenen Abstraktionsebenen verwendet: nicht nur für die Anforderungsanalyse («Business» Use Cases) sondern auch für die Entwicklung von Systemkomponenten («System» Use Cases) und sind damit eine ideale Grundlage für die Systementwicklung. Auf einen Use Case lassen sich verschiedene Sichtweisen definieren, die sich in Aspekten wie Zweck, Formalität, Vielfachheit und Struktur unterschieden und für eine konkrete Definition festzulegen sind.

- Zweck: Ist der Zweck nur die Sammlung von Benutzungsabläufen oder die Erstellung von Anforderungen?
- Formalität: Müssen Use Cases konsistent sein, oder sind Widersprüche erlaubt? Wenn konsistent, werden sie natürlichsprachlich oder mit formaler Sprache beschrieben?
- Vielfachheit: Ist ein Use Case einfach dasselbe wie ein Ablauf oder kann ein Use Case mehrere Abläufe enthalten?
- Struktur: Ist eine Sammlung von Use Cases einfach eine unstrukturierte Sammlung oder hat sie eine (semi)formale Struktur?

Zweck von Use Cases

Use Cases dienen als Leitfaden, um Anforderungen an das System auszuwählen:

Zitat Z11: [CarMee06]
„The crucial benefit of use cases is the way they encourage a directed method of considering project requirements. From the very beginning, we are designing a product by concentrating upon the needs and wants of those who will use it. "

Zitat Z11: folgend besteht der Vorteil im Einsatz von Use Cases darin, dass ein kunden- und nutzenorientiertes Design bereits aus den Anfängen heraus möglich gemacht wird, da in Gestalt der Use Cases bereits der Fokus darauf gelegt wird, wer das System wofür einsetzen wird.

Zitat Z12: [Martin03]
„The purpose of describing use cases is emphatically not to fully specify the exact nature of what a new site will contain and how it is to be built. Instead, use cases define goals and purpose: the problems we are trying to solve. A well-constructed use-case model can be understood by all the stakeholders in a project: developers, managers and clients. It's a powerful aid to collaborative development. Use cases ensure that scope is under control from the outset. The identification of use cases and their dependencies make it easy to distinguish between core goals that must be satisfied and subsidiary enhancements that may be postponed as. Scoping in this

manner allows for better planning and prioritization. It's an implementation-neutral picture of a project. No assumptions about tools and technologies are made, nor should they be. It's transportable. No special tools are required — sticky notes, a whiteboard, pencil and paper, or your favorite graphics application can all be used to document your vision. "

Zitat Z12: sieht den Vorteil in Use Cases darin, dass ein Use-Case-Modell von allen Beteiligten in einem Projekt verstanden werden kann, so dass hier eine einheitliche Transparenz bzgl. der Systemziele erreicht wird bzw. werden soll. Es wird hier darauf hingewiesen, dass es auch wichtig ist, Abhängigkeiten unter den Use Cases zu identifzieren, um Kernziele von weiteren Zielen unterscheiden und in der Projektvorgehensweise entsprechend priorisieren zu können. Bei einem Use-Case handelt es sich nach Z9 um ein Abbild des Projekts. Hier wird ausserdem betont, dass zur Erstellung von textuellen Use Cases nicht notwendigerweise ein bestimmtes Tool vorhanden sein muss. Dies ist aber für die automatisierte Weiterverarbeitung von Vorteil, wie später dargestellt wird.

Zitat Z13: [Cockbu01]
"Use Cases: Hold Functional Requirements in an easy to read, easy to track text format. Represents the goal of an interaction between an actor and the system. The goal represents a meaningful and measurable objective for the actor. Records a set of paths (scenarios) that traverse an actor from a trigger event (start of the use case) to the goal (success scenarios). Records a set of scenarios that traverse an actor from a trigger event toward a goal but fall short of the goal (failure scenarios). Are multi-level, one use case can use/extent the functionality of another. Use Cases Do Not...Specify user interface design. They specify the intent, not the action Detail Specify implementation detail (unless it is of particular importance to the actor to be assured that the goal is properly met) Use Cases are used during many stages of software development. To Capture the Requirements of the systems. To act as a spring board for the software design. To validate the software design against. For Software Test and Quality Assurance. (Tests are performed to validate proper and complete implementation of the use cases) Potentially as an initial framework for the on line help and user manual."

In Zitat Z13: wird erneut die wichtige Funktion von Use Cases als Beschreibung funktionaler Anforderungen und Repräsentation des Ziels einer Interaktion zwischen Aktor und System erwähnt. Dann wird detaillierter beschrieben, was ein Use Case enthalten sollte und hierbei ist sehr interessant, dass nach Cockburn ein Use Case aus Szenarios aufgebaut ist, die ausgehend vom Start des Use Cases bis hin zum Ende des Use Cases (Cockburn nennt dies im Fall des Erreichens des Ziels das Erfolgs-Szenario und bei Nicht-Erreichen des Ziels das Fehlverhaltens-Szenario) dem Aktor begegnen.

Entwicklung von Use Cases

In dem Zusammenhang ist interessant zu betrachten, was nach Meinung weiterer Personen, die sich ausführlich mit Use Cases und deren Erstellung befasst haben, Use Cases beinhalten sollten und wie man sie entwickeln sollte.

Zitat Z14: [CarMee06]
„ To define a project's use cases, we need to consider two concepts, and how they relate:

-the actors

-the goals

Actors are everyone and everything that will use (or be used) by our new website. Goals are what one, some, or all of the actors want to achieve. To be complete, every use case must describe a specific goal and the actors that will perform tasks to achieve that goal. "

Zitat Z15: [Martin03]
„ The real trick to doing use cases is to keep them simple. Don't worry about use case forms, just write them on blank paper, or on a blank page in a simple word processor, or on blank index cards. Don't worry about filling in all the details. Details aren't important until much latter. Don't worry about capturing all the use cases, that's an impossible task anyway. We write use cases, we don't draw them. Use cases are not diagrams. Use cases are textual descriptions of behavioral requirements; written from a certain point of view. Typically, a use case is broken up into two sections. The first is the primary course. This section describes how the system responds to the stimulus of the user and assumes that nothing goes wrong. Al-ternate Courses Some of those details are concerning things that can go wrong."

Zitat Z16: [Cockbu01]
„ Use Case Definitions: Primary Actors: The Actor(s) using the system to achieve a goal. The Use Case documents the interactions between the system and the actors to achieve the goal of the primary actor. Secondary Actors: Actors that the system needs assistance from to achieve the primary actors goal. Use Case: A collection of possible scenarios between the system under discussion and external actors, charac-terized by the goal the primary actor has toward the system's declared responsibilities, showing how the primary actor's goal might be delivered or might fail. Use cases are goals (use cases and goals are used interchangeably) that are made up of scenarios. Scenarios consist of a sequence of steps to achieve the goal, each step in a scenario is a sub (or mini) goal of the use case. As such each sub goal represents ei-ther another use case (subordinate use case) or an autonomous action that is at the lowest level desired by our use case decomposition. "

Während bis hierhin immer von einer rein textuellen Beschreibung der Use Cases ausgegangen wird, werden an anderen Stellen Use Cases als typische Kandidaten für eine graphische Modellierung eingeführt. Der UML2 Standard stellt ein Use Case Diagramm zur Verfügung, um die wichtigsten Akteure und Ereignisse auf einem abstrakten Level darstellen zu können. Im UML-Umfeld ist ein Use Case folgendermaßen definiert:

Zitat Z17: [OMG06]
"Use cases are a means for specifying required usages of a system. Typically, they are used to capture the requirements of a system, that is, what a system is supposed to do. The key con-cepts associated with use cases are actors, use cases, and the subject. The subject is the system under consideration to which the use cases apply. The users and any other systems that may in-teract with the subject are represented as actors. Actors always model entities that are outside the system. The required behavior of the subject is specified by one or more use cases, which are de-fined according to the needs of actors. Strictly speaking, the term "use case" refers to a use case type. An instance of a use case refers to an occurrence of the emergent behavior that conforms to the corresponding use case type. Such instances are often described by interaction specifications."

Allen Zitaten folgend ist gemein, dass ein Use Case immer initiiert ist von einem Benutzer, der ein spezifisches Ziel im Kopf hat und vollständig erfolgreich ist, wenn dieses Ziel zufriedengestellt wurde. Eine graphische oder textuelle Darstellung von Use Cases wird sich in ihrer Beurteilung dementsprechend daran messen lassen müssen, wie verständlich die Beschreibung der Ziele ist und wie sorgfältig die Vorgehensweise ist, mit der die Use Cases entwickelt werden.

Zusammenhang Use Case und Geschäftsprozessmodelle

Use Cases (bzw. Anwendungsfälle) können oft in Geschäftsprozessmodellen wiedergefunden werden. So sind Geschäftsprozessmodelle eine Möglichkeit zur Beschreibung von Abfolgen innerhalb eines Unternehmens. Dafür existieren diverse Notationen. Diese Geschäftsprozessmodelle bestehen jeweils aus Aktivitäten (in UML Aktionen genannt), die in zeitlichem Zusammenhang aneinander gehängt werden. Dabei können Pfade in diesem Geschäftsprozessmodell parallel ablaufen oder auch alternativ ausgeführt werden. Die Aktivitäten können dabei vorhandene Use Cases referenzieren oder eigenständig sein. Geschäftsprozessmodelle können in der Regel in unterschiedlichen Sichten angegeben werden: die fachliche Sicht beschreibt nur den groben Ablauf sowie die Verantwortlichkeiten, wohingegen die technische Sicht Applikationen, Daten, etc. zusätzlich beinhaltet.

Use Cases beinhalten typischerweise Akteure sowie Szenarien.

Definition Akteur

Alle Personen, Gruppen, Institutionen, Rollen und Systeme, die an einer Erledigung einer Aufgabe beteiligt sind, heißen Akteure. Diese können in externe / interne, primäre / sekundäre sowie inizierende / partizipierende Akteure unterschieden werden. Dabei ist ein primärer Akteur ein (externer) Akteur, der ein Ziel hat, das die Unterstützung des entwickelten Systems verlangt, wohingegen ein sekundärer Akteur vom System benötigt wird. Der inizierende Akteur ist derjenige Akteur, dessen Ziel der Use Case erfüllen soll.

Das Vorgehen der Entwicklung von Anwendungsfällen könnte wie folgt aussehen:

- erster Schritt: Erstellen einer Grobbeschreibung. Diese beinhaltet Name, inizierender Akteur, beteiligte primäre Akteure, Typisierung, Kurzbeschreibung, Querverweise auf benötigte Systemfunktionen und –attribute
- zweiter Schritt: Erstellung einer detaillierten Beschreibung für wichtige Use Cases anhand eines Kommunikationsmodells

Definition Szenario

Eine Aktion verknüpft dabei das Ziel eines Akteurs mit der Verantwortlichkeit eines anderen Akteurs. Dabei entsteht eine Interaktion. Eine Interaktion kann auch eine Sequenz von Interaktionen sein. Eine Sequenz von Interaktionen heißt auch Szenario und hat keine Alternativen und beschreibt genau eine Entwicklung von Vergangenheit nach Zukunft unter definierten Bedingungen. Eine Sequenz kann auch parallele Abarbeitungsschritte beinhalten. Ein Szenario findet unter gewissen Bedingungen statt, dient dem Ziel des inizierenden Akteurs, beginnt mit einer auslösenden Aktion und endet mit einem Resultat (entweder Erreichung des Ziels oder Aufgabe).

Ein Use Case ist also eine Sammlung von Szenarios zwischen dem betrachteten System und externen Akteuren und wird charakterisiert durch ein Ziel, welches der inizierendeAkteur erreichen möchte. Ein Use Case enthält typischerweise sehr viele Szenarien, die sich in wenigen Punkten unterscheiden. Dabei gibt es meist ein oder wenige Hauptszenarien, denen eine Variation oder eine Erweiterung zugeordnet wird.

3.5 Reuse im Software Engineering

Vorgehensmodelle

Anforderungen, Requirements und Use Cases wie in Kapitel 3.3 beschrieben, werden im Rahmen des Software Engineering verwendet, um daraus Analyse- und

Designdokumente zu erstellen, Code zu implementieren, diesen zu testen und nach Fertigstellung das fertige Produkt an den Kunden auszuliefern. Dabei gibt es verschiedene Software-Entwicklungsprozesse und Vorgehensmodelle, die das Vorgehen genauer beschreiben.

Wasserfall-modell

Das einfachste ist das Wasserfallmodell. Hier wird angenommen, dass zuerst alle Anforderungen analysiert werden. Nach Abschluss der Analysephase erstellt man Designdokumente, die das zu entwickelnde System näher beschreiben. Auf Basis dieser Designdokumente startet schließlich die Implementierung. Nach Fertigstellung kommt die Phase des Testens und abschließend die Wartung. Sollte in einer Phase ein Problem aufgetreten sein, kann ab dem iterierten Wasserfallmodell direkt in die vorherige Phase zurückgegangen werden, eine direkte Verknüpfung zwischen allen Phasen ist nicht vorhanden. Im ursprünglichen Wasserfallmodell hingegen war kein Rücklauf in die vorherige Phase angedacht.

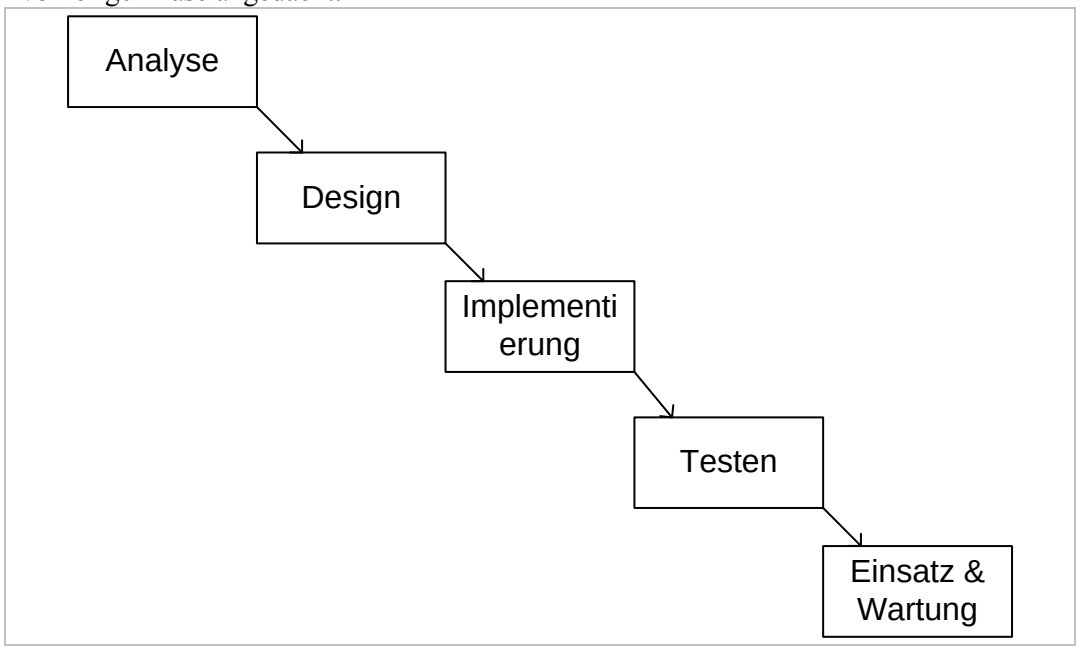


Abb. 14 Wasserfallmodell

V-Modell

In erweiterter Form stellt sich dagegen das Vorgehensmodell V-Modell dar: hier wird die Analyse mit einer Definitionsphase kombiniert, danach folgt erst der grobe Entwurf, dann der Feinentwurf, bevor die Implementierung startet. Alle bisherigen Phasen können im weiteren Verlauf wieder verwendet werden: die Beschreibungen der Analyse werden als Szenarien für die Abnahme der Software verwendet, alle anderen Phasen zeichnen sich durch eine automatische Generierung von Testfällen aus. Das V-Modell (97) wurde in den letzen Jahren zum V-Modell XT erweitert. Dort nehmen Anforderungen nun eine zentrale Rolle ein. Durch eine Trennung zwischen Anforderungen aus Sicht des Kunden (Lastenheft) und der daraus durch den Auftragnehmer abgeleiteten detaillierten Anforderungsmenge (Pflichtenheft) ist es möglich, den verschiedenen Projektbeteiligten die für sie jeweils wichtige Perspektive auf die Anforderungen zu liefern. Diese Trennung der Sichtweisen ist neu im Vergleich zu bisherigen Vorgehensmodellen wie dem V-Modell (97) oder anderer. Im V-Modell XT existieren zwei Vorgehensbausteine (Anforderungsfestlegung und Systemerstellung), die diese unterschiedlichen Sichtweisen von Auftragnehmer und Auftraggeber näher beschreiben.

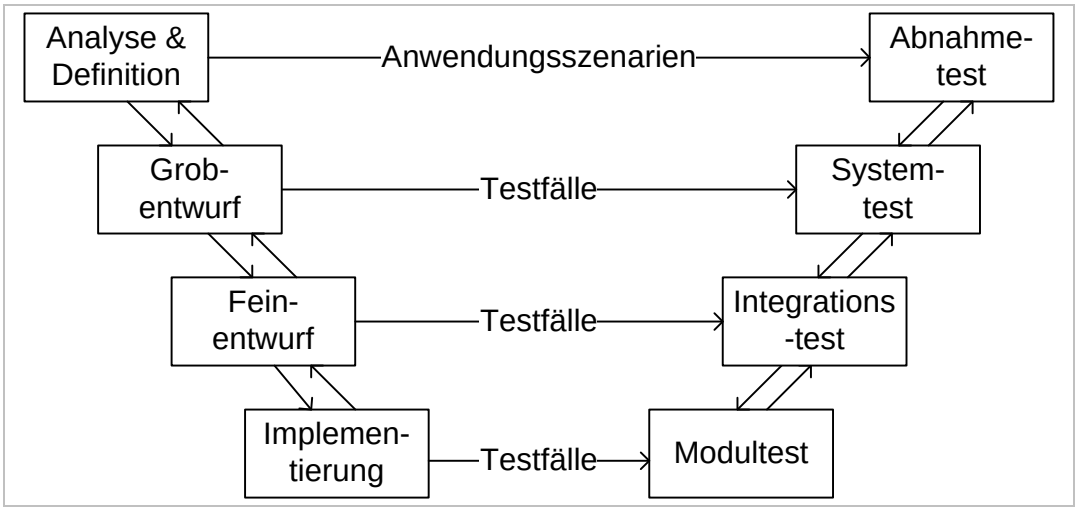
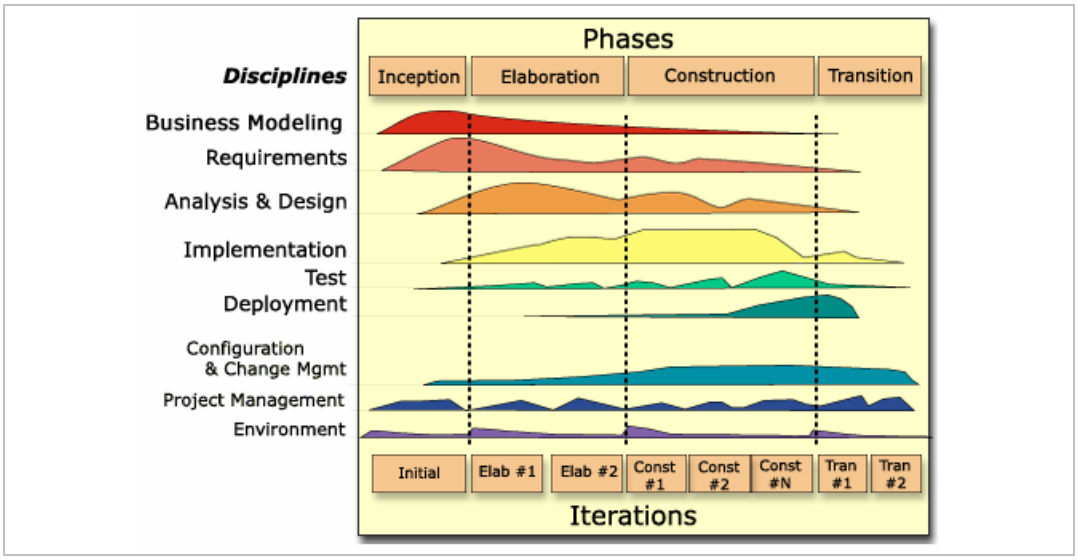


Abb. 15 V-Modell 97

Spiralmodell

Meistens wird heutzutage beispielsweise wie im Spiralmodell eine iterative Entwicklung bevorzugt. Dazu werden einzelne Kernbestandteile des Systems programmiert und wenn diese vollständig umgesetzt sind, findet eine Besprechung mit dem Auftraggeber statt. Wird die bisherige Software abgenommen, wird auf dieser Basis das Produkt weiterentwickelt, ansonsten nachgebessert. Dadurch ist es nicht möglich, dass am Ende ein Produkt entstanden ist, in welchem die Anforderungen des Auftraggebers falsch verstanden oder gar nicht umgesetzt wurden. Häufig wird diese iterative Vorgehen noch verfeinert. Im Vorgehensmodell eXtreme Programming wird diese Kundeninteraktion beispielsweise noch verstärkt, in dem direkt im Projektteam immer Mitarbeiter des Auftraggebers sitzen und die Implementierung überwachen bzw. selbst mitgestalten. Ausserdem sind im eXtreme Programming immer zwei Programmierer gleichzeitig an einem Quellcode am Arbeiten, was die Fehleranfälligkeit von vornherein reduziert.



Rational Unified Process

Abb. 16 Rational Unified Process

Der Rational Unified Process (RUP) unterscheidet als Vorgehensmodell neben den Phasen noch einzelne Disziplinen. Als Disziplinen sind die uns schon bekannten

Bereiche Requirements Engineering, Analyse & Entwurf, Implementierung / Test, Einsatz & Verteilung, Konfigurations- und Änderungsmanagement, Management und Projektmanagement zu nennen. Als Phasen wird hier zwischen der Vorbereitungsphase, der Software-Strukturierung, der Software-Erstellung und der Übergabe unterschieden. Die folgende Abbildung zeigt den typischen Verlauf eines Software-Entwicklungsprozesses im RUP.

Verarbeitung der Anforderungen im Software Engineering

Wie wir in Kapitel 3.1 bei der Beschreibung, welche Dokumente im Rahmen des Requirements Engineering erstellt werden, gesehen haben, werden Anforderungen durch folgende Dokumente aufgenommen:

- eine Liste wesentlicher Systemfunktionalitäten,
- eine Liste wesentlicher Systemattribute,
- Anwendungsfälle,
- ein konzeptuelles Modell und
- ein Glossar (Data Dictionary), welches die verwendeten wesentlichen Begriffe beschreibt**.**

Den wichtigsten Stellenwert nehmen dabei die Anwendungsfälle oder Use Cases ein. Dort werden die Akteure und Szenarien detailliert in natürlichsprachlicher Form beschrieben. Diese Akteure sind Personen oder Systeme, die mit dem zu erstellenden Produkt oder System später interagieren. Die Szenarien stellen eine Abstraktion über die möglichen Interaktionen dar. Es wird also durch die Anforderungsanalyse nur ein Verständnis des Problembereichs bereitgestellt, die exakte Ausführung in allen Details wird hingegen nicht beschrieben. Allerdings werden bestimmte Dokumente erstellt, die für die Entwicklung von essentieller Notwendigkeit sind. Hier wären z.B. das Vokabular (Glossar) des Produkts oder Systems zu nennen, welches Konzepte der realen Welt, die für die Entwicklung des Produkts wichtig sind, sowie deren Attribute beinhaltet und natürlichsprachlich beschreibt.

Generell kann gesagt werden, dass nirgends eine 1:1-Wiederverwendung von Anforderungen im Software Engineering erfolgt. Die oben angegebenen Dokumente werden vielmehr dazu verwendet, um über eine ausgefeilte Architektur des Software-Systems zu diskutieren, um alle Stakeholder zu berücksichtigen und die geplante Verwendung der Software zu kennen. Hier sind die Anwendungsfälle inkl. Akteure wie bereits erwähnt die wichtigsten Elemente. Dort wird beschrieben, in welcher Reihenfolge die Interaktion mit dem System zukünftig geschehen soll und dadurch können typische Komponenten gefunden werden. Verweisen beispielsweise mehrere Aktionen in einem Haupt- oder Nebenszenario auf eine Systemfunktionalität (z.B. Login) kann diese im Design als eigene Komponente realisiert werden. Dies sollte gemäß dem Divide-and-Conquer-Prinzip (Modularisierung) als fundamentale Strategie angewandt werden, um das meist große Problem in verständliche und handhabbare Teile aufzuteilen. Innerhalb einer Komponente (oder Package, neuerdings auch oft mit Web Service gleichgesetzt) sollte diese Strategie wiederum angewandt werden, um einzelne Klassen zu identifizieren.
Bei den oben genannten Diskussionen werden meist auch die Auftraggeber (Kunden) mit einbezogen bzw. diesen die Ergebnisse vorangegangener Diskussionen präsentiert. Ein großes Problem, das sich hierbei ergibt, ist Art der Darstellung: die Modelle der Software-Architekten basieren meist auf der graphischen Modellierungssprache Unified Modeling Language (UML), wohingegen Personen aus Fachabteilungen nur ARIS-Modelle oder gar keine Darstellungsform kennen. Hier wird oftmals viel Zeit benötigt, um den Gesprächspartner in die verwendete Notation einzuweisen. Auf Basis dieser

Modelle wird dann über die Struktur diskutiert, wobei die Wünsche der Auftraggeber oft nicht technisch realisierbar sind oder umgekehrt die Auftraggeber einen besonderen Kniff der SW-Architekten nicht verstehen. Dies resultiert aus dem unterschiedlichen Fokus dieser Personengruppen.
Diskussionen werden auch oft notwendig, weil die Anforderungen zwar in Kurzbeschreibungen in dem Lasten-/ Pflichtenheft gesammelt wurden, diese aber in ausführlicher Form nur in den Köpfen der Auftraggeber existieren. Dies wird oftmals noch durch eine ungenaue, informelle Beschreibungsweise verstärkt („es muss ***einfach*** sein“).

Genauere Beschreibung von Use Cases

Es wäre notwendig, diese (versteckten) Informationen ebenfalls in den Use Cases (oder in einem anderen Dokument im Rahmen des Requirements Engineering) zu notieren. Wie bei näherer Untersuchung existierender Beispielanwendungsfälle festgestellt wurde, besteht ein Szenario in einem Use Case dabei aus:

- Name des Szenarios,
- Beschreibung,
- einem Standardablauf / Hauptszenario,
- diversen (oder auch keine) Varianten / Nebenszenarien,
- Anwendungsumfeld,
- Auslöser bzw. Vorbedingungen,
- Eingangsdaten sowie verwendete Daten im gesamten Ablauf,
- Nachbedingungen bzw. Ereignisse,
- Beteiligte Akteure (Benutzertyp) und Geschäftsobjekte (inkl. deren Attribute),
- Fehlermeldungen und Referenzen,
- Nichtfunktionale sowie ggf. weitere funktionale Anforderungen,
- Systemfunktionen,
- Masken und Dialogabläufe,
- Validierungslogik und Fehlerbehandlung,
- Abnahmekriterien,
- Priorität des Anwendungsfalls / des Szenarios,
- verwendete Begrifflichkeiten und deren Zusammenhänge (siehe auch Glossar) und
- zusätzliche Anmerkungen.

Der Name eines Szenarios beschreibt, was genau passieren soll und könnte zusammen mit der Beschreibung des Szenarios auf jeden Fall für die Dokumentation von Klassen und Packages verwendet werden, gegebenenfalls sogar für den Namen eines Unterpackages (wobei hier die Abhängigkeit mit anderen Szenarien zuerst zu klären wäre). Dabei können das Hauptszenario sowie eventuelle Nebenszenarien

Aktivitäts-diagramme

verwendet werden, um den Geschäftsprozess sowie die technischen Details in UML 2 Aktivitätsdiagrammen oder ereignisgesteuerten Prozessketten zu modellieren. Diese können ihrerseits verwendet werden, um die Orchestrierung von Methoden innerhalb diverser Klassen zu bestimmen. Dabei müssen auch die Vorbedingungen und Auslöser berücksichtigt werden, da diese erfüllt sein müssen, bevor das Hauptszenario begonnen werden kann.

Klassen

Die beteiligten Geschäftsobjekte und Attribute können ihrerseits selbst in Klassen und Klassenattribute überführt werden, wobei der Typ der Attribute noch nachgepflegt werden muss. Die beteiligten Akteure können in der Nutzerverwaltung als Rollen angelegt werden, in welcher dann die jeweiligen Benutzer das System verwenden dürfen. Wurden Masken und Dialogabläufe erstellt, könnten diese wieder verwendet

werden, um die später in Wirklichkeit zu verwendenden Oberflächen zu gestalten. Dazu müssten diese allerdings nicht nur in einem Graphikprogramm erstellt werden, sondern detaillierter beschrieben sein. Die Abnahmekriterien müssten ebenfalls so formuliert werden, dass diese nicht nur natürlichsprachlich sondern auch maschinen-auswertbar vorliegen.
Werden die verwendeten Begrifflichkeiten und deren Zusammenhänge beschrieben, könnten diese automatisch für die Erstellung von Klassendiagrammen oder auch von Ontologien hergenommen werden. Die nicht-funktionalen Anforderungen beschreiben sowohl zeitliche Constraints als auch Sicherheitsaspekte. Diese könnten auch automatisch zur Erstellung vordefinierter Klassen für die Verschlüsselung, etc. übernommen werden.
In [Rupp07] werden neben den hier in Prosatext geschriebenen Anforderungen auch noch weitere Möglichkeiten zur Darstellung und Verfeinerung der Anforderungen genannt: Das System-Use-Case-Diagramm (eine Verfeinerung des Business-Use-Cases) liefert einen ersten Überblick über die Funktionen, die das neue System realisieren soll, und wird später durch Komponenten-Use-Cases verfeinert. Es können bereits direkt beim Aufnehmen der Anforderungen Aktivitäts- oder Sequenzdiagramme oder auch Zustandsautomaten erstellt werden, die dann im weiteren Verlauf (in der Designphase) mehr und mehr verfeinert werden können. Dabei wird empfohlen Diagramme nur dann einzusetzen, wenn der Aufwand sie zu erstellen und zu pflegen nicht unverhältnismäßig groß ist und der Sachverhalt einfacher durch natürliche Sprache auszudrücken wäre. Sollten sowohl Prosatext als auch Diagramme eingesetzt werden, sollten diese verlinkt sein, um bei Änderungen stets beides aktualisieren zu können.

Wie oben beschrieben kann ein Szenario in ein Haupt- und mehrere Nebenszenarien unterschieden werden. Das Hauptszenario (Standardablauf) wird in einem ersten Zug dazu verwendet, um Designmodelle wie Aktivitätsdiagramme oder Zustandsautomaten zu entwerfen. Abb. 17 stellt ein beispielhaftes Anwendungsfalldiagramm dar, welches durch ein Aktivitätsdiagramm umgesetzt werden könnte.

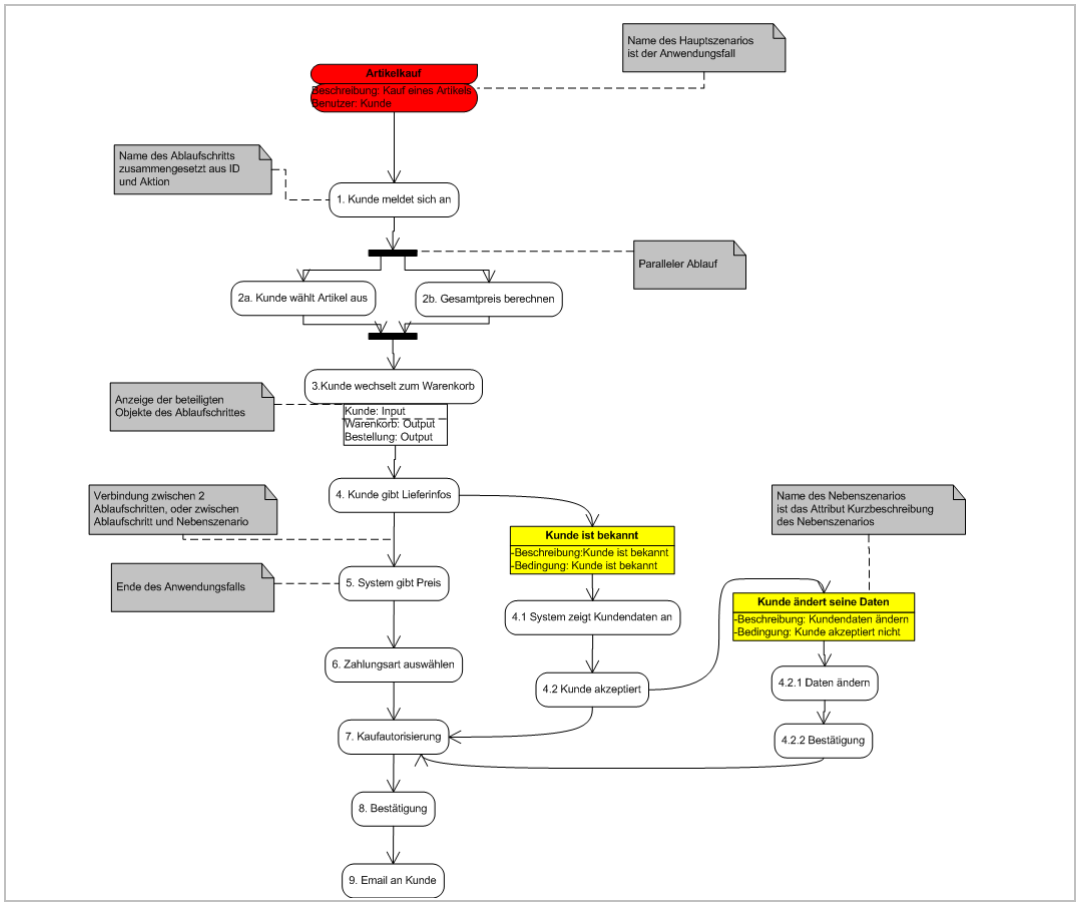


Abb. 17 Anwendungsfalldiagramm

Nutzung der Szenarien

Aktivitätsdiagramme bzw. Zustandsautomaten beschreiben welche Aktionen resp. Zustände eines Systems existieren können und unter welchen Bedingungen bzw. durch welche Aktionen man von einem Zustand zum nächsten gelangen kann. Zur Erstellung von Aktivitätsdiagrammen werden auch die Nebenszenarien berücksichtigt, um alternative Abläufe ebenfalls mit darzustellen.

Reuse im Software-Engineering

Durch die obige Beschreibung der Software-Entwicklungsprozesse und detailliertere Betrachtung der einzelnen Bestandteile von Use Cases kann gesehen werden, dass die Use Cases primär in der ersten Phase des jeweiligen Software-Entwicklungsprozesses erstellt werden und dann in nachfolgenden Phasen erweitert werden. Hier können die anfangs sehr abstrakt gehaltenen Use Cases durch systemspezifische Punkte verfeinert werden und auch nach Abschluss der Analysephase in der Designphase (bzw. vom Grobentwurf zum Feinentwurf) noch genauer beschrieben werden. Ursprünglich textuell beschriebene Anforderungen werden im Laufe eines Projekts graphisch modelliert und durch verschiedene Diagrammtypen verfeinert (begonnen mit einem Use Case-Diagramm über Aktivitätsdiagramme, Sequenzdiagramme oder Zustandsautomaten bis hin zu Klassendiagrammen). Ein Reuse von Anforderungen, Analyse- oder Designdokumenten über die Grenzen eines einzelnen Projekts hinweg findet allerdings derzeit meist nicht statt.

Hingegen werden auf Implementierungsebene existierende Komponenten, Middleware-Ansätze, Architekturen und Frameworks sehr wohl in unterschiedlichen Projekten immer wieder verwendet. Hier ist allerdings bisher das Fachwissen des Software-

Entwicklers gefragt, der genau wissen muss, welche Frameworks denn existieren und welche er für welche Problemstellungen einsetzen kann. Auch auf Testebene werden existierende Frameworks (wie bspw. jUnit) immer wieder verwendet, wobei hier allerdings die Testfälle selbst jeweils auf das Projekt angepasst werden müssen.

3.6 Reuse im Documentation Engineering

Industrialisierung des Schreibens

Wie bereits in Kapitel 1.1 ausgeführt, gehören zur Technischen Dokumentation sowohl interne als auch externe technische Dokumentation. Beide Bereiche sind stark geprägt von vorgeschriebenen Regelwerken, die sich in Anlehnung an Produkthaftung und qualitätssichernde Maßnahmen (oft begleitet von Zertifizierungsbestrebungen) herausgebildet haben und eingehalten werden müssen. Im Bereich der externen Dokumentation haben sich in den letzten Jahren durch den Einsatz neuer Technologien und neuer redaktioneller Systeme Ansätze zur Modularisierung von Dokumenten herausgebildet, dessen Triebfeder die kosteneffiziente und fehler-resistente Erstellung externer technischer Dokumente war. Verstärkt werden nun als Redaktionssysteme Content Management Systeme eingesetzt, die Dokumente nicht mehr als abgeschlossene Einheiten vorhalten, sondern die eigentlichen Inhalte getrennt von Layout und Struktur verwalten. Diese Separation ermöglicht eine flexiblere Wiederverwendung der Inhalte. Als zentrale Methoden des Content Management sind dabei Single Source und Cross Media Publishing anzusehen.

Single Sourcing

Die Begriffe *Single Source Publishing* (SSP) und *Cross Media Publishing* (CMP) werden im angelsächsischen Raum nicht differenziert betrachtet, sondern unter dem Begriff „Single Sourcing“ zusammengefasst.

> Zitat Z18: [Rockle03]
> *„Single sourcing implies that there is a single source for content; content is written once, stored in a single source location, and reused many times.“*

Im deutschen Raum wird unter Single Source Publishing und Cross Media Publishing sehr wohl differenziert. Single Source Publishing wird hier wie folgt definiert:

> Zitat Z19: [Ziegle04]
> *„Mehrfachverwendung von Quellinformationen in verschiedenen ‚Anwendungen’“*

Ziegler folgend kann man sich Single Source Publishing wie in Zitat Z19: beschrieben beispielsweise als das Wiederverwenden von Textbausteinen, die in einer Quelle gehalten werden, in unterschiedlichen Dokumenten vorstellen, wobei das Verwenden des Begriffs Quellinformationen hier nicht unserer Auffassung und unserem Verständnis von content entspricht. Überträgt man Zitat Z19: auf unser Verständnis lässt sich festhalten, dass es sich bei Single Source Publishing um das Wiederverwenden einer Dateninstanz handelt. Eine Dateninstanz, die digitalisiert vorgehalten wird (unserem Verständnis folgend also eine mögliche konkretisierte Repräsentation eines entsprechenden Datums) wird gebunden an eine gewünschte Anwendung wiederverwendet.
Dagegen wird beim Cross Media Publishing ausgehend von einer digitalisiert vorliegenden Dateninstanz gebunden an ein gewünschtes Medium eine weitere Dateninstanz erzeugt, die *inhaltlich* dasselbe Datum in derselben oder einer anderen gewünschten Repräsentation konkretisieren soll.
Die Begrifflichkeiten von content, Inhalt, Daten und Informationen gehen in diesen Anwendungsbereichen oft fließend ineinander über und es ist sehr wichtig, im jeweiligen Disput genau zu erörtern, was im konkreten Fall darunter zu verstehen ist.

Das unterschiedliche Verständnis dieser Begriffe in Kombination führt oft zu einer abgeleiteten falschen Erwartungshaltung, was jeweilige Systeme zu leisten haben.

Produktdaten-integration

Ein anderer Trend, der zu beobachten ist, ist der zunehmende Wunsch, den Gedanken des Single Sourcing auf den gesamten Produktlebens- bzw. entwicklungsprozess zu übertragen und hier drängen Themen wie Produktdaten-integration und -management in den Vordergrund. Die Brücke zwischen Produkt und Dokumentation soll durch die Integration dieser Produktdaten geschlagen werden. Im Zusammenspiel von Produktdatenbanken und neuartigen Redaktionssystemen, die Single Sourcing ermöglichen und unterstützen, rein technisch betrachtet, eine lösbare Herausforderung.

