Institut für Produktion und Industrielles Informationsmanagement

Universität Essen
Fachbereich 5: Wirtschaftswissenschaften
Universitätsstraße 9, D – 45141 Essen
Tel.: ++49 (0) 201/ 183–4006, Fax: ++49 (0) 201/ 183–4017

KOWIEN–Projektbericht 1/2003- V. 1.0

Analyse von Vorgehensmodellen aus dem Software, Knowledge und Ontologies Engineering

Susanne Apke, Lars Dittmann

Susanne.Apke@pim.uni-essen.de

Lars.Dittmann@pim.uni-essen.de

KOWIEN
KOOPERATIVES WISSENSMANAGEMENT IN ENGINEERING-NETZWERKEN

Das Projekt KOWIEN (“Kooperatives Wissensmanagement in Engineering-Netzwerken”) wird mit Mitteln des Bundesministeriums für Bildung und Forschung (BMBF) gefördert (Förderkennzeichen Hauptband 02 PD1060) und vom Projektträger Produktion und Fertigungstechnologien (PFT), Forschungszentrum Karlsruhe GmbH, betreut.
Die Mitglieder des Projektteams danken für die großzügige Unterstützung ihrer Forschungs- und Transferarbeiten.

März 2003

Inhaltsverzeichnis

Seiten

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungs- und Akronymverzeichnis

AAAI	American Association for Artificial Intelligence
ACM	Association for Computing Machinery
AIAI	Artificial Intelligence Applications Institute
AIFB	Angewandte Informatik und formale Beschreibungssprachen
BMBF	Bundesministerium für Bildung und Forschung
bspw.	beispielsweise
DAML+OIL	DARPA Annotated Markup Language and Ontology Inference Layer
DARPA	Defense Advanced Research Projects Agency
DIAM	Département Intelligence Artificielle et Médecine
DV	Datenverarbeitung
d.h.	das heißt
ECAI	European Conference on Artificial Intelligence
EPK	Ereignisgesteuerte Prozesskette
ER	Entity-Relationship
et al.	et alius oder et alii (und weitere)
etc.	et cetera (und so weiter)
f.	folgende (Seite)
ff.	fort folgende (Seiten)
Hrsg.	Herausgeber
IDEF	Integrated Computer Aided Manufacturing Definition
i.d.R.	in der Regel
IEEE	Institute of Electrical and Electronics Engineers, Inc.
IFIP	International Federation for Information Processing
IJCAI	International Joint Conference on Artificial Intelligence
IT	Informationstechnologie
KACTUS	modelling Knowledge About Complex Technical systems for multiple USe
KADS	Knowledge Analysis and Design System
KAW	Knowledge Acquisition Workshops
KBSI	Knowledge Based Systems Incorporated
KI	Künstliche Intelligenz
KIF	Knowledge Interchange Format
KM	Konfigurationsmanagement
KOWIEN	Kooperatives Wissensmanagement in Engineering-Netzwerken
ONIONS	ONtologic Integration Of Naïve Sources
ONTOS	Ontology-Driven Web Service Composition Platform

PIM	Produktion und Industrielles Informationsmanagement
PM	Projektmanagement
QS	Qualitätssicherung
RDF	Resource Description Framework
S.	Seite
SE	Softwareentwicklung
SEU	Softwareentwicklungsumgebung
s.u.	siehe unten
Tf	Testfälle
TOVE	Toronto Virtual Enterprise
UML	Unified Modeling Language
URL	Unified Resource Locator
vgl.	vergleiche
Vol.	Volume (Jahrgang)
vs.	versus
WebODE	Web Ontology Design Environment
WESCON	Western Electronic Show and Convention
www	World Wide Web
z.B.	zum Beispiel

1 Einleitung

Im Rahmen des Verbundprojekts KOWIEN (Kooperatives Wissensmanagement in Engineering-Netzwerken)[1] wird unter anderem geplant, ein generisches Vorgehensmodell zur Konstruktion von Ontologien, die in einem Kompetenzmanagementsystem zum Einsatz gebracht werden sollen, zu entwickeln. Aus diesem Grunde untersucht der vorliegende Projektbericht das Themenumfeld in der Fachliteratur, in dem das generische Vorgehensmodell entstehen soll.

Vorgehensmodelle werden als allgemeine Anleitungen für die Abwicklung von Aufgaben eingesetzt, um die Planung, Durchführung und Kontrolle von vergleichbaren Problemlösungsprozessen zu erleichtern und die Wiederverwendung von Wissen zu unterstützen. Sie definieren die bei einem Prozess durchzuführenden Aktivitäten, ihre Reihenfolge, die dabei entstehenden Artefakte, die daran beteiligten Personen und weitere benötigte Ressourcen und legen somit den organisatorischen Rahmen für die Abwicklung von Aufgaben, insbesondere von Projekten, fest. Insbesondere beziehen sich die Autoren auf Vorgehensmodelle, die die Aktivitäten in den verschiedenen Phasen eines Software-Projekts über dessen Lebenszyklus widerspiegeln.

Der Einsatz von Vorgehensmodellen lässt folgende Punkte der Nutzenstiftung anführen:

- Planungssicherheit (ermöglicht Vergleichbarkeit),
- Effizienz (Reduzierung von Redundanzen und Nachbearbeitungen),
- Kostenreduzierung (ermöglicht durch ein frühzeitiges Reagieren),
- Qualität (durch kontrolliertes Vorgehen),
- Transparenz (bei Eigenentwicklungen).

In der Erforschung softwaretechnischer Systeme lässt sich eine Dreiteilung bezüglich der Anwendung von Vorgehensmodellen erkennen:

- Vorgehensmodelle aus dem Software Engineering,
- Vorgehensmodelle aus dem Knowledge Engineering,

---

1) KOWIEN wird mit Mitteln des Bundesministeriums für Bildung und Forschung (BMBF) gefördert. Förderkennzeichen Hauptband 02 PD 1060. Für weitere Informationen zu KOWIEN vgl.: http://www.kowien.uni-essen.de.

- Vorgehensmodelle aus dem Ontologies Engineering.

Das Software Engineering befasst sich allgemein mit der Entwicklung von Software auf der Grundlage ingenieurmäßiger Entwicklungsmethoden. Innerhalb der Softwareentwicklung konzentriert sich das Knowledge Engineering auf die Entwicklung wissensbasierter Systeme, d.h. auf Systeme, die zur Zielsetzung haben, Wissen mittels Computer zu verarbeiten. Innerhalb der Entwicklung wissensbasierter Systeme konzentriert sich wiederum das Ontologies Engineering auf die Konzeptualisierung des Wissens einer speziellen Domäne und die formalsprachige Repräsentation dieser Konzeptualisierung. Ontologien werden dabei als formalsprachige Spezifikation der sprachlichen Ausdrucksmittel einer Konzeptualisierung verstanden.[2] Die erwähnte Dreiteilung spiegelt den Aufbau dieser Arbeit wieder.

2) Für eine genauere Definition des Begriffs „Ontologien“ vgl. GRUBER (1993) und für eine Definition aus Sicht der Autoren vgl. ALPARSLAN (2002), S. 46 oder Zelewski (2002), S. 66 f.

2 Vorgehensmodelle des Software Engineerings

2.1 Einführung

Das Software Engineering befasst sich mit der Anwendung von ingenieurmäßigen Prinzipien bei der Erstellung von Software.[3] Durch den Einsatz von ingenieurmäßigen Prinzipien werden bei der Erstellung von Software die Ziele verfolgt, die vorgenannten nutzenstiftenden Punkte zu erreichen. Von der Forschungsdisziplin des Software Engineerings wird erwartet, dass sie Methoden und Werkzeuge bereitstellt, die es erlauben,

- *einerseits technische* Probleme
  (Spezifikation, Entwurf und Implementierung von Software) und
- *andererseits organisatorische* Probleme
  (Projektorganisation und Schnittstellenmanagement),

die bei der Erstellung von Software auftreten können, zu lösen und damit die intendierten Ziele zu erreichen.