Lean Documen-tation

Ein neues Schlagwort, das zunehmend ins Zentrum des Interesses rückt, verbirgt sich hinter dem Namen *Lean Documentation*, das angelehnt ist an das Konzept der Lean Production. Das Konzept der *Lean Production* wurde nach dem Zweiten Weltkrieg bei der Toyota Motor Company vom damaligen Präsidenten Eiji Toyota und seinem Produktionsleiter Taiichi Ohno entwickelt, weshalb es auch als Toyota-Produktionssystem (TPS) bezeichnet wird. Die Grundlage für dieses neue Produktionssystem bildete ein ganzheitlicher Ansatz, der aus dem Zusammenwirken einzelner Konzepte besteht – genau so wie sich ein Puzzlebild aus vielen einzelnen Teilen zusammensetzt.
Nach [Traege94] lässt sich die schlanke Produktion in folgende Ansätze untergliedern:

- Kaizen (Prozess der stetigen Verbesserung)
- Jidoka (Automatisierung bzw. Autonomation)
- Muda (Vermeidung von Verschwendung)
- Just-in-Time (Verzicht auf Lagerhaltung)
- Lieferantenbeziehung
- Kundenorientierung
- Betriebliche Aspekte

Nach dem Ansatz des Total Quality Managements ist Qualität als Fehlerfreiheit und Verbesserung des Kundennutzens definiert. Während die Fehlerfreiheit die objektive Qualität bezüglich der Standards und Richtlinien zur Texterstellung beschreibt, stellt der Kundennutzen das subjektive Qualitätsempfinden aus Sicht der Zielgruppe dar. Durch diese duale Definition des Qualitätsbegriffes werden Verbesserungen überhaupt erst möglich, da Qualität nicht länger eine Liste definierter Eigenschaften darstellt, sondern die realen Bedürfnisse der Kunden einbezieht.

> Zitat Z20: [Schott06]
> *„Die schlanke Dokumentation ist ein ganzheitlicher Ansatz zur Optimierung der Prozesse, Werkzeuge und Produkte der Dokumentation und setzt sich genau wie die schlanke Produktion aus verschiedenen Konzepten zusammen, die jeweils einen Beitrag zur Steigerung der Gesamteffizienz leisten. Ziel der schlanken Dokumentation ist ein höherer Ausstoß an qualitativ hochwertigen Inhalten in kürzerer Zeit.“*

Wiederver-wendung als Prozess

Ein Ansatz, der im Rahmen einer schlanken Dokumentation eine Rolle spielt, ist eine konsequente Wiederverwendung.

Reuse nach Rockley

Nach Rockley können Inhalte entweder automatisch (Systematic Reuse) oder manuell (Opportunistic Reuse) übernommen werden. Für beide Fälle existieren drei Anwendungsszenarien (vgl. [Rockle03,29ff]):

- Locked: Die wiederverwendeten Inhalte können nicht geändert werden.

- Nested: Die Inhalte werden in einem Modul gesammelt und mit Gültigkeiten ausgezeichnet.
- Derivative: Die wiederverwendeten Inhalte können verändert werden, abgeleitete Inhalte bleiben über eine Eltern-Kind-Beziehung mit der Quelle in Verbindung (z.B. Kopiernachweis).

Bei diesen Reuse-Methoden handelt es sich um grundlegende Konzepte. Im Hinblick auf das effektive Management feingranularer Inhaltsbausteine stehen noch präzisere Reuse- Möglichkeiten zur Verfügung. Verschiedene Produktvarianten unterscheiden sich häufig nur in einzelnen Funktionalitäten oder Komponenten, was prinzipiell eine hohe Wiederverwendbarkeit von Modulen gewährleistet. Zusätzlich können aber Module existieren, die sich inhaltlich beispielsweise nur durch einzelne Sätze unterscheiden. Für das Management dieser modularen Content-Varianten sind feingranulare Reuse-Methoden nötig, um die redundante Verwaltung weitgehend gleicher Module zu minimieren.

Reuse nach Ziegler

Eine detaillierte Darstellung von Reuse-Methoden für das modulare Variantenmanagement bietet Ziegler (in [Ziegle05,42ff]). Die verschiedenen Ansätze werden nachfolgend kurz dargestellt.

- **Vererbung**: Für Variantenmodule, die eine konstante Schnittmenge an Elementen aufweisen, kann eine Modulklasse definiert werden. Klasseninformationen können automatisch an Module derselben Klasse vererbt werden (Systematic Reuse). Jedes von dieser Klasse abgeleitete Modul erbt somit automatisch diese Basiselemente als einzelne unveränderbare Wiederverwendungen (Locked Reuse). Abgeleitete Module können durch zusätzliche Inhalte ergänzt werden. Die Vererbungsmethode eignet sich auch zur Vererbung von Strukturvorlagen für Modulklassen.
- **Reuse-Pool**: Eine Methode, die auf der manuellen Wiederverwendung (Opportunistic Reuse) von Elementen basiert, ist die Verwendung eines Reuse-Pools zum Aufbau von Modulen. Der Reuse-Pool enthält hierbei „[…] alle Elemente, die für die Verwendung in verwandten Modulen zur Verfügung stehen sollen“ [Ziegle05]. Die benötigten Elemente werden einzeln selektiert und als „Locked Reuse“ in das Modul eingefügt. Wie bei der Vererbungstechnik kann das Modul auch hier ergänzt werden.
- **Variantensammlung**: Content-Varianten können auch in einem einzigen Modul verwaltet werden (Nested Reuse). Die Elemente werden bereits während der Erfassung durch Gültigkeiten für die unterschiedlichen Varianten gekennzeichnet. Gültigkeiten können sich beispielsweise auf Produktvarianten, Zielgruppen oder Medientypen beziehen. Die automatische Auswertung der Gültigkeit erfolgt erst zum Zeitpunkt der Publikation des Moduls.
- Eine weitere Möglichkeit für das Variantenmanagement bieten **lebende Kopien**. Diese stellen allerdings keinen Reuse im eigentlichen Sinne dar. Von einem Modul wird eine Kopie angelegt, in dem Änderungen unabhängig vom Quellmodul vorgenommen werden können (Derivative Reuse). Das kopierte Modul ist mit einem Kopiernachweis versehen, wodurch sich die Kopierquelle feststellen lässt. Änderungen an der Kopierquelle können so manuell im kopierten Modul nachgeführt werden. Der Änderungsdienst kann aktiv durch Meldungen unterstützt werden, die am kopierten Modul automatisch auf Änderungen am Quellmodul hinweisen und den Abgleich der Module ermöglichen.

- Die Kombination flexibler Eingriffsmöglichkeiten (lebende Kopie) und kontrollierter Reuses ist durch den so genannten „**Fragmentierten Reuse**“ möglich. Das wiederverwendete Modul enthält weitere Elemente, ist also fragmentiert. Die einzelnen Elemente können bei Bedarf gelöscht oder angepasst werden. Nur unveränderte Elemente werden weiterhin als Reuses verwaltet.

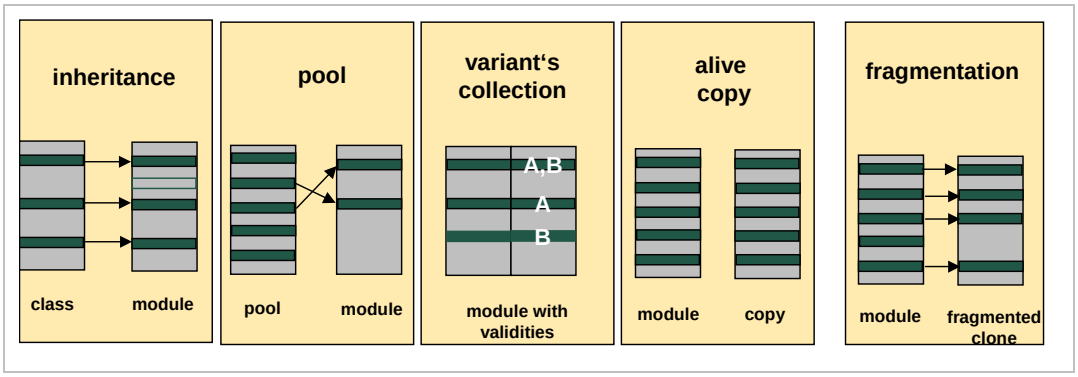


Abb. 18 Reuse nach Ziegler – Übersicht (aus [SAPSie06])

Wiederverwendung dokumentierter Anforderungen

Die Reuse-Definitionen nach Rockley und Ziegler setzen voraus, dass bereits eine modularisierte Art und Weise stattfindet, wie Dokumentation umgesetzt wird. Dies ist entlang eines Produktentwicklungsprozesses nicht durchgängig der Fall und wird in verstärktem Maß dato ausschließlich im Bereich der externen technischen Dokumentation in großem und konsequentem Stil angedacht und umgesetzt. Wir betrachten hier den Fall, dass an einer Stelle im Produktentwicklungsprozess ein internes technisches Anforderungsdokument erstellt wurde, das für nachfolgende Prozesse zur Verfügung steht.
Bei der Analyse, inwieweit eine Wiederverwendung hierbei stattfinden kann, sind Dokumentmodularisierungen und dadurch ermöglichte Wiederverwendungen von content i.S. von Rockley/Ziegler ein möglicher Ansatz.

Intelligente Content Syndizierung

Ein anderer Ansatz kann sein, die Dokument-Erstellungs-und Weiterverarbeitungs-Prozesse genauer zu beschreiben. Es ist dabei denkbar, dass Prozesscharakteristiken semantisch beschrieben werden wie zum Beispiel das Skill-Profil der Personen, die diesen Prozess ausführen können oder die Art der angestrebten Veränderung, die mit dem Dokument bzw. einem kleinen Bestandteil des Dokuments erfolgen soll. Durch eine solche differenzierte Art der Betrachtung der Prozesse lassen sich für das Document Engineering ganz neue Arten einer möglichen Content Automatisierung ableiten: es entsteht die formale Möglichkeit, über einen Pull-Mechanismus und über regelbasierte Prozess- und Skilldefinitionen eine Art Marktplatz anzusprechen mit exakt der Anforderung nach der Anreicherung einer vorhandenen Dateninstanz, die aktuell gebraucht wird. Ein solcher Ansatz hat sicherlich seine Schwächen in denjenigen Bereichen der externen technischen Dokumentation, die de facto bei Auslieferung eines Produkts produktbegleitend geliefert werden müssen, sie werden sich wahrscheinlich mehr im Bereich von derzeit oft ausgelagerten externen technischen Dokumentationsprozessen eignen, die losgelöst vom eigentlichen Produkt oft als eigene Produkte erstellt und verkauft werden. Ein Beispiel hierfür wäre die Erstellung von Trainingsunterlagen, die im Zusammenhang mit Schulungsmaßnahmen als Produkt zum Produkt verkauft werden. Im Rahmen dieser Untersuchung können wir aus zeitlichen Gründen nicht näher auf diesen Ansatz eingehen, möchten aber darauf hinweisen, dass

dieser Ansatz in der laufenden Dissertation ‚Ontology-based Modeling and Re-Use of Technical Documentation' (Tanja Sieber, University of Miskolc) genauer analysiert und vorgestellt wird.

Reuse von Wissen

Ebenfalls ein weiterer Ansatz, der bei zu starrem Festhalten an den Dokumenten oft verloren geht, ist die Frage nach dem Wissen, das bei den Personen vorhanden ist, die das betreffende Dokument erstellen. Inwieweit ist hier das Wissen in geeigneter Form verdatet? Gibt es evtl. dort vorhandenes Wissen, was durch ein Nicht-Verdaten nachfolgend nicht ausgeschöpft werden kann?

3.7 Enterprise Content Integration

Nach Definition des Begriffs ‚Content' und der Untersuchung, wie Artefakte im Software Engineering und Document Engineering wiederverwendet werden können, wird nun der Begriff der Enterprise Content Integration definiert.

Def. 8 Enterprise Content Integration

Unter ‚Enterprise Content Integration' verstehen wir die Nutzung, Wiederverwendung und Integration von jeglichem existierenden Content in einem Unternehmen.

Dies beinhaltet sowohl Anforderungen, Use Cases, vorhandene Dokumente in jeglicher Phase eines Entwicklungsprozesses, technische Dokumentationen jeder Art aber auch Quellcode, welcher implementiert wurde und im Unternehmen eingesetzt wird.
Die Integration von Content in einem Unternehmen kann beispielsweise durch die Nutzung eines Content Management Systems sichergestellt werden. Dort werden die jeweiligen Dateninstanzen über Metadateninstanzen beschrieben

4 Beispiele für Use-Cases

Ausgehend von drei Beispielen für konkret vorkommende Use Cases aus der Industrie wurde hier analysiert, wie in konkreten exemplarischen Fällen Use Cases dargestellt werden und welche Art von Unterstützung und Vorgaben für die Erstellung der Use Cases dem Requirements Engineer an die Hand gegeben wird. Parallel dazu erfolgte eine Analyse, wo wir Potential einer Integration des content sehen, der im Zusammenhang mit der Darstellung von Use Cases anfällt, um daraus abzuleiten, wie im Idealfall welche Art von Semantik erhalten bleiben sollte, um welche Integrationseffekte ausnutzen zu können. Aus Gründen der unterzeichneten NDA-Vereinbarungen mit den Partnerunternehmen können die Analyseergebnisse hier nur abstrahiert dargestellt werden. Eine ausführlichere Version dieses Kapitels kann bei den Autoren angefragt werden.

4.1 Zusammenfassung der Analyse

Aus den untersuchten Beispielen lassen sich einige interessante Beobachtungen zusammenfassend notieren:

- In keiner der betrachteten Fälle wurde die Verdatung des Wissens und der Wissensaustausch betrachtet, der beim Austausch über Dokumente stattfinden soll. Es findet kein Bezug auf ein existierendes Datenmodell bzw. einem Verständnis der Austausch-Aktion als Sender-Empfänger-Modell statt. Dies zeigt sich zum einen in den Dokumenten selbst und zum anderen in der Methodologie, die zukünftigen UseCases als Grundlage dienen soll.
- Allen Beispielen gemein ist daher ein vorausprogrammierter Semantikbruch. Dies wird in einem Fall ansatzweise angegangen, indem für die UseCase-Daten eine Datenmatrix erstellt werden soll. Als UseCase-Daten werden hierbei jedoch nur die verwendeten Objekte betrachtet, für die verwendeten Subjekte und Verben aus den Beschreibungen erfolgt hier kein Abgleich.
- Das Erfassen der UseCases erfolgt in den beiden anderen betrachteten Fällen über tabellarische Templates, die jedoch durch mangelnde linguistische Modellierung über keinerlei weiter zu verwendende Auszeichnungselemente verfügen.In beiden Fällen können wir nicht beurteilen, wie die Abläufe im reellen Alltag ablaufen und ob es hier noch zusätzliche Spezifikationen oder Methodologien gibt, die uns nicht vorliegen.

Zur weiteren Vorgehensweise ziehen wir dementsprechend aus den betrachteten Beispielen ein abstrahiertes Beispiel eines UseCases heran.
Als konkreten Use-Case führen wir' den Use Case ‚Kreditkarte sperren' ein. Diesen sowie die Zielgruppe stellen wir im Weiteren kurz vor und werden diesen Use Case dann zur Evaluation der Modellierungsmethodologien und Ontologie-Methodologien heranziehen.

4.2 Beispiel Use Case ‚Kreditkarte sperren'

Sender: Requirements Engineer
Zielgruppe / Empfänger: Software-Entwickler und technischer Redakteur. Zielgruppe des technischen Redakteurs ist einerseits der Bankberater und andererseits der Endkunde.

Voraussetzungen für den Endkunden: Verlust oder Diebstahl der Kreditkarte
Voraussetzungen für den Bankberater: Anruf des Kunden

- Anruf des Kunden bei der Bank
- Authentifizierung des Kunden (eigener Use Case)
- Bankberater sperrt Karte
- Entweder:
    - Kunde fordert neue Karte an. Bank sendet Kunde neue Kreditkarte zu. Kunde aktiviert Kreditkarte (eigener Use Case)
- Oder: Bank wartet 2 Wochen ab. Erneute Nachfrage beim Kunden.
    - Danach: Kunde macht Sperrung der Kreditkarte rückgängig.
    - Kunde fordert neue Karte an. Bank sendet Kunde neue Kreditkarte zu. Kunde aktiviert Kredikarte (eigener Use Case)

5 Grundlagen der Kommunikation

Content wird ja normalerweise nicht grundlos, sondern unter dem Vorsatz, anderen Personen etwas mitzuteilen, erstellt. Daher betrachten wir im Folgenden nun die Grundlagen der Kommunikationstheorie, evaluieren verschiedene dafür existierende Modelle und entwickeln ein eigenes (semantisches) Kommunikationsmodell, welches uns im weiteren Verlauf bei der Evaluation von Methodologien helfen wird.

5.1 Basiskonzepte der Kommunikationstheorie

Aristoteles

Die Rhetorik (griechischer Titel: τέχνη ρητορική, téchne rhetoriké) des Philosophen Aristoteles (384-322 v. Chr.) enthält eine systematische Abhandlung der Kunst, durch Rede zu überzeugen.

Zitat Z21: [Underw03]

"Like most of the other models in this section of simple models, the model proposed by Aristotle is a linear one. In his Rhetoric*, Aristotle tells us that we must consider three elements in communication:*

the speaker

the speech

the audience"

Hierbei wird als ein Kommunikationsmodell ein lineares Modell vorgestellt, das aus 3 Elementen besteht: dem Sprecher, der Rede und dem Zuhörer bzw. der Zuhörerschaft. Aristoteles befasst sich in der Rhetorik mit den verschiedenen Gattungen und Techniken von Reden und nimmt unter anderem auch Bezug auf die Rolle des Redestils.

Zitat Z22: [Honeyc07], Book III, Chapter 1
"Our next subject will be the style of expression. For it is not enough to know what we ought to say; we must also say it as we ought; much help is thus afforded towards producing the right impression of a speech. The first question to receive attention was naturally the one that comes first naturally -- how persuasion can be produced from the facts themselves. The second is how to set these facts out in language. A third would be the proper method of delivery; this is a thing that affects the success of a speech greatly; but hitherto the subject has been neglected."

Ein wichtiger Bestandteil innerhalb einer kommunikativen Situation sieht Aristoteles also darin, dass der Redner wissen muss, WIE er etwas sagen muss und nicht nur, WAS er sagen soll.

Lasswell

Die Lasswell-Formel ist ein vom US-amerikanischen Politik- und Kommunikationswissenschaftler Harold Dwight Lasswell 1948 in einer griffigen Formel festgehaltenes Modell der Massenkommunikation: wer sagt was durch welches Kommunikations-Mittel zu wem mit welcher Wirkung. Diese Formel wird oft ergänzt durch die fehlenden Fragen nach dem Grund bzw. der Ursache für eine Aussage (warum?) und nach dem 'wie' der Aussage.

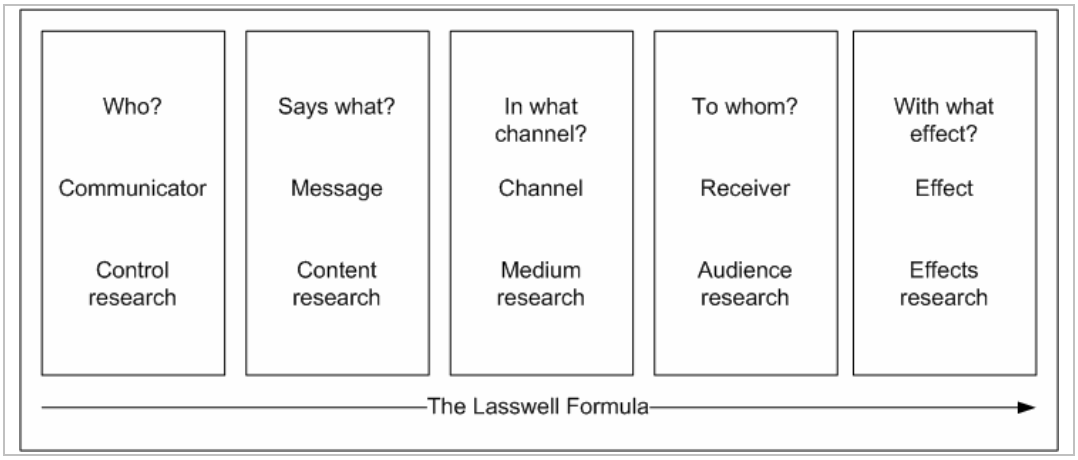


Abb. 19 Laswell-Formel (analog zu [Underw03])

Gerbner

Das Kommunikationsmodell von Gerbner ist auf eine Massenkommunikation hin ausgerichtet. Wir finden darin eine horizontale und vertikale Ebene: es steht zunächst das Ereignis (event), das wir mental verarbeiten müssen, erst dann werden wir kommunizieren, d.h. wir machen uns ein mentales Bild des Geschehens (siehe Graphik: E^1 ist nicht das Ereignis selbst, sondern unsere Vorstellung über das Ereignis). Wir kommunizieren also nicht über das Ereignis, sondern über unsere Interpretation des Events. Das, was letztendlich als Botschaft (in der Graphik SE = statement of event) entsteht, ist nur das Ergebnis unseres mentalen Abbildes.

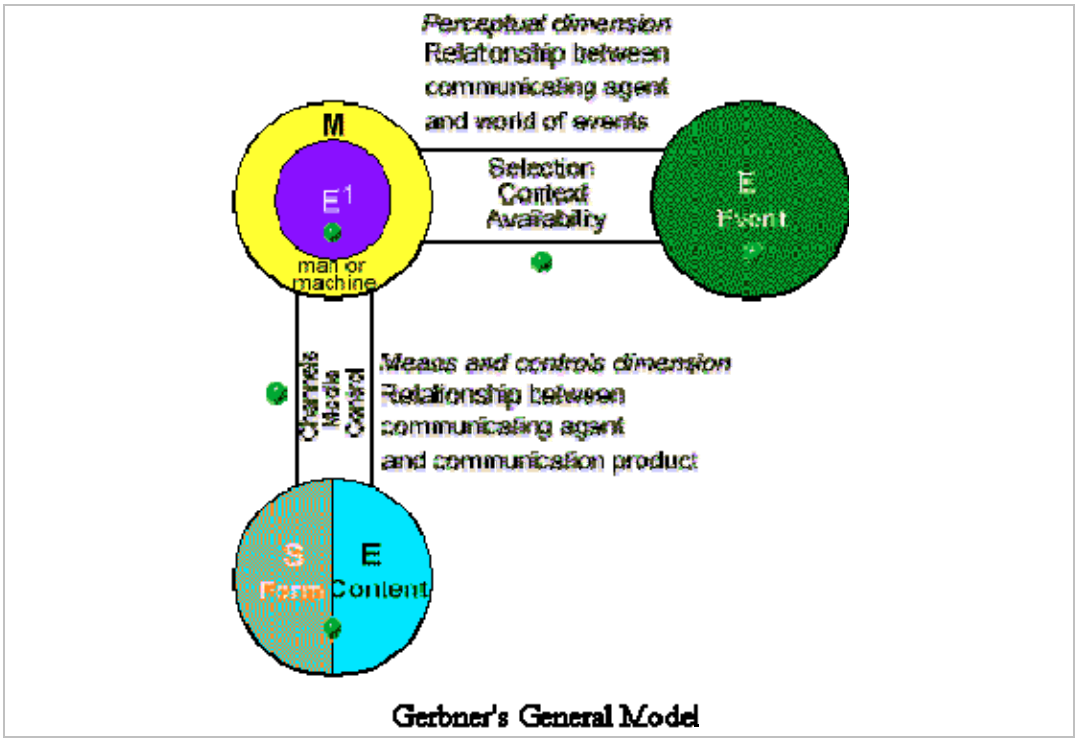


Abb. 20 Kommunikationsmodell nach Gerbner (aus [Underw03])

Der Prozess der Event-Wahrnehmung ist dabei nach Gerbner ein Prozess der aktiven Interpretation des Events: dabei findet eine Selektion statt, das Event wird in einen

Kontext eingebunden und die Art der Wahrnehmung ist zudem abhängig davon, wie viele Events zum Zeitpunkt gleichzeitig verfügbar sind. S steht auch für Signal und kann rein theoretisch auch ohne ein Event präsent sein, allerdings handelt es sich dann um ein Rauschen.

Osgood& Schramm

Zitat Z23: [Underw03]
"The Osgood and Schramm circular model is an attempt to remedy that deficiency: The model emphasizes the circular nature of communication. The participants swap between the rôles of source/encoder and receiver/decoder."

Osgood und Schramm (1954) beschreiben Kommunikation als einen Kreislaufprozeß. Beginn und Ende sind hierbei nicht auszumachen. Als einen weiteren zentralen Bestandteil in diesem Modell rückt auch hier wiederum die Interpretation einer Nachricht in den Vordergrund mit einer Forderung nach einer möglichst effektiven Reduzierung semantisches Rauschens, um eine erfolgreiche Kommunikation durchführen zu können.

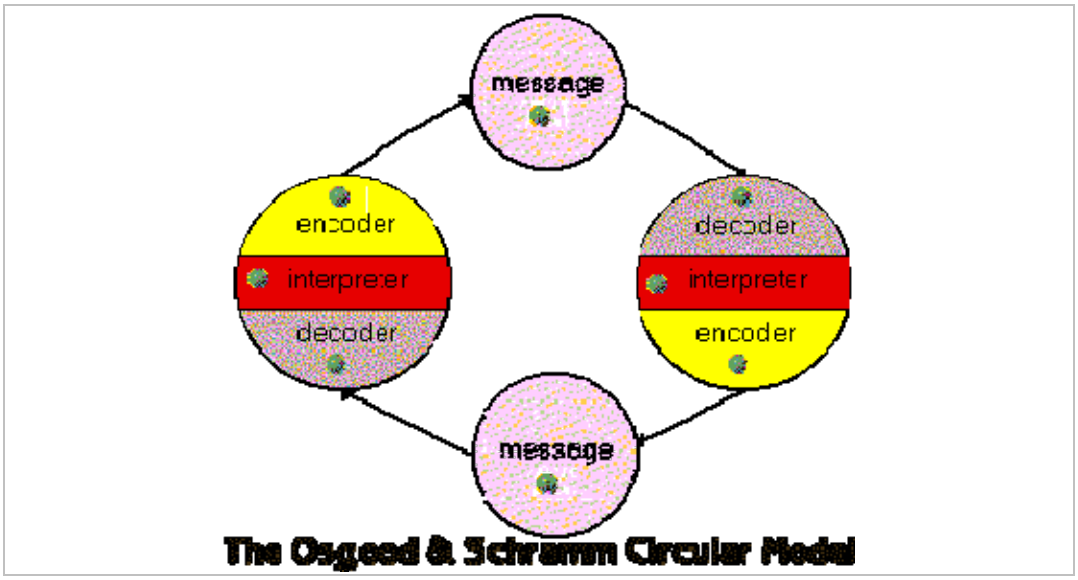


Abb. 21 Kommunikationsmodell nach Osgood und Schramm (aus [Underw03])

Shannon& Weaver

Claude E. Shannon (1949) untersuchte schon in den vierziger Jahren die mathematischen Grundlagen technischer Kommunikationsprozesse, also Telefon, Radio und andere Fernmeldeeinrichtungen. Dazu entwickelte und benutzte er ein Modell allgemeiner Kommunikationssysteme. Das Modell besteht im Wesentlichen aus 5 Einheiten (Siehe Abb. 22):

(1) Eine Informationsquelle. Hierbei ist zu berücksichtigen, dass Shannon ausdrücklich einen rein mathematischen Informationsbegriff verwendet, der, grob gesprochen, ein Maß für die Anzahl der Möglichkeiten ist, aus denen eine Botschaft ausgewählt werden kann. Die Informationsquelle wählt Botschaften aus dieser Menge von möglichen Botschaften aus.

(2) Ein Transmitter oder Sender. Er formt die von der Informationsquelle ausgewählte Botschaft in ein Signal um, das zur Übertragung geeignet ist. Diesen Vorgang nennt man auch Kodierung.

(3) Ein Kanal. Ein (technisches) Medium, das das Signal vom Sender zum Empfänger transportiert. Auf diesen Kanal können Störquellen einwirken, die Rauschen verursachen.

(4) Ein Empfänger. Er ist das Gegenstück zum Sender und führt Umformungen durch, die invers zu denen des Senders sind, um so aus dem Signal die ursprüngliche Botschaft zurück zu gewinnen. Dies nennt man auch Dekodierung.

(5) Ein Ziel. Die Person (oder das Ding) für die (das) die Botschaft bestimmt ist.

Shannon wollte dieses Modell rein technisch verstanden wissen und nicht als Modell für menschliche Kommunikation.

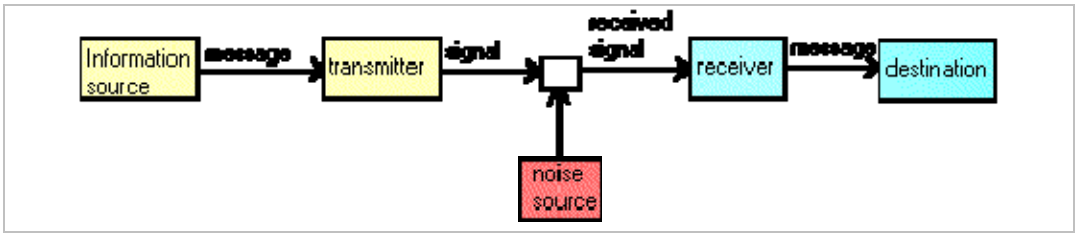


Abb. 22 Kommunikationsmodell Shannon&Weaver (aus [Underw03])

Warren Weaver weitete den Anwendungsbereich des Modells stark aus:

Zitat Z24: Shannon, Weaver 1949, S. 95, zitiert aus [Underw03]
"The word communication will be used here in a very broad sense to include all of the procedures by which one mind may affect another. ´[...]. In some connections it may be desirable to use a still broader definition of communication, namely, one which would include the procedures by means of which one mechanism [...] affects another mechanism [...]"

Zitat Z25: Shannon, Weaver 1949, S.98f, aus [Underw03]
"In oral speech, the information source is the brain, the transmitter is the voice mechanism producing the varying sound pressure (the signal) which is transmitted through the air (the channel). [...] When I talk to you, my brain is the information source, yours the destination;"

Demnach ist die Kommunikation ein linearer Prozeß, in dessen Mittelpunkt das Signal steht. Das Prinzip dieses Shannon & Weaver-Modells ist, daß jede menschliche Kommunikation eine Quelle (information source) hat. Diese Quelle ist der Sender, der seine Nachricht (message) in Form eines Codes über einen Kanal (transmitter) weitergibt.

Berlo

Zitat Z26: [Underw03]
“Berlo's approach is rather different from what seems to be suggested by the more straightforward transmission models in that he places great emphasis on dyadic communication, therefore stressing the role of the relationship between the source and the receiver as an important variable in the communication process.“

Berlo entwickelt ein dyadisches Kommunikationsmodell, in dem er einen Fokus darauf legt, dass die Beziehung zwischen Sender und Empfänger eine zentrale Rolle bei einer stattfindenden Kommunikation spielt. Er defniert jeweils 5 Elemente, die innerhalb der Quelle/Kodieren und innerhalb des Empfängers/Dekodieren die Wiedergabetreue

beeinflussen (siehe Abb. 23): Die Kommunikationsfähigkeiten, das Wissen, das Sozialsystem, die Kultur und Einstellungen.

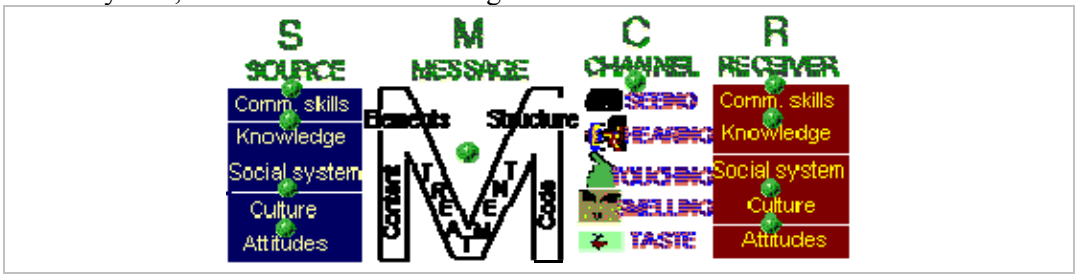


Abb. 23 Kommunikationsmodell nach Berlo (aus [Underw03])

Die Quelle/der Sender wählt bestimmte Elemente aus einem gegebenen Code und möglichem Content-Vorrat (Code und Content verfügen jeweils über spezifische Elemente and Strukturen) aus und präsentiert diese in der einen oder anderen ‚Behandlung' (treatment). Wenn ein Empfänger eine Nachricht entschlüsselt, schätzt er den Sender je nach Art der vom Sender gewählten ‚Behandlung' bzgl. dessen kommunikativen Fähigkeiten etc. ein.
Nach Berlo steckt dementsprechend Bedeutung in den Menschen. Kommunikation ist als das Übermitteln von Nachrichten zu verstehen, die ihrerseits keine Bedeutung beinhalten, sondern nur in der Interpretation in den Nachrichten-Benutzern entstehen.

Organon-modell von Bühler

Karl Bühler [Bühler69] beschreibt in seinem Organonmodell (von griech. für Hilfsmittel), dass jeder Sender (analog zu Shannon & Weaver) zunächst Sinnesreize aus seiner Umgebung (Gegenstände oder Sachverhalte) erhält. Diese kombiniert er daraufhin mit seinen subjektiven Eindrücken (dabei werden bestimmte Teile herausgefiltert und andere subjektive Wertungen hinzugenommen) und gibt dieses Konstrukt an den Empfänger weiter. Tut dieser dasselbe, entsteht ein Kommunikationsprozess. Die Kommunikation funktioniert dabei nach Bühler lediglich über Zeichen. Das sprachliche Zeichen soll dabei allerdings den Kausalzusammenhang zwischen dem Sachverhalt und dem Sprechen beibehalten. Bühler hat dieses Grundmodell weiterentwickelt, indem er neben den Akteuren auch die Übertragungsmodi beschrieben hat, wobei die Sprache darüber hinaus noch drei weitere Aspekte berücksichtigen muss: Gegenstände werden durch Sprachzeichen symbolisch dargestellt; Zeichen können auch Emotionen ausdrücken; Sprache berücksichtigt auch Meinungen des Senders (vgl. „Hund" vs. „Köter"). Bühler beschreibt desweiteren zwei wichtige Prinzipien:

- *Prinzip der abstraktiven Relevanz*: Alle diejenigen Eigenschaften, die für das Zeichen als wesentlich, d.h. als relevant für seine Funktion vereinbart worden sind, dabei wird von allen anderen (weil unwesentlichen) Eigenschaften abgesehen (abstrahiert). Beispiel: Bei einer Ampelanlage sind die wesentlichen Zeichen die Farben "rot", "gelb" und "grün".
- *Prinzip der apperzeptiven Ergänzung*: In einer konkreten Situation stellen wir uns keine abstrakte Ampelanlage vor, sondern ergänzen das Zeichen nach den uns wichtigen Merkmalen.

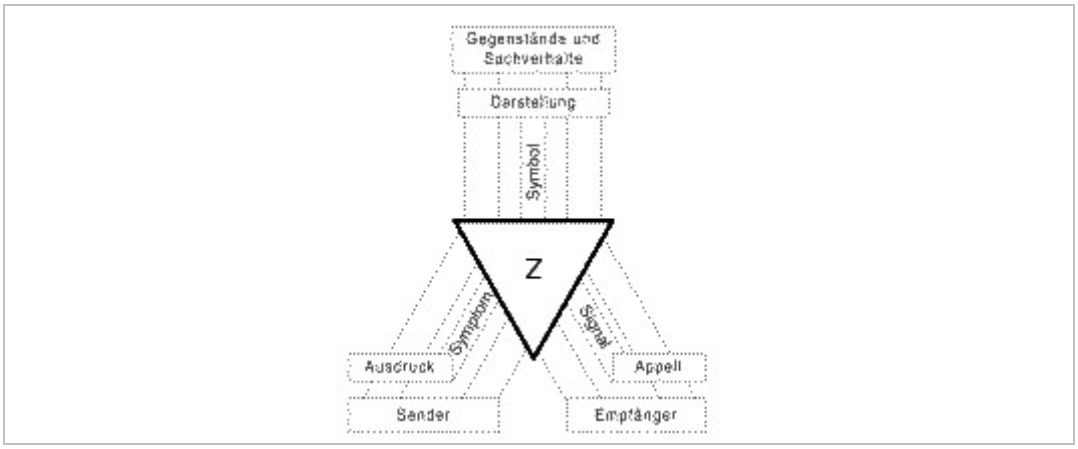


Abb. 24 Organonmodell nach Bühler

Per Flensburg

Per Flensburg hat das Kommunikationsmodel von Shannon und Weaver (siehe Abb. 25) noch erweitert [Flensb07]. Er definiert dazu Daten als Zeichenstrom in einem definierten Format und nennt Informationen als Kombination von Daten mit Struktur. Er bezeichnet desweiteren Content als Sammlung von Information und Metadaten und schließlich Wissen als Kombination aus Content und dem dazugehörigen Kontext.

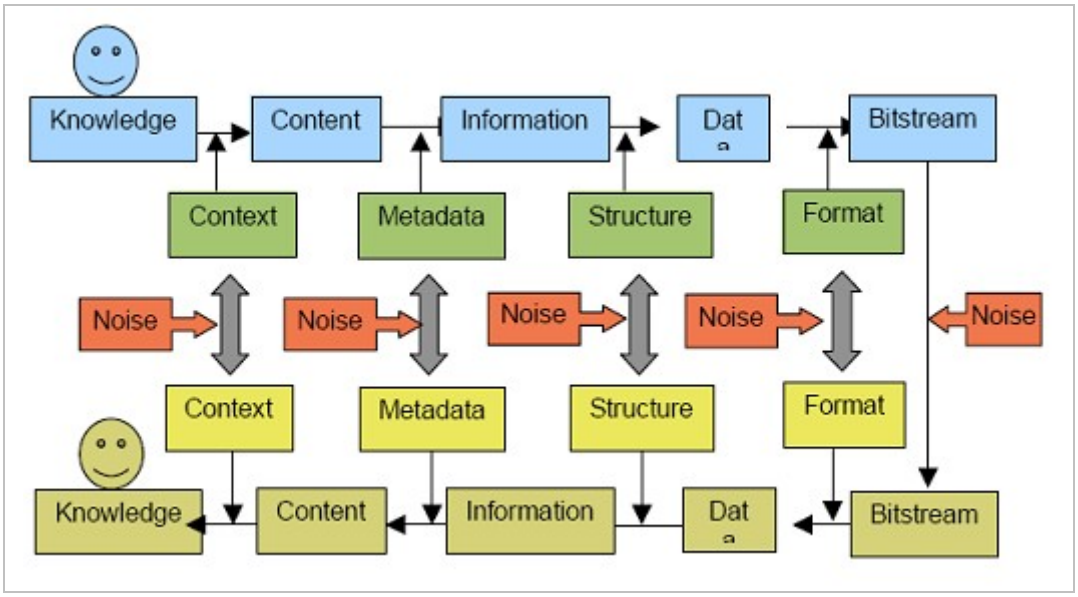


Abb. 25 Begriffsmodell von Flensburg (aus [Flensb07])

Seiner Meinung nach liegt der Schlüssel zu Verständnis darin, genügend Kontext mit anderen zu teilen, damit diese auch dasselbe Verständnis und Wissen erreichen können. Als einzige Möglichkeit dafür sieht er im Konzept des Storytelling. Um beispielsweise einen Geschäftsprozess zu beschreiben, benötigt man erst jemand, der anschaulich den Rahmen und die Kunden der Firma beschreibt und dann Schritt für Schritt die einzelnen Prozessaktionen ausführlich diskutiert. Dies muss allerdings so geschehen, dass immer das mögliche Zielpublikum vor den Augen des Erzählers steht, da seiner Auffassung nach zwar der Empfänger eine große Rolle einnimmt, aber vor allem der Sender einer

Nachricht (der Erzähler) die Verantwortlichkeit trägt, eine passende Form, Struktur und den richtigen Kontext zu wählen, um die Bedeutung der Nachricht so verständlich wie möglich zu machen.

Dance's schraubenförmige Spirale

Im Gegensatz zu bisherigen Kommunikationsmodellen, die lediglich zweidimensional (meist linear, manchmal kreisförmig) beschrieben wurden, steht Dance's schraubenförmige Spirale. Frank Dance [DanLar76] beschreibt die Kommunikation zwischen zwei Parteien als sich stetig ändernde Beziehung. Ein einmal gesprochener Satz verändert die bisherige Kommunikation zwischen zwei Personen und die darauf gegebene Antwort bringt die Spirale wieder einen Schritt weiter. So ist keine Rückkehr zu einem früheren Zeitpunkt möglich. Der Kommunikationsprozess bewegt sich so wie die Spirale immer vorwärts, beruht aber zu jedem Zeitpunkt auch gleichzeitig auf Ereignissen aus der Vergangenheit, die die Gegenwart und die Zukunft beeinflusst. Hier sind also weniger Aspekte wie Nachricht, Kanal, Sender oder Empfänger beschrieben, sondern vielmehr der zeitliche Ablauf einer Kommunikation auf einem höheren Abstraktionslevel. Somit könnte dieses Modell jederzeit mit anderen Modellen (wie z.B. das Shannon&Weaver-Modell) kombiniert werden.

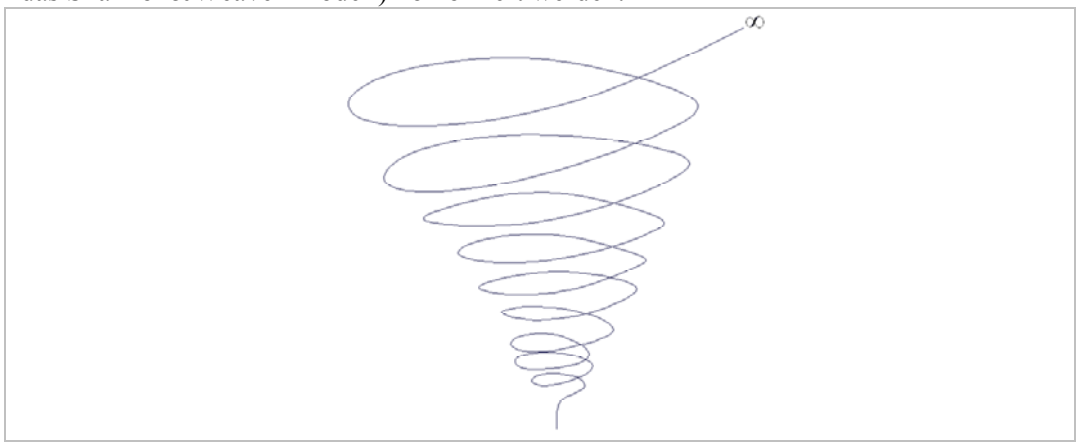


Abb. 26 Spiralmodell von Dance

Beckers Mosaikmodell

Eine dritte Dimension fügt ausserdem Sam Becker in seinem Mosaikmodell hinzu. In *The prospect of rhetoric* [Becker68] präsentiert er ein mosaikförmiges Modell der Kommunikation, welches berücksicht, dass jedes Gespräch durch eine Vielzahl von vorhergeschehenen Ereignissen beeinflusst wird. Jeder Mensch bewegt sich durch die Welt und kann dabei zu einer bestimmten Thematik immer nur gewisse Bruchteile als Wissen aufnehmen. Diese Mosaiksteine vervollständigen sich nach und nach, bis man ein mögliches genaues Bild über das zu erlangene Wissen erhalten hat. Dabei wird es nie möglich sein, alle Mosaiksteine bezüglich einer Thematik zu erreichen. Die einzelnen Mosaikstücke, die man in einem Gespräch anderen mitteilt, müssen aber ausreichen, um dem Gegenüber das Verständnis der Thematik zu ermöglichen.

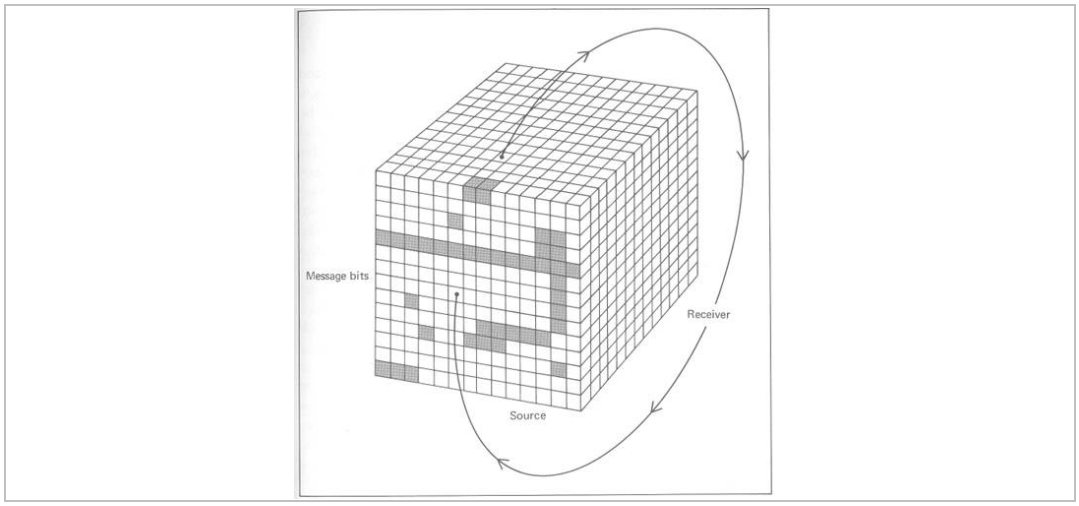


Abb. 27 Mosaikmodell nach Becker (analog zu [Becker68])

Funktionales Modell von Ruesch und Bateson

Als weiteres multidimensionales Modell sei das funktionale Modell von Ruesch und Bateson [RueBat94] genannt. Diese beschreiben Kommunikation als simultanes Ereignis auf vier verschiedenen Stufen: Die unterste (Level 1) ist der intrapersonale Prozess, die nächste passiert interpersonal (und damit auch extrapersonal) und fokusiert die überschneidenen Erfahrungsbereiche zweier Akteure. Level 3 beschreibt eine Gruppeninteraktion, die schließlich auf der höchsten Stufe noch durch ein kulturelles Level als Sammlung von Bewohnern ganzer Landstriche erweitert wird. Auf allen Ebenen passieren vier kommunikative Funktionen: Nachricht senden, Nachricht empfangen, kanalisieren und evaluieren. Dadurch fokusiert das Modell weniger auf die strukturellen Attribute der Kommunikation (Sender, Nachricht, Empfänger), sondern mehr auf die involvierten Personen(gruppen).

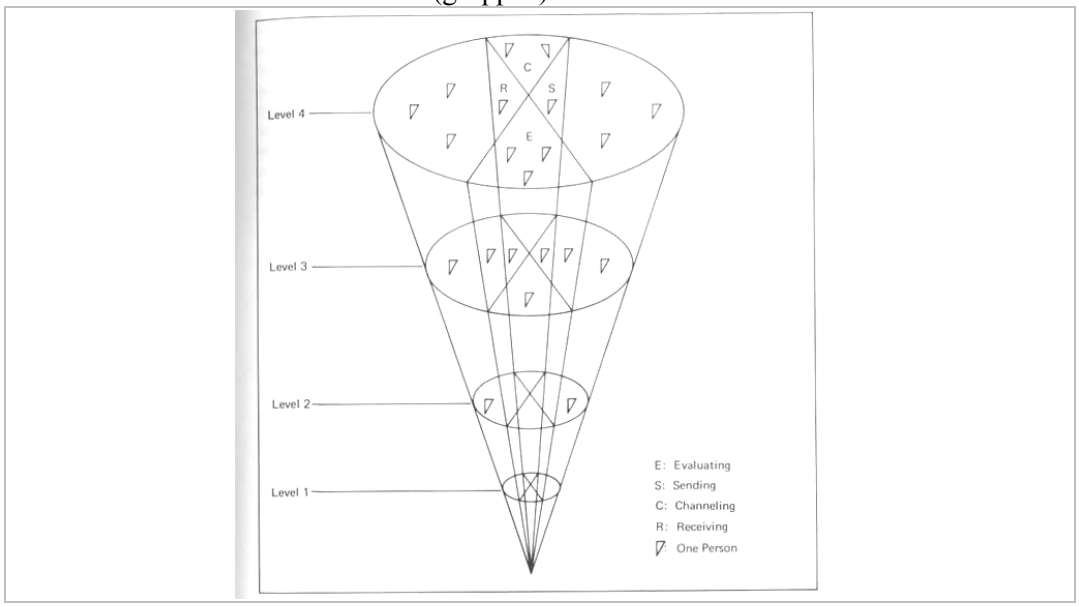


Abb. 28 Modell von Ruesch und Bateson (aus [RueBat94])

5.2 Semantisches Kommunikationsmodell

Hinsichtlich einer Weiterentwicklung des semantischen Datenmodells (Sieber/Kammerer) ausgehend von der Fragestellung „Wie findet im Umfeld der internen technischen Dokumentation Kommunikation statt?“ erweisen sich die vorgestellten Kommunikationsmodelle nach Shannon&Weaver, Berlo, Osgood&Schramm und Per Flensburg als hilfreich.

Nachricht

Wir werden im folgenden bei der Herleitung unseres semantischen Kommunikationsmodells die Begrifflichkeit einer *Nachricht* (engl.: *message*) in Relation zu den im semantischen Datenmodell verwendeten Terminus *Dateninstanz* vorstellen, so dass deutlich wird, dass es im übertragenen Sinne auch bei der technischen Dokumentation letztendlich um eine Nachrichtenübertragung geht, die als Ziel eine Wissenskommunikation zwischen Sender und Empfänger vorsieht.

Rauschen

Wie diese Wissenskommunikation im semantischen Kommunikationsmodell schematisch dargestellt werden kann und an welchen Stellen Probleme bei der Kommunikation auftauchen können, werden wir herleiten. Hierzu werden wir aus den vorgestellten Kommunikationsmodellen nach Shannon&Weaver den Begriff des *Rauschens* (und hierdurch bedingte syntaktische Störungen bei einer Nachrichtenübermittlung) übernehmen und nach Berlo im Sinne eines *semantischen Rauschens* ergänzen.

Interpretation

Dieses semantische Rauschen spielt in unserem Kommunikationsmodell eine Schlüsselrolle, da wir eine Reduktion dieses semantisches Rauschens bei einer Nachrichtenübertragung anstreben werden, um eine möglichst fehlerfreie *Interpretation* (und hier beziehen wir uns auf das Kommunikationsmodell nach Osgood&Schramm) zu ermöglichen.

Im weiteren Verlauf wollen wir diese Problemstellungen an einem konkreten Fallbeispiel einer Anforderungsspezifikation genauer ansehen. Dafür betrachten wir als groben Kontext ein Buchungssystem für Lehrgangs-/Schulungsräume. Bei der Anforderung soll es um das Löschen einer periodischen Buchung gehen. Die Detailbeschreibung sieht wie folgt aus:

Fallbeispiel: Löschen einer periodischen Buchung

> Zu einer periodischen Buchung können einzelne Termine gelöscht werden. Sollte jedoch nur noch eine einzelne Buchung existieren, wird diese automatisch in eine Einzelbuchung umgewandelt. Ausserdem ist selbstverständlich das Löschen der kompletten periodischen Buchung denkbar, dann werden alle existierenden Buchungen, die zu dieser periodischen Buchung gehören, aus dem System entfernt.

Bezugnehmend auf die vorgestellten Kommunikationsmodelle (und das zu entwickelnde semantische Kommunikationsmodell) gibt es hier einen Sender (in unserem Fall der Ersteller der Anforderungen, also meist der Kunde) sowie einen oder mehrere Empfänger (diese werden im Weiteren genauer beleuchtet). Die zu übertragende Anforderung ist „Löschen der periodischen Buchung“.

Interessant ist natürlich, für welche Zielgruppen diese Anforderung beschrieben wird. So interessieren einen Software-Entwickler sicher andere Aspekte als den leitenden Projektmanager. Der Software-Entwickler muss beispielsweise noch folgende Anforderungsspezifikationen zusätzlich erhalten, um die oben genannte Anforderung auch vollständig implementieren zu können:

Eine periodische Buchung besteht aus mindestens einer Buchung. Jede Buchung findet an einem bestimmten Termin statt, hat einen Titel, bezieht sich auf einen Raum sowie besitzt ggf. einen Kommentar.

Der Termin besteht aus einem Datum, einer Anfangszeit sowie einer Endzeit. Die Zeitabstände sind im Halbstundentakt anzugeben.

Jeder Raum hat eine eigene Nummer, einen Gebäudenamen und ist durch weitere Eigenschaften gekennzeichnet: die Anzahl seiner Plätze, die Verfügbarkeit von Tafel, Beamer, Projektor, Flipchart oder Computer. Es existieren verschiedene Raumarten: Besprechungsraum, Konferenzraum, Plenum oder Seminarraum.

Jegliche Buchung kann nur von Personen vorgenommen werden, die am System eingeloggt sind. Dadurch sind dem System sowohl Benutzerkennung, Passwort als auch Vorname, Nachname, Raum, Gebäude, eMail-Adresse sowie Telefonnummer bekannt. Bestimmte Personen haben zusätzlich Adminrechte und dürfen Buchungen nicht nur anlegen sondern beispielsweise auch löschen.

Eine periodische Buchung kann verschiedene Wiederholungsintervalle besitzen: täglich, wöchentlich, monatlich oder jährlich. Zusätzlich sind auch mehrere Tage je Woche möglich (jeden Montag und jeden Mittwoch von 10:00 – 12:00 Uhr).

Hingegen ist es für einen Projekt Manager sicher nicht wichtig, dass eine periodische Buchung automatisch in eine Einzelbuchung umgewandelt werden muss, wenn nur noch ein Termin dieser periodischen Buchung existiert. Des weiteren ist für ihn vermutlich auch nicht von Interesse, dass die Räume im Halbstundenrhythmus buchbar sind. Für einen Software-Entwickler hingegen sind dies ganz entscheidende Kriterien, die bei der Implementierung berücksichtigt werden müssen. Genauso sind beliebige weitere Rollen denkbar, die ebenfalls einen unterschiedlichen Detailgrad der Anforderungen benötigen.

Erstellen einer Anforderungs spezifikation

Allgemein lässt sich feststellen, dass beim Erstellen einer Anforderungsspezifikation im Vorfeld die folgenden Fragen abgeklärt werden müssen:

- Wer ist der zukünftige Leser der Spezifikation?
- Welche Handlungen sollen auf Grundlage der Anforderungsspezifikation ermöglicht werden?
- Wie granular müssen die Anforderungen dazu dargestellt werden?
- Welches Wissen braucht der Anwender maximal, um die gewünschten Handlungen ausführen?
- Welches Wissen kann man dabei voraussetzen?
- Wie kann das zu vermittelnde Wissen – das dem Delta zwischen angenommenen vorausgesetzten und dem zu vermittelndem Maximal-Wissen entspricht – geeignet versprachlicht werden, so dass es dazu beiträgt, zu einer bestimmten gewünschten Handlung zu führen?

Wissen kommuni-zieren

Bei der Suche nach entsprechenden Antworten tritt unter der besonderen Berücksichtigung der zur Verfügung stehenden Kommunikationskanäle wieder die „Verdatung“ von Wissen in den Vordergrund, die dem „Wissenstransfer“ natürliche Grenzen setzt:

- Wissen muss verdatet werden, ehe es kommuniziert werden kann.

- Es stehen ausschließlich Dateninstanzen auf der Objekt- und Metaebene zur Verfügung, um Wissen zu kommunizieren.

Dies wird deutlich, wenn man sich anhand des Sender-Empfänger-Grundmodells klarmacht, dass das einzige, das von einem Sender produziert und von einem Empfänger wahrgenommen werden kann, Dateninstanzen sind.

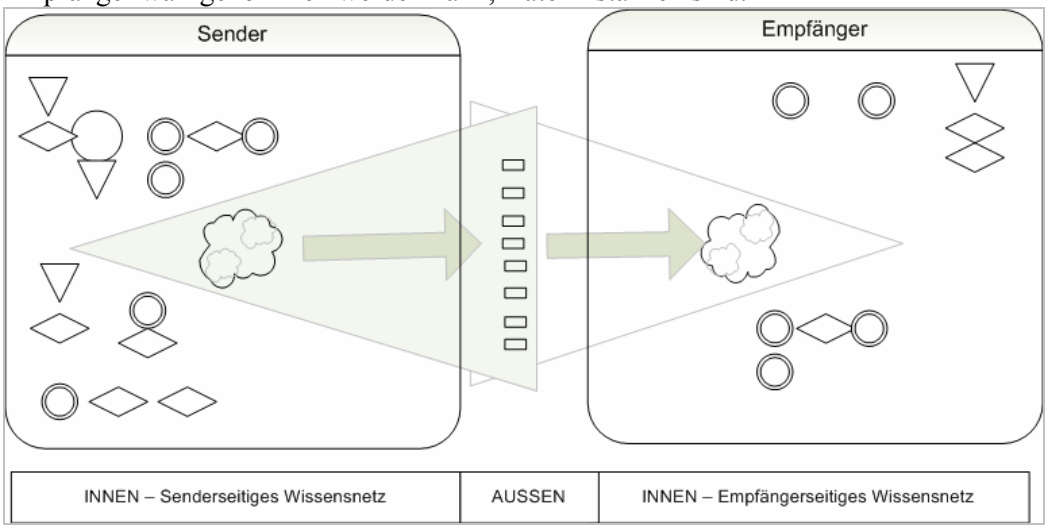


Abb. 29 Sender-Empfänger-Grundmodell

Verdatung von Wissen

Bei der Verdatung von Wissen geht es in der technischen Dokumentation darum (vgl. [SieKam07]),

- wie die Explizierung des vorhandenen Wissens bei der Verdatung möglich ist (siehe Abb. 30),
- welcher Wissensstand beim Empfänger vorausgesetzt wird und
- welcher Wissensbaustein übermittelt werden soll (vgl. Abb. 31).

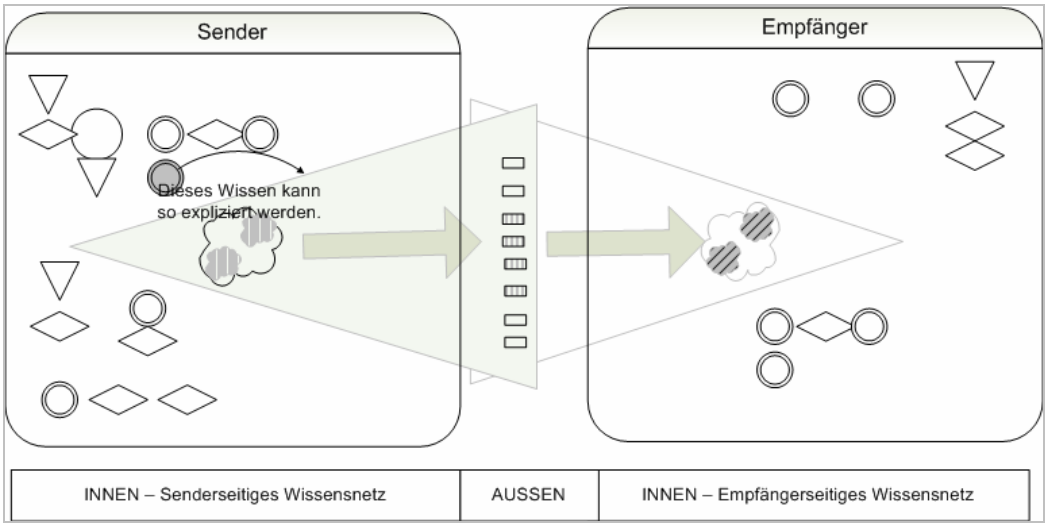


Abb. 30 Sender-Empfänger-Grundmodell mit möglicher Explizierung von Wissen

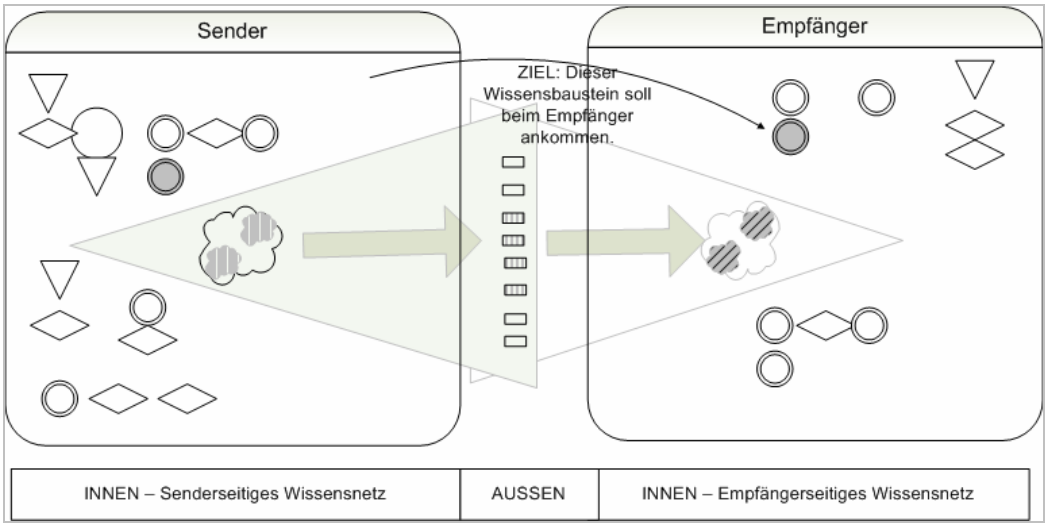


Abb. 31 Zielsetzung bei der Wissensverdatung in der technischen Dokumentation

Probleme treten hier zum einen durch die Tatsache auf, dass senderseitig eine Art vernetzte Wissensstruktur vorhanden ist, die sich wesentlich von der auf der Empfängerseite unterscheiden kann und wird. Ein Sender kann dementsprechend nur durch Annahmen und Vermutungen an das empfängerseitig vorhandene Wissensnetz anknüpfen und auf dieser Grundlage festlegen, welcher Wissensbaustein wahrscheinlich angebracht ist. Zum anderen ist empfängerseitig nicht garantiert, dass aufgrund der gewählten Verdatung und der daraus resultierenden Dateninstanzen auch tatsächlich das Datum abstrahiert werden kann, welches senderseitig konkretisiert wurde (siehe Abb. 32) und ob ein und – wenn ja – welcher Wissensbaustein aus dem abstrahierten Datum erzeugt wurde (siehe Abb. 33).

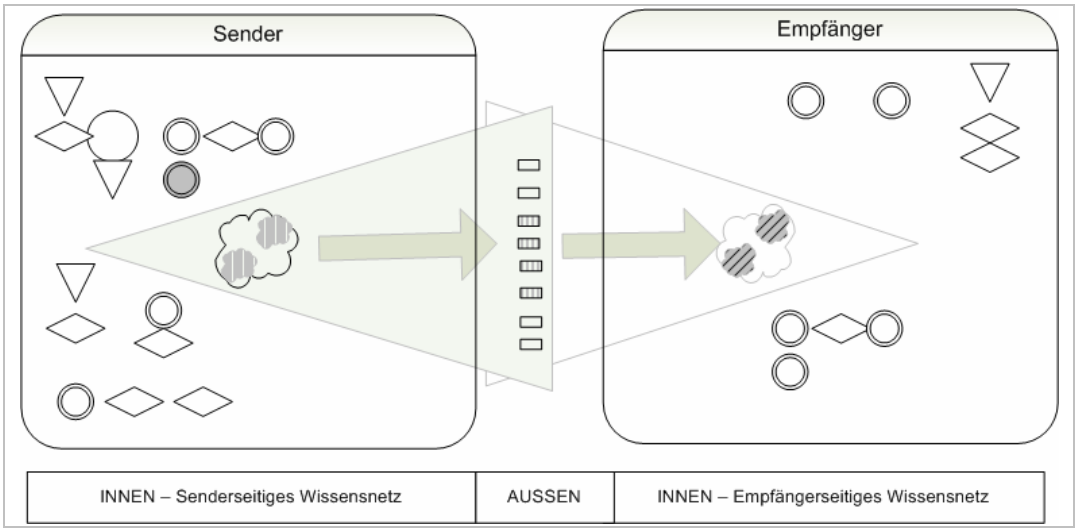


Abb. 32 Effekt bei der Wissensverdatung in der technischen Dokumentation: anderes Datum beim Empfänger konkretisiert

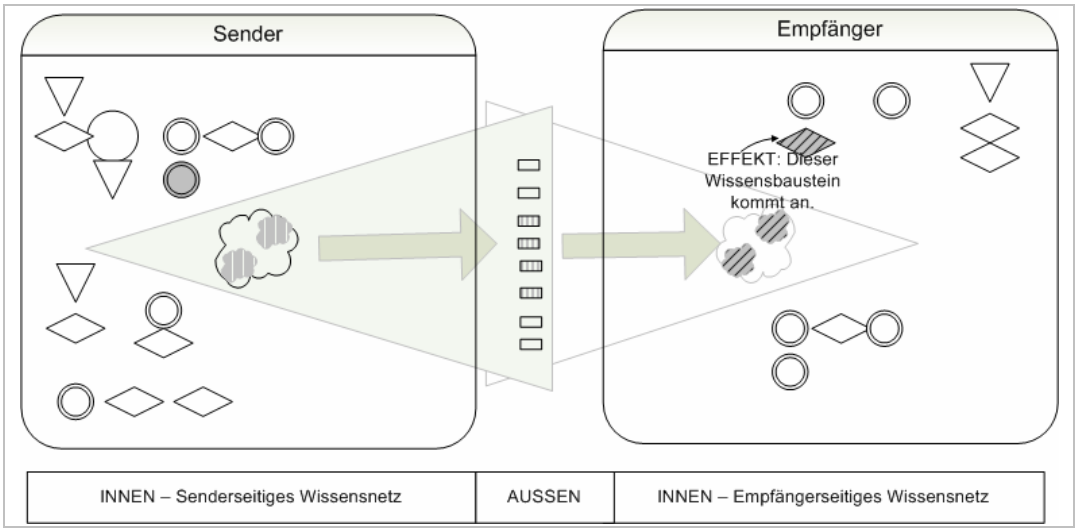


Abb. 33 Effekt bei der Wissensverdatung in der technischen Dokumentation: anderer Wissensbaustein aus Datum abgeleitet.

Folgt man dem semantischen Datenmodell, ist das, was übermittelt werden kann, ohne dass eine Interpretation eines Empfängers stattfinden muss, die Ebene der Dateninstanzen. Können diese korrekt abstrahiert werden, kann man von Daten sprechen. Nicht jedes Datum erscheint einem wahrnehmenden Subjekt aber bereits als Information. Hierzu ist ein Prozess notwendig, der das Datum mit bereits vorhandenen Wissenseinheiten in Beziehung setzt.
Vergleicht man dies wiederum mit dem oben entwickelten Sender-Empfänger-Modell (vgl. Abb. 29, Abb. 30 und Abb. 31), stellt Information genau den Wissensbaustein dar, der im technischen Dokumentationsprozess optimalerweise beim Empfänger ankommen soll. Der Empfänger soll aus den übermittelten Dateninstanzen im ersten Schritt die korrekten Daten abstrahieren und aus den Daten einen neuen Wissensbaustein – eine Information – generieren können.
Wir schließen uns hier den in [SieKam07] vorgestellten Lemmata für Wissen und Information an:

Lemma 1 Wissen

- Wissen ist intrapersonal.
- Wissen ist die Basis für Problemlösungen und Handlungen.
- Wissen muss für die Kommunikation verdatet werden.

Lemma 2 Information

- Information ist neues Wissen in Verortung bestehenden Wissens.
- Information ist system-relativ und damit subjektiv.

Die Herausforderung eines technischen Autors im Umfeld der technischen Dokumentation ist dementsprechend die Auswahl und Gestaltung der Dateninstanzen der Art, dass das Ergebnis beim zukünftigen Leser eine solche Veränderung in den Wissensstrukturen bewirkt, die den hypothetischen Leser dazu befähigt, die gewünschte Handlung auszuführen. Dabei muss der technische Autor von einem dynamischen Modell des beim zukünftigen Leser vorhandenen Wissenszustands ausgehen und diesem Modell auf der einzig ihm vorhandenen Ebene der Dateninstanzen begegnen.

Die Dateninstanzen, die der Sender auswählt sind unter anderem abhängig davon, welchen Komplexitätsgrad des Datums er anstrebt, den er konkretisieren wird. Es kann sein, dass an einer bestimmten Stelle im Anforderungsdokument ausschließlich ein atomares Datum in seiner konketisierten Form als atomare Dateninstanz auf einen potentiellen Leser treffen wird. Es kann aber auch sein, dass ein komplexes Datum, das aus verschiedenen atomaren Daten aufgebaut ist, in einer aus atomaren Dateninstanzen aufgebauten konkretisierten Form als komplexe Dateninstanz auf den Leser trifft.
Um die Bedeutung der Daten zu beschreiben, wird zusätzlich zu diesen Möglichkeiten der Komplexitäts-Granularität an einigen Stellen auf die beschreibende Benennung über ein Metadatum zurück gegriffen, das dann auch wiederum zusätzlich in seiner konkretisierten Form als (Meta-)-Dateninstanz auf den Leser trifft. Ein Päckchen, das nun also aus zusammenhängenden Dateninstanzen auf Objekt- und Metaabene besteht, wollen wir in unserem semantischen Kommunikationsmodell als Nachricht bezeichnen (siehe folgende Abbildung).

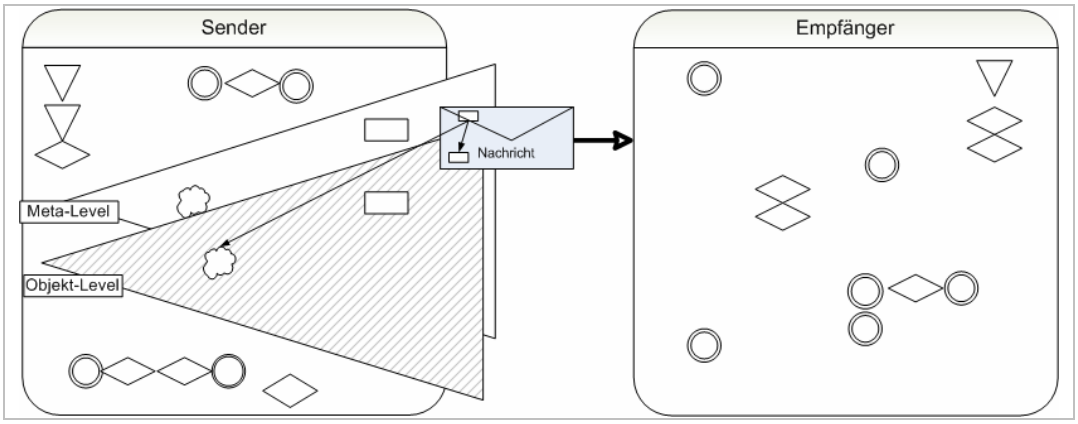


Abb. 34 Zum Begriff der Nachricht im semantischen Kommunikationsmodell: Atomare Dateninstanz mit beschreibender Benennung des atomaren Datums über atomare Metadateninstanz

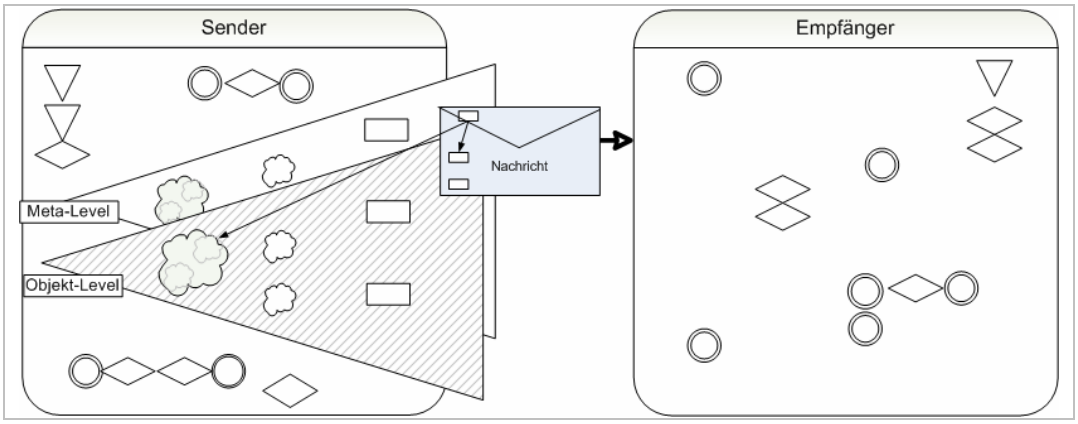


Abb. 35 Zum Begriff der Nachricht im semantischen Kommunikationsmodell: Komplexe Dateninstanzen mit beschreibender Benennung des komplexen Datums über atomare Metadateninstanz

Nach unserem Verständnis von Kommunikation im Umfeld der technischen Dokumentation kann dementsprechend in Anlehnung des semantischen

Kommunikationsmodells sowohl der Prozess der Wissenskommunikation verdeutlicht werden als auch die Vielschichtigkeit und Komplexität der daraus entstehenden ‚Daten-Ebenen'. Wir wollen im folgenden untersuchen, auf welcher Basis/auf Grundlage welcher Modelle heute im Bereich der technischen Dokumentation eingesetzten Modellierungsmethodologien aufgesetzt sind und welche Bereiche in Relation zu unserem semantischen Kommunikationsmodell dadurch, was eine erfolgreiche Datenmodellierung angeht, abgedeckt werden.

5.3 Definition Methodologien

Um die Kommunikation zwischen Sender und Empfänger sicherzustellen, sollte es eine vordefinierte Herangehensweise geben, wie die Nachrichten aufgebaut sein müssen, die der Sender dem Empfänger schickt.
Es existieren vielfache Begrifflichkeiten um die Herangehensweise beim Erstellen eines Modells zu beschreiben: Methodologie, Methode, Technik, Prozess, Aktivität, etc.
Wir wollen im weiteren Verlauf den Begriff der Methodologie verwenden und uns dabei auf die Definition von IEEE beziehen:

> Zitat Z27: IEEE 1990 [IEEE610]
>
> „A methodology is a comprehensive, integrated series of techniques or methods creating a general systems theory of how a class of thought-intensive work ought to be performed."

Dabei ist eine Methode bzw. eine Technik definiert wie folgt:

> Zitat Z28: IEEE 1990 [IEEE610]
>
> „A method is a set of orderly processes or procedures used in the engineering of a product or performing a service. […] A technique is a technical and managerial procedure used to achieve a given objective."

Wir beschreiben im Folgenden die Methodologien, die es im Bereich linguistische Modellierung und zur Erstellung von Ontologien derzeit gibt. Dabei ist interessant, dass es im Bereich der linguistischen Modellierung stärker um die Modelle an sich geht und die Methodologie meist nur implizit beschrieben wird, bei Ontologien sieht es im Gegensatz dazu genau umkehrt aus: Hier werden stärker die Schritte beschrieben, die unternommen werden müssen, um eine Ontologie zu erstellen. Eine genaue Beschreibung aus welchen Konzepten (Modell) diese Ontologie nun bestehen soll, ist hingegen meist gar nicht vorhanden und wird nur in wenigen Werken ansatzweise überhaupt beschrieben.

6 Linguistische Modelle für Dokumentation

6.1 Lineare vs. Modulare Dokumentationserstellung

Lineare Dokumentationserstellung ist charakterisiert durch eine hierarchische und sequentielle Vorgehensweise.
Unter Modularem Schreiben kann man das Aufbauen für sich stehender Inhaltsmodule verstehen, die für verschiedene Dokumentformate und Lesesequenzen eingesetzt werden können und Sinn machen. Dementsprechend gipfeln sämtliche Bestrebungen im Rahmen des modularen Ansatzes bei der Dokumenterstellung darin, die Wiederverwendung von diesen Dokumentationsbausteinen zu erhöhen.Im folgenden werden Methodologien vorgestellt, die Hilfestellungen bieten wollen für eine entsprechend vorzunehmende Modularisierung.

6.2 Ament: Single Sourcing

Zitat Z29: [Ament03]
„Single Sourcing is a documentation method that enables you to re-use the information you develop.[…] Single Sourcing is a methodology, not a technology."

Ament

Nach Ament schlägt sich die modulare Dokumentationserstellung in drei Hauptbereichen nieder:

- „Chunking"
- „Labeling" und
- „Linking"

Unter Chunking versteht Ament das Aufteilen von Information basierend auf diversen Informationstypen in alleinstehende Module. Diese Module müssen eindeutig, standardisiert und kontext-unabhängig über ein Label benannt werden. Im letzten Schritt – im Linking – werden die ausgezeichneten Module über Verweise verknüpft.
Er schlägt als Methodologie für einen durchgreifenden Single Sourcing Prozess dementsprechend folgende Reihenfolge vor:

(1) Das Aufbauen modularer Inhaltsbausteine, die genau eine der folgenden Fragen beantwortet: Wer? Was? Wann? Wo? Warum? Wie?
(2) Aggregieren des Dokuments: die alleinstehenden Inhaltsmodule in verschiedene Dokumentformate für unterschiedliche Zielgruppen zu unterschiedlichen Zwecken zusammenstellen. Dieser Schritt ändert nichts am Inhalt der bereits existierenden Inhaltsmodule, sondern beeinflusst lediglich die Reihenfolge und das Ausgabeformat der Module im Dokument.
(3) Verlinken über Verweise.

Als zentrale Bestandteile eines Single-Sourcing Ansatzes sieht Ament weiter das „Re-purposing" und „Re-assembly". Bei beiden gibt es nach Ament Prozesse, die rein maschinen-basiert durchgeführt werden können (Konvertieren in andere Datenformate etc.) und Prozesse, die kognitive Voraussetzungen erfordern und deswegen von Menschen durchgeführt werden müssen (dies fängt bereits beim Erstellen wiederverwertbarer Informationsbausteine an und greift bis hin zu den Entscheidungen darüber, welche Modulreihenfolge für welche Zielgruppe geeignet ist). Ament gibt dazu in seinem Buch eine Schritt-für-Schritt „Anleitung" für das Umsetzen seiner vorgestellten Single Sourcing Strategie.

Gerade im Bereich, wie man nun aber zu den Modulen kommt und wie diese gestaltet sein sollen, wird sehr stark davon ausgegangen, dass dann in den nachfolgenden Verarbeitungsschritten Menschen die Aufgabe übernehmen, die Inhaltsbausteine zusammen zu stellen.

6.3 Strukturelle Textkontrolle: Ein linguistisches Informationsmodell für technische Kommunikation (Ley)

Ley nimmt als Ausgangssituation für die Entwicklung seines *linguistischen Informationsmodells* die Tatsache, dass so genannte Informationsprodukte der Technischen Kommunikation in einem Spannungsfeld zwischen Kundenfreundlichkeit und Kundenzufriedenheit auf der einen und marktwirtschaftlichen Faktoren auf der anderen Seite stehen [Ley06b]. Hinsichtlich einer optimalen Erfüllung beider Aspekte – dem Produktionsaspekt und dem Verstehensaspekt – spielt nach Ley die Informationsstrukturierung die zentrale Rolle.
Als zwei mögliche Strukturierungsprinzipien führt er das Funktionsprinzip und das Inhaltsprinzip an.

Funktions-prinzip

Zitat Z30: [Ley06b]

„Die funktionale Informationsstrukturierung lässt sich am besten an einem kurzen Beispiel erläutern:

(1) Der Videorekorder ist eingeschaltet.

(2) Drücken Sie die Taste REC.

(3) Im Display erscheint das Menü Startzeit.

Die Funktion des zweiten Satzes ist unmissverständlich. Als Autor fordere ich meine Leser dazu auf, eine bestimmte Handlung zu vollziehen, indem ich äußere „Drücken Sie die Taste REC". Bei der Äußerung von Satz (2) handelt es sich also um eine (Handlungs-)Aufforderung. Diese Funktion wird durch die Verwendung des Imperativsatzes unterstrichen. Satz (1) und (3) hingegen sind beides einfache Aussagesätze. Worin aber besteht deren Funktion? Diese ergibt sich aus ihrem Verwendungszusammenhang. Wenn ich Satz (1) beispielsweise in dem oben dargestellten Kontext äußere, so kann ich damit die Voraussetzung nennen, die für den erfolgreichen Vollzug der Handlung in (2) notwendig ist. Analog dazu kann ich das Resultat der Handlung anführen, indem ich äußere „Im Display erscheint das Menü Startzeit". Drehe ich Satz (1) und Satz (3) jedoch um, so verändert sich auch deren Funktion im Verhältnis zu Satz (2). Die Funktion eines Satzes ist also abhängig von seinem Äußerungskontext."

Inhalts-prinzip

Diese einzelnen Dokument-Teile müssen jedoch keineswegs immer einen handlungsanleitenden Charakter haben. Auch beschreibende oder erklärende Teile haben eine Funktion: nämlich die, ETWAS zu beschreiben beziehungsweise zu erklären.