Im Bereich des Software Engineerings werden Vorgehensmodelle in vielfältigen Variationen beschrieben.[4] Dabei wird in diversen Vorgehensmodellen ein idealisierter Ablauf des Softwareentwicklungsprozesses vorgestellt. In diesen Vorgehensmodellen wird der Entwicklungsprozess in überschaubare Phasen zerlegt (es lässt sich anmerken, dass sich sämtliche Vorgehensmodelle in der Regel auf die vier groben Prozessphasen *Analyse, Entwurf, Realisierung* und *Einführung* zurückführen lassen oder auf Teilphasen hiervon), wodurch die Planung, Durchführung und Kontrolle des Software-Projekts ermöglicht werden soll. Die gesamten Phasen und die Ordnung ihrer zeitlichen Reihenfolge beschreibt man als *Software-Life-Cycle*.[5]

---

3) Vgl. SOMMERVILLE (2001), S. 6-7; BULLINGER (1997), S. 6.

4) Vgl. für einen Überblick über Vorgehensmodelle aus dem Software Engineering BULLINGER (1997), S. 6-14; POMBERGER (1993), S. 17-32; SOMMERVILLE (2001), S. 42-69 und 171-190; WANG (2002), S. 280-300.

5) Vgl. zum Begriff *Software-Life-Cycle* POMBERGER (1993), S. 18-19.

Die Vorgehensmodelle aus dem Software Engineering sind eine spezielle Form von Referenzmodellen,[6] denn Vorgehensmodelle enthalten *Empfehlungen.*

Das Ziel dieses Kapitels ist es, einen Überblick über Vorgehensmodelle aus dem Software Engineering zu geben. Hierzu werden die charakteristischen Merkmale ausgewählter Vorgehensmodelle (*sequentielles Software-Life-Cycle-Modell, Wasserfall-Modell, protoypingbasiertes Software-Life-Cycle-Modell* und *Spiral-Modell*) erläutert und hinsichtlich ihrer Vor- und Nachteile diskutiert. Diese Analyse dient als Grundlage für das im Projekt KOWIEN zu entwickelnde Vorgehensmodell zur Konstruktion eines ontologienbasierten Wissensmanagementsystems.

6) Vgl. zum Begriff Referenzmodell ROSEMANN (1999), S. 23-24; SCHEER (1999), S. 4-6 und SCHÜTTE (1998).

2.2 Ausgewählte Ansätze des Software Engineerings

2.2.1 Das sequentielle Software-Life-Cycle-Modell

Mit dem Begriff des *Software-Life-Cycle* ist die Vorstellung einer Zeitspanne von der Entwicklung über den Einsatz von Software bis zum Ende der Benutzung der Software verbunden. In Abbildung 1 ist das klassische sequentielle Phasenmodell[7], der Software-Life-Cycle wiedergegeben.

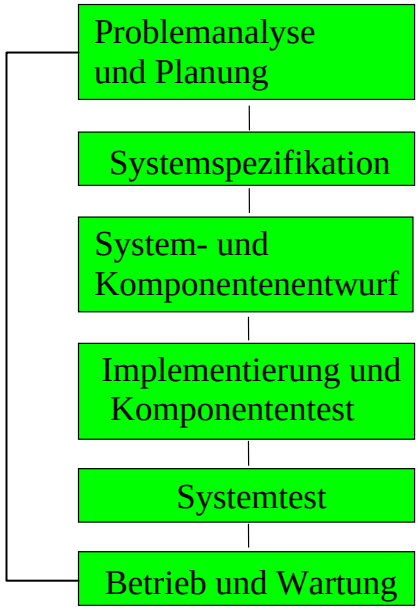

Abbildung 1: Das Software-Life-Cycle-Modell[8]

Das sequentielle Software-Life-Cycle-Modell besteht aus den Phasen: *Problemanalyse und Planung*, *Systemspezifikation*, *System- und Komponentenentwurf*, *Implementierung und Komponententest*, *Systemtest*, *Betrieb und Wartung*. Der Grundgedanke des sequentiellen Software-Life-Cycle-Modells ist, dass für jede der oben aufgeführten Phasen klar

7) Vgl. dazu POMBERGER (1993), S. 18-23; SCACCHI (2001), S. 5-7.

8) In Anlehnung an POMBERGER (1993), S. 18.

zu spezifizieren ist, welche Ergebnisse erwartet werden. Eine neue Phase wird erst dann begonnen, wenn die vorhergehende Phase abgeschlossen ist.

Im Folgenden werden die Ziele der Phasen beschrieben:[9]

(1) *Problemanalyse und Planung*

Das Ziel der Problemanalyse und Planung ist es, den Aufgabenbereich der zu entwickelnden Software festzustellen und zu dokumentieren. Zu der Problemanalyse und Planung gehört auch die Bestimmung der *finanziellen*, *personellen*, *technischen* sowie *zeitlichen* Ressourcen, die zur Realisierung des Software-Projekts erforderlich sind.

(2) *Systemspezifikation*

In der Phase der Systemspezifikation wird zwischen dem Auftraggeber und dem Softwareentwickler festgelegt, was die zu entwickelnde Software leisten soll und welche Vorbedingungen für ihre Realisierung gelten. Die Anforderungen an die Software lassen sich in *funktionale* und *nicht-funktionale* Anforderungen unterteilen. Die funktionalen Anforderungen definieren die vom Anwender erwarteten Systemfunktionen. Die nicht-funktionalen Anforderungen enthalten Anforderungen, wie z.B. die Zuverlässigkeit, Sicherheit und Portabilität sowie gewünschte Antwort- und Verarbeitungszeiten der Software. Darüber hinaus enthalten die nicht-funktionalen Anforderungen die Spezifikation der Formen, in der die Anwender mit der Software kommunizieren (*Benutzerschnittstelle*).

Als ein Kontrakt zwischen Auftraggeber und Auftragnehmer muss die Systemspezifikation für die Beteiligten *verständlich* formuliert sein. Als Grundlage für die Kommunikation zwischen Auftraggeber und Auftragnehmer, die aus unterschiedlichen Erfahrungskontexten stammen können (z.B. betriebswirtschaftlich ausgebildeter Manager und Informatiker) wurden Artefakte entwickelt.[10]

---

9) Vgl. für die einzelnen Phasen POMBERGER (1993), S. 18-20.

10) Ein Beispiel für ein derartiges Artefakt ist die Unified Modeling Language (UML). UML stellt semi-formale sprachliche Ausdrucksmittel bereit, mit denen der Auftraggeber und der Auftragnehmer die Anforderungen an die zu entwickelnde Software eindeutig beschreiben können; vgl. zu UML im Rahmen der Anforderungsspezifikation für das Projekt KOWIEN ALAN (2002), S. 13-15.

(3) *System- und Komponentenentwurf*

Das Ziel dieser Phase ist es, festzulegen, welche Systemkomponenten entwickelt oder erworben werden müssen, um die in der Systemspezifikation identifizierten Anforderungen abzudecken. Dabei stellt der Entwurf der Systemarchitektur die hierbei wichtigste Tätigkeit dar. Die Systemkomponenten werden durch

- den Entwurf ihrer Schnittstellen,
- die Festlegung ihrer Interdependenzen,
- die Spezifikation ihres zugrunde liegenden *logischen Datenmodells* und
- den Entwurf ihre algorithmischen Struktur

spezifiziert.

(4) *Implementierung und Komponententest*

Das Ziel der Implementierung ist die im System- und Komponentenentwurf erhaltenen Ergebnisse formalsprachlich umzusetzen und damit auf dem Rechner ausführbar zu machen.