Zitat Z31: [Ley06b]

„Doch wie können wir uns dieses „Etwas" vorstellen? Ich kann zum Beispiel die Funktionsweise einer Maschine, einen Handlungszusammenhang oder die Ursachen für Schäden an einem Produkt erklären. Beispiele für den Inhalt meiner Beschreibung sind der Aufbau einer Maschine, die Eigenschaften einzelner Komponenten oder der Anwendungsbereich einer bestimmten Prüfprozedur. Kurz:

Bei diesem „Etwas“ handelt es sich um die Objekte, auf die sich das Informationsprodukt beziehungsweise Teile davon beziehen.“

Einen Weg, diese Objekte für einen Gegenstandsbereich systematisch zu beschreiben, stellen Ontologien dar. Dabei kann die inhaltliche Strukturierung eines Gegenstandsbereichs mittels Ontologien ihren Ausgangspunkt in allgemeinen Konzepten wie „Abstrakte Objekte“, „Konkrete Objekte“, „Qualitäten“, „Quantitäten“ oder „Situationen“ nehmen. Bei zunehmender Beschreibungstiefe werden die Konzepte immer spezifischer, bis die Domäne schließlich vollständig (beziehungsweise für die beabsichtigten Zwecke ausreichend) erschlossen ist. Das „Etwas“, das in einem Informationsprodukt beschrieben oder erklärt werden soll, lässt sich auf diese Weise systematisch identifizieren.

Textbegriff

In seiner Dissertation geht Ley auch auf den Textbegriff genauer ein und untersucht hierbei sowohl kognitionspsychologische als auch kommunikationsorientierte Ansätze. Demnach sieht er den Textbegriff an ein kommunikatives Handlungsmodell gebunden, das gekennzeichnet ist durch das Vorhandensein einer Intention des Autors, ein Handlungsziel und ein Handlungsresultat. Eine kommunikative Handlung des Autors muss daher immer abgestimmt sein auf den kommunikativen Rahmen, in dem sie stattfindet.
Ausgehend von seinen Untersuchungen zu Textstrukturen definiert Ley als Charakteristika für Texte der technischen Kommunikation folgende Eigenschaften:

- Strukturelle Dreidimensionalität
- Multimediale Komplexität und
- Eine Trennung von Erfassung und Publikation.

Strukturelle Dreidimensionalität

Die Strukturelle Dreidimensionalität hat dabei folgende Dimensionen:

- *Syntaktische Dimension*: aus welchen Einheiten besteht ein Schema, wie lassen sich mehrere Schemata zu einem Gesamtschema verknüpfen, welche Einheiten dürfen in welcher Reihenfolge und Häufigkeit auftreten.
- *Semantische Dimension*: welche Art von Einheiten werden kommuniziert. Die Semantik muss dabei eine adäquate Repräsentation der zu beschreibenden Welt bzw. Domäne liefern, ihre Beschreibung muss praktikabel, homogen und konsistent sein.
- *Pragmatische Dimension*: alle Einheiten eines Schemas haben eine Funktion, so dass das Schema über eine illokutionale Binnenstruktur verfügt.

Strukturelle Textkontrolle

Aus diesen Vorüberlegungen heraus entwickelt Ley seine *Strukturelle Textkontrolle* mit dem Hintergrund, dass eine solche es erlauben soll, eine klar umrissene Informationsmenge angemessen, vollständig und konsistent zu modellieren.

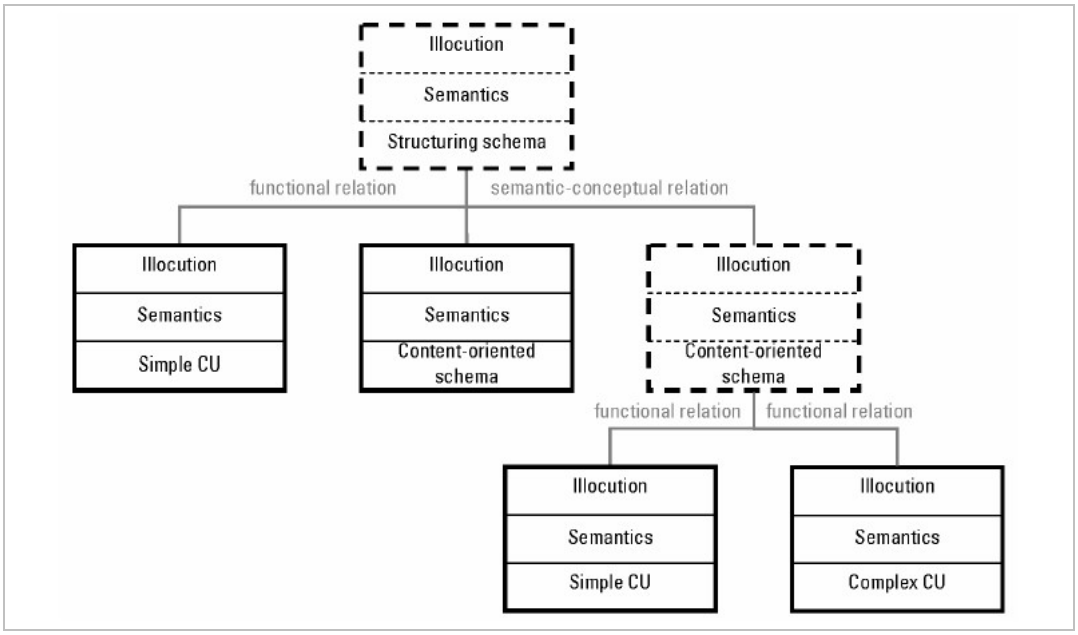


Abb. 36 Strukturelle Textkontrolle [Ley06b, S. 139]

Diese strukturelle Textkontrolle ist gekennzeichnet durch

- der Definition von Textkonstituenten,
- die für jeden Textkonstituenten geforderte Angabe
  - des illokutiven Grundtypen,
  - der semantischen Struktur,
  - der syntaktischen Struktur und
- die Angabe der Relationen zwischen den Textkonstituenten.

Illokutions-struktur

Die Angabe einer Illokutionsstruktur sieht Ley darin begründet, dass unter sprechakttheoretischer Perspektive das Vollziehen von Illokutionen das konstituierende Merkmal der Kommunikation ist. Ein Text enthält dementsprechend eine Textillokution, die das kommunikative Potenzial des Textes als Ganzes enthält. Ley schränkt die Auswahl der dabei zu verwendenden illokutiven Grundtypen und ihrer entsprechenden Untertypen bereits für den Gebrauch im Rahmen der technischen Kommunikation ein. Mögliche Illokutionstypen sind danach:

- Assertionen
  - Feststellungen
  - Beschreibungen
  - Erklärungen
  - Indikationen
- Direktiva
  - Aufforderungen
  - Warnungen
    - Gefahr
    - Warnung
    - Vorsicht

Semantische Struktur

Was die Angabe der semantischen Struktur angeht, bleibt Ley sehr vage in seinen Aussagen. Er fordert, dass jede Textkonstituente, die von einer Illokution getragen wird,

mit einem begrifflichen Repräsentanten verknüpft wird. Wie man letzten Endes aber auf diesen kommt, spezifiziert er nicht näher.

Syntaktische Struktur

Als letzte Forderung sieht Ley die Angabe der syntaktischen Struktur vor und bietet hierfür wiederum Modellierungsprimitiven zur Verwendung an:

- *Einfache Kommunikative Einheiten*, die den Autor inhaltsorientiert bei der Informationserfassung unterstützen.
- *Komplexe Kommunikative Einheiten*, die strukturierenden Charakter haben und bereits erfasste Texte zu größeren textuellen Einheiten zusammenfassen können.
- *Inhaltsorientierte Schemata*, durch die der Autor eine in sich geschlossene Informationsmenge erfassen kann, die für sich alleine aussagekräftig ist und damit losgelöst von anderen textuellen Einheiten produziert und rezipiert werden kann.
- *Strukturierende Schemata*, über die der Autor isolierte Informationen funktions- und bedarfsgerecht zur Verfügung stellen kann.

Ley zieht dann Parallelen zur Textstruktur von Informationsprodukten, die seiner Meinung nach nicht nur in eine Makro- und Mikrostruktur eingeteilt werden können, sondern zusätzlich noch über eine Meso- und Metastruktur verfügen:

Mikro- und Mesostruktur

Zitat Z32: [Ley06b]

„Die unterste Ebene ist die Mikrostruktur. Dieser Ebene schreibe ich den Satzbau, also die Formulierungsmuster zu. Aber auch der Wortschatz und die so genannten Inline-Elemente wie Produktbezeichnung oder Hervorhebung sind Gegenstand der Mikrostruktur. Die darüber liegende Ebene bezeichne ich als Mesostruktur. Ihr kommt eine ganz zentrale Rolle bei der Erstellung von Informationsprodukten zu. Entlehnt ist diese Ebene der Kommunikationsanalyse, wo Mesostrukturen für Frage-Antwort-Sequenzen oder Vorwurf-Rechtfertigungs-Sequenzen stehen. Es ist das funktionale Strukturierungsprinzip, das auf dieser Ebene am stärksten zum Tragen kommt. Nicht von ungefähr wird diese Ebene auch gerne mit einem (fiktiven) Dialog zwischen Technischem Redakteur und Leser gleichgesetzt. Ihre Elemente werden daher auch als Funktionale Einheiten oder Kommunikative Einheiten bezeichnet.“

Nach Ley ist die Mikrostruktur also die Binnenstruktur einer einfachen Kommunikationseinheit, während die Mesostruktur die Ebene der Kommunikativen Einheiten wiederspiegelt. Es handelt sich dabei nach Ley um Strukturen, die nicht notwendigerweise den Rang eines eigenen Kommunikationstyps besitzen.

Makro- und Metastruktur

Zitat Z33: [Ley06b]

„Mehrere Funktionale Einheiten oder Kommunikative Einheiten stehen allerdings (fast) nie isoliert. Sie sind vielmehr in größere Elemente eingebettet, die als Sequenzmuster, Schemata oder Maps/Blöcke bekannt sind. Textteile dieser makrostrukturellen Beschreibungsebene sind weniger funktional, als vielmehr inhaltlich geprägt. Im Idealfall ist jedes Sequenzmuster oder Schema und jeder Block/jede Map in sich kohärent und bildet ein inhaltliches Ganzes.Die oberste Ebene eines Informationsprodukts schließlich ist die Metastruktur. Auf dieser Ebene werden die Elemente der Makrostruktur in eine ganz bestimmte Reihenfolge gebracht und publiziert. Hier wiederum können beide Strukturierungsprinzipien zum Einsatz kommen. Ein Informationsprodukt kann beispielsweise objektorientiert aufgebaut sein, wie es häufig bei Softwaredokumentation der Fall ist: Erst wird Menü 1 mit seinen einzelnen Funktionen beschrieben, dann Menü 2.“

Über die Makrostruktur wird nach Ley die Ebene der inhaltsorientierten Schemata definiert. Diese Struktur liefert dementsprechend eine Vorstellung des komplexen Zusammenhangs und der Textbedeutung. Die Metastruktur als dazu nochmals übergeordnete Ebene lässt sich als abstrakte Organisationsform größerer textueller Zusammenhänge verstehen. Sie ist prinzipiell im Bereich der technischen Kommunikation sehr stark geprägt durch vorhandene Normen und Richtlinien, die bei der Erstellung technischer Dokumentation berücksichtigt werden muss.
Festgehalten werden die Strukturfestlegungen im Rahmen einer Grammatik, über die Ley folgendes sagt:

Dokumentationsgrammatiken

Zitat Z34: [Ley06b]

„Allerdings unterstützen einige Dokumentgrammatiken den Erstellungsprozess, während andere Dokumentgrammatiken hier kaum Entlastung für Technische Redakteure bringen. Dies betrifft in erster Linie die Mesostruktur. Wenn auf dieser Ebene nur Elemente wie <Paragraphxx, <Textxx oder <Listexx vorkommen, fällt es mir als Autor in der Regel sehr viel schwerer, die Informationen in eine sach- beziehungsweise handlungslogische Reihenfolge zu bringen. Enthält die Dokumentgrammatik statt dessen Elemente wie <Voraussetzungxx, <Warnungxx, <Handlungxx und <Resultatxx, so nimmt mich die Dokumentgrammatik dank ihrer „sprechenden Elemente" an die Hand und leitet mich Schritt für Schritt durch den Erstellungsprozess. Ich kann mein Augenmerk verstärkt auf den eigentlichen Schreibprozess richten. Über die richtige Sequenzierung der einzelnen Elemente mache ich mir nur noch wenige Gedanken. Ähnliches gilt übrigens auch für die Makro- und Metastruktur. Auch hier kann eine Dokumentgrammatik dazu beitragen, dass bestimmte Informationen in ganz bestimmten Elementen erfasst werden. Dies stellt sicher, dass ein und dieselbe Information nicht an mehreren Stellen im Informationsprodukt gleichzeitig erscheint und Redundanz vermieden wird. Es stellt aber auch sicher, dass die entsprechende Information an genau der Stelle publiziert wird, wo die Leserinnen und Leser diese erwarten."

Relationen

Als letzte geforderte Angabe tauchen bei Ley Relationen als sinnstiftende Elemente zwischen den Textkonstituenten auf und er unterscheidet hier

- *Semantisch-konzeptuelle Relationen*, die abstrakt sind, über ihre Kernbedeutung definiert sind und bei denen es sich entsprechend um semantisch primitive Relationen handelt, die unabhängig von ihrem Äußerungskontext identifiziert werden können. Hierdurch werden hierarchische Beziehugen, Teil-Ganzes-Beziehungen oder Klassenzugehörigkeiten definiert.

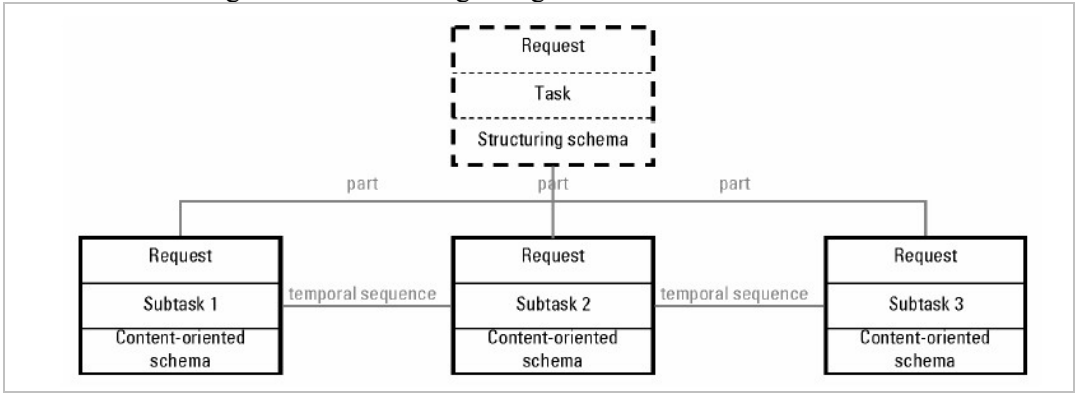


Abb. 37 Beispiel: semantisch-konzeptuelle Relation nach Ley [Ley06b, S. 137]

- *Funktionale Relationen*, die sich auf die Illokution beziehen und zu deren Bestimmung vor allem der Kontext berücksichtigt wird, in dem die Textkonstituenten geäußert werden, d.h. das kommunikative Potenzial einer Textkonstituente wird auf den jeweiligen Kontext bezogen und konkretisiert. Diese Art der Relationen ist in erster Linie auf der Mesostruktur lokalisiert, da funktionale Relationen vor allem zwischen kommunikativen Einheiten vorhanden sind.

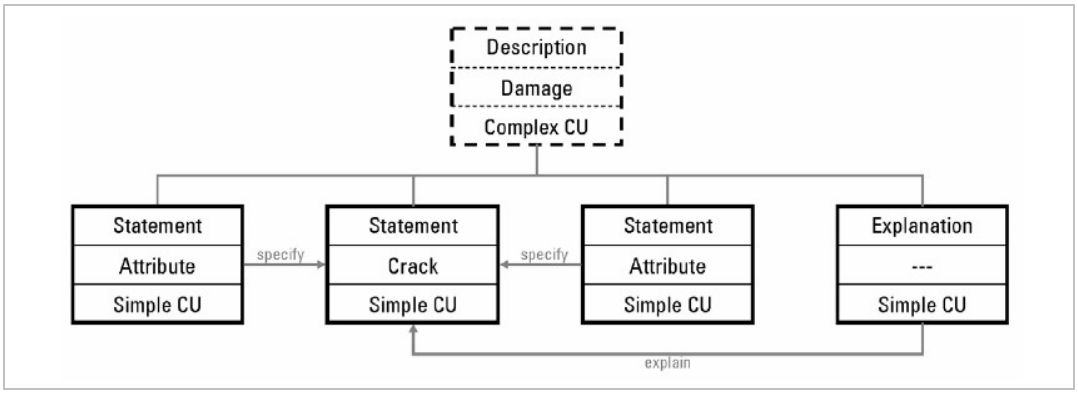


Abb. 38 Beispiel: funktionale Relation nach Ley [Ley06b, S. 138]

6.4 DITA (Darwin Information Typing Architecture)

DITA ist eine Informationsarchitektur auf XML-Basis, die speziell für die Technische Dokumentation konzipiert wurde. DITA definiert eine Menge an Informationstypen, die topic-orientiert erstellt und verwaltet werden können, und stellt mögliche Mechanismen zur Verfügung, um die Topics zu kombinieren und existierende Informationstypen mittels eines Spezialisierungsprozesses zu erweitern. Nach dem Prinzip der Vererbung werden die für die Ausgangstypen vorgenommenen Festlegungen an die abgeleiteten neuen Typen weitervererbt und können bei Bedarf spezifisch angepasst oder erweitert werden.

Basiselemente von DITA

Die Basiselemente von DITA sind:

- *topics:* Ein Topic ist eine 'Informationseinheit' mit einem Titel und einem Inhalt, kurz genug um ein einzelnes Subjekt zu spezifizieren oder eine Antwort auf eine einzelne Frage zu sein, aber lang genug, um selbständig Sinn zu ergeben und als eine Einheit behandelt zu werden. Topics sind in sich geschlossene, möglichst kontextunabhängige Inhaltsblöcke. Ein einfaches Beispiel für ein Topic ist eine Begriffserklärung. Wenn eine Begriffserklärung als eigenständiges Topic organisiert ist, kann sie an allen Stellen, an denen der Begriff vorkommt, (wieder)verwendet werden.
- *maps:* DITA Maps sind Dokumente, die Referenzen zu Topics zusammenfassen und organisieren, um die Beziehungen zwischen den Topics anzuzeigen.

Ziele der Topic Organisation

Autoren können durch das Organisieren des Inhalts in Topics drei Ziele erreichen:

- Inhalte sind lesbar und das auch dann, wenn auf sie von einem Index oder von einer Suche aus zugegriffen wird und nicht nur als ein Bestandteil, der in einer bestimmten Lesesequenz auftreten muss. Da die meisten Leser technische Dokumentation nicht wie einen Roman von Anfang zu Ende lesen, verspricht der topic-orientierte Ansatz ein wertvolles Mittel des Informationsdesigns, um

sicherzustellen, dass jede Informationseinheit selbständig sinnbringend gelesen werden kann.
- Inhalte können zielgerichtet zusammengestellt werden.
- Inhalte können in unterschiedlichen Zusammenstellungen wieder-verwendet werden.

Topic-orientiertes Schreiben erfordert jedoch die Einhaltung folgender Kriterien:
- Konsistenz,
- lokale Unabhängigkeit,
- Unabhängigkeit von der Navigation und
- textuelle Übergangskonstruktionen.

Generische Topics

DITA liefert generische Informationstypen als Basis für weitere spezialisierte Typen.
- *Concept topics* antworten auf ″Was ist...″ Fragen und stellen Hintergrundwissen dar, das dem Leser hilft, essentielle Informationen über das beschriebene Produkt zu erfahren.
- *Reference topics* beschreiben geordnete Eigenschaften eines Subjekts oder Produkts. In technischer Dokumentation werden sie oft benutzt, um geordnete Inhalte aufzunehmen, wie z.B. bibliographische Listen oder Kataloge. Reference topics stellen einen schnellen Zugriff auf diese Fakten sicher.
- *Glossary topics* definieren einzelne Bedeutungen eines Begriffs. Neben dem Identifizieren eines Begriffs kann das Topic auch verwandte Begriffe defiieren. Durch diese Art der Terminologiefestlegung ist sichergestellt, dass ein Team von Autoren dieselben Begrifflichkeiten verwendet.

Reuse

Eines der Hauptziele von DITA ist die Reduktion der weit verbreiteten copy&paste-Wiederverwendung.

> Zitat Z35: [DITAAB07]
> *“Because of the non-nesting structure of topics, a topic can be reused in any topic-like context. Information designers know that when they reuse a topic in a new information model, the architecture will process it consistently in its new context.[…] DITA instead leans toward a different SGML reuse technique and provides each element with a conref attribute that can point to any other equivalent element in the same or any other topic. This referencing mechanism starts with a base element, thus assuring that a fail-safe structure is always part of the calling topic (the topic that contains the element with the conref attribute). The new content is always functionally equivalent to the element that it replaces.”*

Wiederverwendung innerhalb von DITA geschieht dementsprechend auf zwei Ebenen: Topic und Content Wiederverwendung.

6.5 Mumasy: Informationsmodell für den Maschinenbau

Mumasy (Multimediales Maschineninformationssystem) ist ein Verbundprojekt des deutschen Maschinen- und Anlagenbaus. Ziel des Projekts war es, ein herstellerunabhängiges Informationsmodell für den Maschinenbau zu schaffen für den Datenaustausch zwischen Zulieferern, Herstellern und Kunden bzw. den verschiedenen Phasen des Produktlebenszyklus.
Dementsprechend sollen über den gesamten Produktlebenszyklus hinweg, Informationen aller Art über eine Maschine erfasst und diese den Informationsnutzern für ihren ganz individuellen Zweck zur Verfügung gestellt werden. Informationsnutzer betrachten eine Maschine aus unterschiedlichen Perspektiven. Je nachdem, wie die

vermittelte Information gedanklich umgesetzt wird, entstehen Sichtweisen. Führt die Umsetzung der Informationen zum Handeln, spricht man von einer handlungsorientierten Sichtweise. Führt die Umsetzung der Informationen zum Verstehen eines Funktionsablaufs, spricht man von der funktionalen Sichtweise. Beschreiben die Informationen die geometrische Struktur einer Maschine, spricht man von der baulichen Sichtweise. Zusätzlich können die Informationen aus der Dokumentensicht (z.B. Betriebsanleitung) betrachtet werden. Das Mumasy-Informationsmodell repräsentiert Strukturen für jede der vier Sichtweisen, und ermöglicht eine umfassende Darstellung der Maschine für die Nutzer.

Linguistische Modellierungs method.

6.6 Wiegand: funktional-positionale Segmentierung

Wiegand beschäftigte sich mit der Strukturanalyse von Wörterbuchartikeln und entwickelt ursprünglich dafür die Segmentationsmethodologie bestehend aus der *funktionalen Segmentation* und der *funktional-positionalen Segmentation*.

Funktionale Segmentation

Dabei besteht die funktionale Segmentation von Wörterbuchartikeln aus der Ermittlung der funktionalen Textsegmente eines Artikels, was diese Methodologie in ihrem Ansatz auch für technische Dokumente interessant macht. In der Betonung auf die Funktionalität von Textsegmenten findet sich wiederum die Verbindung zur Sprechakt-Theorie wieder. Die exhaustive funktionale Segmentation ermittelt in einem weiteren Schritt Strukturanzeiger – funktionale lexikographische Textsegmente, deren genuiner Zweck darin besteht, die Wahrnehmung des potentiellen Benutzers dadurch zu unterstützen, dass sie ihm solche Ausschnitte aus der vollständigen Artikelstruktur anzeigen, deren Kenntnis etwas dazu beitragen kann, daß er die Angaben besser identifizieren, unterscheiden und systematisch - und damit schneller - auffinden kann. Dabei unterscheidet Wiegand zwischen typographischen und nicht-typographischen Strukturanzeigern. Als typographische Strukturanzeiger gelten die Schriftart und der Schriftschnitt. Zu den nicht-typographischen Strukturanzeigern gehören z.B. Satzzeichen, Klammern oder Pfeile.
Unter der exhaustiven funktionalen Textsegmentation versteht Wiegand die stufenweise Segmentation eines Wörterbuchartikels nach Angaben und Strukturanzeigern zusammen mit einer Darstellung aller Teil-Ganzes-Beziehungen in einer Beschreibungssprache, deren formale Eigenschaften bekannt sind (vgl. [Wiegan89a] und [Wiegan89b]). Diese Variante der funktionalen Textsegmentation ist dann erfüllt, wenn alle Textelemente bestimmt und benannt sowie den übergeordneten Textsegmenten zugewiesen worden sind. Damit wäre eine Zuordnung der Elemente zu Angaben oder Strukturanzeigern und eine Darstellung der Teil-Ganzes-Beziehungen gegeben.

Funktional-positionale Segmentation

Eine zweite Variante, die funktional-positionale Segmentation, ermittelt nicht nur die funktionalen Textsegmente der Wörterbuchartikel, sondern auch deren Position innerhalb der linearen Reihenfolge aller Textsegmente. Entsprechend der exhaustiven funktionalen Segmentation kann für diese Variante eine exhaustive funktional-positionale Segmentation erfolgen. Der auf diese Weise ermittelte hierarchische Aufbau von Wörterbuchartikeln wird in Form von baumartigen partitiven Strukturgraphen dargestellt.

Einsatz Technische Dokumen-tation

Die funktional-positionale Segmentierung wird im Umfeld der technischen Dokumentation eingesetzt, wenn bereits Dokumente bestehen und man diese funktional ‚zerlegen' möchte, um zukünftig beispielsweise auf XML umzusteigen. Das Ganze funktioniert in der Praxis, indem ein Dokument ausgedruckt wird und dann versucht wird, „Kästchen" zu zeichnen, die sich nirgends überschneiden dürfen (sie dürfen aber

ineinander verrschachtelt sein). Beim Kästchenziehen orientiert man sich an der bereits verwendeten typographischen Auszeichnung und erhält so eine Hierarchie, die man dann nur noch benennen muss (welche Funktion erfüllt genau das Wort, was ich jetzt gerade eingekästelt habe und das sich typographisch abhebt von den umherliegenden Wörtern, welche Funktion schreibe ich dem Absatz zu, der eine Hierarchie darüber der nächste Kasten ist etc.) und dementsprechend einfach in XML überführt werden kann.

6.7 Schäflein-Armbruster/Muthig: Funktionsdesign®

Konsequent umgesetzt ist das Funktionsprinzip (siehe Kapitel 6.3) in der Strukturierungsmethodologie Funktionsdesign® [MutSch99]. Der Grundgedanke dieser Methodologie besteht darin, dass jedem Teil eines Informationsprodukts – einem Satz, einem Abschnitt oder noch größeren Einheiten – eine eindeutige Funktion zugewiesen werden kann. Nach dieser von Muthig und Schäflein-Armbruster entwickelten Methodologie müssen für eine Dokumentart die für den kommunikativen Zweck erforderlichen Elemente (Sequenzmuster, Funktionale Einheiten und Auszeichnungselemente) bestimmt sowie die Regeln bei ihrer Verwendung festgelegt werden.

Situation des Empfängers

Dazu muss sich der Anwender des Funktionsdesigns® in die Situation des Empfängers – z.B. des Lesers eines Warnhinweises – versetzen. Der Leser erwartet im Warnhinweis Antworten auf Fragen wie „Worin besteht die Gefahr, wie groß ist die Gefahr, wie kann ich es vermeiden, mich in Gefahr zu begeben".

Der Anwender des Funktionsdesigns® muss detailliert prüfen, welches Informationsbedürfnis der Anwender kategorial hat, und muss dann den zukünftigen Autoren die erforderlichen Funktionalen Einheiten zur Verfügung stellen, um dieses Informationsbedürfnis zu befriedigen. Dabei kommt es vor, dass eine Funktionale Einheit Funktionale Untereinheiten aufweist.

Für jede auf diese Weise ermittelte Funktionale Einheit und Untereinheit müssen dann noch inhaltliche Festlegungen getroffen werden, um eine Konsistenz auf funktionaler und inhaltlicher Ebene zu erreichen.

Einsatz Technische Dokumentation

Wird im Umfeld der externen technischen Dokumentation sehr weit eingesetzt, weil es im Gegensatz zum Information Mapping™ (siehe 6.8) kostenlos ist und im Gegensatz zu anderen Methodologien an diversen Hochschulen im Curriculum für Technische Redakteure enthalten ist. Die Methodologie lehnt sich vollkommen an die Sprechakttheorie an und hat als Basis neben der Definition von Dokumentarten und Sequenzarten vor allem das Entwickeln Funktionaler Einheiten über die Anwenderfragen-Simulation im Fokus. Es schreibt dann außerdem vor, dass identifizierte Funktionale Einheiten typographisch konsistent ausgezeichnet werden müssen (also z.B. mit Formatvorlagen).

6.8 Information Mapping™

Bei der Information-Mapping-Technik handelt es sich um eine Methodologie zur Informationsstrukturierung. Sie basiert auf Allgemeinwissen aus den Bereichen der Kognitions-, Medien- und Lernpsychologie ist.

Wahrnehmungsoptimierung

Bei der Methodologie geht es im Wesentlichen darum, Informationen in *wahrnehmungs-optimierte Informationsgrößen und -strukturen* zu verwandeln. Bei der Information-Mapping-Technik werden diese Kenntnisse in einer einzigen Methodologie zusammengefasst. Die Information wird so aufbereitet, dass einzelne Informationsarten

in sich abgeschlossen sind. Mit dieser Methodologie kann ein Technischer Redakteur technische Dokumentation, einen Artikel oder andere Texte so schreiben, dass die in dem Dokument enthaltenen Informationen in einer kürzeren, prägnanteren und verständlicheren Form vorliegen. Die Inhalte werden dadurch leichter verständlich, bestimmte Inhalte können leichter wiedergefunden werden. Die Dokumente sind schneller zu lesen, führen zu weniger Fehlhandlungen und sind eindeutiger verständlich. Die Information-Mapping-Methodologie ist ein Basis-Standard, der sich sehr leicht an firmenspezifische Bedürfnisse anpassen lässt und sich somit zu einem eigenen Redaktionsstandard im Unternehmen ausbauen lässt. Leider ist diese Methodologie bzw. ihr Erlernen kostenpflichtig und Nachfragen bzgl. kostenloser akademischer Teilnahmen an E-Learning Kursen blieben unbeantwortet.

6.9 Textuelle Informationsmodellierung (Lobin)

Zitat Z36: [Lobin00, S. 3]
„Textuelle Informationsmodellierung ist also die Modellierung von Information anhand von Gesetzmäßigkeiten, die wir in der Sprache finden.[…] SGML und XML sind vielmehr Instrumente für die Modellierung von strukturierter Information."

Lobin beschreibt in seinem Buch, wie SGML und XML genutzt werden können, um Informationen an der Leitlinie von Textstrukturen zu modellieren. Der Idee von strukturierter Information liegen dabei nach Lobin folgende Beobachtungen zugrunde:

Konkrete und abstrakte Einheiten

- In einem Text gibt es unterschiedliche Arten von Ebenen, so genannte *konkrete Einheiten* textueller Art (Abfolge von Buchstaben) können von *abstrakten Einheiten* („Kategorien" wie beispielsweise „Überschrift") unterschieden werden. Die Abhebung der unterschiedlichen abstrakten Einheiten erfolgt dabei oft über spezifische Darstellungsmuster, wie beispielsweise typographischer Abhebung der Überschrift eines Kapitels vom übrigen Text.

Grammatik

- Die Anordnung dieser abstrakten und konkreten Einheiten ist nicht beliebig, sondern folgt festen Regeln, die zum einen das hierarchische Verhältnis von abstrakten Einheiten zu untergeordneten abstrakten oder konkreten Einheiten wieder spiegelt und zum anderen die lineare Anordnung gleichrangiger Einheiten. Diese Regeln lassen sich zu einer Grammatik zusammenfassen.
- Eine solche Grammatik kann derart aufgebaut werden, dass die Einheiten als Baumform angeordnet werden können, wodurch das hierarchische und lineare Verhältnis zwischen den Informationseinheiten veranschaulicht werden kann. Bei der Umsetzung in SGML/XML werden die entsprechend ermittelten verschiedenen Typen von Informationseinheiten nicht nur bei der Konstruktion von Bäumen verwendet, sondern erlauben es auch, die weitere Verarbeitung des Baumes zu bestimmen. Ein Hinzufügen weiterer Information in den einzelnen Knoten durch Attribute ermöglicht es, jeweilige Teilmengen von Informationseinheiten gleichen Typs weiter zu untergliedern

Einsatz Technische Dokumentation

Diese Methodologie wird oft in Kombination mit dem Funktionsdesign® verwendet. Sie ist stark ausgerichtet auf SGML-XML-Umsetzung, kann die im Funktionsdesign® ermittelten Funktionalen Einheiten als Ausgangspunkt nehmen und hat dann seine Stärke darin, dass es die Weiterverarbeitung stark an die XML-Welt anlehnt. Prinzipiell leistet sie aber nicht mehr als das Funktionsdesign®, da dieses auch vorschreibt, dass man Sequenzmuster festlegen muss.

7 Methodologien für Ontologien

Beim Erstellen von Ontologien für das Requirements Engineering bzw. Nutzen von Ontologien im Requirements Engineering führt kein Weg an einer Betrachtung und Beschäftigung mit bestehenden Ontologie-Methodologien vorbei. Ontologie-Methodologien beschreiben, wie denn eine Ontologie erstellt werden kann. Gleichermaßen soll in diesem Kapitel auch betrachtet werden, wie denn bestehende Ontologien verwendet werden können, um ein Dokument (bspw. eine Requirements Specification) semantisch zu annotieren.

7.1 Existierende Ontologie-Methodologien

Es existieren prinzipiell drei verschiedene Herangehensweisen, um eine Ontologie zu erstellen:

(1) entweder existieren keinerlei Basis-Dokumente, die für die Erstellung verwendet werden können und die Ontologie muss von Grund auf neu erstellt werden. Damit beschäftigen sich Methodologien wie die von Uschold & Grüninger, SENSUS, On-To-Knowledge, METHONTOLOGY, etc.

(2) Alternativ dazu kann auch bereits Wissen in mehr oder weniger strukturierter Form vorhanden sein (bspw. in Datenbanken, Komponentenontologien, etc.). Dies beschreiben die Ansätze DILIGENT, InfoSleuth, KRAFT, Observer, IPROMPT, etc.

(3) Die letzte Möglichkeit besteht darin, Ontologien aus komplett unstrukturierten Textdokumenten unter Nutzung von Information Extraction sowie Natural Language Processing (NLP) Techniken zu erstellen. Dies wird unter dem Begriff Ontology Learning beschrieben und beispielhaft seien hier die Ansätze von Kietz et al. sowie Aussenac-Gilles beschrieben.

Uschold & Grüninger

Uschold & King [UscKin95] bzw Uschold & Grüninger [UscGrü96] stellen basierend auf den Erfahrungen aus der Enterprise Ontology, die sie vorab erstellt haben, eine Methodologie vor, die aus vier Hauptphasen besteht:

- → Bestimmung des Zwecks der Ontologie
- → Erstellung der Ontologie
  - o Ontologieerfassung (top-down, bottom-up oder middle-out)
  - o Ontologiecodierung
  - o Ontologieintegration
- → Evaluation
- → Dokumentation

Grüninger & Fox [GrüFox95] bzw. Uschold & Grüninger haben basierend auf den obigen Schritten und aufgrund der im TOVE Ontologie Projekt gesammelten Erfahrungen die Schritte für die Entwicklung einer Ontologie nochmals erweitert:

- → Motivationsszenarien
- → Informale Kompetenzfragen
- → Terminologiedefinition in Prädikatenlogik
- → Formale Kompetenzfragen
- → Axiomdefinition in Prädikatenlogik
- → Vollständigkeitstheoreme

SENSUS

Swartout et al. [SwEtAl96] haben eine Methodologie auf der Basis von SENSUS entwickelt, wobei sie allerdings lediglich eine Taxonomie aufbauen und keine richtige Ontologie. Dabe werden die folgenden Schritte festgeschrieben:

→ Identifikation der Saatbegriffe (Konzepte der Ontologie)
→ Manuelles Verbinden der Saatbegriffe mittels SENSUS
→ Hinzufügen der Konzepte auf dem Pfad zur Wurzel
→ Hinzufügen neuer Begriffe der Domäne
→ Hinzufügen ganzer Unterbäume
→ Entfernen der nicht verwendeten SENSUS-Begriffe

On-To-Knowledge

Staab et al. (2001) [StEtAl01] sowie Sure [Sure03] haben die Ontologie-Methodologie On-To-Knowledge (OTKM) entwickelt, die aus den folgenden Schritten besteht:

→ Machbarkeitsstudie:
  - o Problemidentifikation
  - o Auswahl einer möglichen existierenden Lösung

→ Ontologie-Kickoff:
  - o Anforderungsaufnahme
  - o Analyse der vorhandenen Quellen
  - o Erstellung einer Taxonomie

→ Verfeinerung:
  - o Verhandlungen über die Konzepte mit Domänen-Experten
  - o Taxonomie erweitern
  - o Formalisieren der Taxonomie
  - o Beziehungen hinzufügen

→ Evaluation:
  - o Probleme und Lösungsmöglichkeiten evaluieren

→ Wartung

In [StrHar05] wird ausserdem ein weiterer Ontologie-Entwicklungsprozess beschrieben, der folgende Schritte beinhaltet:

→ Verbreitete und gemeinsam genutzte Konzepte identifizieren
→ Definition von Eigenschaften dieser Konzepte
→ Ermitteln von Wertebereichen für diese Eigenschaften
→ Ontologiehierarchie adaptieren
→ Definitionen verfeinern

METHON-TOLOGY

Bereits 1997 wurde durch Fernandez-Lopez et al. die Grundlagen für die Ontologie-Methodologie METHONTOLOGY gelegt (siehe dazu [GomFer04]). Diese beschreibt die Erstellung einer Ontologie als eine Menge von Aktivitäten, die in drei übergeordnete Gruppen eingeordnet werden können: Management-, Entwicklungs- und Unterstützungsaktivitäten. Diese wiederum können in die folgenden Unteraktivitäten aufgeteilt werden:

→ Managementaktivitäten:
  - o Planung
  - o Kontrolle

- Qualitätssicherung

→ Entwicklungsaktivitäten:
- Spezifikation
- Konzeptualisierung
- Formalisierung
- Implementierung
- Unterhalt

→ Unterstützungsaktivitäten:
- Wissensbeschaffung
- Integration
- Evaluation
- Dokumentation
- Konfigurationsmanagement

Bei der Konzeptualisierung in METHONTOLOGY müssen die folgenden Schritte berücksichtigt werden:
→ Glossar der Begriffe erstellen
→ Konzepttaxonomien erstellen
→ Binäre Ad-hoc-Beziehungsdiagramme erstellen
→ Konzeptwörterbuch erstellen
→ Binäre Ad-hoc-Beziehungen beschreiben / Instanzattribute beschreiben / Klassenattribute beschreiben / Konstanten beschreiben
→ Formale Axiome definieren / Regeln definieren
→ Instanzen erstellen

Hingegen umfasst das Reengineering einer Ontologie gemäß METHONTOLOGY die folgenden Arbeitsschritte:

→ Implementierte Ontologie → Reverse Engineering zu konzeptuellem Modell
→ Konzeptuelles Modell → Restrukturierung zu neuem konzeptuellen Modell
→ Konzeptuelles Modell neu → Forward Engineering zu implementierter Ontologie
→ Implementiere Ontologie

Eine ausführlichere Beschreibung von METHONTOLOGY findet sich in [GomFer04].

Pinto et al.

Gemäß [PinMar04] besteht die Erstellung einer Ontologie aus prinzipiell mehreren Stufen: Spezifikation, Konzeptualisierung, Formalisierung, Implementierung und Wartung. In jeder dieser Stufen können mehrere Aktivitäten durchgeführt werden. Parallel zu diesen Stufen existieren die unabhängigen Aktivitäten Wissensgewinnung, Evaluation und Dokumentation. Die Methodologie zeigt also eine starke Ähnlichkeit zu der zuvor vorgestellten METHONTOLOGY. Die Entwicklung einer Ontologie findet dabei gemäß der Entwicklung eines Prototypen statt: in jeder Phase kann zu einer beliebigen vorherigen Phase zurückgekehrt werden. Der Prototyp der Ontologie wird in jedem Zyklus weiter verfeinert, bis er schließlich Produktreife erlangt hat.