(5) *Systemtest*

Das Ziel des Softwaretests ist es, die Interdependenzen der Softwarekomponenten unter realen Bedingungen zu prüfen, möglichst viele Fehler zu entdecken und zu beheben sowie zu gewährleisten, dass die Systemimplementierung die funktionalen und nichtfunktionalen Anforderungen erfüllt. Es wird sowohl verifiziert als auch validiert.

(6) *Betrieb und Wartung*

Bei dieser Phase handelt es sich zeitlich gesehen um die längste Teilphase des Software-Life-Cycles. Nach Abschluss des Systemtests wird die Software zur Anwendung freigegeben. Die Aufgabe der Wartung ist es, Fehler, die erst während des Betriebs der Software auftauchen, zu beheben sowie Erweiterungen der Software vorzunehmen.

2.2.2 Das Wasserfall-Modell

2.2.2.1 Allgemeine Darstellung

Das Wasserfall-Modell ist eine Modifikation des sequentiellen Software-Life-Cycle-Modells.[11] Abbildung 2 zeigt das Wasserfall-Modell mit einem Top-Down-Vorgehen.

Wie auch im sequentiellen Software-Life-Cycle-Modell werden hier die einzelnen Phasen als Gruppierungen von Aktivitäten verstanden, die jeweils vollständig bearbeitet und dokumentiert werden.

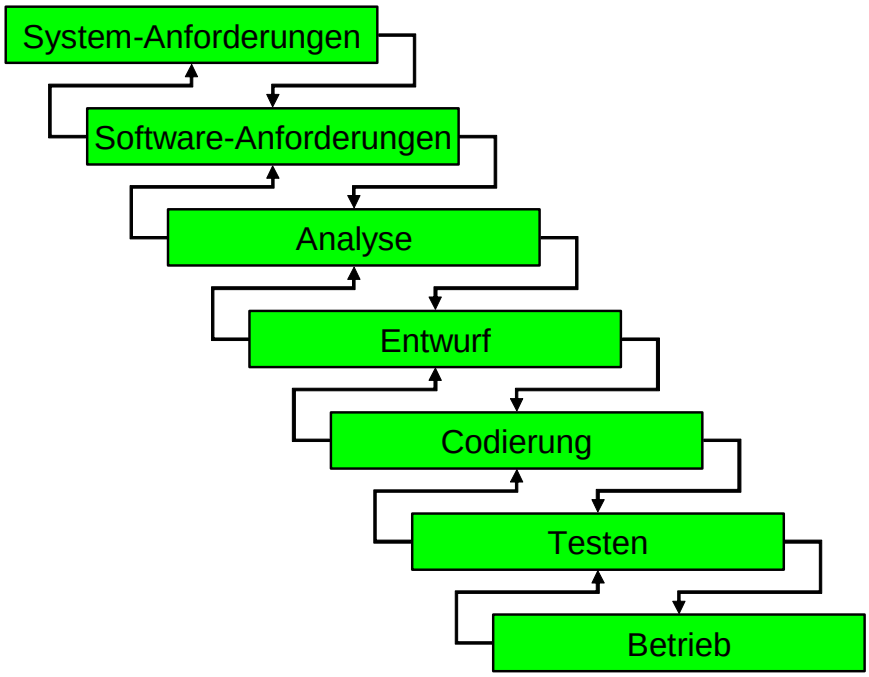

Abbildung 2: Das Wasserfall-Modell[12]

Das Wasserfall-Modell als eine Verfeinerung des sequentiellen Software-Life-Cycle-Modells enthält zwei grundlegende Modifikationen:[13]

11) Vgl. zum Wasserfall-Modell ROYCE (1970).

12) Vgl. BIETHAHN (2000), S. 206.

13) Vgl. POMBERGER (1993), S. 24.

- *Rückkopplung* zwischen den Phasen: Auf die strenge Sequentialisierung wird verzichtet. Stattdessen sind Rückkopplungen zwischen *aufeinander folgenden* Phasen vorgesehen.
- Validierung: Der Software-Life-Cycle wird um die (experimentelle) Validierung der Phasenergebnisse erweitert.

Mit diesen beiden Modifikationen wird die streng sequentielle Vorgehensweise des klassischen Software-Life-Cycle-Modells aufgeweicht. Die Rückkopplungen insbesondere für die Systemspezifikation und den System- und Komponentenentwurf sowie die phasenweise Validierung sollen es ermöglichen, den Softwareentwicklungsprozess besser kontrollierbar zu gestalten.

2.2.2.2 V-Modell

Seine große Bedeutung für die Öffentliche Hand berücksichtigend, wird an dieser Stelle kurz die Systematik des V-Modells der Vollständigkeit halber erläutert. Des Weiteren soll es dem Leser zur Veranschaulichung des bisher Geschriebenen dienen.

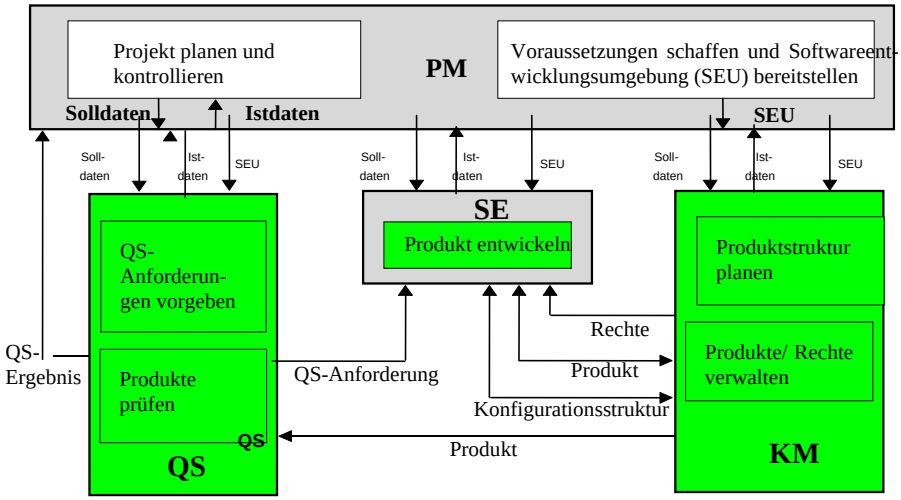

Abbildung 3: Interaktion der Submodelle innerhalb des V-Modells[14]

Aufgrund großem unkontrolliertem Wachstums und sich verkürzender Innovationszyklen kam es zu zunehmender Unübersichtlichkeit in den IT-Systemen der Bundesverwaltung, so dass sämtliche DV-Ressourcen in einigen Bereichen der Bundesverwaltung vollständig für die Softwarepflege eingesetzt werden mussten und für neue Entwicklungen keine Kapazitäten mehr verfügbar waren.[15] Eine Standardisierungsbemühung - anfänglich angestrebt vom Bundesministerium der Verteidigung - führte zur Formulierung des V-Modells. Das V-Modell legt die Vorgehensweise bei der Softwareentwicklung in seiner vom Bundesministerium des Inneren vorgesehenen Form verbindlich als Standard für den gesamten öffentlichen Bereich fest. Dabei lässt sich das Modell kurz anhand der folgenden Merkmale charakterisieren:

- Das Modell erhebt den Anspruch auf Allgemeingültigkeit.

14) Das V-Modell zum Download findet sich unter: http://www.informatik.uni-bremen.de/gdpa/, Zugriff am 28.01.2003. Die Abbildung wurde der Seite 2-9 entnommen.

15) Vgl. BRÖHL (1995), S.15.

- Es werden 25 Rollen für Managementaufgaben definiert.
- Es muss eine Anpassung an konkrete Anforderungen erfolgen.
- Es ist gedacht als Entwicklungsmodell für ein Gesamtsystem.
- Innerhalb des Modells existiert eine Aufteilung in Sub-Modelle (siehe Abbildung 3):
    - Systemerstellung (SE),
    - Qualitätssicherung (QS),
    - Konfigurationsmanagement (KM),
    - Projektmanagement (PM).

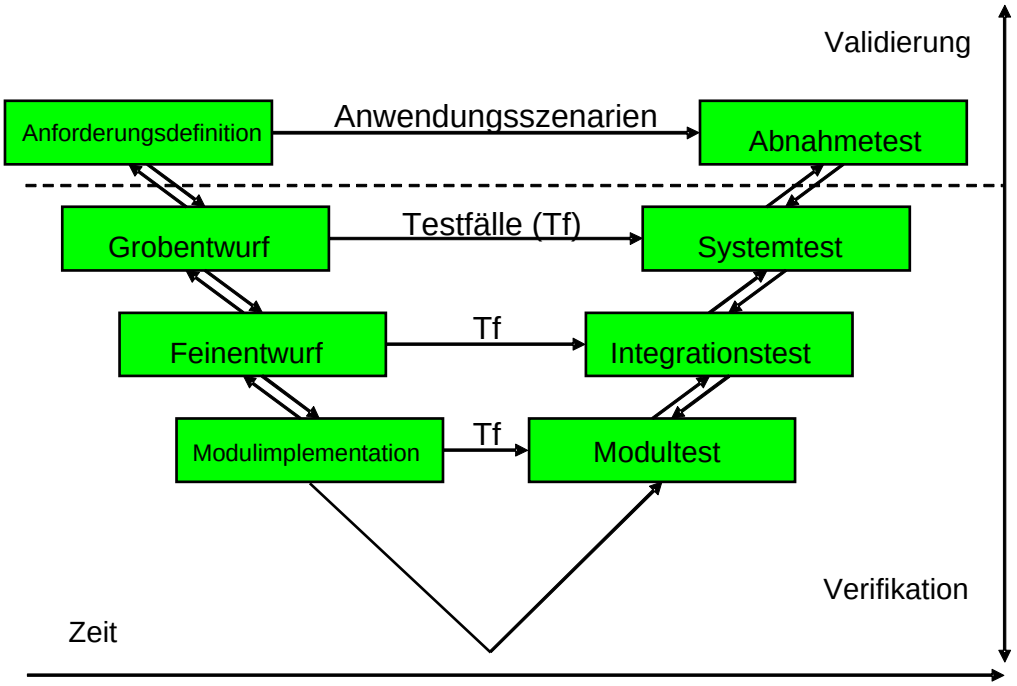

Abbildung 4: Übersicht über das V-Modell[16]

Insgesamt erfolgt die Software Entwicklung im Zeitverlauf von der Anforderungsdefinition zum Abnahmetest über Verifikation und anschließende Validation (siehe Abbildung 4). Hierbei ergibt sich in der Darstellung das charakterisierende „V“. Es entspricht somit in der Grundstruktur dem zuvor in Kapitel 2.2.2 beschriebenen Wasserfall-Modell, geht jedoch durch seine semi-formalisierten Vorgaben darüber hinaus.

16) Vgl. Quelle wird nachgereicht

2.2.3 Das protoypingbasierte Software-Life-Cycle-Modell

Im Unterschied zu den bisher vorgestellten Vorgehensmodellen unterscheidet sich das protoypingbasierte Vorgehen durch die Überlappung bestimmter Phasen und der Ergebnisse in den Phasen. Die Phaseneinteilung bleibt zwar erhalten, jedoch mit dem Unterschied, dass die Phasen der Problemspezifikation und Planung sowie Systemspezifikation zeitlich überlappt ablaufen und die restlichen Phasen bis auf Betrieb und Wartung integriert werden.[17] Die Phasen sind somit keine Teilabschnitte einer diskreten Softwareentwicklung.

Die Erstellung einer vereinfachten Version der Software oder Softwarekomponente (Prototyp) ist ein iterativer Prozess[18]: Dabei werden die Phasen Prototyp-Spezifikation, -Konstruktion und -Test solange wiederholt, bis der Anwender den Prototyp akzeptiert.

Anhand dieses Prototyps wird durch, realen Einsatzbedingungen entsprechende, Experimente untersucht, ob die Anforderungen erfüllt werden. Der Vorteil dieser Vorgehensweise ist nahe liegend: Schon frühzeitig kann der Anwender ausprobieren, ob der Prototyp die Anforderungen an die Software erfüllt und bei Bedarf können frühzeitig Änderungen vorgenommen werden. Damit wird das Risiko einer fehlerhaften und nicht vollständigen Spezifikation der Anforderungen reduziert.

Es wird ein wesentlicher Unterschied zwischen klassischem Vorgehen und dem protoypingbasierten Vorgehen deutlich: Während beim Erstgenannten sehr spät – nachdem alle Spezifikations- und Entwurfsprozesse abgeschlossen sind – implementiert wird, wird dagegen beim Letztgenannten sehr früh implementiert.

2.2.4 Das Spiral-Modell

Im Spiral-Modell[19] werden die bislang vorgestellten Vorgehensmodelle kombiniert oder als Sonderfälle integriert. Hierdurch wird die Möglichkeit geboten, eine den Anforde-

---

17) Vgl. POMBERGER (1993), S. 25; WANG (2002), S. 282.

18) Im Allgemeinen werden beim Prototyping (1) *exploratives* Prototyping, (2) *experimentelles* Prototyping und (3) *evolutionäres* Prototyping unterschieden; vgl. dazu SOMMERVILLE (2001), S. 171-189.

19) Das Spiral-Modell geht auf BOEHM zurück; vgl. BOEHM (1988) und SOMMERVILLE (2001), S. 53-55.

rungen eines Projekts entsprechende Vorgehensweise auszuwählen. Die folgende Abbildung zeigt diese Variante des Software-Life-Cycle-Modells.

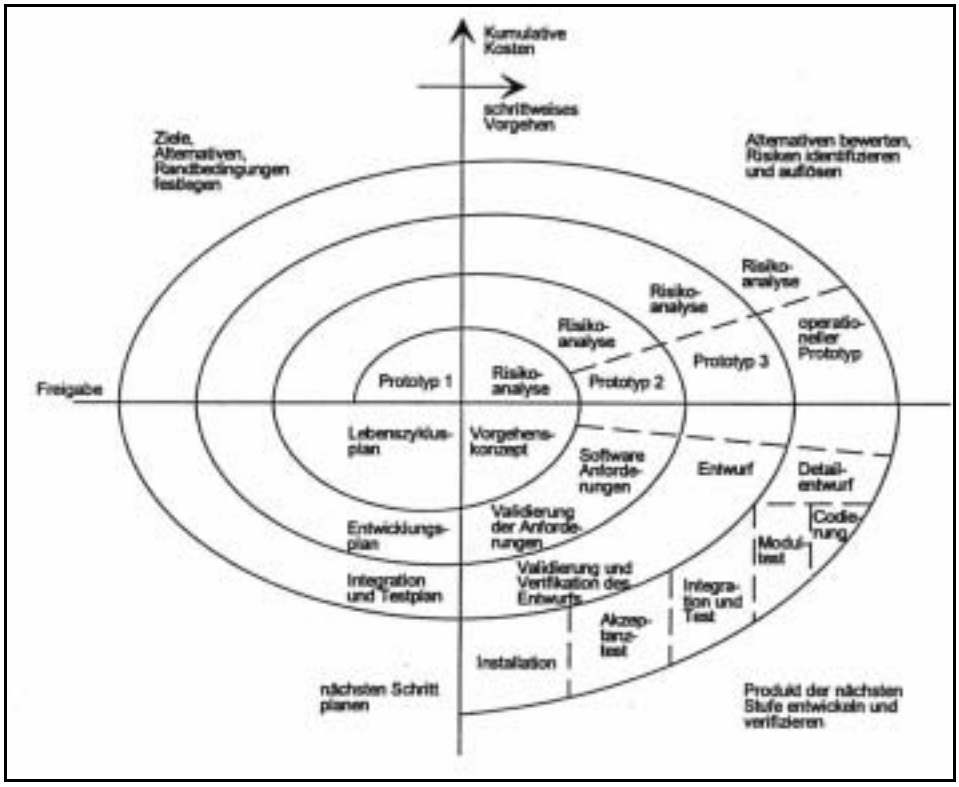

Abbildung 5: Das Spiral-Modell[20]

Während die Ausdehnung der Spirale in Abbildung 5 den bis zu einem bestimmten Zeitpunkt entstandenen Gesamtaufwand wiedergibt, beziehen sich die Winkeldimensionen auf den Projektschritt in den jeweiligen Spiralzyklen. Das Spiral-Modell fasst den Entwicklungsprozess als iterativen Prozess auf, wobei jeder Zyklus folgende Aktivitäten enthält:

- Festlegung von Zielen für und Identifikation von Anforderungen an das (Teil-)Produkt,
- Identifikation von Alternativen zur Realisierung des (Teil-)Produkts,
- Identifikation von Restriktionen (Kosten- und Zeitrestriktion sowie Interdependenzen mit anderen organisatorischen Einheiten).