Im Nachfolgenden werden Methodologien behandelt, die nicht davon ausgehen, dass eine Ontologie erst von Grund auf neu erstellt werden muss, sondern dass bereits Bausteine (Datenbanken, Ontologieteile, etc.) vorhanden sind, die es erlauben, eine (größere) Ontologie zu entwickeln bzw. eine existierende Ontologie weiterzuentwickeln.

DILIGENT

Bei DILIGENT (DIstributed, Loosely-controlled and evolvInG Engineering of oNTologies) [VrEtAl05] handelt es sich um eine Methodologie, die prinzipiell aus fünf Schritten besteht:

- → Erstellen,
- → lokale Adaption,
- → Analyse,
- → Revision,
- → lokales Update.

Dies fünf Schritte werden von unterschiedlichen Rollen vorgenommen: der Ontologie-Entwickler erstellt die Ontologie primär, die lokale Adaption findet durch den Benutzer statt. Dies wird von einem Control Board analysiert, ggf. verbessert und wieder freigegeben und der Benutzer erstellt gegebenenfalls wieder ein lokales Update. Dabei wird davon ausgegangen, dass die Ontologie am Anfang noch nicht komplett ist, sondern nur bruchstückhaft vorliegt, auf den sich die Entwickler der Ontologie am Anfang geeinigt haben. Bei der lokalen Adaption können die Benutzer auf die erstellte Kernontologie zugreifen und sich lokale Kopien anfertigen und diese auf ihre Wünsche anpassen. Die Veränderungen der verschiedenen Benutzer werden dann als Change Requests gesammelt und dem Kontrollboard zur Analyse vorgelegt. Das Kontrollboard muss dann entscheiden, welche lokalen Änderungen in der Kernontologie vorgenommen werden sollen und welche abgelehnt werden. Im nächsten Schritt, der Revision, wird die Kernontologie mit den lokalen Ontologien verglichen und beurteilt, ob die Kernontologie noch einen Mehrwert für die Nutzung der lokalen Ontologien hat oder ob sie eventuell wieder angepasst werden muss. Wurde die Kernontologie angepasst und wieder freigegeben, haben die Benutzer die Möglichkeit ihre lokalen Ontologien an die Veränderungen der Kernontologie anzupassen.

Infosleuth

Dabei kann die Ontologie beispielsweise aus Datenbanken konstruiert werden. Dies könnte durch das System von Infosleuth geschehen [StrHar05], allerdings bedarf es hierbei eines Experten für die Evaluation der vorgeschlagenen Begriffe, die aus der Datenbank automatisch generiert werden. Bei Infosleuth handelt es sich um ein agentenbasiertes System, das die Erstellung von komplexen Ontologien aus kleineren Komponentenontologien unterstützt.

KRAFT

In der Methodologie KRAFT [StrHar05] geht es mehr um die Entwicklung von gemeinsam genutzten Ontologien. Dabei stellt KRAFT eine agentenbasierte Middleware dar, die eine Menge an Techniken umfasst, um Ontologien zu kombinieren (Klassenmapping, Attributmapping, Beziehungsmapping und Compoundmapping). Die Methodologie von KRAFT umfasst die folgenden vier Schritte:

- → Rahmen der Ontologie bestimmen
- → Domänenanalyse
- → Formalisierung der Ontologie
- → Erstellen der Top-Level-Ontologie

OBSERVER

Auch bei OBSERVER (ontology-based system enhanced with relationships for vocabulary heterogenity resolution) [AlEtAl05] handelt es sich um ein System für das Erstellen einer größeren Ontologie aus kleineren Komponenten, wobei hier speziell das Ontology Merging betrachtet wird. Bei Ontology Merging entsteht eine neue Ontologie, die mindestens zwei andere Ontologien umfasst im Gegensatz zu Ontology Alignment, welche keine zusätzliche Ontologie als Ergebnis hat, sondern in die bisherigen Ontologien Verbindungen einbaut, um die gemeinsamen Konzepte beider Ontologien zu beschreiben. Dabei gibt es folgende Begrifflichkeiten zu unterscheiden:

(i) Synonyme: Zwei Begrifflichkeiten in unterschiedlichen Ontologien haben die selbe Semantik;
(ii) Hyponyme: ein Begriff ist spezieller als ein anderer in einer anderen Ontologie;
(iii) Hyperonyme: ein Term ist genereller als ein anderer;
(iv) Überschneidungen;
(v) Disjunktheit: es existieren keine Überschneidungen und
(vi) Überdeckungen: Konzepte in der einen Ontologie besitzen ein unterschiedliches Abstraktionsniveau als Konzepte der anderen Ontologie.

IPROMPT

Auch bei der IPrompt-Methodologie (Noy, Musen 2003 [NoyMus03]) geht es darum, mehrere Ontologien zu einer zu verbinden. Dabei basiert die Methodology auf SMART (Moy, Musen 2000 [NoyMus00]), einem Algorithmus um semi-automatisch Ontologien zu verbinden. Die Methodologie IPROMT beinhaltet folgende Schritte:

→ Eine Liste von möglichen Zusammenlegungsvorschlägen erstellen
→ Nächste Operation auswählen
→ Operation ausführen
→ Inkonsistenzen und Probleme finden
→ Lösungsvorschläge unterbreiten

Die oben beschriebenen Ontology-Mapping-Technologien (Infosleuth, KRAFT, IPROMPT, etc.) werden in [AlEtAl05] noch genauer beschrieben und verglichen. Der interessierte Leser sei daher auf diese Quelle verwiesen.

Ontology Learning

Neben den bisher betrachteten Ansätzen eine Ontologie, entweder von Beginn an zu erstellen oder Bausteine von kleinen Ontologien zu einer großen zu verbinden, besteht auch noch die Möglichkeit, eine Ontologie auf Basis vorhandener Texte zu erlernen. Dafür werden Natural Language Processing (NLP) Ansätze mit Machine Learning-Algorithmen kombiniert, um aus vorliegenden Texten eine Ontologie zu erhalten. Dabei wird nach [GomFer04] unter folgenden Möglichkeiten unterschieden bzw. diese teilweise auch kombiniert:

→ Lernen der Ontologie aus vorliegenden Texten:
  - Pattern-basierte Extraktion: vorhergehende Silben beschreiben eine Verfeinerung eines bereits vorhandenen Konzepts (z.B. Dampfschiff ist Unterklasse von Schiff)
  - Assoziationsregeln: Kommt ein Wort häufig in der Nähe eines anderen Wortes vor, werden diese vermutlich miteinander irgendwie in Beziehung stehen.

- Konzeptuelles Clustern: Stehen Subjekte mit denselben Verben in Sätzen, werden diese Subjekte eine Verbindung oder gemeinsame Oberklasse haben (Anna fährt mit dem Rad, Berta fährt mit dem Auto).
- Ontology pruning: Erweitern eine Kernontologie um domänenspezifische Aspekte unter Nutzung eines Corpus an vorliegenden Texten.
- Konzeptuelles Lernen: Eine vorhandene Taxonomie wird durch zusätzliche Konstrukte erweitert.

→ Lernen der Ontologie aus Instanzen
→ Lernen der Ontologie aus Schemata
→ Lernen der Ontologie aus vorhandenen Ontologie-Mappinginformationen

Kietz et al.

Die Methodologie nach Kietz, Maedche und Volz [KiMaVo00] beschreibt das Lernen einer Ontologie durch Nutzung von bereits identifizierten Konzepten eines bestehenden Dokuments sowie natürlicher Sprachanalyse auf diesem Dokument. Dies geschieht zyklisch wobei die folgenden 5 Schritte immer wiederholt werden:

→ Auswahl der Quellen
→ Lernen der Konzepte
→ Fokussierung der Domäne
→ Lernen der Relationen
→ Evaluation

Die Methodologie wurde dabei im Tool Text-To-Onto implementiert und im Rahmen der bereits beschriebenen Methodologie On-To-Knowledge verwendet.

Aussenac-Gilles

Die Methodologie von Aussenac-Gilles und Kollegen [AuBiSz00] erlaubt das Erstellen eines Domänenmodells durch die Analyse eines Corpus (einer Sammlung von Dokumenten) durch NLP-Tools. Die Methodologie kombiniert dabei Wissensammlungstools (basierend auf Linguistic) mit Modellierungstechniken, um Verknüpfungen zwischen Modell und Text zu erhalten. Nach Auswahl des Textes wird linguistisches Wissen (Begriffe, lexikalische Beziehungen, Gruppen von Synonymen) gesammelt. Dies wird im nächsten Schritt in ein semantisches Netzwerk transformiert. Dieses Netzwerk beinhaltet dann Konzepte, Beziehungen zwischen den Konzepten sowie Konzeptattribute. Dafür werden die folgenden vier Schritte vorgeschlagen:

→ Corpus constitution: Dies beschreibt die Auswahl der notwendigen Dokumente. Diese sollte vollständig sein, also alle Begrifflichkeiten der Domäne abdecken.
→ Linguistic study: Nutzung der linguistischen Techniken (wie vorher beschrieben)
→ Normalization: Erstellung eines Konzeptmodells. Dies besteht aus zwei Schritten: einem linguistischen (Auswahl von Hyperonymen, Hyponymen, etc.) sowie einem modellierungstechnischen Schritt (Normalisieren der Konzepte und ihrer Verbindungen).
→ Formalization: Konzepte und Relationen werden in einer formalen Sprache implementiert.

Für diese Methodologie stehen verschiedene Tools zur Verfügung: Lexter, Géditerm und Terminae.

Weitere Informationen siehe [GomFer04]. Einen sehr guten Überblick über das Lernen von Ontologien aus Texten bietet außerdem [Bieman05]. In diesem Bericht werden auch gleichzeitig die Grenzen des Ontologie Learning aufgezeigt: so ist die Idee zwar prinzipiell sehr gut, aber eine automatische Durchführung steht derzeit noch in den Sternen. Es wird nach der Konzeptfindung, Auffinden von Relationen, Clustern, Prunen und Erstellen von Hierarchien derzeit immer eines Fachmanns benötigen, der über die erstellte Ontologie nochmals darübersieht und diese korrigiert. Aber gerade bei der Erstellung von großen Ontologien ist eine semi-automatische Erstellung immer noch einer komplett händischen vorzuziehen.

DBPedia

Was bisher noch nicht berücksichtigt wurde, ist das Erstellen von Ontologien aus bereits stärker strukturierten Dokumenten. So wird beispielsweise in DBPedia [AueLeh07] gezeigt, dass aus strukturierten Seiten von Wikipedia eine Ontologie erstellt werden kann inkl. Konzepte, Attribute, Relationen und Instanzen. Diese Ontologie kann dann recht einfach dazu verwendet werden, um Zusammenhänge zwischen vorher nicht verbundenen Seiten zu erkennen („Was haben Innsbruck und Leipzig gemeinsam?“). Dazu werden Wiki-Templates verwendet, die beispielsweise für Städte, Personen, etc. bereits existieren und die Struktur dieser Seiten angeben. Diese Templates werden zwar nicht auf allen Wikipedia-Seiten verwendet, aber stellen doch eine Grundlage für eine Vielzahl von Seiten dar. Die Inhalte dieser Templates werden als RDF-Tripel gespeichert und in einer Ontologie gesammelt, die im weiteren Verlauf für das Reasoning verwendet werden kann. Eine genaue Methodologie hierfür wurde bisher aber nicht angegeben.

Bei den bisher betrachteten Methodologien wurde nicht betrachtet, wie man zu Konzepten kommen soll. In den bisher beschriebenen Methodologien wurde zwar immer gesagt, dass man Konzepte finden soll, aber wie dies geschehen soll wurde nicht genauer ausgeführt.

Ontology Development Guide 1.0.1

Im Ontology Development Guide 1.0.1 [NoyGui01] wird das Vorgehen zum Finden von Konzepten und deren Relationen erstmalig beschrieben. Dabei wird vorausgeschickt, dass es keinen einheitlich korrekten Weg gibt, um eine Domäne zu modellieren – es gibt vielmehr verschiedene Möglichkeiten. Die Entwicklung der Ontologie (also der Konzepte, Attribute und Relationen) sollte immer iterativ erfolgen und die identifizierten Konzepte in der Ontologie sollten möglichst nahe an den (physikalischen oder logischen) Objekten der Realwelt liegen. Diese sind dabei meist Substantive (Konzepte) oder Verben (Relationen) in Sätze, die die Domäne beschreiben.

Um eine Ontologie zu erstellen, sollte man daher immer Kompetenzfragen (competency questions) aufstellen, die domänenspezifisch sind. Diese Kompetenzfragen dienen dann später als Leitfaden, ob die erstellte Ontologie ausreichend ist. Dies gehört bereits zum ersten Schritt:

→ Bestimme die Domäne oder den Einsatzbereich der Ontologie
→ Berücksichtige die Wiederverwendung bereits existierender Ontologien
  - Dieser Schritt wurde bereits ausführlich besprochen.
→ Liste wichtige Konzepte in der Ontologie auf

- Im dritten Schritt soll man eine Liste aller wichtigen Begrifflichkeiten der Domäne herausschreiben, sowie all ihrer Attribute, die für die Domäne von Wichtigkeit sind.

→ Definiere die Klassen und die Klassenhierarchie

- Auch hier wird wieder von Top-Down, Bottom-Up oder Middle-Out gesprochen. Eine weitere Möglichkeit wäre, sich über die Relationen von existierenden Konzepten Gedanken zu machen und zu evaluieren, wie denn verschiedenen Konzepte in Beziehung stehen. Kommt die Beziehung in Abb. 39 vor, sind beide Konzepte sowie die Beziehung in die Ontologie aufzunehmen.

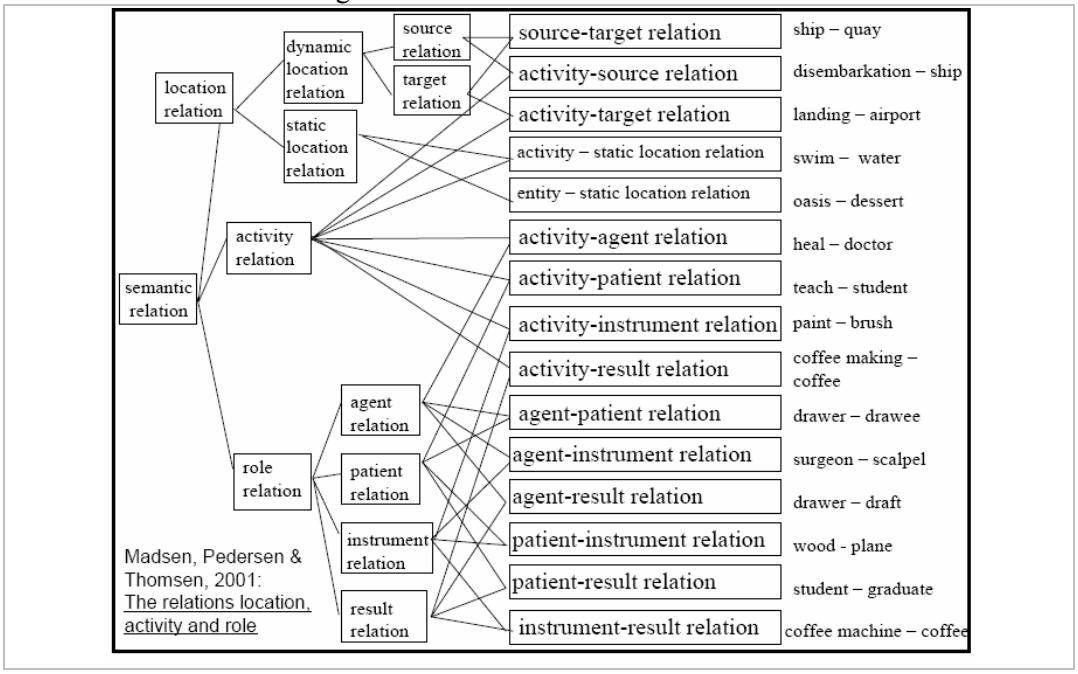


Abb. 39 Relations- und Beziehungstypen

→ Definiere die Eigenschaften der Klassen (slots)

- Dieser Schritt ist stark mit dem vorhergehenden verbunden.

→ Definiere die Datentypen, etc. dieser

- Hier werden Datentypen, Wertebereiche, Kardinalitäten, etc. von Eigenschaften (Slots) definiert.

→ Erstelle Instanzen

Beim Erstellen der Konzepte sowie der Konzepthierarchien sollten folgende Ratschläge aus [NoyGui01] berücksichtigt werden:

Zitat Z37: [NoyGui01]

- Wenn die Domäne einer Property bestimmt werden soll, suche die generellste Klasse die für diese Domäne herangenommen werden kann. Andererseits definiere keine zu allgemeinen Eigenschaften, da diese sonst im weiteren Verlauf ggf. nutzlos sind (z.B. eine Eigenschaft mit Domäne THING, diese wäre zwar allgemeingültig, aber nicht wirklich vorteilhaft).

- Wenn eine Domäne eine Liste von Klassen und darin sowohl eine Klasse als auch ihre Unterklasse enthält, entferne die Unterklasse.
- Wenn eine Domäne eine Liste von Klassen und darin alle Unterklassen einer Oberklasse enthält, aber nicht die Oberklasse selbst, dann entferne alle Unterklase und füge die Oberklasse hinzu.
- Wenn für eine Domäne nur ein paar Unterklassen einer Klasse enthalten sind, prüfe, ob nicht die Oberklasse selbst eventuell eine bessere Wahl wäre.
- Die Unterklasse einer Klasse beschreibt immer Konzepte, die auch die Oberklasse allgemeiner beschreiben könnte.
- Klassen repräsentieren Konzepte in einer Domäne und nicht die Wörter, die diese Konzepte beschreiben könnten.
- Synonyme für dasselbe Konzept beschreiben eine und nicht mehrere Klassen.
- Alle Nachbarn in einer Hierarchie sollten dieselbe Abstraktionsstufe besitzen.
- Wenn eine Klasse nur eine Unterklasse besitzt, gibt es vermutlich ein Modellierungsproblem oder die Ontologie ist nicht vollständig. Wenn mehr als ein Duzent Unterklassen für eine Klasse existieren, könnte die Einführung von weiteren Abstraktionsstufen dazwischen sinnvoll sein.
- Unterklassen einer Klasse haben entweder (1) zusätzliche Eigenschaften als die Oberklasse (2) andere Einschränkungen als die Oberklasse oder (3) andere Beziehungen als die Oberklasse.
- Klassen in terminologischen Hierarchien müssen keine zusätzlichen Eigenschaften einführen.
- Wenn die Konzepte mit unterschiedlichen Werten Restriktionen für unterschiedliche Klassen werden, wäre das Hinzufügen einer zusätzlichen Klasse für diese Unterscheidung sinnvoll.
- Wenn eine Unterscheidung in der Domäne sinnvoll erscheint und die Objekte mit unterschiedlichen Werten auch unterschiedliche Dinge in der Domäne repräsentieren, dann sollten auch eigene Klassen dafür in der Domäne erstellt werden.
- Eine Klasse zu der Instanzen existieren, sollte nur mit größter Sorgfalt verändert werden.
- Instanzen sind die am meisten spezifischen Konzepte, die in einer Wissensbasis enthalten sind.
- Wenn Konzepte eine natürliche Hierarchie darstellen, sollten diese auch als Klassen dargestellt werden.
- Eine Ontologie muss nicht alle möglichen Informationen über eine Domäne enthalten: eine zu große Generalisierung oder Spezialisierung über den Nutzen der Applikation hinaus ist meist nicht sinnvoll.
- Die Ontologie muss nicht alle möglichen Eigenschaften und Unterscheidungen zwischen Klassen enthalten (nur soweit sinnvoll).
- Beschreibe am Anfang eine Herangehensweise und Regeln für die Namensgebung für Klassen, Eigenschaften und Instanzen und bleibe bei diesen.

Eine andere Herangehensweise um Konzepte für Ontologien zu identifizieren sowie nützliche Tipps beim Erstellen einer Ontologie beschreibt [ReEtAl04]:

- Pragmatisch: Sollen Dinge automatisch unter die angegebene Klasse klassifiziert werden?
- Starte mit Primitiven und verfeinere diese später.

- Philosophische Sichtweise: Kann das Konzept komplett beschrieben werden? Viele Konstrukte aus dem täglichen Leben sind nicht aufzählbar (z.B. Menschen, Sprachen, etc.)

7.2 Parallele Erstellung der Ontologie beim Schreiben der Requirements Spezifikationen

Wie im vorherigen Teilkapitel gesehen, existieren derzeit sehr viele Ansätze, um Ontologien aus strukturierten oder unstrukturierten Quellen zu erhalten und es existieren auch einige Vorgehensmodelle, wie denn eine Ontologie von Beginn an erstellt werden soll.
Im semantischen Kommunikationsmodell haben wir bereits beschrieben, wie die Sender auf Basis ihres Wissens Dateninstanzen erstellen und diese dann in einer Nachricht (bei uns beispielsweise die Requirements Spezifikation) an den oder die Empfänger senden. Dieser hat aber eine andere Wissensbasis und kann daher gegebenenfalls die Dateninstanzen nicht so rezipieren, wie diese vom Sender gemeint waren. Daher müssen die (Objekt-)Dateninstanzen sorgfältig ausgewählt werden und ggf. um Meta-Dateninstanzen angereichert werden, um auf Grundlage der anzunehmenden Wissensbasis des Empfängers ein möglichst erfolgreiches Gelingen der kommunikativen Situation erreichen zu können. Hierfür bieten sich Ontologien förmlich an. Diese werden beim Erstellen des Dokuments ebenfalls entwickelt und dann verwendet, um den kommunikativen Rahmen abzubilden, indem sowohl Wissen über das anzunehmende empfängerseitige Wissensnetz, als auch Wissen über das Produkt – in diesem Fall die Requirements – in einer formalen Art und Weise beschrieben werden, die eine zukünftige Wiederverwendung der erstellten Dateninstanzen erlauben.
Zur Erstellung dieser Ontologien sei auf die in Kapitel 7.1 beschriebenen Ontologie-Methodologien verwiesen, die eine grundsätzliche Beschreibung, wie an die Entwicklung der Ontologie herangegangen werden soll, beinhalten. Genauer betrachtet werden sollen hier METHONTOLOGY, On-To-Knowledge sowie die Methodologie von Noy & McGuinness (siehe dazu Kapitel 8).

Zuerst muss entschieden werden, ob man eine top-down, bottom-up oder middle-out Strategie verwenden möchte (meist wird wohl eine middle-out-Strategie am sinnvollsten sein). Danach sollte man beginnen, eine Taxonomie der Begriffe festzulegen, die auch teilweise im Dokument selbst vorkommen, aber natürlich auch spezialisiert und generalisiert werden müssen. Hierbei kann man sich im Rahmen des Requirements Engineering an den parallel entwickelten Glossaren orientieren, die auch alle Begriffe beinhalten, die für das zukünftige System von Interesse sind.
Zusätzlich sollten auch die Begrifflichkeiten in die Ontologie integriert werden, die die Wissensbasis des Senders (Schreibers) widerspiegeln. Also die Informationen über das Umfeld des zukünftigen Systems, die zukünftigen Nutzer des Systems, sowie alle Informationen die beim Aufnehmen der Requirements durch verschiedene Personen geäußert werden.

Wurden die Begriffe in Hierarchien eingeordnet (die Taxonomie umfasst also alle entscheidenden Konzepte), werden unter Rücksprache mit Domänen-Experten Verbindungen zwischen diesen Konzepten gesucht und beschrieben sowie die Ontologie formalisiert, dokumentiert, evaluiert und getestet. Im folgenden Schritt wird die erstellte Ontologie dann verwendet um das Dokument zu annotieren.

7.3 Semantische Annotation von Dokumenten

Für die semantische Annotation von Dokumenten (Texte, Bilder, Videos, etc.) existieren bereits einige Software-Systeme (Tools): Annotea, S-CREAM, Onto-Mat, KIM, OnTeA, etc. (Einen Überblick vermittelt [Annot07]). Dabei gehen diese Tools davon aus, dass die Ontologie bereits vollständig vorhanden ist und auch das Dokument selbst nicht mehr verändert wird. Andernfalls muss die Annotation ggf. erneut durchgeführt werden. Die Tools untersuchen den Text auf Existenz von Konzepten aus der Ontologie und annotieren den Text entweder automatisch oder nach Rückfrage durch den Benutzer.

Primär lässt sich zwischen drei Arten von Annotationstools unterscheiden: Pattern-basiert, Regel-basiert und MachineLearning-basiert. Regelbasierte semantische Annotationsprozessoren wären beispielsweise KIM, MUSE oder SemTag. Auf Basis von Patterns gäbe es OnTeA, Armadillo oder On-To-Mat/PANKOW als Beispiel. Beispiele für Machine-Learning-Ansätze sind MnM und AmilCare. Im Folgenden wird je einer von jeder Art beispielhaft kurz beschrieben.

KIM KIM (Knowledge and Information Management) beschreibt eine Plattform für das semantische Annotieren und Indizieren von Dokumenten. KIM analysiert einen vorliegenden Text und erkennt Referenzen auf bekannte Entitäten (Personen, Orte, etc.). Diese Referenzen werden versucht mit Konzepten in der Ontologie zu matchen und dann in den Text eindeutige URIs einzufügen, die diese Konzepte genauer beschreiben. Ausserdem wird die Ontologie um Instanzen und Properties aus dem Dokument erweitert.

OnTeA OnTeA [LaSeBa06] lädt beispielsweise das zu bearbeitende Dokument sowie eine Menge regulärer Ausdrücke und versucht unter Zuhilfenahme dieser regulären Ausdrücke neue Instanzen für die Ontologie zu erkennen. Dies wird für alle definierten regulären Ausdrücke durchgeführt und anschließend noch zusätzliche Properties hinzugefügt.

MnM MnM stellt ein Tool dar, um eine semantische Annotation von Texten vorzunehmen. Dazu müssen zuerst mehrere annotierte Texte als Corpus eingelesen werden. Diese dienen als Grundlage für das Machine-Learning und einem Lazy-NLP genannten Algorithmus, der versucht die Konzepte im Dokument zu erkennen.

Einen guten Überblick über existierende Semantische Annotationen und Plattformen dafür liefert auch [ReeHan05].

Allerdings wurde in den bisher genannten Plattformen nicht beschrieben, welche Konzepte eigentlich semantisch annotiert werden sollen. [Reif01] beschreibt, dass es prinzipiell drei Arten von Annotationen gibt:

- Die Annotationen werden eingebettet in das Dokument
- Das Dokument verweist auf ein externes Dokument, das die Annotationen beinhaltet.
- Die Annotationen verweisen auf das annotierte Dokument; das Dokument selbst enthält keine Hinweise auf die Annotationen.

Beim Annotieren können folgende Probleme auftreten:

- Unterschiedliche Granularität

- Implizite Information
- Unterschiedliche Datenformate

Wie [OrEtAl06] weiter ausführt, können semantische Annotationen als Moleküle betrachtet werden, die die Atome (Dokumente und URLs) zusammenführen. Als semantische Annotation kommen informelle, formale oder ontologische Annotationen in Frage. Diese können auf verschiedenen Quellen durchgeführt werden: Dokumente, Wikis, Blogs, Grafiken, Prozessmodelle, etc. Aber auch hier wird keine Antwort auf die Frage gegeben, was denn speziell annotiert werden soll.

8 Evaluation der Methodologien

Im Folgenden wollen wir die in den vorigen Kapiteln dargestellten linguistischen Modellierungsmethodologien und Ontologiemethologien unter Berücksichtigung des entwickelten semantischen Kommunikationsmodells evaluieren.
Evaluation wird gemeinhin als die gezielte Bewertung von materiellen oder immateriellen Gegenständen unter Rückgriff auf Kriterien und Verfahren verstanden, deren Angemessenheit erläutert oder begründet werden sollte [House93]. Bei der Evaluation müssen daher vorher bestimmte Artefakte unter Berücksichtigung von ebenfalls vorher definierten Faktoren verglichen werden. Es existiert schon seit einigen Jahren eine rege Evaluationsforschung, die eine besondere Rolle im Bildungs- und Gesundheitswesen spielt [Frank00]. Diese ist vor allen Dingen auf Evaluationsverfahren ausgerichtet, die der Politikberatung dienen.

Design Science

Vor allem in Bereichen der Wirtschaftsinformatik wird die Evaluation mittlerweile noch stärker berücksichtigt und auch im Rahmen der sogenannten ***Design Science*** [HeEtAl04] als wichtiger Bestandteil jeglicher Forschung beschrieben.
Es existieren auch bereits einige Evaluationsansätze zur Bewertung von Modellen und Methodologien. So wird in [MooSha94] beispielsweise eine Evaluation der Qualität von Entity-Relationship-Diagrammen (ER-Diagrammen) beschrieben. In [HoGoBr93] auf der anderen Seite werden Methodologien zur objekt-orientierten Analyse und Design verglichen und evaluiert. Diese und andere Quellen dienten uns als Hilfestellung bei der Auswahl der von uns gewählten Evaluationskriterien, die in Form von konkreten Fragen erstellt wurden. Diese Fragen wurden für jede Methodologie durchgegangen und mit Punkten bewertet. Aufgrund eines vorher festgelegten Priorisierungsfaktors wurden abschließend die Punkte miteinander verrechnet und so eine Entscheidungshilfe gegeben.

8.1 Aufbau der Evaluationsmatrix und des Evaluationsleitfadens

Grundsätzlicher Aufbau der Evaluations-matrix

Die Anwendung der Methodologie führt wie in Abb. 40 zu sehen ist zu einem (Daten-) Modell, wovon zur Laufzeit Instanzen gebildet werden. Aus diesem Grunde haben wir uns bei der Evaluation auch die verschiedenen Bereiche angesehen: zuerst wird untersucht wie die Methodologie aufgebaut und dokumentiert ist. Hier legen wir speziellen Wert auf die Verständlichkeit der Dokumentation der Methodologie. Dann wird die Methodologie angewendet, d.h. wir evaluieren die Applikation der Methodologie. Anschließend sehen wir uns das Ergebnis (das Daten-Modell) an und untersuchen dessen Qualität. Zuletzt betrachten wir noch, wie die Instanzen des Datenmodells aussehen und wie diese personalisiert und wiederverwendet werden können.

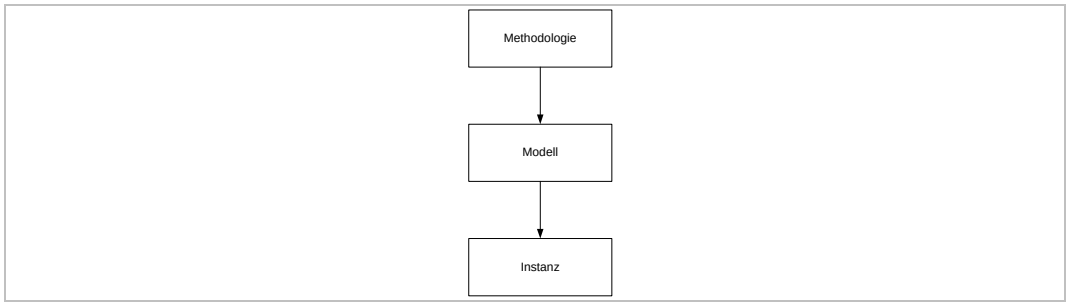


Abb. 40 Zusammenhang Methodologe, Modell und Instanz

Die Evaluationsmatrix besteht also aus fünf Bereichen, die im Weiteren genauer beschrieben werden. Jede Frage wurde mit einem Gewichtungsfaktor zwischen eins und drei versehen. Auf jede Frage kann mit einem Wert zwischen eins und fünf geantwortet werden, wobei diese Faktoren in einem Evaluationsleitfaden vorher festgelegt wurden. Dieser Evaluationsleitfaden kann in Anhang A eingesehen werden.

Die Fragen innerhalb des Evaluationsleitfadens wurden dabei unter Anlehnung an [HeEtAl04], [MooSha94] und [HoGoBr93] definiert, wobei einige (speziell für die Bereiche Reuse und Personalisierung) auch auf Basis unseres semantischen Kommunikationsmodells entstanden sind.

Verständlichkeit der Dokumentation der Methodologie

Hier betrachten wir die uns zur Verfügung stehenden Dokumente, die die Methodologie beschreiben. Sind diese gut strukturiert und einfach zu lesen? Fördert also deren Art, Aufbau und Umfang die Lesbarkeit und Verständlichkeit der Methodologie? Eine Beschreibung auf 100 Seiten kann gut sein, da sie sehr ausführlich ist, wenn sie aber schwer verständlich geschrieben ist, wären 10 Seiten besser, die das wichtigste Grundprinzip veranschaulichen. Werden für die Anwendung der Methodologie spezifische Vorkenntnisse vorausgesetzt? Oder kann man diese ohne jegliches Grundwissen sofort anwenden? Wenn Vorkenntnisse benötigt werden, beschreiben wir auch, welche genauer benötigt werden. Desweiteren betrachten wir die Vollständigkeit der uns vorliegenden Dokumentation: Sind alle Teilbereiche ausreichend beschrieben oder fehlt an der einen oder anderen Stelle etwas, was zur Verständlichkeit und zur Ausführung der Methodologie benötigt würde?

Applikation der Methodologie

Bei der Applikation der Methodologie sind diverse Aspekte für uns interessant: Kann diese direkt angewandt werden oder muss sie zuerst an eine Domäne angepasst werden? Sind Tools für die Anwendung vorhanden, die direkt auf die Methodologie zugeschnitten sind oder ggf. dafür verwendet werden können? Andererseits soll aber auch die Toolabhängigkeit der Methodologie unter den Prüfstand kommen. Ist die Methodologie kostenlos verfügbar und handelt es sich vielleicht auch um einen anerkannten Standard? Ist die Methodologie im industriellen und akademischen Umfeld bekannt und weit verbreitet? Desweiteren wird die Interoperabilität untersucht: wie ist die Integration mit anderen Methodologien beschrieben bzw. möglich? Wird eine spezielle Ausgangsbasis vorausgesetzt und wird eine bestimmte Auszeichnungssprache in diesem Zusammenhang vorgeschrieben oder empfohlen? Ist die Methodologie bei der Anwendung konsistent oder stellen sich verschiedene Inkonsistenzen heraus?

Qualität des Ergebnisses

Im Weiteren wurde dann das Ergebnis der Methodologie und dessen Qualität untersucht: Kann das Ergebnis direkt im anzunehmenden Arbeitsumfeld eingesetzt werden? Ist es leicht zu pflegen und zu warten, also auch auch flexibel und erweiterbar? Wird die Methodologie öfters angewandt, ist dann auch eine Verknüpfung der Ergebnisse möglich oder scheint dies eher ausgeschlossen? Wie sieht es mit der Weiterverarbeitung des Ergebnisses aus: sind dafür Tools vorhanden und kann das Ergebnis leicht unter Nutzung von Standards weiterverarbeitet werden?

Personalisierung

Die Kriterien für die Untersuchungsbereiche Re-Use und Personalisierung stehen in direktem Zusammenhang zum verwendeten Referenzmodell – dem semantischen Kommunikationsmodell – und können hierin sehr gut aufgezeigt werden.
Für zukünftige möglichst automatisiert ablaufende Personalisierungsmöglichkeiten ist demnach wichtig, dass der kommunikative Rahmen in den Teilen beschrieben ist, die

zukünftig Ableitungen der Art zulassen, welche Instanzen in welchem kommunikativen Rahmen entstanden sind und sich über logische Regelbildung für welche Art von anderen kommunikativen Rahmen ebenfalls anbieten, verwendet zu werden. An dieser Stelle ist auch ein großer Deckungsbereich mit Kriterien, die für den Bereich Re-Use erarbeitet werden, vorhanden, da es sich prinzipiell um eine spezielle Art des Re-Use handelt, nämlich den, Instanzen *personalisiert* weiter zu verwenden. Als Kriterien definierten wir infolgedessen Fragen zur Definition der Wissensnetze auf Empfänger- und auf Senderseite und die Frage zur Existenz eines Rollenkonzepts, das heisst – auf das semantische Kommunikationsmodell angewendet – das Vorhandensein eines Metadatums, welches für die Definition des Rollenkonzepts vorgesehen ist, nicht mit der Nachricht an den Empfänger geschickt wird, sondern systemintern für Steuerungs- und Ableitzwecke verwendet werden kann.

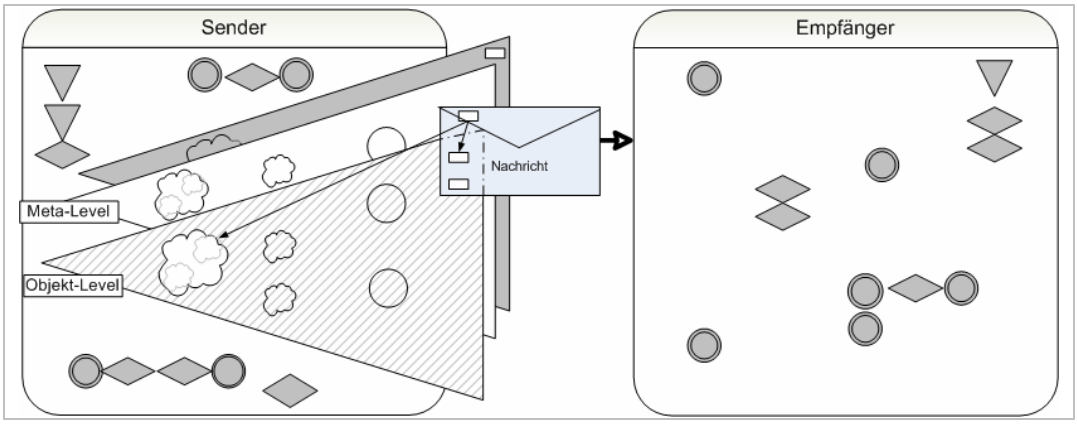


Abb. 41 Kriterien für optimierte Personalisierungmöglichkeiten im semantischen Kommunikationsmodell

Die Bereiche, die für Personalisierung optimalerweise formal beschrieben sein sollten, sind in Abb. 41 im semantischen Kommunikationsmodell grau hinterlegt:
- das vorausgesetzte Wissensnetz des Empfängers,
- das Wissensnetz des Senders und die
- Aufnahme des Rollenkonzepts auf einer Meta-Datenebene.

Re-Use

Bei der Kriterienfindung für einen möglichst zukünftig optimierten Re-Use Prozess ist uns wichtig, dass die Methodologie Anregungen dazu gibt, die Verankerung der zukünftigen Instanzen in Raum und Zeit ableiten zu können und auf eine standardisierte Terminologie und innere Struktur hin zu wirken. Optimalerweise sollte dementsprechend die Methodologie dazu anleiten, Module (die dazu gehörigen Instanzen sind die Nachrichten) zu definieren, deren räumliche und zeitliche Einordnung definiert wird.
Anhand des semantischen Kommunikationsmodells kann man sich das wiederum sehr gut veranschaulichen (siehe Abb. 42): das Ergebnis der Methodologie – das entstehende Datenmodell – muss für jedes in der Modulbildung verwendete komplexe/atomare Datum (auf Meta- und Objektebene!) festlegen, wie dieses innerlich strukturiert sein soll, d.h. im Falle eines komplexen Datums aufnehmen, aus welchen Daten es besteht, in welcher Reihenfolge und wie oft diese vorkommen können und im Falle eines atomaren Datums, wie dieses repräsentiert werden kann/soll und wie die innere sprachliche Struktur der zukünftigen Instanz sein soll (Schreibregeln, Benennungsregeln). Desweiteren muss die Methodologie vorsehen, dass die Struktur

der Nachricht, die an den zukünftigen Empfänger gesendet wird, eindeutig aus der Struktur der Moduldefinition ableitbar ist, so dass eine Rückführung möglich ist, aus welchen Daten auf der Objekt- und aus welchen Daten auf welcher Meta-Ebene sich die Nachricht aus den darüber konkretisierten Instanzen zusammensetzt.
Desweiteren muss das entstehende Modell Aussagen darüber zulassen können, welches komplexe/atomare Datum für sich alleine stehen kann und bei zukünftiger Verwendung dementsprechend unabhängig davon, welcher kommunikative Rahmen vorliegt, einsatzbereit und verständlich bleibt.
Für die Verankerung in der Zeit sollte die Methodologie optimalerweise den Anwender der Methodologie dabei unterstützen, bereits bei der Modellbildung Überlegungen zu treffen, bei welchen Daten zeitlich abhängige zukünftige Instanzen vorliegen werden und dies entsprechend virtuos im Modell bereits vorgesehen werden kann. Dies kann auf einer dafür zum Einsatz kommenden Metaebene geschehen, die die Verwendung einer Variablen an den entsprechenden Stellen vorsieht und beispielsweise systemintern dann verwendet werden kann, um die Instanzen durch Import aus anderen Systemen bei Laufzeit zu generieren.