20) Vgl. BIETHAHN (2000), S. 210.

Im Anschluss erfolgt die Evaluation der identifizierten Lösungsalternativen hinsichtlich der Projektziele und unter dem Aspekt der Identifikation potenzieller Risikoquellen. Werden Risikoquellen identifiziert, schließen sich Maßnahmen zur Reduzierung des Risikos an. Hierbei kann insbesondere das Prototyping eingesetzt werden.

Dieses Vorgehen wird im Uhrzeigersinn fortgesetzt und es folgt jeweils der nächste Schritt entsprechend des klassischen Phasenmodells mit den zusätzlichen Aktivitäten je Zyklus: Risikoanalyse, protoypingbasierte Validierung und Festlegung des Projektplans für den nachfolgenden Zyklus. Die Spirale endet, wenn der inkrementelle Entwicklungsprozess eine fertige Software ergibt.

2.3 Zusammenfassende Gegenüberstellung

Das Ziel dieses Kapitels ist es, ausgewählte Vorgehensmodelle aus dem Software Engineering vorzustellen und dabei auf ihre Besonderheiten einzugehen. Im Folgenden werden die vorgestellten Vorgehensmodelle hinsichtlich ihrer Vor- und Nachteile diskutiert.

Das sequentielle Software-Life-Cycle-Modell bildet eine wichtige Grundlage für die ingenieurmäßige Entwicklung von Software. Der Vorteil dieses Vorgehensmodells liegt in seiner übersichtlichen Struktur, in der die wichtigsten Aktivitäten des Entwicklungsprozesses definiert und gegeneinander abgegrenzt werden.

Dem stehen jedoch mehrere gravierende Nachteile gegenüber:

Das sequentielle Software-Life-Cycle-Modell basiert auf der Prämisse, dass die Anforderungen an die Software nach Abschluss der Spezifikationsphase vollständig erfasst sind. Angesichts der Komplexität der zu erstellenden Software ist eine vollständige Erfassung der Anforderungen der Anwender zu diesem Zeitpunkt oft unmöglich. Änderungs- und Erweiterungswünsche ergeben sich erst mit dem wachsenden Verständnis für den Anwendungsbereich während des Softwareentwicklungsprozesses.[21]

Die strikte Trennung der einzelnen Phasen stellt zudem oft eine ungerechtfertigte Idealisierung dar. Denn in der Praxis lassen sich vielfältige Überlappungen der einzelnen Phasen feststellen und die Interdependenzen sind komplexer als es in dem sequentiellen Software-Life-Cycle-Modell dargestellt wird.

---

21) Vgl. BIETHAHN (2000), S. 202.

Die sequentielle Vorgehensweise hat zur Konsequenz, dass erste spät „greifbare“ Ergebnisse erstellt werden, die unter realen Bedingungen evaluiert werden können. Dies hat zur Folge, dass Software-Fehler oder Änderungswünsche der Auftraggeber erst sehr spät identifiziert und nur unter hohen Kosten berücksichtigt werden können.

Das Wasserfall-Modell hat gegenüber dem sequentiellen Software-Life-Cycle-Modell zum einen den Vorteil, dass am Ende einer jeden Phase eine Validierung vorgesehen ist. Hierdurch können die Phasenergebnisse hinsichtlich ihrer Korrektheit und Vollständigkeit überprüft und bei Bedarf angepasst werden. Zum anderen sind im Wasserfall-Modell Rückkopplungen zu vorangegangenen Phasen vorgesehen, so dass nachträglich neue Erkenntnisse in vorangegangene Phasenergebnisse integriert werden können.

Wie auch am sequentiellen Software-Life-Cycle-Modell lässt sich am Wasserfall-Modell kritisieren, dass die Spezifikation der Anforderungen nur in der Entwurfsphase vorgesehen ist und dass die Anwender in den Softwareentwicklungsprozess nicht weiter involviert werden.

Das protoypingbasierte Vorgehen bietet gegenüber den zuvor vorgestellten Vorgehensmodellen den Vorteil, dass gerade eine frühzeitige Entwicklung der Software oder von Softwarekomponenten beabsichtigt wird. Hierdurch kann schon frühzeitig der Prototyp unter realen Bedingungen hinsichtlich der gewünschten Anforderungen der Anwender getestet und bei Bedarf modifiziert werden. Jedoch kann die wiederholte Entwicklung von Prototypen für eine Systemkomponente insbesondere bei großen Software-Projekten erhebliche Kosten verursachen.

Im Spiral-Modell wird versucht, die Stärken der anderen Vorgehensmodelle durch Integration oder Behandlung als Sonderfälle zu berücksichtigen. Die Vorteile des Spiral-Modells liegen in der expliziten Berücksichtigung des Risikos innerhalb des Softwareentwicklungsprozesses. So können schon frühzeitig durch die Entwicklung von Prototypen ungeeignete Lösungsvarianten identifiziert und Software-Fehler beseitigt werden.

In der folgenden Tabelle sind die vorgestellten Vorgehensmodelle des Software Engineerings synoptisch gegenübergestellt:

	Ent-wicklungs-prozess	Vorteile:	Nachteile:
Sequentielles Software-Life-Cycle-Modell	sequentiell	• lungsprozesses in zeitlich getrennte Abschnitte. • chung werden leichter kontrollierbar. • gung durch Aufgabenteilung.	• kein sequentieller Prozess, sondern jede einzelne Phase kann zu Rückwirkungen auf frühere Phasen führen. Das sequentielle Vorgehen spiegelt nicht die Gegebenheiten der Softwareentwicklung wider. • rekten und abgeschlossenen Anforderungsspezifikation ausgegangen. Hierbei wird nicht berücksichtigt, dass es zu Missverständnissen zwischen Auftraggeber und -nehmer kommen kann, die nachträglich revidiert werden müssen. • sion erst am Ende des Entwicklungsprozesses vorliegt, wird die Beseitigung von Fehlern aufwendig und teuer.
Wasserfall-Modell	sequentiell	• ergebnisse. • hergehenden Phasen bei Vorliegen von neuen Erkenntnissen und Identifikation von Fehlern.	• sequentiellen Software-Life-Cycle-Modell.
Prototypingbasiertes Software-Life-Cycle-Modell	iterativ	• eines vorläufigen Prototyps, so dass Änderungen und Fehler identifiziert und angepasst oder behoben werden können.	• tematik, die den Entwicklungsprozess mit Erfahrungswissen unterstützt.
Spiral-Modell	iterativ	• ergebnisse. • sikos des Softwareentwicklungsprozesses.	• Durchführung sehr komplexer, großer Entwicklungsprojekte.

Tabelle 1: Synopse Modelle des Software Engineerings

3 Vorgehensmodelle des Knowledge Engineerings

3.1 Einführung

Die klassischen Phasenmodelle lassen sich insgesamt für die Neuentwicklung eines wissensbasierten Systems durchaus anwenden. Jedoch sind sie nicht besonders geeignet für die Anforderungen, die sich bei der Wissensakquisition, d.h. beim Füllen des Systems mit „Leben“, ergeben. Das Knowledge Engineering befasst sich immer auch zu einem großen Teil mit der Wissensakquisition.

Prinzipiell lassen sich hier zwei Ansätze des Knowledge Engineerings unterscheiden. Zum einen wird der Ansatz des (Rapid-)Prototyping und zum anderen der modellorientierte Ansatz in der Literatur unterschieden.

Die vorgestellten Modelle des Software Engineerings lassen sich, wie bereits erwähnt wurde, prinzipiell auch zur Entwicklung wissensbasierter Systeme einsetzen. Sie werden jedoch oftmals nicht den spezifischen Anforderungen gerecht, die bei wissensbasierten Systemen auftreten können. Dem Leser sei deshalb empfohlen, die Modelle aus Kapitel 2 zum Verständnis zu berücksichtigen.

3.2 Ausgewählte Ansätze des Knowledge Engineerings

3.2.1 Ansatz des (Rapid-)Prototyping

Aus erkennbarem, d.h. zu repräsentierendem, Wissen wird bei diesem Ansatz frühzeitig ein Prototyp als erste Implementierung für die Wissensbasis und die Wissensakquisition entwickelt.

Ein typisches Life-Cycle-Modell beinhaltet in der Regel folgende Phasen:[22]

1. Identifikation

Das gesamte Projekt wird identifiziert und analysiert. Die Projektziele werden festgelegt.

22) Vgl. HAUN (2000), S. 202-203. Die Vor- und Nachteile des Prototypings wurden bereits in ähnlicher Form in Kapitel 2 berücksichtigt, aus Übersichtlichkeitsgründen sei diese Redundanz bitte zu entschuldigen.

2. Konzeptualisierung

Es wird ein Konzeptualisierung für das spätere wissensbasierte System erarbeitet. In dieser oftmals sehr zeitaufwendigen Phase werden möglichst umfassend die Anforderungen an den Prototypen festgelegt.

3. Formalisierung

Unter Berücksichtigung der entwickelten Konzeptualisierung wird eine formale Beschreibung des Wissens durchgeführt. Die Konzeptualisierung wird formalsprachlich spezifiziert, um im Expertensystem verarbeitet werden zu können. Hierzu ergeben sich Anforderungen an die Wissensrepräsentation, die sich teilweise aus der Konzeptualisierung vererben.

4. Implementierung

An die Formalisierung des Wissens schließt sich die Erstellung des Prototyps an. Auf die erste Entwicklung folgt die Verfeinerung gemäß den aufgestellten Anforderungen (siehe Punkt 5).

5. Verifikation und Validierung

Wenn im Testbetrieb entsprechend der aufgestellten Anforderungen Abweichungen festgestellt werden, so wird der Prototyp gegebenenfalls verändert. Streng genommen durchläuft die Entwicklung somit einen schleifenförmigen Prozess der Schritte 4 und 5.

Als Vorteile lassen sich für diesen Ansatz nennen:

- Wissensingenieur und Experte können Wissenstransfer direkt abstimmen,[23]
- Der Experte kann das Verhalten des Systems überprüfen,
- Kürzere Feedback-Zyklen und rasche Ergebnisse erhöhen Motivation der Beteiligten.

23) Der Wissensingenieur entwickelt das System und das Wissen des Experten wird mittels Akquisitionstechniken im System hinterlegt.

Als Nachteile lassen sich ins Feld führen:

- Repräsentationsart des Wissens wird zu einem sehr frühen Zeitpunkt bestimmt,
- Keine standardisierte Phaseneinteilung des Entwurfsprozesses,
- Sehr frühe Konzentration auf Implementierungsdetails,
- Sehr schwierig mentales konzeptionelles Bild zu revidieren.

3.2.2 Modellorientierter Ansatz

Historisch betrachtet ist der modellorientierte Ansatz eine Weiterentwicklung des (Rapid-)Prototypings. Durch ein systematischeres Vorgehen als beim Prototyping soll das wissensbasierte System erstellt werden.

In der Praxis findet sich häufig folgendes Life-Cycle-Modell:[24]

1. Anforderungsdefinition

Die Anforderungen der späteren Anwender und der Entwickler werden eruiert und dokumentiert.

2. Terminologie

Die vorkommenden Begriffe werden definiert oder aus der Fachterminologie übernommen. Hierbei bezieht man sich auf das Wissen der Experten.

3. Modellierung der Problemlösung

Die Problemlösungsmethoden, die in der Praxis Anwendung durch den Experten erfahren, werden für die Verwendung innerhalb des wissensbasierten Systems abgebildet.

4. Repräsentation und Implementierung

Das akquirierte Wissen wird formalsprachlich spezifiziert und im System implementiert.

---

24) Vgl. HAUN (2000), S. 204 ff.

5. Validierung und Weiterentwicklung

In Tests wird untersucht, ob gegebenenfalls das System weiterentwickelt werden muss, um die aufgestellten Anforderungen zu erfüllen.

Der hauptsächliche Unterschied zum herkömmlichen Software Engineering liegt in einer in einzelnen Punkten verfeinerten Arbeitsmethodik, die insbesondere Wissensanalyse, generische Problemlösungsmodelle und heuristische Klassifikationen umfasst.[25]

Als Vorteile lassen sich erkennen:

- Klare Trennung von Analyse und Implementierung,
- Das Modell ist nahe an der Terminologie des Experten,
- Der Interpretationsprozess wird transparent.

Als Nachteile sind jedoch zu nennen:

- Modelle sind i.d.R. nicht operational, d.h. es ist keine direkte Überprüfung des dynamischen Ablaufs möglich,
- Modelle verfügen i.d.R. nicht über eine formale Semantik.

3.2.3 CommonKADS

Ein bekanntes modellorientiertes Vorgehensmodell stellt KADS und die Weiterentwicklung zu CommonKADS dar.[26] Die Abbildung 6 zeigt die so genannte Model Suite, d.h. die einzelnen Modelle in ihrer obersten Ebene, die zusammen den CommonKADS-Ansatz bilden. Nach Durchlaufen eines *Organisationsmodells*, eines *Aufgabenmodells* und eines *Agentenmodells* werden innerhalb dieser Modelle die besonders wissensintensiven Abläufe eruiert und anschließend in einem Wissensmodell und einem Kommunikationsmodell analysiert. Für das Design-Modell werden Anforderungen für Schlussfolgerungsfunktionen aus dem Wissensmodell und Anforderungen zur Spezifikation für Interaktionsfunktionen aus dem Kommunikationsmodell abgeleitet. Der Bereich des Ontologies Engineerings findet in diesem Zusammenhang innerhalb des *Wissensmo-*

25) Vgl. HAUN (2000), S. 205 f.