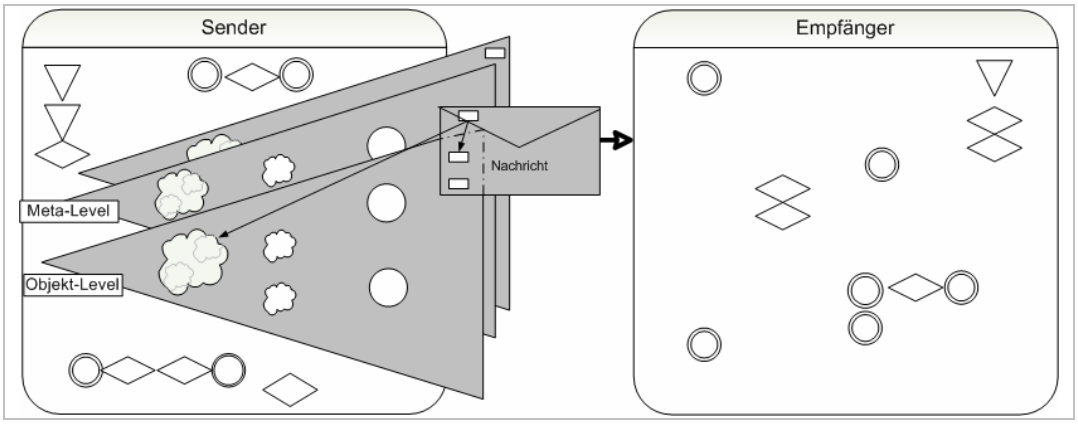


Abb. 42 Kriterien für optimierte Re-Use-Möglichkeiten im semantischen Kommunikationsmodell

8.2 Erfahrungen beim Anwenden der linguistischen Modellierungsmethodologien

Information Mapping™

Das Ziel der Methodologie ist es, Informationen jeder Art so zu strukturieren, dass die daraus entstehenden Dokumente zu wirkungsvollen Werkzeugen für ihre Benutzer (= Leser) werden. Sie führt daher den Autor zu einer Schreibweise, die immer den Benutzer und seine Bedürfnisse in den Mittelpunkt stellt. Durch die Anwendung von Information Mapping™ sollen auch bei großem Dokumentationsumfang kleine, in sich verstehbare Informationseinheiten entstehen, die problemlos in unterschiedlichen Kontexten wiederverwendet werden können. Aufgrund ihrer spezifischen Struktur und Gestaltung können die Dokumente in verschiedenen Medien ohne zusätzliche Umstrukturierung jeweils optimal produziert werden.
Information Mapping™ basiert auf der Anwendung eines modularen Systems:

- Analyse der Informationen unter Berücksichtigung der Leserbedürfnisse (zielgruppen- und objektorientiertes Vorgehen).
- Klassifizierung der Informationen in sieben Informationsarten, für die jeweils empfohlene Darstellungsarten gegeben werden.

- Aufbereitung der Informationen mit zwei neuen Informationseinheiten (Blocks und Maps)
- Verarbeitung der Informationen unter Einhaltung von Sieben Prinzipien (Regeln auf der Basis wissenschaftlicher Erkenntnisse).

Analysieren:

Grundlage der Methodologie ist die Analysephase, in der die Bedürfnisse der Zielgruppe bestimmt und die zu vermittelnden Informationen darauf abstimmt werden. Folgende Fragen werden beantwortet:

- Welche Informationen benötigt die Zielgruppe?
- Welchen Zweck muss die Information erfüllen?
- Wie muss die Information strukturiert sein?
- Welche Informationstiefe benötigt die Zielgruppe?“

Dies entspricht der analytischen Vorgehensweise beim Funktionsdesign® (vgl. 6.7), wobei das Funktionsdesign® hier stärker in der Definition verschiedener Dokumentarten ist, während bei Horn bereits in Richtung Strukturierung und Tiefe der Information analysiert werden soll.

Da keine hinreichende Dokumentation bei Anwendung der Methodologie vorliegt, bleibt unklar, wie das Ergebnis aus dieser Analysephase konkret auszusehen hat. Vor allem in den letzten Punkten – Informationsstruktur- und tiefe ist nicht verständlich, wie das aussehen soll und in welcher Art und Weise und ob überhaupt das schriftlich festgehalten werden muss.

Kritikpunkt: Beim Anwenden der Methodologie steht man vor dem bereits bekannten Problem der eher nicht so trivialen Fälle aus den Bereichen der technischen Dokumentation: was tun, wenn es verschiedene Zielgruppen gibt? Was tun, wenn eine zukünftige Zielgruppe zur Erfüllung ihres Informationsbedürfnisses mehrere verschiedene Dokumente aus unterschiedlichen Quellen bezieht.

Ergebnis aus der Analysephase:

Zielgruppen und benötigte Informationen:

Ein *Requirements Engineer* (= AUTOR) beschreibt einen *Use Case* in der Anforderungsspezifikation. Ein *Software-Engineer* möchte auf Basis der Daten, die er der Anforderungsspezifikation entnehmen kann, ein Design Dokument erstellen, d.h. er benötigt diejenigen Daten, die es ihm ermöglichen, das zukünftige System zu strukturieren und Abläufe exemplarisch zur erneuten Rücksprache mit Requirements Engineer und Kunden bereitzustellen. Er trifft weiterhin auf Basis der Anforderungen technologische Entscheidungen.Ein *Technischer Redakteur* verwendet sowohl die Anforderungsspezifikation als auch das Design Dokument und entwickelt daraus zum einen ein Benutzerhandbuch für den *Bankangestellten* und eine Kundenbroschüre für den *Endkunden*. Daten, die er dazu braucht, sind unterschiedlich von ihrer Tiefe her: für das Erstellen der Kundenbroschüre reicht ihm ein Datensatz, der beschreibt, welche Telefonnummern in welchen Ländern gewählt werden müssen und welche Daten der Kunde bereits vorher bereithalten sollte, um seine Karte sperren lassen zu können. Falls die Daten zum Zeitpunkt der Erstellung der verwendeten Dokumentationen nicht bereitlagen, benötigt der Technische Redakteur einen Hinweis, woher er die relevanten Daten beziehen kann. Für das Erstellen des Benutzerhandbuchs muss ihm neben den Dokumenten entweder das System zur Verfügung gestellt werden oder gültige screenshots der entsprechenden Abläufe. Aus den Dokumenten benötigt er eine Übersicht über die Ablauffolge der

Handlungen, die der Bankangestellte mit dem System ausführen muss, sobald ein Kunde anruft, um die Kreditkarte sperren lassen zu wollen.

Organisieren: Entwickeln von Informations-strukturen

Information Mapping™ bietet einen Ansatz zum Entwickeln von Informationsstrukturen durch die Anwendung von 7 Prinzipien, die darauf aufbauen, wie Menschen lernen und lesen.

Diese 7 Prinzipien sind:

- Gliederung
- Betitelung
- Relevanz
- Einheitlichkeit
- Gleichwertigkeit der Informationsträger
- Verfügbarkeit von Einzelheiten
- Systematische Gliederung und Betitelung

Hier tritt durch die vorliegende mangelnde Dokumentation das Problem auf, dass an dieser Stelle unklar bleibt, wie diese 7 Prinzipien umgesetzt werden sollen.

Existierende Inhalte sollen in so genannte Informationsblöcke eingeteilt werden, die betitel werden sollen. Für diese Betitelung hat Robert Horn ca. 200 unterschiedliche Arten definiert wie z.B. Analogie, Checkliste, Kommentar, Definition, Theorem etc. Diese Arten wiederum basieren alle auf den folgenden sieben grundlegenden Informationstypen:

- Prozeduren – Schritte
- Prozessbeschreibungen – Erklärungen
- Struktur – Beschreibungen
- Konzepte – Definitionen und Beispiele
- Prinzipien – Regeln
- Fakten – Physikalische Eigenschaften
- Klassifikationen – Typen und Kategorien

Ergebnis Organisieren

Dies entspricht unserer Meinung nach genau dem, was man beim Funktionsdesign® bei „Ermitteln der Funktionalen Einheiten“ macht mit dem Unterschied, dass man jetzt den Vorteil hat, dass 200 bereits ermittelte Informationsblöcke vorliegen. Da uns hier leider keine Liste vorliegt, gehen wir davon aus, dass wir genau die Funktionalen Einheiten aus dem Funktionsdesign® nun auch hier als Informationsblöcke nehmen würden.

Präsentieren

Am Ende muss der Inhalt präsentiert werden. Dieser letzte Schritt findet nach der Analyse der Bedürfnisse der Zielgruppe, dem Organisieren des Inhalts in Informationsblöcke und deren Benennung statt. Erst dann wird der Inhalt in ein ‚bedeutungsvolles Ganzes’ zusammengestellt, das der Zielgruppe präsentiert werden soll. Vier grundlegende Prinzipien liegen den hierbei stattfindenden Formatier-Entscheidungen zugrunde:

“Chunking” - ist jegliche Information in nicht-teilbare Blöcke aufgeteilt? Labeling – Sind alle Informationsblöcke mit einem geeigneten Label versehen? Relevanz – ist alles innerhalb eines Informationsblocks wirklich essentiell, um die Bedürfnisse der Zielgruppe zu erfüllen? Hierarchie – sind hierarchische Label einheitlich dazu verwendet, um dem Leser verständlich zu machen, wo er gewesen ist und wohin er

geleitet wird? Da eine lange Sequenz an Informationsblöcken nicht mehr Klarheit vermittelt als eine Serie von Paragraphen, empfiehlt Horn zu einem Topic Sektionen zu bilden, die zwischen 1 und 9 Blöcke beinhalten. Diese Gruppierungen nennt er Information Maps.

Das Problem für uns war an dieser Stelle erneut, dass wir keine weiterführenden Materialien hatten. Hier bleibt vieles anhand der verwendeten Kurzdokumentationen unklar, beispielsweise, ob unterschiedlichen Typen von Information bei Horn bereits deren gestalterische Umsetzung zugrunde liegt. Wesentlich an Erkenntnis hier ist für uns, dass prinzipiell das, was hier noch an Strukturierung und Labeling hineingearbeitet werden soll, verknüpft ist mit dem Zusammenfassen von blocks zu maps (wobei nach Horn eben nicht mehr als 9 blocks verwendet werden sollen) und so ein map quasi einem topic entspricht, der auch mit einem label versehen werden soll. Es geht jedoch nicht hervor, ob dieses Label als Instanz dann irgendwo im Text erscheinen soll und ob es für das Labeling der maps auch bereits vorderfinierte von Horn gibt, die bereits entsprechend geeignete blocks bündeln und so als Modul zur Verfügung stehen.

Funktions-design

Das Funktionsdesign® setzt auf der Makro-Ebene mit einer geforderten Spezifikation der Dokumentarten an. Dabei soll der Funktionsdesign-Entwickler die Gesamtmenge des Produktwissens betrachten. Im nächsten Schritt klärt er, welche Zielgruppen welche Informationen zu welchem Zeitpunkt benötigen. Auf diesem Hintergrund ordnet er das Produktwissen einzelnen Dokumentarten zu. Dabei gilt es, eine sinnvolle und trennscharfe Klassifikation aller Dokumentarten zu entwickeln.

Ergebnisse Makroebene

Zielgruppen und Dokumentarten:

Requirements Engineer beschreibt Use Case in der Anforderungsspezifikation.

→*Software-Engineer* entwickelt daraus ein Design Dokument, d.h. er strukturiert das zukünftige System und stellt Abläufe exemplarisch zur erneuten Rücksprache mit Requirements Engineer und Kunden bereit. Er trifft weiterhin auf Basis der Anforderungen technologische Entscheidungen..

→*Technischer Redakteur* verwendet Anforderungsspezifikation und Design Dokument und entwickelt daraus zum einen ein Benutzerhandbuch für den Bankangestellten und eine Kundenbroschüre für den Endkunden

→ →*Endkunde*

→ →*Bankangestellter*

Auf der Mikro-Ebene sollen danach die Funktionalen Einheiten ermittelt werden.Die Dokumentarten sind im Funktionsdesign® aus Elementen zusammengesetzt, deren Definition primär an deren (kommunikativer) Funktion ausgerichtet ist. Zu unterscheiden sind dabei die Ebenen Dokumente, Sequenzmuster, Funktionale Einheiten, Auszeichnungselemente. Dabei besteht jede Dokumentart aus einer überschaubaren Menge solcher Elemente. Den schreibtechnischen Kern bilden die Funktionalen Einheiten.[…] Die grundlegende Neu- und Weiterentwicklung der notwendigen Funktionalen Einheiten läßt sich sehr gut als fiktiver Dialog zwischen Funktionsdesign-Entwickler und potentiellem Anwender beschreiben. Dabei steht im Hintergrund immer das kommunikative Ziel der Dokumentart. Als einKritikpunkt an dieser Stelle ist festzustellen, dass das Funktionsdesign® keine Antwort auf die Frage bietet, wie vorgegangen werden soll, wenn es bei einer Dokumentart wie oben mehrere

Empfänger gibt (Anforderungsspezifikation) oder wenn ein Empfänger mehrere Dokumentarten zur Verfügung gestellt bekommt (Technischer Redakteur).

Ergebnisse Mikroebene

Fiktiver Dialog zwischen Requirements Engineer und Software *Engineer* →
Funktionale Einheiten, die zur Verfügung gestellt werden müssen:
Angaben zum Anwendungsfall:
- Name des Anwendungsfalls
- Voraussetzung für Inkrafttreten des Anwendungsfalls
- Ablauf des Anwendungsfalls für den Systembenutzer
  - Zielangabe
  - Voraussetzung
  - Handlungsaufforderung
  - Handlungsschritt
  - Resultat
- Möglicher Problemfall
- Verhalten bei Auftreten des Problemfalls
- Restriktionen für den Ablauf des Anwendungsfalls im System
  - Systemabhängigkeit
- Angaben zum Systembenutzer:
  - Alter
  - Sprache
  - Eigenschaft
  - Zu beachtende Auswirkung (auf Abfolge oder Layout)

Fiktiver Dialog zwischen Requirements Engineer und Technischem Redakteur →
Funktionale Einheiten, die zur Verfügung gestellt werden müssen:

Angaben zum Anwendungsfall:
- Name des Anwendungsfalls
- Voraussetzung für Inkrafttreten des Anwendungsfalls
- Ablauf des Anwendungsfalls für den Systembenutzer
  - Zielangabe
  - Voraussetzung
  - Handlungsaufforderung
  - Handlungsschritt
  - Resultat
- Möglicher Problemfall
- Verhalten bei Auftreten des Problemfalls
- Ablauf des Anwendungsfalls für den Endkunden:
  - Zielangabe
  - Voraussetzung
  - Handlungsaufforderung
  - Handlungsschritt
  - Resultat
- Möglicher Problemfall
- Verhalten bei Auftreten des Problemfalls
- Angaben zum Systembenutzer:
  - Eigenschaft
  - Zu beachtende Auswirkung (auf Abfolge oder Layout)

Angaben zum Endkunden:
Eigenschaft
Zu beachtende Auswirkung (auf Abfolge oder Layout)

Für die ermittelten Funktionalen Einheiten müssen Festlegungen getroffen werden, die die Autoren bei der Verwendung verbindlich befolgen müssen:

- Verwendung,
- Sequenzierung,
- Innere Struktur und Formulierung/syntaktisches Muster,
- Gestaltung/explizite Kennzeichnung.

Der Funktionsdesign-Entwickler muss für jede einzelne Funktionale Einheit prüfen, ob es für jede dieser Kategorien sinnvolle, die konsistente Textproduktion unterstützende Festlegungen gibt.

Ergebnis Festlegungen

Funktionale Einheit Handlungsaufforderung:

- Verwendung: In der Regel soll eine Handlungsaufforderung nur zu einer Handlung anleiten, nicht zu mehreren gleichzeitig. Machen Sie Ihre Entscheidung abhängig von der Komplexität der auszuführenden Handlungen und den Wissens- und Könnensvoraussetzungen des Empfängers.
- Sequenzierung: Handlungsaufforderungen sind der Kern von Handlungssequenzen. Zulässige Vorgänger von Handlungsaufforderungen sind die Funktionalen Einheiten Zielangabe, Handlungsvoraussetzung, Resultat. Zulässige
- Nachfolger sind die Funktionalen Einheiten Resultat, Zielangabe, Auftretender Problemfall, Verhalten im Problemfall.
- Syntaktische Struktur: Eine Handlungsaufforderung besteht immer mindestens aus dem Handlungsverb und dem handlungsrelevanten Gegenstand. Im einfachsten Fall also: Button drücken. Zulässig sind auch handlungsspezifizierende Informationen, z.B. Button CTRL drücken. Formulieren Sie Handlungsaufforderungen immer im Imperativ mit Verb-Letztstellung. Beispiel: Eingabe bestätigen. Alternative Formulierungen sind nicht zulässig. Prüfen Sie darum die Handlungsaufforderungen besonders sorgfältig. Vermeiden Sie unbedingt die verbreitete ist-zu-Formulierung, z. B. Die Eingabe ist zu bestätigen.oder die Passiv-Formulierungen, z. B. Die Eingabe wird bestätigt.
- Gestaltung: Die optische Gestaltung von Handlungsaufforderungen übernimmt das Erfassungswerkzeug MS Word automatisch, wenn Sie die Formatvorlage Handlungsaufforderung wählen.Festgelegt ist eine automatische Numerierung aller Handlungsaufforderungeneiner Handlungssequenz. Die Nummer erscheint farbig (rot). Der Absatzabstand nach oben beträgt 4 Pt, nach unten 2 Pt.“

DITA

Bei DITA sollen die Inhalte in Topics aufgeteilt werden

Zitat Z38: [DITAAB07]

„A topic is a unit of information with a title and some form of content, short enough to be specific to a single subject or answer a single question, but long enough to make sense on its own and be authored as a unit.

Topic structure

All topics have the same basic structure, regardless of topic type: title, description or abstract, prolog, body, related links, and nested topics.“

Ergebnis Topics

Anwendungsfall-Beschreibung für den Systembenutzer

Beschreibung des Systembenutzers

Anwendungsfall-Beschreibung für den Endkunden:

Beschreibung des Endkunden:

Im nächsten Schritt sind nach DITA die Inhaltskategorien zu definieren.

Zitat Z39: [DITAAB07]
„Information typing is the practice of identifying types of topics that contain distinct kinds information, such as concepts, tasks, and reference information. Topics that answer different kinds of questions can be categorized as different information types. The base topic types provided by DITA provide a usable starter set that can be adopted for immediate authoring. Information typing is part of the general authoring approach called structured writing, which is used across the technical authoring industry to improve information quality. It is based on extensive research and experience, including Robert Horn’s Information Mapping, and Hughes Aircraft’s STOP (Sequential Thematic Organization of Proposals).

- *Concepts: DITA concept topics answer "What is..." questions. They include a body-level element with a basic topic structure, including sections and examples.*
- *Tasks: Task topics answer "How do I?" questions, and have a well-defined structure that describes how to complete a procedure to accomplish a specific goal.*
- *Reference: Reference topics describe regular features of a subject or product, such as commands in a programming language. “*

Ergebnis: Inhaltskate-gorien

T1 Task:Anwendungsfall-Beschreibung für den Systembenutzer

C1 Concept: Beschreibung des Systembenutzers

T2 Task: Anwendungsfall-Beschreibung für den Endkunden:

C2 Concept: Beschreibung des Endkunden

Darauf aufbauend werden die Topics in so genannten Maps organisiert.

Zitat Z40: [DITAAB07]

„DITA maps are documents that collect and organize references to DITA topics to indicate the relationships among the topics. They can also serve as outlines or tables of contents for DITA deliverables and as build manifests for DITA projects.

Maps describe the context in which the topics will be read – the audience, platform, relationships, requirements of the information set. In this way, the topics themselves become relatively context-free, and can be more easily used and reused in many different contexts, as defined by maps.“

Ergebnis: Organisation in Maps

Beispiel Use Case-Beschreibung aus Sicht des Endnutzers

Zielgruppe: technischer Redakteur

Ausgabeplattform: Autorenumgebung

Verknüpfungen: T2 C2

Voraussetzungen: Import gültiger Telefonnummern erfolgt.

Strukturelle Textkontrolle

Die strukturelle Textkontrolle nach Ley wurde in der uns zugrunde liegenden Dissertation von Ley für die Domäne Prüfprozeduren an Flugzeugen entwickelt. Die hierbei entwickelten Modellierungsprimitive und Informationsklassen sind speziell für diese Domäne hervorragend geeignet. Das Anwenden der Strukturellen Textkontrolle in unserem Kontext setzt das Anpassen an diese völlig andere Domäne voraus, was unseren zeitlichen Rahmen dieses Projekts sprengen würde. Es sollte aber erwähnt werden, dass dies Gegenstand einer Untersuchung im Rahmen einer Abschlussarbeit sein könnte.

8.3 Erfahrungen beim Anwenden der Ontologie-Methodologien

In diesem Abschnitt werden die Ontologie-Methodologien nicht erneut beschrieben, sondern lediglich die Erfahrungen und Anmerkungen, die bei der Applikation der Methodologien gemacht wurden, erläutert.

OTKM

On-To-Knowledge: Eine Feasability Study wurde nicht gemacht, da im Rahmen unserer Evaluation bereits beschlossen wurde, eine Ontologie zu erstellen. In der Kickoff-Phase wurde eine Requirements Specification erstellt für den Use Case ‚Kreditkarte Sperren' (siehe Kapitel 4.3). Ausserdem sollte untersucht werden, was die Ontologie beschreiben und unterstützen soll. Die Ontologie beschreibt die wichtigsten Konzepte im Anwendungsfall "Kreditkarte sperren" und unterstützt die maschinelle Weiterverarbeitung dieses Anwendungsfalls im Software Engineering und der technischen Dokumentation. Folgende weitere Informationen wurden gesammelt:

Domain:	Finanzbereich, Kreditkartenmanagement
Date:	2007-11-20
Ontology Engineer:	Florian Lautenbacher & Tanja Sieber

Goal of the Ontology:	Unterstützt den Software-Entwicklungs- und Dokumentationsprozess
Domain and Scope:	Sperren von Kreditkarten, Finanzbereich, Kreditkartenmanagement
Supported Applications:	existierende Banksysteme, Software-Entwicklung

Knowledge Sources:	Existierende Ontologien, Kreditkartenbeschreibungen, eGK-Anforderungsdokumente
Users and Use Cases:	Software-Entwickler, technische Redakteure, teilweise: Endanwender, Bankangestellter
Competency Questions:	-
Potentially reusable ontologies:	SUMO_Financial.owl, CongoProcess.owl

Zitat Z41: [StEtAl01]

„It should also guide an ontology engineer to decide about inclusion, exclusion, and the hierarchical structure of concepts in the ontology.“

Wie genau dies geschehen soll, wird aber nicht in der Quelle beschrieben. Der Output des ersten Schrittes ist also eine Taxonomie. Dabei liegt aber keine Hinführung vor, wie denn die Taxonomie erstellt werden soll: welche Begrife inkludiert werden sollen und welche nicht, ob top-down, bottom-up oder ein middle-out-Verfahren angewandt werden soll.
Die erstellte und dann gleich mit Eigenschaften verbundene Taxonomie sieht aus wie in Abb. 43 dargestellt.

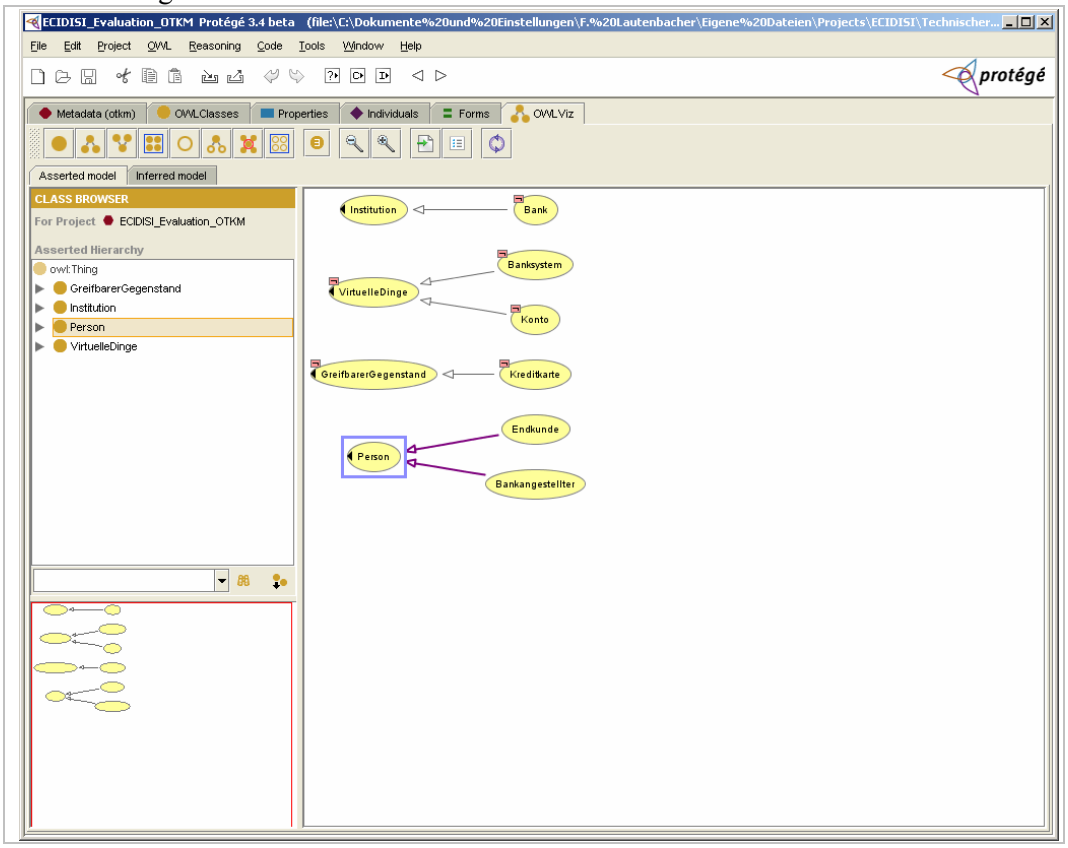


Abb. 43 Ontologie gemäß OTKM

Diese wurde vom Autor nach dem Prinzip Middle-Out erstellt. Eine Suche nach existierenden Ontologien in diesem Bereich erfolgte ebenfalls und brachte die CongoProcess.owl (Englische Ontologie zu Credit Cards) und SUMO_Finance.owl (allgemein zu Finanzen) zu Tage. Bei der Suche sollte eigtl. bereits feststehen, welches

Ontologieformat (OWL, WSML, F-Logic, etc.) später gewählt werden soll, da sonst im Nachhinein eine Wiederverwendung eher schwierig wird, wenn eine WSML-Ontologie gefunden wurde, aber eine OWL-Ontologie erstellt werden soll. Dann können nur die Konzepte der Ontologie wiederverwendet werden (wenn der Ersteller der Ontologie die Sprache der Ontologie kennt), im anderen Fall wäre es sogar möglich direkt auf die Ontologie (beispielsweise in OWL) aufzusetzen.
Bei der Suche lieferte "credit card ontology" ganz andere Suchergebnisse als beispielsweise "credit card filetype:owl"; "cancel credit card filetype:owl" (was für unseren Use Case die notwendigen Dinge beinhalten würde) liefert keinerlei Ergebnisse. Die gefundenen existierende Ontologien (CongoProcess.owl) beinhalten bereits Konzepte, die für das Verständnis einer Kreditkarte wichtig sind: CreditCardType, ValidityType, CreditCardNumber, cardExpiration, etc. sagen aber nichts über Zielgruppen, Sender, Besitzer, etc. aus.
Beim Erstellen der Taxonomie werden oft einige Punkte vergessen, die man anfangs als Attribut auffasst und nicht in die Taxonomie übernimmt. Erst beim Hinzufügen der Attribute (wie z.B. Konto bei Endkunde) stellt man fest, das diese eigentlich ein eigenes Konzept wären, und daher in die Ontologie bzw. Taxonomie aufgenommen werden sollten. Am Ende wurde eine schemahafte Taxonomie erstellt mit einigen Konzepten, die ausser "Thing" allerdings keine gemeinsame Oberklassifikation haben.
Am Ende der Erstellung der Ontologie bleiben immer noch viele Fragen offen: wurde die Ontologie so richtig dargestellt? Sind alle Konzepte beinhaltet? Was fehlt? Dies kann so unter Nutzung der Methodologie nicht beantwortet werden.

Bei der Erstellung der Ontologie wurde auch bereits direkt in den nächsten Schritt (Refinement) übergegangen und dabei die Konzepte mit Domänenexperten diskutiert und Relationen und Attribute hinzugefügt. Eine Formalisierung wird erst nach Angabe der Relationen und Attribute angestrebt, wobei sich dann wieder einige zusätzliche Punkte ergeben, die berücksichtigt werden müssen. Also fand eine Wiederholung des Refinements statt wie dies bereits in der Methodologie vorgegeben war.
Beim Kickoff sollte auch angegeben werden, wer denn die Zielgruppe dieser Ontologie ist sowie welche möglichen Nutzungsszenarien denn bestehen könnten. Zielgruppen sind hier der Software-Entwickler und der technische Redakteur. Diese können die Ontologie einsetzen, um verwendete Begrifflichkeiten fachlich sauber und exakt zu definieren und in verschiedenen Bereichen auf dieselben Definitionen zuzugreifen.
Es werden keine Vorgaben für eine spezielle Sprache getroffen. Soweit möglich soll in diesem Use Case aber die Web Ontology Language OWL eingesetzt werden. Daher erfolgte in einem nächsten Schritt die Umsetzung der Ontologie in OWL mittels des Tools Protégé.
In Protégé wurde zuerst die Taxonomie eingepflegt sowie die disjointClasses-Informationen (Banksystem und Konto sind sicher zwei unterschiedliche Dinge, wohingegen ein Bankangestellter durchaus auch Kunde bei der Bank sein kann). Anschließend wurden die ObjectProperties und DatatypeProperties hinzugefügt. Dabei werden weitere Ungenauigkeiten in der zuvor auf dem Papier/Whiteboard erstellten Ontologie entdeckt und gleich ausgemerzt.
Bei der Erstellung der Ontologie wurden die Konzepte existierender Ontologien (CongoProcess, SUMO_Finance) berücksichtigt, eine direkte Wiederverwendung fand unter anderem wegen unterschiedlichen Sprachen der Ontologie (die gefundenen waren Englisch, die zu erstellende Deutsch) nicht statt.

Die Evaluationsphase brachte folgende Ergebnisse:

Die Ontologie beschreibt das Sperren einer Kreditkarte sowie weitere darum herum notwendigen Konzepte und scheint daher die gestellten Anforderungen zu erfüllen. Ob sie allerdings auch für die Software-Entwicklung und technische Dokumentation im weiteren Verlauf nützlich sein wird, wird sich erst noch herausstellen. Die erstellte Ontologie kann in OWL-Quelltext in Anhang B gefunden werden.
Die Rollout- und Maintenance-Phase wurden bei dieser Evaluation nicht mehr durchgeführt, allerdings bei der Evaluation berücksichtigt.

METHON-TOLOGY

Neben [GoFC04] sehen wir uns für METHONTOLOGY ausserdem die Quelle [GomFer05] an, die METHONTOLOGY im Zusammenspiel mit dem Tool WebODE beschreibt.
METHONTOLOGY ist an sich wesentlich umfangreicher als beispielsweise die On-To-Knowledge-Methodologie (OTKM). Sie umfasst zusätzliche Managementaktivitäten und Unterstützungsaktivitäten, die hier nur teilweise betrachtet werden, sofern sie unsere Evaluation betreffen. Die interessanten Aspekte sind vor allem die Entwicklungsaktivitäten und hier schwerpunktmäßig die Konzeptualisierung.

METHONTOLOGY umfasst im Gegensatz zu anderen Methodologien den kompletten Lebenszyklus einer Ontologie: von der Erstellung, über die Pflege, Unterhalt, etc. Ähnlich zu OTKM muss auch hier anfangs (in der Phase der Spezifikation) angegeben werden, wer denn die Zielgruppe der Ontologie ist, was der Zweck und die Einsatzbereiche sind sowie wie die Ontologie später genutzt werden soll. Neben den Entwicklungsaktivitäten finden parallel die Management- und Unterstützungsaktivitäten statt. Die Entwicklungsaktivitäten sind Spezifikation, Konzeptualisierung, Formalisierung, Implementierung und Wartung.

Im ersten Schritt soll ein Glossar von Termen erstellt werden. Dieses umfasst bei unserem Use Case die folgenden Begriffe: Kreditkarte, Bank, Endkunde, Bankangestellter, Konto, Sperren.
Als Nächstes soll eine Taxonomie aus diesen Begriffen erstellt werden. Hier werden Bankangestellter und Endkunde zu Person abstrahiert, Kreditkarte und Konto zu Finanzkonzept sowie Bank zu Firma.
Beim Erstellen der Taxonomie werden vier Möglichkeiten vorgeschrieben: Subklassenbeziehungen, eine Disjoint-Decomposition, Exhaustive-Decomposition und Partitionierung. Eine Disjoint-Decomposition beinhaltet Konzepte in denen deren Instanzen überschneidungsfrei sind (so kann eine Firma kein Finanzkonzept sein). Exhaustive-Decomposition beschreibt eine Unterklassenzerlegung, so dass jede Instanz der Oberklasse immer gleichzeitig eine der angegebenen Unterklassen ist. Dies ist in unserem Beispiel nicht existent, ebensowenig wie die Partitionierung.
Im nächsten Schritt sollen ad-hoc-Beziehungen zwischen je zwei Konzepten definiert werden. Diese wären beispielsweise „Bankangestellter bedient Endkunde“, „Endkunde besitztKarte Kreditkarte“, „Kreditkarte gehörtZu Konto“, etc.
Danach müssen für jedes Konzept die Klassen- und Instanzattribute sowie die möglichen Instanzen selbst festgehalten werden. Die möglichen Instanzen sind teilweise nicht aufzählbar (z.B. für Person oder Firma, da nicht alle Personen und Firmen der Erde aufgezählt werden können). Es werden desweiteren Klassenattribute als Attribute mit festen Werten je Unterklasse und Instanzattribute, die variabel für jede Instanz sind, aufgezählt.
Daneben sollen noch Tabellen für Konstanten festgelegt werden sowie Axiome und Formeln definiert und beschrieben werden.

In unserem Beispiel konnten nur Instanzattribute gefunden werden, Instanzen wurden nur beispielhaft aufgezählt (Sparkasse Augsburg, Raiffeisenbank Miskolc) und Axiome und Formeln keine entdeckt. Aber an sich eine gute Hilfestellung in METHONTOLOGY, was alles beachtet werden sollte.
Die Dokumentation, die als Teilaktivität der Unterstützungsaktivitäten beschrieben wurde, ist direkt in der Konzeptualisierung enthalten. Anschließend folgt die Formalisierung und Implementierung. Beide Schritte sind unseres Wissens meist miteinander verwoben, da durch die Implementierung bereits eine Formalisierung stattfindet und nicht vorab zusätzlich eine formalisierte Darstellung der Konzepte erfolgen muss (erhöht nur den Arbeitsaufwand, bringt aber keinen zusätzlichen Mehrwert).
Zur Implementierung wird das Tool WebODE in [CoEtAl05] vorgeschlagen, dies ist aber zur Zeitpunkt der Evaluation [2007-11-21] nicht verfügbar. Die Implementierung erfolgte daher in Protégé ähnlich wie bei der letzten Evaluation von OTKM.

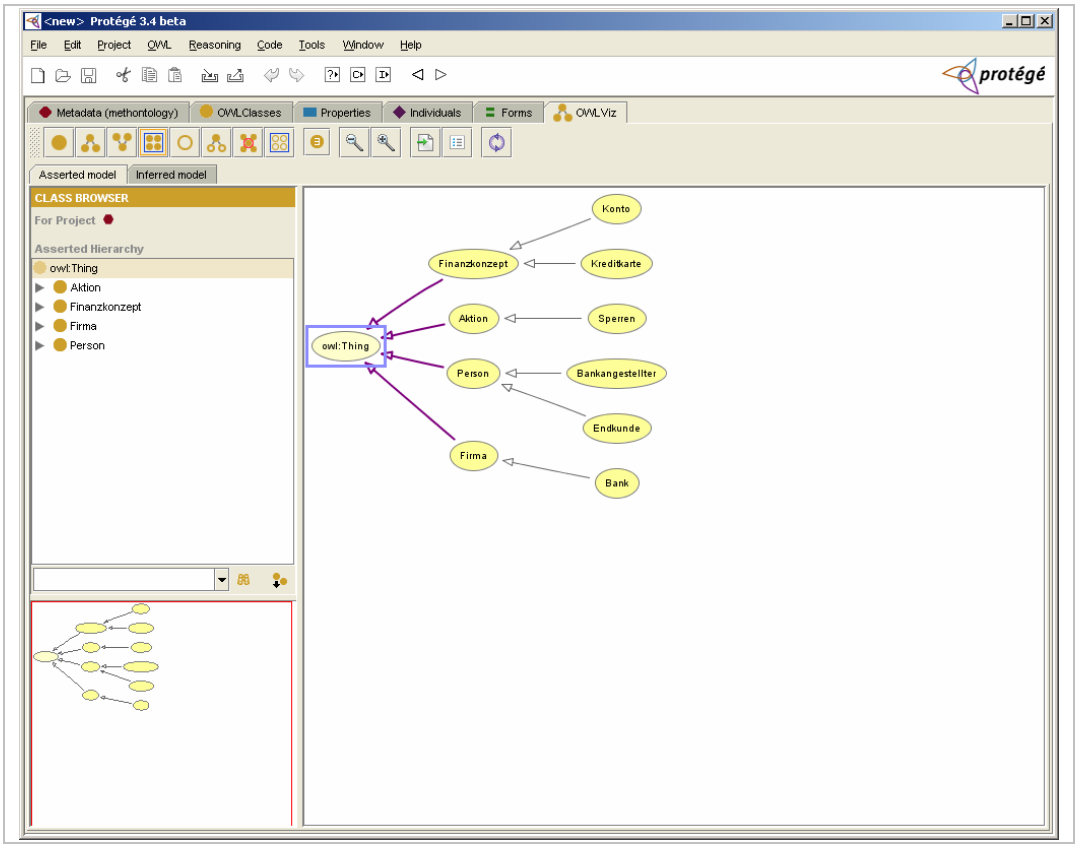


Abb. 44 Ontologie gemäß METHONTOLOGY

Noy & McGuinness

Noy und McGuiness beschreiben im Ontology Development Guide 101 eine Methodologie zum Erstellen von Ontologien. Dabei wird nicht der komplette Lebenszyklus betrachtet, sondern nur beschrieben wie die Konzepte für die Ontologie gefunden werden können.

Zuerst soll die Domäne und der Rahmen der Ontologie definiert werden: in unserem Fall Finanzdienstleistungen, Kreditkartenmanagement sowie Sperren einer Kreditkarte. Der genaue Rahmen ist (wie bereits bei den anderen Ansätzen) schwer zu begrenzen. Es muss ausserdem angegeben werden, wer der zukünftige Nutzer der Ontologie sein wird (Zielgruppe) und wer die Ontologie später warten wird. Desweiteren muss nun wieder eine Reihe von Kompetenzfragen erstellt werden, die die Ontologie im weiteren Verlauf beantworten soll.
Im zweiten Schritt soll untersucht werden, ob Ontologien existieren, die in diesem Bereich wiederverwendet werden können. Dies wurde bereits in den vorherigen Evaluationen vorgenommen und diese Ergebnisse hier berücksichtigt.