26) Zu dieser Methodologie vgl. SCHREIBER (2001).

dells (Knowledge Models) seine Berücksichtigung.[27] SCHREIBER ET AL. verwenden den Begriff „Ontologien“ jedoch abweichend zu dem hier gebrauchten Verständnis. Vielmehr sprechen sie stattdessen von „domain schemas“.[28] Ontologien stellen für die Autoren in diesem Zusammenhang „generalized domain schemas“ dar.[29]

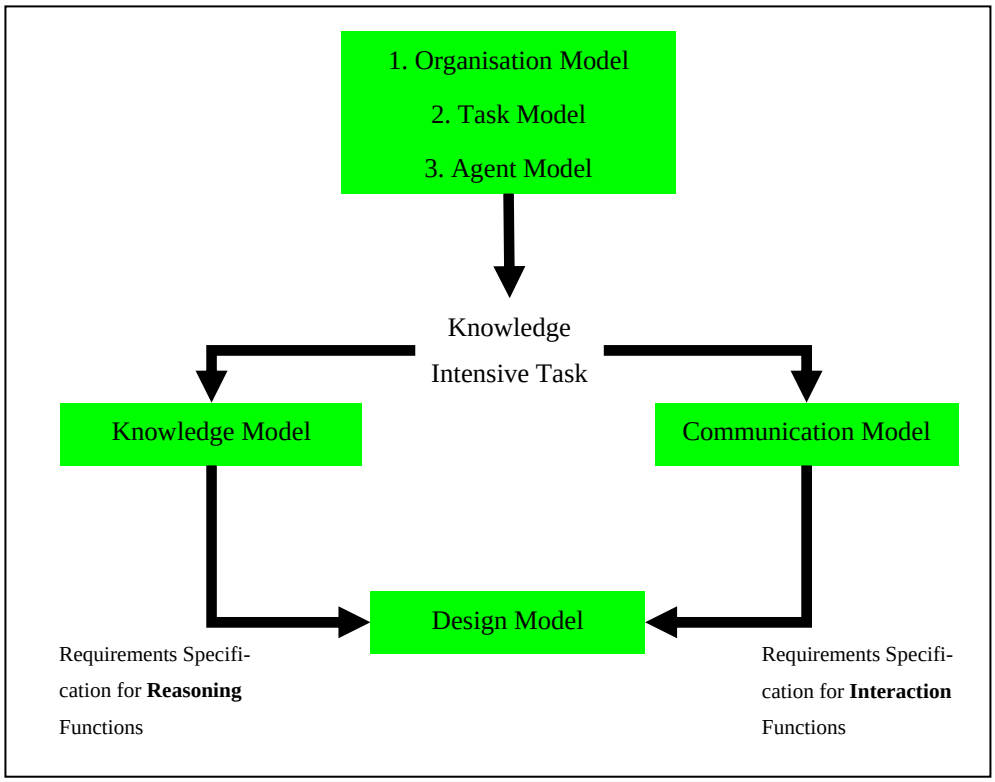

Abbildung 6: The CommonKADS Model Suite[30]

27) Vgl. zum Begriff des *Ontologies Engineerings* Kapitel 4, Seite 29 ff..

28) Vgl. hierzu insbesondere SCHREIBER (2001), S. 91.

29) Vgl. SCHREIBER (2001), S. 331.

30) Vgl. SCHREIBER (2001), S. 18 f.

3.3 Zusammenfassende Gegenüberstellung

Die Abbildung 7 stellt den Zusammenhang zwischen Prototyping und modellorientiertem Ansatz schematisch dar. Deutlich wird, dass der modellbasierte Ansatz ein explizites Modell in der Phase der Wissensakquisition berücksichtigt, während beim Prototyping ein mentales konzeptionelles Modell repräsentiert wird.

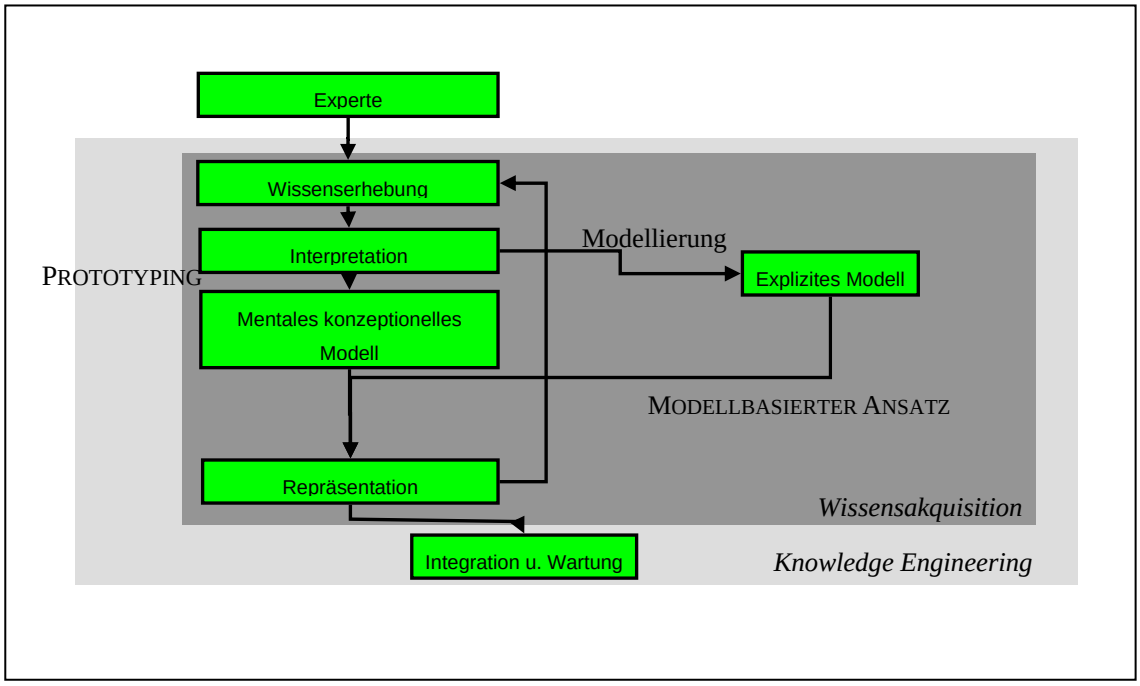

Abbildung 7: Prototyping vs. Modellierung[31]

31) In Anlehnung an HAUN (2000), S. 197.

4 Vorgehensmodelle des Ontologies Engineerings

4.1 Einführung

Um die Vorgehensmodelle aus dem Bereich des Ontologies Engineerings zu untersuchen, werden im Folgenden zuerst Anforderungen aufgestellt, anschließend werden bekannte Vorgehensmodelle vorgestellt und an Hand der Anforderungen analysiert. Nach einer Zusammenfassung der Analyseergebnisse werden noch einige für das Verbundprojekt KOWIEN relevante Vorgehensmodelle mit einem spezifischen Teilfokus zur Thematik vorgestellt.

4.2 Anforderungen

Die wichtigsten Anforderungen [32] an das Vorgehensmodell, das im Kontext des KOWIEN-Projekts eingesetzt werden soll, um die Entwicklung von Kompetenz-Ontologien zu beschreiben und zu strukturieren, werden im folgenden Abschnitt dargestellt.

- Generizität
- Anwendungsbezogenheit
- Vollständigkeit
- Begründung, Dokumentation
- Einfachheit
- Klarheit
- Werkzeugunterstützung

32) Ähnliche Zusammenstellungen von Kriterien für die Konstruktion von (Vorgehens-) Modellen und für ihre Bewertung sind beispielsweise in VERLAGE (1998), S.73 oder in MOODY (1994), S.101 f. zu finden. Die hier genannten Anforderungen wurden aufgrund ihrer Bedeutung in der Literatur bzw. wegen ihrer Relevanz für die im KOWIEN-Projekt vorliegenden spezifischen Gegebenheiten (wie z.B. Hintergrund der beteiligten Personen) ausgewählt.

4.2.1 Generizität

Das Vorgehensmodell soll insofern allgemeine Gültigkeit besitzen, als dass es nicht nur ausschließlich für ein bestimmtes Projekt (hier: KOWIEN) konzipiert ist, sondern auch im Rahmen ähnlicher Vorhaben, also etwa grundsätzlich für die Entwicklung ontologienbasierter Kompetenzmanagementsysteme, zum Einsatz kommen kann. Eine generelles, auch domänenunabhängiges Vorgehensmodell ist eine Grundvoraussetzung für die allgemeine Anerkennung und die Reife der Disziplin der Ontologienentwicklung.[33] Dadurch wird einerseits eine Wiederverwendung von Wissen ermöglicht, andererseits kann so ein Entwicklungsprozess leichter an projektspezifische Gegebenheiten angepasst werden.[34]

4.2.2 Anwendungsbezogenheit

Damit es im Anwendungsfall eine Hilfestellung zur Umsetzung eines Softwareentwicklungsprojekts leistet, soll das Vorgehensmodell auch konkrete Beispiele und Handlungsempfehlungen für den Anwender des Vorgehensmodells umfassen. Dazu zählt zum Bespiel eine detaillierte Spezifizierung von Aktivitäten, um das methodische Fundament des Ansatzes zu erläutern.[35] Diese Anforderung steht nicht zwangsläufig im Gegensatz zum Kriterium der Generizität; sie könnte etwa dadurch realisiert werden, dass zusätzlich zu einer allgemeinen Phasenbeschreibung jeweils Beispiele eine mögliche Umsetzung verdeutlichen.