Im dritten Schritt werden die wichtigsten Kernkonzepte festgehalten und nieder geschrieben.
Beim Erstellen der Begrifflichkeiten muss (im Gegensatz zum Glossar anderer Ansätze wie METHONTOLOGY) keine Beschreibung zu den Begriffen hinzugefügt werden. Lediglich die Begriffe werden notiert. Darauffolgend soll aus diesen Begriffen eine Taxonomie erstellt werden. Es ist dem Entwickler selbst überlassen, ob er eine Top-Down / Bottom-Up oder Middle-Out-Strategie anwenden möchte. Aufgrund der vorher erstellten Kernkonzepte wird vermutlich eine Middle-Out-Strategie am besten greifen. In unserem Beispiel war allerdings eine Bottom-Up-Strategie am meisten erfolgsversprechend.
Als nächstes werden die Eigenschaften jeder Klasse definiert. Name zu Person, Prüfziffer für Kreditkarte, etc. Dazu die Typen dieser Attribute (String, Boolean, Number). Daraufhin werden für alle Eigenschaften (hier Slots genannt) die Domäne und der Bereich dieser Attribute definiert, sollte es sich nicht um primitive Datentypen sondern um weitere ontologische Klassen handeln. Im nächsten Schritt werden die Instanzen definiert.

Speziellen Wert legt der Ontology Development Guide 101 auf das Finden von Konzepten / Klassen in der Ontologie. Hier werden einige typische Fehler beschrieben und wie man diese vermeiden kann, bzw. auf was alles geachtet werden sollte.
Es wird ausserdem beschrieben, wie Eigenschaften gefunden werden und auch dass man sich auf die Schreibweise der Klassen und Eigenschaften ebenso wie der erstellten Instanzen vorher einigen sollte sowie welche Möglichkeiten es denn dazu gibt. Dies geht über die Beschreibungen von OTKM und METHONTOLOGY hinaus. Allerdings werden hier leider nur die Alternativen vorgestellt (Eigenschaften mit has- zu beginnen oder dass man sowohl MealCourse, Meal-Course als auch Meal_Course schreiben kann). Eine ***Handlungsrichtlinie*** wäre hier wünschenswert gewesen.

Zusammenfassend kann man feststellen, dass der Ontology Development Guide 101 zwar wesentlich genauer auf die Erstellung von Konzepten und Eigenschaften eingeht und hier auch einige Möglichkeiten aufzeigt und Tipps gibt, allerdings keinen kompletten Lebenszyklus der Ontologie beschreibt und daher auch keine Versionierungsmöglichkeiten, etc. kennt.

8.4 Evaluationsmatrix für linguistische Modellierungsmethodologien

Auf den nächsten Seiten (im Querformat) ist die Evaluationsmatrix für ausgewählte linguistische Modellierungsmethodologien zu sehen.

8.5 Evaluationsmatrix für Ontologie-Methodologien

Direkt darauf folgend ist ebenfalls die Evaluationsmatrix der vorher beschriebenen Ontologie-Methodologien dargestellt.

Modellierungsmethodologie:		*Information Mapping®*		*Funktionsdesign®*		*Strukturelle Textkontrolle*		*DITA*	
	Priorität	**Bepunktung**	**Berechnete Punktzahl**	**Bepunktung**	**Berechnete Punktzahl**	**Bepunktung**	**Berechnete Punktzahl**	**Bepunktung**	**Berechnete Punktzahl**
Kriterien	**(1=niedrig, 2=mittel, 3=hoch)**	**(1=niedrig, 5=hoch) siehe Evaluationsleitfaden**	**(Priorität * Bepunktung)**	**(1=niedrig, 5=hoch) siehe Evaluationsleitfaden**	**(Priorität * Bepunktung)**	**(1=niedrig, 5=hoch) siehe Evaluationsleitfaden**	**(Priorität * Bepunktung)**	**(1=niedrig, 5=hoch) siehe Evaluationsleitfaden**	**(Priorität * Bepunktung)**
Verständlichkeit der Dokumentation der Methodologie			**28**		**37**		**27**		**31**
Werden spezifische Vorkenntnisse vorausgesetzt?	2	5	10	5	10	3	6	5	10
Welche Zielgruppe ist gemeint?		jede Art von Autor		technische Redakteure, Product Engineers mit Schwerpunktausrichtung Dokumentation und Kommunikation		Informations-/Dokumentationsarchitekten		Informationsarchitekt	
Welche Vorkenntnisse werden benötigt?		nein		grundlegendes Interesse an Strukturierungsmethoden von Vorteil		linguistische Kenntnisse notwendig		Grundkenntnisse über XML, Architekturen und objektorientierten Ansätzen können sicherlich nicht schaden	
Fördert Art, Aufbau und Umfang der Dokumentation Lesbarkeit und Verständlichkeit?	3	3	9	5	15	3	9	3	9
Benutzte / Referenzierte Quellen angeben		Information Mapping - Die Methode (Internetseite: http://www.information-mapping-online.de/imap/methode.html); Information Mapping (http://coe.sdsu.edu/eet/articles/infomap/index.htm)		sar_Problemtypologie.pdf (Vorlesungsskript FHF Product Engineering von Prof. Schäflein-Armbruster WS2004/2005) und 03_FD_Die Methode1.pdf (Vorlesungsskript FHF Product Engineering von Prof. Schäflein-Armbruster SS2005)		Ley_Informationsmodell.pdf (Dissertation Ley)		DITA_1_1_ArchitecturalSpecification_OASIS_August_2007.pdf	
Ist die Methodologie vollständig dokumentiert?	3	3	9	4	12	4	12	4	12
Was fehlt gegebenenfalls?								Fragen, die sich direkt bei Anwenden der Architektur bzw. dazu notwendiger Strukturierungsschritte ergeben, bleiben offen.	
Applikation der Methodologie			**78**		**80**		**66**		**92**
Handelt es sich um einen anerkannten Standard?	**3**	3	9	2	6	1	3	5	15
Welche Standardisierungsorganisation?		? Information Mapping Inc. Sitzt beispielsweise in USA, es gibt auch eine Information Mapping Institution, die für Europa zuständig ist etc.						OASIS	
Domänenspezifisch? Ist es direkt verwendbar oder muss es zuerst an eine Domäne angepasst werden?	3	5	15	5	15	4	12	5	15
Welche Domäne?									
Tools = Sind Tools zur Nutzung der Methodologie vorhanden?	2	1	2	1	2	1	2	5	10
Welche Tools?									
Methodologie kostenlos verfügbar?	2	1	2	5	10	5	10	5	10
Integration mit anderen Methodologien möglich? Interoperabilität	1	3	3					3	3
Wird eine bestimmte Ausgangsbasis vor Anwendung der Methodologie vorausgesetzt?	2	5	10	5	10	5	10	5	10
Welche Voraussetzungen?									
Wurde die Methodologie speziell im Zusammenhang mit einer bestimmten Auszeichnungssprache definiert? (XML, OWL, etc.)	1	5	5	5	5	5	5	3	3
Welche Auszeichnungssprache?									
Ist die Anwendung der Methodologie toolunabhängig?	2	5	10	5	10	5	10	3	6
Welche Tools?									
Ist die Methodologie im industriellen oder akademischen Umfeld bekannt und weit verbreitet?	2	5	10	5	10	1	2	4	8
Anmerkungen:									
Ist die Methodologie in sich konsistent?	3	4	12	4	12	4	12	4	12
Anmerkungen:									

Qualität des Ergebnisses			**28**		**27**		**28**		**33**
Kann das Ergebnis nach Anwenden der Methodologie direkt im anzunehmenden Arbeitsumfeld eingesetzt werden?	3	1	3	1	3	1	3	1	3
Ist das Ergebnis leicht zu pflegen und zu warten? (erweiterbar, flexibel)	3	3	9	3	9	3	9	4	12
Sind unterschiedliche Ergebnisse nach Anwendung der gleichen Methodologie leicht zu kombinieren?	1	3	3	2	2	3	3	3	3
Kann das Ergebnis leicht durch Standards weiterverarbeitet werden (z.B. XML)?	2	5	10	5	10	5	10	5	10
Sind Tools zur Weiterverarbeitung des Ergebnisses vorhanden?	1	3	3	3	3	3	3	5	5
ZWISCHENSUMME:			**134**		**144**		**121**		**156**
Personalisierung			**14**		**14**		**14**		**14**
Ist der kommunikative Rahmen vollständig beschrieben zur formalen Umsetzung? (Struktur und Umfang einer zukünftigen Nachricht, Wissensbasis von Sender und Empfänger, Daten & Metadaten)									
Ist das Wissensnetz des Senders umfassend dargestellt?	2	1	2	1	2	1	2	1	2
Ist das Wissensnetz des Empfängers umfassend dargestellt?	3	1	3	1	3	1	3	1	3
Wird ein Rollenkonzept für die Methodologie zur Personalisierung entwickelt / ist vorgesehen? (Für welchen Empfänger schreibt der Sender?)	3	3	9	3	9	3	9	3	9
Re-Use			**44**		**45**		**51**		**66**
Ist der kommunikative Rahmen vollständig beschrieben zur formalen Umsetzung? (Struktur und Umfang einer zukünftigen Nachricht, Wissensbasis von Sender und Empfänger, Daten & Metadaten)									
Ist eine gewisse Granularität vorhanden? Zugriffsmöglichkeiten?	3	5	15	5	15	5	15	5	15
Unterstützt mich die Methodologie darin, eine syntaktische Struktur für einzelne Bausteine festzulegen?	2	3	6	3	6	5	10	5	10
Unterstützt mich die Methodologie bei einer standardisierten anwendungsfallübergreifenden Einführung und Verwendung zu benennender Begriffe?	3	5	15	4	12	4	12	5	15
Unterstützt mich das Modell bei der Festlegung von anwendungsfallübergreifenden Standards für die auf dem resultierenden Modell beruhenden zukünftigen Instanzen? (Schreibregeln, Benennungsregeln)	2	2	4	4	8	1	2	3	6
Unterstützt mich die Methodologie die Gültigkeit der Module in einem anderen kommunikativen Rahmen zu spezifizieren? (Nachrichtenpakete in anderem Kontext?)	2	1	2	1	2	5	10	5	10
Unterstützt die Methodologie eine flexible Versionierungsfähigkeit? (Variablen angeben) {Ist die Haltbarkeit der Wiederverwendung angegeben? Wann kann etwas nicht mehr verwendet werden? (Versionierung)}	2	1	2	1	2	1	2	5	10
GESAMTSUMME:			**192**		**203**		**186**		**236**

Modellierungsmethodologie:		*On-To-Knowledge-Method (OTKM)*		*METHONTOLOGY*		*Ontology Development Guide 101*	
	Priorität	**Bepunktung**	**Berechnete Punktzahl**	**Bepunktung**	**Berechnete Punktzahl**	**Bepunktung**	**Berechnete Punktzahl**
Kriterien	**(1=niedrig, 2=mittel, 3=hoch)**	**(1=niedrig, 5=hoch) siehe Evaluationsleitfaden**	**(Priorität * Bepunktung)**	**(1=niedrig, 5=hoch) siehe Evaluationsleitfaden**	**(Priorität * Bepunktung)**	**(1=niedrig, 5=hoch) siehe Evaluationsleitfaden**	**(Priorität * Bepunktung)**
Verständlichkeit der Dokumentation der Methodologie			**27**		**30**		**29**
Werden spezifische Vorkenntnisse vorausgesetzt?	2	3	6	3	6	4	8
Welche Zielgruppe ist gemeint?		Ontologie-Entwickler, Software-Entwickler, Knowledge Engineer		Prinzipiell alle Ontologie-Entwickler Die spezifizierten: Software-Entwickler und		Ontologie-Entwickler mit wenig Erfahrung, Software Entwickler, technische Redakteure	
Welche Vorkenntnisse werden benötigt?		Formale Sprache wie OWL, F-Logic, etc. Wie werden Konzepte für die Ontologie gefunden. Kein Leitfaden dazu		Logiken in der Formalierungsphase		Nutzung von Tools wie Protégé	
Fördert Art, Aufbau und Umfang der Dokumentation Lesbarkeit und Verständlichkeit?	3	4	12	4	12	5	15
Benutzte / Referenzierte Quellen angeben		[SSSS01]		[GoFC] und [CFGL05]		[NoyGui01]	
Ist die Methodologie vollständig dokumentiert?	3	3	9	4	12	2	6
Was fehlt gegebenenfalls?		Wie wird die Taxonomie erstellt		Erstellen der Taxonomie nicht ganz klar, allerdings zusätzliche Handlungen beschrieben, die zur späteren Ontologie führen können (Axiome, Regeln, Konstanten, etc.). Allerdings middle-out-Ansatz empfohlen für die Taxonomie. Keine Suche nach existierenden Ontologien berücksichtigt.		Nur die Erstellung von Konzepten und Eigenschaften beschrieben, restlicher Lebenszyklus von Ontologien und zusätzlich notwendige Aktivitäten nicht berücksichtigt	
Applikation der Methodologie			**79**		**90**		**84**
Handelt es sich um einen anerkannten Standard?	**3**	1	3	2	6	1	3
Welche Standardisierungsorganisation?		Keine, Uni Karlsruhe		Keine Standardisierungsorganisation dahinter, aber sehr weit verbreitet und wird oft eingesetzt		Keine.	
Domänenspezifisch? Ist es direkt verwendbar oder muss es zuerst an eine Domäne angepasst werden?	3	5	15	5	15	5	15
Welche Domäne?		Kann direkt eingesetzt werden, da domänenunbhängig		Kann direkt eingesetzt werden, da domänen unabhängig.		Kann direkt eingesetzt werden.	
Tools = Sind Tools zur Nutzung der Methodologie vorhanden?	2	3	6	4	8	4	8
Welche Tools?		OntoEdit, Protégé zur formalen Ontologie-Entwicklung		WebODE, ODE, Protégé, etc. WebODE speziell für diese Methodologie entwickelt, aber zum Zeitpunkt der Evaluation nicht verfügbar		Tools wie Protégé vorhanden, keine komplette Unterstützung der Ontologie-Entwicklung	
Methodologie kostenlos verfügbar?	2	5	10	5	10	5	10
Integration mit anderen Methodologien möglich? Interoperabilität	1	3	3	3	3	5	5
Wird eine bestimmte Ausgangsbasis vor Anwendung der Methodologie vorausgesetzt?	2	5	10	5	10	5	10
Welche Voraussetzungen?		keine		keine		Keine.	
Wurde die Methodologie speziell im Zusammenhang mit einer bestimmten Auszeichnungssprache definiert? (XML, OWL, etc.)	1	5	5	5	5	4	4
Welche Auszeichnungssprache?		Keine Abhängigkeiten, kann auch unabhängig von OWL verwendet werden		Primär für Ontologien in RDF, OWL oder OntoLingua entwickelt, kann aber auch für andere Sprachen verwendet werden.		Speziell für OWL und Protégé entwickelt und beschrieben (z.B. Slots)	
Ist die Anwendung der Methodologie toolunabhängig?	2	5	10	5	10	5	10
Welche Tools?		unabhängig		unabhängig		Zwar für Protégé beschrieben, aber prinzipiell toolunabhängig	
Ist die Methodologie im industriellen oder akademischen Umfeld bekannt und weit verbreitet?	2	4	8	4	8	2	4
Anmerkungen:		Bekannt, wird teilweise im akademischen Umfeld eingesetzt. Keine Kenntnisse über Industrie		Relativ bekannt, sehr oft zitiert und in Akademie und Forschungsprojekten gleichermaßen eingesetzt.		Wohl schon bekannt, weite Verbreitung den Evaluierern allerdings unbekannt.	
Ist die Methodologie in sich konsistent?	3	3	9	5	15	5	15
Anmerkungen:		Bei der Suche nach existierenden Ontologien wäre es hilfreich gewesen gleich die später verwendete Sprache zu kennen, um gezielter suchen zu können. Diese wird aber erst in einem späteren Schritt definiert.		Konsistent, keine inhaltlichen Fehler entdeckt, sehr ausführlich beschrieben.		Der Bereich der Konzept- und Eigenschaftenfindung ist konsistent beschrieben und auch die Ausführung ist konsistent	

Qualität des Ergebnisses			**29**		**29**		**30**
Kann das Ergebnis nach Anwenden der Methodologie direkt im anzunehmenden Arbeitsumfeld eingesetzt werden?	3	1	3	1	3	1	3
Ist das Ergebnis leicht zu pflegen und zu warten? (erweiterbar, flexibel)	3	3	9	3	9	3	9
Sind unterschiedliche Ergebnisse nach Anwendung der gleichen Methodologie leicht zu kombinieren?	1	2	2	2	2	3	3
Kann das Ergebnis leicht durch Standards weiterverarbeitet werden (z.B. XML)?	2	5	10	5	10	5	10
Sind Tools zur Weiterverarbeitung des Ergebnisses vorhanden?	1	5	5	5	5	5	5
ZWISCHENSUMME:			**135**		**149**		**143**
Personalisierung			**12**		**12**		**12**
Ist der kommunikative Rahmen vollständig beschrieben zur formalen Umsetzung? (Struktur und Umfang einer zukünftigen Nachricht, Wissensbasis von Sender und Empfänger, Daten & Metadaten)							
Ist das Wissensnetz des Senders umfassend dargestellt?	2	3	6	3	6	3	6
Ist das Wissensnetz des Empfängers umfassend dargestellt?	3	1	3	1	3	1	3
Wird ein Rollenkonzept für die Methodologie zur Personalisierung entwickelt / ist vorgesehen? (Für welchen Empfänger schreibt der Sender?)	3	1	3	1	3	1	3
Re-Use			**14**		**18**		**20**
Ist der kommunikative Rahmen vollständig beschrieben zur formalen Umsetzung? (Struktur und Umfang einer zukünftigen Nachricht, Wissensbasis von Sender und Empfänger, Daten & Metadaten)							
Ist eine gewisse Granularität vorhanden? Zugriffsmöglichkeiten?	3	1	3	1	3	1	3
Unterstützt mich die Methodologie darin, eine syntaktische Struktur für einzelne Bausteine festzulegen?	2	1	2	1	2	1	2
Unterstützt mich die Methodologie bei einer standardisierten anwendungsfallübergreifenden Einführung und Verwendung zu benennender Begriffe?	3	1	3	1	3	3	9
Unterstützt mich das Modell bei der Festlegung von anwendungsfallübergreifenden Standards für die auf dem resultierenden Modell beruhenden zukünftigen Instanzen? (Schreibregeln, Benennungsregeln)	2	1	2	1	2	1	2
Unterstützt mich die Methodologie die Gültigkeit der Module in einem anderen kommunikativen Rahmen zu spezifizieren? (Nachrichtenpakete in anderem Kontext?)	2	1	2	1	2	1	2
Unterstützt die Methodologie eine flexible Versionierungsfähigkeit? (Variablen angeben) {Ist die Haltbarkeit der Wiederverwendung angegeben? Wann kann etwas nicht mehr verwendet werden? (Versionierung)}	2	1	2	3	6	1	2
GESAMTSUMME:			**161**		**179**		**175**

8.6 Zusammenfassung der Evaluation

Evaluation Linguistische Methodologie

Bei der Evalation der Methodologien haben wir als erste ‚Rubrik' die Verständlichkeit der Methodologie aufgenommen. Hierbei sticht eindeutig mit 37 von 40 erreichbaren Punkten das Funktionsdesign® hervor, was dadurch zu erklären ist, dass diese Methodologie bereits seit über 10 Jahren in diversen Hochschulen in der Lehre eingesetzt wird und die dazu verfügbaren Materialien hervorragend aufgebaut sind. Schlussreiter sind fast gleichauf die Strukturelle Textkontrolle von Ley (27 erreichte Punkte) und das Information Mapping™ von Horn (28 erreichte Punkte). Bei der Anwendung der Methodologie bildet wiederum die Stukturelle Textkontrolle von Ley das Schlusslicht, was zum einen damit zusammenhängt, dass Ley in seiner Dissertation Modellierungsprimitive und Informationsklassen für eine spezielle Domäne definiert und damit die Methodologie an und für sich domänenspezifisch ist, da vor dem eigentlichen Anwenden diese Informationsklassen an die entsprechende Domäne anzupassen sind. Zum anderen durchdringt diese Methodologie weder das industrielle noch das akademische Umfeld. Die Vorgehensweise wird in der vorliegenden Dissertation nicht in einer verständlichen Art und Weise vermittelt, was sicherlich auch nicht Sinn und Zweck einer Dissertation sein sollte. Wir schließen nicht aus, dass bei Vorlage einer verständlicheren Anleitung der Methodologie hier bei ‚Applikation der Methodologie' noch mehr Punkte vergeben werden könnten.
DITA als anerkannter Standard setzt sich bei ‚Applikation der Methodologie' mit 92 Punkten (von 105 erreichbaren) klar ab vom Funktionsdesign® mit 80 und Information Mapping™ mit 78 erreichten Punkten. Dieses Bild findet sich ebenfalls wieder in den Rubriken ‚Qualität des Ergebnisses' und ‚Re-Use'. Allerdings muss man einschränkend erwähnen, dass die Ergebnisse dadurch verzerrt sind, dass sehr unterschiedliche Dokumentationen der Methodologien vorlagen. Da uns beim Information Mapping™ beispielsweise nur rudimentäre Materialien zur Verfügung standen, liegen die erreichten Punkte nach einem kostenpflichtigen Training dieser Methodologie evtl. höher. Im Breich ‚Qualität des Ergebnisses' sind alle Ergebnisse gleich schlecht, was einen direkten Einsatz im Arbeitsumfeld angeht. DITA als beste Methodologie erreicht mit 33 von 50 erreichbaren Punkten hier nur 67% Erfüllungsgrad der aufgestellten Kriterien. Da alle Methodologien eine modularisierte Dokumentationserstellung zum Ziel haben, sollte man davon ausgehen, dass die Kriterien bei Personalisierung und Re-Use in hohem Maße von den Methodologien erfüllt werden. Bei ‚Personalisierung' liegen jedoch alle mit 14 von 40 erreichbaren Punkten fernab von den aufgestellten Kriterien. Bei ‚Re-Use' liegt jedoch DITA mit diesmal 66 von 70 erreichbaren Punkten vorne und deckt damit 94% der Kriterien ab. Im Vergleich zu den anderen Methodologien erreicht DITA dies vor allem durch den hohen Grad an Standardisierung und dem Einbetten der Modularisierung in einen kompletten Framework, in dem den standardisierten Blöcken bestimmte Eigenschaften und Verhaltensweisen zugeordnet sind. Hier zeigen sich klare Vorteile gegenüber den reinen Strukturierungsmethodologien Funktionsdesign® und Information Mapping™, die in dieser Rubrik mit 44 bzw. 45 Punkten am Ende liegen.Sehr eklatant bei der Evaluation der Methodologien aus dem linguistischen Umfeld ist, dass, obwohl die Zielgruppe und die Bedürfnisse der Zielgruppe bei der Applikation der Methodologie erarbeitet werden müssen, die entsprechend erarbeitete Wissensbasis über die Zielgruppe, über die Bedürfnisse der Zielgruppe, nirgends formal festgehalten werden. So erklären sich auch die schlechten Ergebnisse aller Methodologien im Bereich ‚Personalisierung'. Im Bereich ‚Re-Use' fällt auf, dass ein zunehmender Grad an sich bietenden Freiheiten, was die Benennung und auch die

Gestaltung zukünftiger Instanzen angeht, für den Re-Use hinderlich sind und deswegen Methodologien wie DITA, die auf den ersten Blick eher ‚rigoros' erscheinen, hier ihre Vorteile haben. Die Entwicklung von Content Management Systemen und der dahinter steckenden Logik und Verarbeitung von Inhalten zeigt sich in der Entwicklung der Methodologien: während weder im Funktionsdesign® noch beim Information Mapping™ Versionierungsfähigkeit und Gültigkeit der herausgearbeiteten Module ein Thema zu sein schien, nehmen sich neuere Methodologien wie die Strukturelle Textkontrolle und DITA diesen typischen Hürden an.

Evaluation Ontologie-Methodologie

Bei Anwendung der Ontologie-Methodologien hat sich gezeigt, dass keine dieser alle Schritte bei der Erstellung einer Ontologie ausreichend beschreibt. OTKM gibt einen guten Überblick, aber teils nur recht vage Handlungsanweisungen, was genau gemacht werden soll. Hier ist METHONTOLOGY schon umfangreicher, zumal auch komplett alle Schritte im Lebenszyklus einer Ontologie bedacht wurden. Beim Aufbau von METHONTOLOGY fühlt man sich an den Rational Unified Process (RUP) erinnert, in welchem ähnlich die unterschiedlichen Stufen bei der Software-Entwicklung beschrieben sind. Aber auch hier wird keine genaue Beschreibung gegeben, wie denn die Konzepte in der Ontologie genau zu finden sind und welche Dinge dabei beachtet werden sollen. Dieses Leck füllt der Ontology Development Guide 101, in welchem ausführlicher Handlungsweisungen gegeben werden, wie Konzepte, Eigenschaften und Instanzen gefunden werden können und was dabei beachtet werden muss. Dies spiegelt sich auch in den Werten der Evaluation wider: OTKM fällt mit 161 Punkten am schlechtesten aus, METHONTOLOGY besitzt mit 179 Punkten am meisten, knapp vor 175 Punkte für den Ontology Development Guide 101. Auffallend ist, dass bei der Personalisierung alle gleich wenig Punkte (12) erhalten haben. Beim Reuse unterscheiden sich die Ansätze lediglich geringfügig, wobei der Ontology Development Guide 101 mit 20 Punkten noch am besten abschneidet.
Eine Kombination von METHONTOLOGY und Ontology Development Guide 101 ist durchaus möglich und scheint uns die vielversprechendste Ontologie-Methodologie zu sein. Allerdings ist bei allen betrachteten Methodologien die Möglichkeit für Reuse und Personalisierung sehr beschränkt und wird oft gar nicht berücksichtigt. Eine Weiterentwicklung unter Berücksichtigung dieser Aspekte wäre daher wünschenswert.

Zusammenfassung

Anhand des semantischen Kommunikationsmodells kann man erkennen, dass keine der betrachteten Methodologien in der Lage ist, alle Bereiche des kommunikativen Rahmens abzudecken und zu beschreiben. Interessant ist auch, dass Ontologien zwar prinzipiell in der Lage sein müssten, die geforderten Wissensnetze formal beschreiben zu können. Es wird nur leider in keiner der Methodologien gefordert. Könnte man dieses Gedanken der möglichen formalen Beschreibung der Wissensnetze und Rollenkonzepte über Ontologien integrieren in die nach unserer Ansicht und Evaluation beste Methodologie aus dem Dokumentationsumfeld – DITA – sind Großteile des kommunikativen Rahmens formal beschreibbar und abgedeckt.

9 Related Work

Bei der Suche nach zum Forschungsrahmen passenden Arbeiten wurden systematisch folgende Begrifflichkeiten verwendet: „Semantic Requirement“, „Requirement Engineering Semantic Web“, „Model-Driven Requirement Engineering“, „Requirement Modeling“ sowie „Requirement Specification“.
Mehrere gefundene Arbeiten beschreiben einen groben Zusammenhang zwischen Requirement Engineering auf der einen Seite und Ontologien oder Techniken des Semantic Web auf der anderen Seite.

9.1 Requirements Engineering

Im Rahmen des V-Modell XT-Entwicklungsprozesses werden beispielsweise Anforderungen festgehalten, wie Requirements zu sammeln und zu notieren sind. Gemäß [UniKai05] enthält das Lastenheft dabei alle an das zu entwickelnde System verbindlich gestellten Anforderungen und ist Bestandteil des Vertrags zwischen Auftraggeber und Auftragnehmer. Kern des Lastenhefts sind die funktionalen und nicht-funktionalen Anforderungen an das System, sowie eine Skizze des Gesamtsystementwurfs. Diese beinhalten eine textuelle und graphische Beschreibung der Anwendungsfälle bzw. sonstiger Anforderungen. Eine eindeutige Semantik wird hierbei nicht vorgeschrieben und eine automatische Weiterverwendung ist normalerweise nicht vorgesehen.
In [AaJoLa05] wird vorgeschlagen die Anforderungen nicht nur textuell, sondern auf Basis eines graphischen Modells (Gefärbten Petri-Netzen) anzugeben, um damit eine automatische Transformation in lauffähigen Code (hier: nach WS-BPEL [Alves06]) zu ermöglichen. Dieser Ansatz ist generell ähnlich zu Techniken der modellgetriebenen Software-Entwicklung (Model-Driven Software Development, MDSD). Damit soll über verschiedene Modelltransformationen und Codegenerierung aus einem allgemeinen Geschäftsprozess über Verfeinerungen letztendlich lauffähiger Code generiert werden.
In [KoZhEs06] wird ein derartiges Web zum Erstellen von Web-Seiten vorgeschlagen. Hier wird ein Anforderungsmodell auf Basis eines wohldefinierten Metamodells in mehrere plattformunabhängige (Platform Independent Model, PIM) Modelle wie z.B. ein Content Model, ein Navigation Model, Process Model oder Presentation Model, transformiert, aus welchen dann schließlich Modelle für verschiedene Zielplattformen (J2EE, .NET) erstellt werden. Diese werden abschließend zu ausführbarem Code transformiert. Modelltransformationen werden dabei meist auf Basis von UML-Modellen (Unified Modeling Language [OMG05]) durchgeführt. Mögliche Sprachen hierzu wären: ATL, MTF, QVT oder das EMF Tranformation Toolkit. In [Tratar06] wird ausserdem ein modellbasiertes Requirements-Engineering (MDRE) vorgeschlagen. Dazu werden alle Anforderungen in einem Modell abgelegt, aus welchem dann ein Pflichtenheft generiert werden kann. Das zugrunde liegende Metamodell bietet eine formale Grundlage für die Strukturierung der Anforderungen.
Ansätze, um die Anforderungen erst mal aufzunehmen, gibt es mehrere: in [Micros06] beschreiben die Autoren die Perspective-Based Architecture Method, wonach durch verschiedene Fragestellungen aus verschiedenen Perspektiven die jeweiligen Anforderungen aufgenommen werden. Desweiteren könnten die Requirements auch durch ein generisches Dokumenten Modell festgehalten werden wie in [StoWeb06] vorgeschlagen, um damit gleichzeitig auch den Prozess der technischen Dokumentation zu automatisieren. Weitere Sprachen zum Requirements Engineering sind wie in den genannten Quellen näher ausgeführt die von der Object Management Group definierte

Unified Modeling Language UML (vgl. [Weilki06a]) sowie die System Modeling Language SysML [Weilki06b].
In [Wolf07] beschreibt der Autor ein modellbasiertes Verfahren um verteilte Software-Entwicklungsprojekte in einer Tool-Suite zu vereinfachen. Dazu werden u.a. alle Anforderungen in einem Tool erhoben und in einem anderen auf Basis desselben Meta-Modells direkt weiterverarbeitet. Dieser so genannte MDSD Light [BaEtAl08] Entwicklungsansatz wird mittlerweile in diversen Projekten verfolgt (u.a. bei der Geschäftsprozessmodellierung).

In [NeStZd02] wurde gezeigt, wie Code mit Metadaten angereicht werden kann, um die Abhängigkeiten zu bestimmten Anforderungen festhalten zu können. Diese Metadaten sind allerdings in natürlich-sprachlicher Form und das System wurde beispielhaft in der selbst-entwickelten Programmiersprache XOTcl implementiert.

9.2 Semantik-basiertes Requirements Engineering

Neben diesen syntaxbasierten Methoden oder Modellen gibt es allerdings auch schon erste Ansätze einer semantikbasierten Beschreibung von Anforderungen. Die Zusammenhänge zwischen RE und Semantic Web wurden im technischen Bericht [SelAus03] erstmals genauer untersucht, wobei der damalige Stand der Semantic Web Sprachen mittlerweile überholt ist. Wie dort genauer beschrieben ist, ist sowohl das Internet als auch der Bereich der Anforderungsanalyse ein virtuelles, chaotisches System von Systemen wie durch Maier und Rechtin definiert. In [SelAus03] wird ausserdem beschrieben, welche Bereiche des Requirement Engineering durch welche Ebenen des sogenannten Semantic Web Layer Cake (Schichtenmodell) umgesetzt werden können. Der Folgereport [MaKoAu04] greift die dort gewonnenen Erkenntnisse auf und beschreibt darauf aufbauend, wie Komponenten semantisch angereichert werden können und zeigt dies beispielhaft. Eine ähnliche Problematik wird in [Kiniry03] beschrieben, wobei hier ausserdem noch auf eine Komposition von semantisch beschriebenen Komponenten eingegangen wird. Dabei wird hier allerdings die programmiersprachliche Semantik von Java verwendet und nicht Sprachen aus dem Semantic Web Umfeld. In [KaiSae05] wird hingegen die Betrachtung auf eine ontologie-basierte Anforderungsanalyse gelenkt und ein eigenes Framework zur Analyse von Anforderungen vorgeschlagen. Leider sind die genauen technischen Details über ein Mapping von Requirements zu Elementen einer Ontologie nur sehr oberflächlich beschrieben und nicht genauer ausgeführt. Es wird lediglich angeführt, dass die Nutzung von Ontologien das Aufdecken von Unvollständigkeiten oder Inkonsistenzen ermöglicht sowie die Messung der Dokumentenqualität. Die Ontologie bestehe dabei aus einem Thesaurus und einer Menge von Inferenzregeln. In [DoLoSo05] wird eine eigene Ontologie auf Basis von OWL-S [Martin05] definiert, um die Anforderungen an die Qualität eines Services genauer beschrieben zu können. Diese Ontologie QoSOnt wird danach verwendet, um bei der Suche nach Services die gewünschten Anforderungen mit den gegebenen zu vergleichen und ein dazu passender Matchmaking-Algorithmus wird beschrieben. Ein dazu vergleichbarer Ansatz [LiFoBi96] fasst ebenfalls alle Anforderungen direkt in einer Ontologie, allerdings umfasst diese Ontologie nun nicht nur die Quality of Service sondern auch weitere organisatorische, strukturelle, funktionale und kostenbedingte Anforderungen. Auf Basis dieser Anforderungen wird eine Deduktion vorgeschlagen, um zu überprüfen, ob die Summe der Anforderungen konsistent ist oder wie diese Anforderungen zusammenhängen.

Wie in [DobSaw06] beschrieben wurden erste Grundlagen zur ontologie-basierten Anforderungsanalyse bereits bei der Entwicklung der Sprache RML (Requirements Modeling Language) gelegt, die Anfang der 80er entworfen wurde. Aufbauend auf RML werden die Zusammenhänge zu den heutigen Semantic Web (Regel-)Sprachen OWL und SWRL beschrieben.

In [HapSee06] wird beschrieben, wie Ontologien im Rahmen des Software-Engineering verwendet werden können. Dabei werden einige Bereiche beispielhaft skizziert (u.a. die Erweiterung von Analyse und Design durch Ontologien) sowie eine Kategorisierung der Ontology-Driven Architecture in 4 Teilbereiche vorgeschlagen, um besser ausdrücken zu können, wo der Nutzen einer Ontologie vorrangig ist.

9.3 Use Cases

Generell sind Use Cases in einer leicht-verständlich erzählenden Art und Weise geschrieben, indem sie auf Vokabular zugreifen, das in der entsprechenden Domäne gängig ist. Hierbei ist bereits interessant, dass es sinnvoll ist, genauer zu spezifizieren, welche Domäne berücksichtigt werden sollte, da nicht davon ausgegangen werden kann, dass alle Beteiligten eines Projekts zu einer Domäne zu zählen sind.
Das Ziel einer Use-Case Darstellung sollte sein, dass sie leicht nachzuvollziehen und zu überprüfen ist und durch den leichten Zugang die Benutzer und Anwender der Use-Case Darstellungen in das Formulieren von Anwendungen und deren Umsetzung in Use Cases besser involviert werden können. Use-Case-Zone [Usecase06] stellt eine gute Übersicht zur Verfügung, wie Use Cases gehandhabt werden können und welche Stolpersteine dabei identifiziert wurden. Ein interessantes Kapitel über Use-Cases findet man in [Larman06].
Einige Forschungsgruppen untersuchten, wie man von unterschiedlichen Quellen zu Use Case Diagrammen kommt: [SanCas01] entwickeln Use Cases aus Organisationsmodellen heraus (i*-framework). Dazu werden im Rahmen einer zielorientierten Analyse erst alle Akteure innerhalb eines Organisationsmodells gefunden, anschließend die Anwendungsfälle für jeden einzelnen Akteur untersucht sowie die einzelnen Szenarien durchgeforstet. [LoLaGa02] benutzen existierende Workflows, um sie für zukünftige Anforderungsmodellierungen wieder zu verwenden. Dazu werden aus administrativen Workflows sogenannte Case Graphs generiert, aus denen dann Business Use Cases bzw. technische Use Cases erstellt werden können. Dies wird durch die Verwendung von Petri-Netzen in den Case Graphs unterstützt. Weiterhin gibt es einen interessanten Ansatz, Use Cases zu konkretisieren von Irwin und Turk [IrwTur05], die eine ontologie-basierte Analyse von Use Case Modeling Grammatiken zeigen. Dazu verwenden sie die Ontologie von Bunge, Wand und Weber (BWW), um die Fähigkeit der Grammatik-Modellierung zu evaluieren. Daraus wird anschließend eine Menge von Hyptothesen generiert.

10 Zusammenfassung und Ausblick

10.1 Problemreduktion von RE auf eine kommunikative Situation

Im Laufe des Requirements Engineering entstehen diverse technische Dokumente, deren Entstehung man sich – wie wir in Kapitel 2.2 aufzeigen konnten - ausgehend von der Sprechakttheorie als Schreibakt vorstellen kann und muss. Probleme, die hierbei entlang des Produktentwicklungszyklus (siehe auch Kapitel 3) auftreten können, sind dementsprechend kommunikativer Art, so dass sich eine Reduktion vornehmen lässt, indem man sich den Austausch von Anforderungen über Dokumente als kommunikative Situation vorstellt. Da keines der gängigen Kommunikationsmodelle (vgl. Kapitel 5.1) unserem Verständnis von Daten (i.S. des semantischen Datenmodells nach Sieber/Kammerer, das wir in Kapitel 2.1 vorgestellt haben) folgt, haben wir auf Basis des semantischen Datenmodells ein eigenes semantisches Kommunikationsmodell (siehe Kapitel 5.2) entwickelt. Dieses haben wir als Referenzmodell für die Evaluation von linguistischen und Ontologie-Methodologien verwendet und unsere Evaluationskriterien auf das Referenzmodell bezogen. Bei der Evaluation konnten wir feststellen, dass weder Methodologien aus dem linguistischen Bereich noch aus dem Ontologie-Sektor in der Lage sind, den kommunikativen Rahmen anhand unseres semantischen Kommunikationsmodells vollständig formal zu beschreiben und abzudecken. Wie in 8.6 bereits angerissen, erscheint uns deswegen eine Kombination von Methodologien aus beiden Bereichen als sinnvoll, um die Lücken zu schließen.

10.2 Kombination linguistische Modellierungsmethodologien / Ontologie-Methodologien als Lösung

Wie man in Kapitel 8.6 sehen konnte, gibt es bereits einige (auch sehr gute) Ansätze für linguistische Modellierungsmethodologien und für Methodologien zur Entwicklung einer Ontologie. Allerdings wurde bislang nicht untersucht, wie beides kombiniert werden konnte.

Nutzt man Natural Language Processing (NLP) Tools, um zu erkennen, ob zwei Artefakte ähnlich sind (z.B. zwei Requirements-Dokumente), benötigt man diverse bereits annotierte Beispiele und kann dann versuchen diese Artefakte automatisch zu vergleichen. Allerdings ist der Erfolgsfaktor eher gering. Daher werden oftmals die Dokumente händisch annotiert, um dann im Nachhinein die Ähnlichkeit dieser Dokumente automatisch ermitteln zu können. Dies erfordert allerdings einen hohen manuellen und zeitlichen Aufwand, weshalb hier eine Automatisierung interessant erscheint.

Hier kommen die vorgestellten linguistischen Modellierungsmethodologien ins Spiel: sind die Dokumente von Beginn an bereits strukturiert und die einzelnen Bereiche annotiert (z.B. durch das Funktionsdesign®), ist die Texterkennung mittels NLP-Techniken nicht mehr notwendig und damit auch eine mögliche Fehlerquelle ausgeschlossen. Die Bereiche und ihre Annotationen können direkt verglichen werden oder aus den einzelnen Annotationen eine Ontologie erstellt werden. Sind die Annotationen so ausführlich, dass auch das Empfängerwissensnetz oder das Wissensnetz des Senders abgebildet ist, können hier ggf. auch bereits mehrere Ontologien erstellt werden und im weiteren Verlauf genutzt werden. Dazu sind die betrachteten Ontologie-Methodologien notwendig, um passende Ontologien für das Sender- und Empfängerwissensmodell zu beschreiben.

10.3 Idee eines Framework

Die Idee, Empfängerwissensmodelle zu beschreiben, kann im Rahmen eines Produktentwicklungszyklus bereits im Frühstadium erfolgen. Betrachtet man beispielsweise die Tätigkeit eines Usabilitity Engineers, dann fällt auf, dass bereits an dieser Stelle das Wissen über die zukünftigen Kunden des Produkts erarbeitet wird. In typischen Prozessabläufen findet jedoch weder eine formale Beschreibung des hier bereits anfallenden Wissens statt noch eine automatisierte Weiterleitung des Wissens über die Kunden an die anderen Bereiche im Unternehmen, die als Sender (z.B. beim Erstellen des Benutzerhandbuchs) eben jene Kunden als Zielgruppe haben und sich das Wissen darüber selbst erarbeiten müssen. Als Vision kann man sich vorstellen, dass hier formale Beschreibungen in Form von Ontologien übergeordnet über Prozessregeln verknüpft und an Rollenkonzepte gekoppelt werden können, so dass im gesamten Produktentwicklungszyklus darauf zugegriffen werden kann und standardisierte Verknüpfungen über Metadaten erfolgen, die einen formalisierten Zugriff auf damit ausgezeichnete Instanzen zulassen, die in ihrer Art der Gestaltung ebenfalls standardisiert sind.

10.4 Ausblick

Durch eine derart gewährleistete standardisierte Art der Dokumentation in Kopplung mit der vervollständigten Abdeckung des kommunikativen Rahmens über das Abbilden von Wissen in Form von Ontologien, ist gewährleistet, dass es zu weniger Wissensverlusten an existierenden Schnittstellen kommt und das Wissen konsistent von vorne nach hinten ‚durchgereicht' wird. Die dadurch und durch eine standardisierte Vorgehensweise bei der Dokumentation erreichte einheitliche formale semantische Beschreibung kann als Grundlage für zukünftige Auseinandersetzungen mit Requirements Engineering gesehen werden und sollte im Fokus für Re-Engineering-Bestrebungen im Software-Engineering stehen. Des Weiteren legt die angestrebte Kombination des Einsatzes beider Methodologien die Basis für Dokumentation-Prozessgestaltungen im Sinne des unter 3.6 aufgezeigten Content Syndication-Konzeptes.

Literatur

AaJoLa05 VAN DER AALST, W.M.P; JORGENSEN, J.B.; LASSEN, K.B. *Let's Go All the Way: From Requirements via Colored Workflow Nets to a BPEL Implementation of a New Bank System.* In: On the Move to Meaningful Internet Systems 2005, OTM Confederated Conferences, 2005.

AlEtAl05 ALEXIEV, V; BREU, M.; DE BRUIJN, J. FENSEL, D.; LARA, R.; LAUSEN, H.: *Information Integration with Ontologies – Experiences from an Industrial Showcase*. John Wiley, Sussex, 2005.

Alves06 ALVES, A. et al.: *Web Services Business Process Execution Language Version 2.0*, Committee Draft, May 2006, available online at http://www.oasis-open.org/committees/download.php/18714/wsbpel-specification-draft-May17.htm

Ament03 AMENT, K.: *Single Sourcing: Building Modular Documentation.* William Andrew Publishing, 2003

Anding04 ANDING, Markus: *Online Content Syndication.* Wiesbaden : Deutscher Universitäts-Verlag, 2004

Annot07 Semantic Web: *Annotation Portal*. Online abgerufen am 6.11.2007 unter http://annotation.semanticweb.org

AueLeh07 AUER, Sören; LEHMANN, Jens (2007): *What have Innsbruck and Leipzig in Common? Extracting Semantics from Wiki Content*. In: Proceedings of European Semantic Web Conference, LNCS 4519, Springer.

AuBiSz00 AUSSENAC-GILLES, Nathalie; BIÉBOW, Brigitte; SZULMANN, Sylvie (2000): *Revisiting Ontology Design: A Method Based on Corpus Analysis*. In: Knowledge Engineering and Knowledge Management, EKAW 2000, Juan-les-Pins, France, Oktober. LNAI 1937, S. 172-188.

Austin62 AUSTIN, J.L. (1962): *How to Do Things with Words*, Cambridge, Mass.; Harvard University Press.

BaBeSt01 BACK, Andrea ; BENDEL, Oliver ; STOLLER-SCHAI, Daniel: *E-Learning im Unternehmen - Grundlagen - Strategien - Methoden - Technologien.* Zürich : Orell Füssli Verlag, 2001.

BaEtAl08 BAUER, Bernhard; LAUTENBACHER, Florian; PALFINGER, Günther; ROSER, Stephan: *MDSD Light and its application*. ACM SAC 2008, Fortaleza, Brasilien, März 2008

Bauman03 BAUMANN, Sabine: Ganzheitliche Unterstützung der Wertschöpfungskette in Medien-Unternehmen mit Content Management Lösungen. In: STAHL, Florian; MAASS, Wolfgang

(Hrsg.). *Content Management Handbuch : Strategien, Theorien und Systeme für erfolgreiches Content Management*. Universität St. Gallen mcm institute, 2003. – ISBN 3906979164, S. 79–88

Becker68 BECKER, Sam (1968): *The prospect of rhetoric*. Zusammenfassung online, abgerufen am 7.10.2007 unter http://zimmer.csufresno.edu/~johnca/spch100/notes.htm.

Bieman05 BIEMANN, Chris (2005): *Ontology Learning from Text: A Survey of Methods*. In: LDV-Forum 2005, Band 20 (2), S. 75-93. Online abgerufen am 06.11.2007 unter http://wortschatz.uni-leipzig.de/~cbiemann/pub/2005/BiemannLDVOntology05.pdf.

Bühler69 BÜHLER, Karl (1965, 1992). Sprachtheorie: Die Darstellungsform der Sprache. 2. Aufl. Jena: G. Fischer. Neudruck: Stuttgart: Fischer. Bühler, Karl (1969). Die Axiomatik der Sprachwissenschaften. Frankfurt/Main: Klostermann.

BurEpp05 BURKHARD, Remo ; EPPLER, Martin: Knowledge Visualization. In: SCHWARTZ, David G. (Hrsg.). *Encyclopedia of Knowledge Management*. B&t Database Management/Cip, 2005.

CarMee06 CARR, Norm; MEEHAN, Tim: *What's the Problem?* Online verfügbar unter http://www.alistapart.com/articles/whatstheproblem/, zuletzt geprüft am 12. Mai 2006.

Cockbu01 COCKBURN, Alistair (2001) : *Writing Effective Use Cases*. Addison-Wesley, 2001.

CoEtAl05 CORCHO, Oscar; FERNANDEZ-LOPEZ, Mariano; GOMEZ-PEREZ, Asuncion; LOPEZ-CIMA, Angel (2005): *Building legal ontologies with METHONTOLOGY and WebODE*. In: Benjamins, R.; Casanovas, P.; Breuker, J. & Gangemi, A. (ed.): "Law and the Semantic Web". Springer-Verlag, LNAI No. 3369, pp. 142-157, 2005, ISBN: 3-540-25063-8

DanLar76 DANCE, Frank E. X.; LARSON, Carl. (1976). *The Functions of Human Communication: A Theoretical Approach*. New York: Holt, Rinehart & Winston.

DavPru98 DAVENPORT, Thomas H. ; PRUSAK, Laurence: *Wenn Ihr Unternehmen wüsste, was es alles weiss. : Das Praxisbuch zum Wissensmanagement.* verlag moderne industrie AG & Co. KG, 1998.

DITAAB07 *DITA from A to B: helping you get started with topic-based structured writing.* Online verfügbar unter http://www.ditausers.com/. 2007

DITAIn07 *DITA Infocenter: A searchable knowledge base of specifications for DITA Users.* Online verfügbar unter http://www.ditainfocenter.com/. 2007

DobSaw06 DOBSON, G.; SAWYER, P. *Revisiting Ontology-Based Requirements Engineering in the age of the Semantic Web*. In: Dependable Requirements Engineering of Computerised Systems at NPPs, Halden, 2006.

DoLoSo05 DOBSON, G.; LOCK, R.; SOMMERVILLE, I.: *Quality of Service Requirements Specification Using an Ontology.* Workshop on Service-Oriented Computing Requirements (SOCCER), Paris, August, 2005.

Flensb07 FLENSBURG, P. *An enhanced communication model.* In: Proceedings of the 30th Information Systems Research Seminar in Scandinavia (IRIS) 2007.

Frank00 FRANK, Ulrich: *Evaluation von Artefakten in der Wirtschaftsinformatik*. In: Häntschel, I; Heinrich, L.J. (Hrsg.): Evaluation und Evaluationsforschung in der Wirtschaftsinformatik. München, Wien: Oldenburg 2000, S. 35 – 48.

Gersdo03 GERSDORF, Rubens: Content Management für die flexible Informationswiederverwendung. In: STAHL, Florian; MAASS, Wolfgang (Hrsg.). *Content Management Handbuch : Strategien, Theorien und Systeme für erfolgreiches Content Management*. Universität St. Gallen mcm institute, 2003. – ISBN 3906979164, S. 61–74

GomFer04 GÓMEZ-PÉREZ, Asunción; FERNÁNDEZ-LÓPEZ, Mariano; CORCHO, Oscar: *Ontological Engineering*, Springer-Verlag, Heidelberg, 2004

GrüFox95 GRÜNNINGER, Michael; FOX, Mark (1995): *Methodology for the Design and Evaluation of Ontologies*. 1995.

HanNeu87 HANSEN, Hans R. ; NEUMANN, Gustaf: *Wirtschaftsinformatik Bd.1 : Grundlagen und Anwendungen.* UTB, 1987.

HapSee06 HAPPEL, H.J.; SEEDORF, S.: Applications of Ontologies in Software Engineering. Workshop SWESE06 at the ISWC06, Athens, USA.

HeEtAl04 HEVNER, Alan R.; MARCH, Salvatore T.; PARK, Jinsoo; RAM, Sudha: *Design Science in Information Systems Research*. In: MIS Quarterly, Vol. 28, No. 1, S. 75-105, März 2004.

HoGoBr93 HONG, S.; GOOR, G., BRINKKEMPER, S.: *A Formal Approach to the Comparison of Object-Oriented Analysis and Design Methodologies*. In: Nunamaker, J.F.; Sprague, R.H. (Hrsg.): Information Systems of HICCS, Los Alamitos, 1993, S. 689-698.

Honeyc07 HONEYCUTT, LEE: *Aristotle's Rhetoric.* online verfügbar unter: http://www.public.iastate.edu/~honeyl/Rhetoric/rhet3-1.html#1404a

House93 HOUSE, E.R.: *Professional Evaluation: Social Impact and Political Consequences*. Newbury Park, Ca. 1993

IEEE610 IEEE STANDARDS BOARD: IEEE St. 610.12-1990, IEEE Standard Glossary of Software Engineering Terminology, IEEE Press 1990.

IrwTur05 IRWIN, G.; TURK, D.: An Ontological Analysis of Use Case Modeling Grammar, Journal of the Association for Information Systems, 2005,

available online at
http://www.biz.colostate.edu/faculty/gretcheni/JAIS04-122.pdf.

KaiFal06 KAINDL, H; FALB, J. *From Requirements to Design: Model-driven Transformation or Mapping?* In: Proceedings of MoRSE 2006

KaiSae05 KAIYA, H.; SAEKI, M.: *Ontology Based Requirements Analysis: Leightweight Semantic Processing Approach*. In: International Conference on Quality Software (QSIC), Melbourne, Australia, September 2005.

KelTer05 KELLER, Tanja ; TERGAN, Sigmar-Olaf: Visualizing Knowledge and Information: An Introduction. In: TERGAN, Sigmar-Olaf (Hrsg.). *Knowledge and information visualization : searching for synergies*. Berlin;Heidelberg : Springer, 2005.

KiMaVo00 KIETZ, Joerg-Uwe; MAEDCHE, Alexander; VOLZ, Raphael (2000): *A Method for Semi-Automatic Ontology Acquisition form a Corporate Intranet*, ECAW 2000.

Kiniry03 KINIRY, J. R.: *Semantic Component Composition*. In: ECOOP 2003, Darmstadt, Germany, July 2003.

Kovács04 KOVÁCS, László: *Adatbázisok tervezésének és kezelésének módszertana.* Budapest : ComputerBooks Kiadó Kft., 2004

KoZhEs06 KOCH, N.; ZHANG, G. ESCALONA, M.J. *Model Transformations from Requirements to Web System Design*, ICWE 06, Palo Alto, California, USA; July 2006.

Kuhlen99 KUHLEN, Rainer: *"Knowledge is becoming hypertextified" : zur zweiten ACM-Hypertext Konferenz November 1989 Pittsburgh.* Konstanz : 1999.

Kuhlen04 KUHLEN, Rainer: Information. In: KUHLEN, Rainer; SEEGER, Thomas; STRAUCH, Dietmar (Hrsg.). *Grundlagen der praktischen Information und Dokumentation : Band 1: Handbuch zur Einführung in die Informationswissenschaft- und praxis. Band 2: Glossar*. Saur, K G, 2004.

Kuntz05 KUNTZ, Andreas: *Daten, Wissen, Information* URL http://server02.is.uni-sb.de/courses/ident/themen/dat_wiss_info/information.php. – Überprüfungsdatum 2006-04-19.

Larman04 LARMAIN, C. *Applying UML and Patterns*, Prentice Hall PTR 2004

Larman06 CRAIG LARMAN: *Use-Case Model: Writing Requirements in Content*, available online at http://www.craiglarman.com/book_applying_2nd/06-use%20cases.pdf, last viewed 2006-05-17

LaSeBa06 LACLAVÍK, Michal; SELENG, Martin; BABÍK, Marián: *OnTeA: Semi-Automatic Ontology based Text Annotation Method*. In: Tools for

Acquisition, Organisation and Presenting of Information and Knowledge, Bratislava, 2006. Online, abgerufen am 06.11.2007 unter http://laclavik.net/publications/itat_nazou_ontea.pdf

Ley06a LEY, M.: Aspekte der Informationsstrukturierung : Über Strukturierungsprinzipien, die Ebenen der Textstruktur und Dokumentgrammatiken. In: technische Kommunikation 28 (2006), Nr. 4/06, S. 51–53.

Ley06b LEY, M.: Kontrollierte Textstrukturen Ein (linguistisches) Informationsmodell für die Technische Kommunikation Dissertation. Justus-Liebig-Universität Giessen. 2006.

LiFoBi96 LIN, J.; FOX, M.S.; BILGIC, T.: *A Requirement Ontology for Engineering Design*. In: Concurrent Engineering, Vol. 4, No. 3 279-291. SAGE Publications, 1996.

Lobin00 Lobin, H.: *Informationsmodellierung in XML und SGML*. Berlin, Heidelberg, New York, Barcelona, Budapest, Hongkong, London, Mailan, Paris, Singapur, Tokio: Springer, 2000

LoLaGa02 O. LOPEZ, M.A. LAGUNA, F.J.GARCIA: Automatic Generation of Use Cases From Workflows: A Petri Net Based Approach, FASE 2002.

MaaSta03 MAASS, Wolfgang ; STAHL, Florian: Content Management als Teil des Kommunikations-Management. In: STAHL, Florian; MAASS, Wolfgang (Hrsg.). *Content Management Handbuch : Strategien, Theorien und Systeme für erfolgreiches Content Management*. Universität St. Gallen mcm institute, 2003. – ISBN 3906979164, S. 37–47

MaKoAu04 MAYANK, V.; KOSITSYNA, N.; AUSTIN, M. *Requirements Engineering and the Semantic Web – Part II*: Representation, Management and Validation of Requirements and System-Level Architectures, Technical Report 2004-14, Institute for Systems Research, February 2004.

Martin03 MARTIN, Robert C. (2003) : *Use Cases – A minimalist's view*. Artima Weblog, July 2, 2003.

Martin05 MARTIN, D. et al.: 2004. *OWL-S: Semantic Markup for Web Services*, W3C Member Submission.

Micros06 MICROSOFT: *Perspective-Based Architecture Method*. In: The Architecture Journal – Vol. 9.

MooSha94 MOODEY, D.L.; SHANKS, S.: What makes a good data model? Evaluating the Quality of Entity Relationship Models. In: Loucopoulos, P. (Hrsg.): Entity-Relationship Approach – ER'94. Business Modelling and Re-Engineering. Berlin, Heidelberg, S. 94-111, 1994.

MutSch99 MUTHIG, J. AND SSCHÄFLEIN-ARMBRUSTER, R.: *Funktionsdesign: eine universelle und flexible Standardisierungstechnik.* In: Felkner,

Christine / Sturz, Wolfgang (Hrsg.) 1995ff: Technische Dokumentation – wirtschaftlich organisieren, systematisch erstellen, kundengerecht gestalten. Augsburg: WEKA-Verlag, 1999.

Namahn NAMAHN: *Information Mapping™ : a research note.* online verfügbar unter: http://www.namahn.com/resources/documents/note-IM.pdf .

NeStZd02 NEUMANN, G. ; STREMBECK, M. ; ZDUN, U. *Using Runtime Introspecitble Metadata to Integrate Requirement Traces and Design Traces in Software Components.* In: Proc. of ECCOP USE, Malaga, Spain, June 2002. Verfügbar online unter: http://wi.wu-wien.ac.at/home/mark/publications/use02.pdf

NoyGui01 NOY, Natalya F.; MCGUINNESS, Deborah L. (2001): *Ontology Development 101: A Guide to Creating your first ontology*. KSL-01-05.

NoyMus00 NOY, Natalia F.; MUSEN, Mark A. (2000): *PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment*. 2000

NoyMus03 NOY, Natalia F.; MUSEN, Mark A. (2003): *The PROMPT Suite: Interactive Tools for Ontology Merging and Mapping,* 2003.

OMG05 OMG SPECIFICATION: *Unified Modeling Language (UML) Specification: Infrastructure, Version 2.0, Final Adopted Specification*, July 2005, formal/05-07-05

OMG06 OBJECT MANAGEMENT GROUP (OMG): The Unified Modeling Language (UML), 2.0 Resource page, available online at http://www.uml.org/, last viewed 2006-05-17

OrEtAl06 OREN, Eyal; MÖLLER, Knud Hinnerk; SCERRI, Simon; HANDSCHUH, Siegfried; SINTEK, Michael (2006): *What are Semantic Annotations?*. In: ISWC 2006.

PinMar04 PINTO, Helena Sofia; MARTINS, Joao: *Ontologies: How can they be built?* Knowledge and Information Systems, 2004.

Probst00 PROBST, Gilbert: *Manager Bilanz.* (2000), Nr. 10.

ReEtAl04 RECTOR, Alan; DRUMMOND, Nick; HORRIDGE, Matthew; ROGERS, Jeremy; KNUBLAUCH, Holger; STEVENS, Robert; WANG, Hai; WROE, Chris: *OWL Pizzas: Practical Experience of Teaching OWL-DL: Common Errors & Common Patterns.* In: Engineering Knowledge in the Age of the Semantic Web, LNCS 3257, 2004

ReeHan05 REEVE, Lawrence; HAN, Hyoil: *Survey of Semantic Annotation Platforms.* ACM SAC 2005. Online abgerufen am 06.11.2007 unter http://www.csd.uoc.gr/~hy566/Survey%20of%20Semantic%20Annotation%20Platforms.pdf

Reif01 REIF, Gerald: *Semantische Annotation*. In: Semantic Web – Wege zur vernetzten Wissensgesellschaft. Springer-Verlag, Berlin, Heidelberg, 2006.

Rockle03 ROCKLEY, Ann: Managing enterprise content : Aunified content strategy. 1. ed. Indianapolis, Ind. : New Riders, 2003 – ISBN 0-7357-1306-5

RueBat94 RUESCH, J. BATESON, G.: *Communication—A systems theoretical point of view* (1994). In: Journal Systemic Practice and Action Research, Vol 7 (1), 1994.

Rupp07 RUPP, CHRIS & DIE SOPHISTEN: Requirements-Engineering und Management – Professionelle, iterative Anforderungsanalyse für die Praxis, 4. Auflage, Hanser Verlag, München, 2007.

SanCas01 V.F.A. SANTANDER, J.F.B. CASTRO: Developing Use Cases from Organizational Modeling. IEEE 2001, available online at http://wer.inf.puc-rio.br/WERpapers/artigos/artigos_WER01/santander.pdf.

Schott06 SCHOTT, Georges: *Entwicklung eines Lean Documentation Konzepts zur Effizienzanalyse und Optimierung der Softwaredokumentation bei der SAP AG*. Karlsruhe, HS Karlsruhe, 31. Dezember 2006

Seeger97 SEEGER, Thomas: Grundbegriffe der Information und Dokumentation. In: BUDER, Marianne (Hrsg.). *Grundlagen der praktischen Information und Dokumentation : Ein Handbuch zur Einführung in die fachliche Informationsarbeit*. 3. völlig neu gefaßte Ausgabe. München et al. : Saur, 1997.

SelAus03 SELBERG, S.A.; AUSTIN, M. *Requirements Engineering and the Semantic Web – Part I*, Technical Report 2003-20, Institute for Systems Research, April 2003.

Sieber99 SIEBER, Tanja: *Funktion von Wissen im Kommunikationsprozess.* Karlsruhe, FH Karlsruhe, Seminararbeit, 1999.

SieKam06 SIEBER, Tanja ; KAMMERER, Matthias: *Sind Metadaten bessere Daten? : Metadaten als Mittler zwischen Daten und Prozessen.* In: *technische Kommunikation* 5 (2006), Nr. 5/06, S. 56–58.

SieKam07 SIEBER, Tanja; KAMMERER, Matthias: *Daten, Wissen und Information. Eine Grundlagenanalyse unter besonderer Berücksichtigung der technischen Dokumentatio*n. Unveröff. Typoskript, Version 16. 2007.

SieKov05 SIEBER, Tanja ; KOVÁCS, László: *Technical Documentation:Terms, problems and challenges in managing data, information and knowledge.* In: UNIVERSITY OF MISKOLC (HRSG.). *5th International Conf. of PhD Students*. , 2005, S. 165–170.

SieKov06 SIEBER, Tanja ; KOVÁCS, László: *Unbemerkt und unsichtbar.* In: *technische Kommunikation* (2006), Nr. 6/06.

StEtAl01 STAAB, Steffen; STUDER, Rudi; SCHNURR, Hans-Peter; SURE, York: *Knowledge Processes and Ontologies*. In: IEEE Intelligent Systems, January/February 2001.

StoWeb06 STOCK, I.; WEBER, M. *Authoring Technical Documentation Using a Generic Document Model*, SIGDOC06, Myrtle Beach, South Carolina, USA, October 2006.

StrHar05 STUCKENSCHMIDT, Heiner; VAN HARMELEN, Frank: *Information Sharing on the Semantic Web*. Springer-Verlag, Heidelberg 2005.

Sure03 SURE, York: *Methodology, Tools & Case Studies for Ontology based Knowledge Management.* Karlsruhe, Universität Friedericiana zu Karlsruhe, Fakultät für Wirtschaftswissenschaften, Mai 2003.

SwEtAl96 SWARTOUT, Bill; PATIL, Ramesh; KNIGHT, Kevin; RUSS, Tom (1996): *Towards Distributed Use of Large-Scale Ontologies*, KAW 1996.

tekom05 TEKOM (HRSG.): *Effizientes Informationsmanagement durch spezielle Content-Management-Systeme : Praxishilfe und Leitfaden zu Grundlagen – Auswahl und Einführung – Systemen am Markt.* 2005

Tratar06 TRATAR, E. *Anforderungen erfassen, aber mit System: Modellbasiertes Requirements Engineering*, ObjektSPEKTRUM 06/2006, verfügbar online unter http://www.gebit.de/Aktuelles/MDRE_OS_06_06.pdf

Underw03 UNDERWOOD, M.: *CCMS –Communicatioin, Cultural and Media Studies*, online verfügbar unter http://www.cultsock.ndirect.co.uk/MUHome/cshtml/index.html.

UniKai05 UNI KAISERSLAUTERN: *V-Modell XT – Anforderungen (Lastenheft) für WiBe 4.0 Musterprojekt*, online verfügbar unter http://ftp.uni-kl.de/pub/v-modell-xt/Release-1.2/Beispielprojekte/WiBe/.

UscGrü96 USCHOLD, Mike; GRÜNNINGER, Michael (1996): *Ontologies: Principles, Methods and Applications*. Knowledge Engineering Vol. 11, Nr. 2.

UscKin95 USCHOLD, Mike; KING, Martin (1995): *Toward a Methodology for Building Ontologies*. IJCAI 1995.

Usecase06 USE CASE ZONE: "Use Case Papers and Links", available online at http://www.pols.co.uk/use-case-zone/use-case-papers.html, last viewed 2006-05-17.

VrEtAl05 VRANDECIC, Denny; SOFIA PINTO, H.; SURE, York; Tempich, Christoph (2005): *The DILIGENT Knowledge Processes*. In: Journal of Knowledge Management 9 (5), S. 85-96, Oktober 2005.

Weilki06a WEILKIENS, T. *Requirements-Engineering mit der UML*, ObjektSPEKTRUM, 01/2006.

Weilki06b WEILKIENS, T. *Systems Engineering mit SysML / UML*, dpunkt-Verlag, 2006.

Wiegan89a WIEGAND, H.: *Aspekte der Makrostruktur im allgemeinen einsprachigen Wörterbuch: alphabetische Anordnungsformen und*

ihre Probleme. In: Wörterbücher. Ein internationales Handbuch zur Lexikographie, hrsg. von Franz Josef Hausmann, Oskar Reichmann, Herbert Ernst Wiegand, Ladislav Zgusta, Bd. 1. Berlin/New York 1989 (Handbücher zur Sprach- und Kommunikationswissenschaft 5), S. 371-409.

Wiegan89b WIEGAND, H.: *Der Begriff der Mikrostruktur: Geschichte, Probleme, Perspektiven.* In: Wörterbücher. Ein internationales Handbuch zur Lexikographie, hrsg. von Franz Josef Hausmann, Oskar Reichmann, Herbert Ernst Wiegand, Ladislav Zgusta, Bd. 1. Berlin/New York 1989 (Handbücher zur Sprach- und Kommunikationswissenschaft 5), S. 409-462.

Willke04 WILLKE, Helmut: *Einführung in das systemische Wissensmanagement.* Carl Auer Verlag, 2004.

Wolf06 WOLF, Timo: *SYSIPHUS: Modellbasierte Kollaboration in Software-Entwicklungsprojekten*, ObjektSPEKTRUM 02/2007.

Ziegle04 ZIEGLER, Wolfgang: *Lohnt sich Content Management?* : Aspekte zum Einsatz von Redaktionssystemen. In: technische Kommunikation 26 (2004), Nr. 6/2004, S. 19–23

Anhang A: Evaluationsleitfaden

Modellierungsmethodologie		Methodologie A				
Kriterien	**Prio (1=niedrig, 2=mittel, 3=hoch)**					
Verständlichkeit der Dokumentation der Methodologie		**1**	**2**	**3**	**4**	**5**
Werden spezifische Vorkenntnisse vorausgesetzt?	2	Umfang und Spezifizität der vorausgesetzten Kenntnisse liegt weit über dem, was bei der angesprochenenen Zielgruppe vorausgesetzt werden kann.				es sind keine spezifischen Kenntnisse vorausgesetzt, die über dem zu erwartenden Kenntnisstand der Zielgruppe liegen
Fördert Art, Aufbau und Umfang der Dokumentation Lesbarkeit und Verständlichkeit?	3	Art, Aufbau und Umfang erschweren in großen Bereichen ein leichtes Lesen und Verstehen der Dokumentation und tragen dazu bei, dass die Methode/das Modell nicht verstanden wird				hervorragend gewählte Struktur, gestalterische verständnisunterstützende Komponenten --> leichtes Lesen und durchgängige sehr gute Verständlichkeit der Dokumentation fördern das Erfassen des Inhalts
Ist die Methodologie vollständig dokumentiert?	3	Nach dem Lesen der Dokumentation ist völlig unklar, was nun konkret gemacht werden soll, um die Methode anzuwenden, evtl. auftretende Fragen/Probleme beim Anwenden der Methode sind nicht im Rahmen der Dokumentation abgehandelt				Es bleiben keine Fragen offen, man kann direkt beim Anwenden der Methode loslegen, die Dokumentation unterstützt bei evtl. auftretenden Fragen/Probleme während des Anwendens der Methode

Applikation der Methodologie		1	2	3	4	5
Handelt es sich um einen anerkannten Standard?	3	nein	ja	ja	ja	ja
Domänenspezifisch? Ist es direkt verwendbar oder muss es zuerst an eine Domäne angepasst werden?	3	muss zuerst komplett auf eigene Domäne angepasst und transferiert werden				kann direkt eingesetzt werden
Tools = Sind Tools zur Nutzung der Methodologie vorhanden?	2	nein, und es ist auch nicht abzusehen, dass welche entwickelt/angeboten werden sollen	nein, aber für die Zukunft vorgesehen	Diverse Tools verfügbar die Teilbereiche abdecken	Durchgängiges Tool von einem Hersteller	Ja, verschiedene Tools von diversen Herstellern, werden ständig angepasst
Methodologie kostenlos verfügbar?	2	nein				ja
Integration mit anderen Methodologien möglich? (Interoperabilität)	1	nein, ist nicht möglich	nein, aber für die Zukunft vorgesehen	Ja, prinzipiell möglich aber noch nicht implementiert	Ja, erste Implementierung	ja, und bereits für Standardkombinationsmöglichkeiten implementiert
Wird eine bestimmte Ausgangsbasis vor Anwendung der Methodologie vorausgesetzt?	2	Ja, im Vorfeld müssen spezielle Tätigkeiten ausgeführt werden, die nicht als Grundlegende in der angesprochenen Domäne vorauszusetzen sind				Nein, kann direkt angewendet werden
Wurde die Methodologie speziell im Zusammenhang mit einer bestimmten Auszeichnungssprache definiert? (XML, OWL, etc.)	1	Sehr viele Abhängigkeiten zu anderen Sprachen		Eine Abhängigkeit zu einer anderen Sprache		Keine Abhängigkeiten
Ist die Anwendung der Methodologie toolunabhängig?	2	Komplett toolabhängig		nein, aber ohne ein bestimmtes Tool wird es schwer		Ist toolunabhängig, einfach mit anderen Tools anzuwenden
Ist die Methodologie im industriellen oder akademischen Umfeld bekannt und weit verbreitet?	2	sehr niedriger Bekanntheitsgrad		Ist bekannt, wird aber nicht eingesetzt	Teilweise eingesetzt	Wird in Lehre und bei Firmen bereits großflächig eingesetzt
Ist die Methodologie in sich konsistent?	3	nein, die Methode/das Modell widerspricht sich selbst an vielen Stellen		größtenteils konsistent, Teile inkonsistent		ja, durchgängige Konsistenz vorhanden und dadurch Transparenz gewährleistet

Qualität des Ergebnisses		**1**	**2**	**3**	**4**	**5**
Kann das Ergebnis nach Anwenden der Methodologie direkt im anzunehmenden Arbeitsumfeld eingesetzt werden?	3	Ergebnis muss nochmals überarbeitet werden				Ergebnis direkt anwendbar
Ist das Ergebnis leicht zu pflegen und zu warten? (erweiterbar, flexibel)	3	Pflege und Wartung sehr komplex		Pflege und Wartung möglich, aber nicht beschrieben		Pflege und Wartung einfach. Bewährte Vorgehensweise bzw. Evolution beschrieben
Sind unterschiedliche Ergebnisse nach Anwendung der gleichen Methodologie leicht zu kombinieren?	1	Ergebnisse nicht kombinierbar	Ergebnisse können kombiniert werden, aber nur manuell		Kombination vorgesehen (durch Tools unterstützt, etc.)	Kombination kann automatisch durchgeführt werden
Kann das Ergebnis leicht durch Standards weiterverarbeitet werden (z.B. XML)?	2	keine Weiterarbeitung durch Standards möglich				Weiterverarbeitung durch Standards möglich
Sind Tools zur Weiterverarbeitung des Ergebnisses vorhanden?	1	keine Tools vorhanden				Tools vorhanden aber Toolunabhängig
Personalisierung		**1**	**2**	**3**	**4**	**5**
Ist der kommunikative Rahmen vollständig beschrieben zur formalen Umsetzung? (Struktur und Umfang einer zukünftigen Nachricht, Wissensbasis von Sender und Empfänger, Daten & Metadaten)						
Ist das Wissensnetz des Senders umfassend dargestellt?	2	Wissensnetz nicht dargestellt		Wissensnetz teilweise dargestellt		Wissensnetz umfassend dargestellt
Ist das Wissensnetz des Empfängers umfassend dargestellt?	3	Wissensnetz nicht dargestellt		Wissensnetz teilweise dargestellt		Wissensnetz umfassend dargestellt
Wird ein Rollenkonzept für die Methode zur Personalisierung entwickelt / ist vorgesehen? (Für welchen Empfänger schreibt der Sender?)	3	nein		ja, informell beschrieben	ja, ist formal beschrieben	ja und ist formal und standardisiert beschrieben

Re-Use		1	2	3	4	5
Ist der kommunikative Rahmen vollständig beschrieben zur formalen Umsetzung? (Struktur und Umfang einer zukünftigen Nachricht, Wissensbasis von Sender und Empfänger, Daten & Metadaten)						
Weist das Ergebnis eine gewisse Granularität auf? Entstehen dadurch definierte Zugriffsmöglichkeiten auf einzelne Bausteine?	3	Ein großer Brocken, keine Granularität		Möglichkeit in Module zu gliedern wäre vorhanden		durch feingranulare Bausteine einfache Zugriffsmöglichkeiten
Unterstützt mich die Methodologie darin, eine syntaktische Struktur für einzelne Bausteine festzulegen?	2	Keinerlei Strukturzwang		Struktur wird informell beschrieben		Struktur wird formal beschrieben
Unterstützt mich die Methodologie bei einer standardisierten anwendungsfallübergreifenden Einführung und Verwendung zu benennender Begriffe?	2	nein		Standardisierung wird nur für konkreten Anwendungsfall unterstützt		Standardisierung wird anwendungsfallübergreifend unterstützt und ist formal abgebildet
Unterstützt mich die Methodologie bei der Festlegung von anwendungsfallübergreifenden Standards für die auf dem resultierenden Modell beruhenden zukünftigen Instanzen? (Schreibregeln, Benennungsregeln)	2	nein		Standardisierung wird nur für konkreten Anwendungsfall unterstützt		Standardisierung wird anwendungsfallübergreifend unterstützt und ist formal abgebildet
Unterstützt mich die Methodologie die Gültigkeit der Module in einem anderen kommunikativen Rahmen zu spezifizieren? (Nachrichtenpakete in anderem Kontext?)	2	Module nur in einem kommunikativen Rahmen gültig		Module ggf. in anderem Kontext gültig		Module in jeden kommunikativen Rahmen gültig da selbstbeschreibend
Unterstützt die Methodologie eine flexible Versionierungsfähigkeit? (Variablen angeben) {Ist die Haltbarkeit der Wiederverwendung angegeben? Wann kann etwas nicht mehr verwendet werden? (Versionierung)}	2	Keine Haltbarkeit angegeben, keine Versionierung möglich				Verschiedene Versionen vorhanden mit Haltbarkeitsdatum versehen

Anhang B: Ontologie in OWL erstellt nach OTKM

```
<?xml version="1.0"?>
<rdf:RDF
    xmlns="http://www.ds-lab.org/ontology_methodologies/otkm#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
  xml:base="http://www.ds-lab.org/ontology_methodologies/otkm">
  <owl:Ontology rdf:about=""/>
  <owl:Class rdf:ID="Institution"/>
  <owl:Class rdf:ID="Bankangestellter">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Person"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="#Person">
    <owl:disjointWith>
      <owl:Class rdf:ID="Bank"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:ID="VirtuelleDinge"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:ID="GreifbarerGegenstand"/>
    </owl:disjointWith>
  </owl:Class>
  <owl:Class rdf:ID="Kreditkarte">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#GreifbarerGegenstand"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Endkunde">
    <rdfs:subClassOf rdf:resource="#Person"/>
  </owl:Class>
  <owl:Class rdf:about="#GreifbarerGegenstand">
    <owl:disjointWith>
      <owl:Class rdf:about="#Bank"/>
    </owl:disjointWith>
    <owl:disjointWith>
      <owl:Class rdf:about="#VirtuelleDinge"/>
    </owl:disjointWith>
    <owl:disjointWith rdf:resource="#Person"/>
  </owl:Class>
  <owl:Class rdf:about="#VirtuelleDinge">
    <owl:disjointWith>
      <owl:Class rdf:about="#Bank"/>
    </owl:disjointWith>
    <owl:disjointWith rdf:resource="#Person"/>
    <owl:disjointWith rdf:resource="#GreifbarerGegenstand"/>
  </owl:Class>
  <owl:Class rdf:ID="Banksystem">
    <rdfs:subClassOf rdf:resource="#VirtuelleDinge"/>
    <owl:disjointWith>
      <owl:Class rdf:ID="Konto"/>
    </owl:disjointWith>
  </owl:Class>
  <owl:Class rdf:about="#Konto">
    <rdfs:subClassOf rdf:resource="#VirtuelleDinge"/>
    <owl:disjointWith rdf:resource="#Banksystem"/>
  </owl:Class>
```

```
<owl:Class rdf:about="#Bank">
  <rdfs:subClassOf rdf:resource="#Institution"/>
  <owl:disjointWith rdf:resource="#VirtuelleDinge"/>
  <owl:disjointWith rdf:resource="#Person"/>
  <owl:disjointWith rdf:resource="#GreifbarerGegenstand"/>
</owl:Class>
<owl:ObjectProperty rdf:ID="arbeitetFuer">
  <rdfs:domain rdf:resource="#Bankangestellter"/>
  <rdfs:range rdf:resource="#Bank"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="besitzt">
  <rdfs:domain rdf:resource="#Person"/>
  <rdfs:range rdf:resource="#GreifbarerGegenstand"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="istKundeBei">
  <rdfs:domain rdf:resource="#Endkunde"/>
  <rdfs:range rdf:resource="#Bank"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="gehoertZu">
  <rdfs:domain rdf:resource="#Konto"/>
  <rdfs:range rdf:resource="#Bank"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="hatKonto">
  <rdfs:domain rdf:resource="#Endkunde"/>
  <rdfs:range rdf:resource="#Konto"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="ObjectProperty_15">
  <rdfs:range rdf:resource="#Banksystem"/>
  <rdfs:domain rdf:resource="#Bank"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="verwaltet">
  <rdfs:range rdf:resource="#Konto"/>
  <rdfs:domain rdf:resource="#Banksystem"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="Benutzer">
  <rdfs:range rdf:resource="#Bankangestellter"/>
  <rdfs:domain rdf:resource="#Banksystem"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="authentifiziertSichBei">
  <rdfs:range rdf:resource="#Banksystem"/>
  <rdfs:domain rdf:resource="#Endkunde"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="sperrt">
  <rdfs:range rdf:resource="#Kreditkarte"/>
  <rdfs:domain rdf:resource="#Bankangestellter"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="aktiviert">
  <rdfs:range rdf:resource="#Kreditkarte"/>
  <rdfs:domain rdf:resource="#Bankangestellter"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="geeignetFuer">
  <rdfs:range rdf:resource="#Konto"/>
  <rdfs:domain rdf:resource="#Kreditkarte"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="bedient">
  <rdfs:range rdf:resource="#Banksystem"/>
  <rdfs:domain rdf:resource="#Bankangestellter"/>
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:ID="Telefonnummer">
  <rdfs:domain rdf:resource="#Bankangestellter"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="Bankadresse">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
```

```
    <rdfs:domain rdf:resource="#Bank"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="Bankname">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:domain rdf:resource="#Bank"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="Adresse">
    <rdfs:domain rdf:resource="#Endkunde"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:ID="Telefonnummer_Kunde">
    <rdfs:domain rdf:resource="#Endkunde"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  </owl:DatatypeProperty>
  <owl:FunctionalProperty rdf:ID="BLZ">
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
    <rdfs:domain rdf:resource="#Bank"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="Kontonummer">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
    <rdfs:domain rdf:resource="#Konto"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="Pruefziffer">
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
    <rdfs:domain rdf:resource="#Kreditkarte"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="Kartennummer">
    <rdfs:domain rdf:resource="#Kreditkarte"/>
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="Ablaufdatum">
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#dateTime"/>
    <rdfs:domain rdf:resource="#Kreditkarte"/>
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="Art">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
    <rdfs:domain rdf:resource="#Kreditkarte"/>
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
  </owl:FunctionalProperty>
</rdf:RDF>

<!-- Created with Protege (with OWL Plugin 3.2.1, Build 365)
http://protege.stanford.edu -->
```