4.2.3 Vollständigkeit

Das Vorgehensmodell soll vollständig sein im Sinne der Anwendungsziele, somit von der Ausgangssituation bis zur Zielerreichung alle relevanten Aspekte darstellen. Das bedeutet auf der einen Seite, dass (beispielsweise aus Sicht der Anwender) alle notwendigen oder sinnvollen Aktivitäten enthalten sein müssen. Auf der anderen Seite ist eine

---

33) Vgl. GUARINO (2002), S. 61.

34) Auch spezifische Modelle können selbstverständlich wieder verwendet werden, wenn die entsprechenden Rahmenbedingungen erfüllt sind. Nur die Wahrscheinlichkeit der Wiederverwendung ist *deutlich* geringer und auch ursprünglich nicht gewollt.

35) Vgl. FERNÁNDEZ LÓPEZ (1999), S. 4-3.

ausreichend genaue Beschreibung der einzelnen Schritte erforderlich, damit keine „Lücken“ entstehen, die bei der konkreten Projektdurchführung noch geschlossen werden müssen. Diese relative Definition von Vollständigkeit[36] führt dazu, dass die Erfüllung dieser Anforderung bei vielen Ansätzen quasi nicht überprüfbar ist, da die Anwendungsziele, Annahmen (im Sinne von Modellprämissen) und Ausgangssituationen oft nur implizit vorhanden oder nicht detailliert genug beschrieben sind. Aus diesem Grund wird das Kriterium der Vollständigkeit an dieser Stelle zwar aufgeführt und erläutert, jedoch nicht zur Beurteilung der in Kapitel 4.3 dargestellten Ansätze eingesetzt.[37]

4.2.4 Begründung und Dokumentation

Während jeder Phase sollen jeweils die ursprünglichen Intentionen und die Design Rationale[38] natürlichsprachlich (und nicht formalsprachlich) ausgeführt werden. Dieses Vorgehen erleichtert später eventuelle Rechtfertigungen gegen Änderungsvorschläge. Außerdem kann durch regelmäßige Dokumentation die Nachvollziehbarkeit der Ontologienentwicklung und somit auch die Akzeptanz unter Benutzern und Entwicklern bedeutend verbessert werden. Ein weiterer Vorteil ist, dass solche zusätzlichen Informationen nachfolgende Änderungen für konkrete Anwendungen vereinfachen. Aus diesen Gründen soll das Vorgehensmodell die Dokumentation des Entwicklungsprozesses berücksichtigen und konkrete Methoden zur Umsetzung nennen.

---

36) Vgl. z.B. SCHÜTTE (1997), S.7. Darin wird ein Modell als vollständig angesehen, wenn es alle im Metamodell beschriebenen Beziehungen zwischen den Objekten auch in der gleichen Form einhält. MOODY (1994), S. 102, betrachten ein (Daten-)Modell als vollständig, sobald es die Fähigkeit besitzt, alle funktionalen Anforderungen der Benutzer umzusetzen. Diese Definitionen von Vollständigkeit richten sich also ebenfalls nicht nach einem *allgemeingültigen* Referenzmodell, sondern sind abhängig von dem vorgegebenen Metamodell oder von den Zielen oder Erwartungen der Anwender des Modells.

37) Zwar wäre, wie bereits erwähnt, eine Bewertung der Vollständigkeit aus Sicht der Anwender des Vorgehensmodells (also der Ontologieentwickler) möglich. Das KOWIEN-Projekt ist jedoch zum Zeitpunkt der Verfassung dieses Textes noch nicht so weit fortgeschritten, dass die tatsächliche Ontologiekonstruktion vorgenommen werden kann.

38) Als Design Rationale wird hier die Zusammenfassung von Merkmalen und deren Begründungen für eine Gestaltungsentscheidung verstanden.

4.2.5 Einfachheit

Simplizität, bezogen auf die Größe und Komplexität eines Modells[39], ist wünschenswert, weil der Inhalt des Vorgehensmodells für alle Projekt-Beteiligten und Endanwender verständlich sein soll. Insbesondere bei der Entwicklung von betriebswirtschaftlichen Applikationen wie Kompetenzmanagementsystemen ist es wichtig, dass das Vorgehensmodell nicht zu komplex aufgebaut ist, weil während eines großen Teils der Projektdurchführung „Nicht-Techniker" beteiligt sind. Simplizität ist deshalb für eine möglichst hohe Akzeptanz und auch Effizienz des Vorgehensmodells von großem Vorteil.[40]

4.2.6 Klarheit

Der Grundsatz der Klarheit bezieht sich auf „[...] die Verständlichkeit und die Eindeutigkeit von Modellsystemen".[41] Das Vorgehensmodell soll klar und eindeutig formuliert sein, um unterschiedliche Interpretationen und eventuell zeitaufwändige Diskussionen unter den Projektverantwortlichen zu vermeiden. Die Anforderung ist wichtig für Anwendbarkeit und Akzeptanz des Vorgehensmodells, denn alle Modelle, insbesondere Vorgehensmodelle, dienen zur Veranschaulichung und als Diskussionsgrundlage, so dass Zweideutigkeiten nicht nur zu Missverständnissen führen, sondern auch die Qualität des Entwicklungsprozesses und der resultierenden Ontologien negativ beeinflussen können. Aus diesem Grund ist die Klarheit auch abhängig von der Konsistenz, also der syntaktischen Korrektheit sowie der Widerspruchsfreiheit eines Modells. Das Kriterium der Klarheit kann realisiert werden, indem die Modellelemente möglichst formalspra-

---

39) Eine ähnliche Erläuterung geben MOODY (1994), S.102. Für die Messung der Komplexität eines Datenmodells schlagen sie vor, die Anzahl der Entitäten und die Menge der Beziehungen im jeweiligen Datenmodell zu addieren. Für Vorgehensmodelle könnten die Anzahl der Phasen und der Detailgrad der Phasenbeschreibungen als Maß für die Simplizität herangezogen werden.

40) Die geforderte Einfachheit bezieht sich somit vornehmlich auf die Reduktion der Komplexität eines Modells, beispielsweise sollte die *Nebenläufigkeit* nur wenn unbedingt notwendig in einem Modell berücksichtigt werden.

41) SCHÜTTE (1997), S. 10.

chig beschrieben werden[42], beispielsweise durch zusätzliche (semi-) formale Modellierungen wie Ereignisgesteuerte Prozessketten (EPKs).[43]

4.2.7 Werkzeugunterstützung

Das Vorgehensmodell soll für den beschriebenen Prozess und die dafür vorgeschlagenen Methoden entsprechende IT-Werkzeuge empfehlen, die die einzelnen Aktivitäten unterstützen und die Qualität der Ergebnisse verbessern können. Die Umsetzung dieser Anforderung trägt ebenfalls zu einer leichteren Anwendung des Vorgehensmodells und einer höheren Akzeptanz unter den Beteiligten bei, da der Komfort und auch die Effizienz des Entwicklungsprozesses durch geeignete Computerunterstützung bedeutend gesteigert werden können. Der Einsatz einer Software zur kollaborativen Textbearbeitung etwa vereinfacht das gemeinsame Arbeiten mehrerer Personen an einem Dokument.

Bei der Umsetzung der dargestellten Anforderungen ist zu beachten, dass diese nicht unabhängig voneinander bestehen, sondern sich gegenseitig in ihrer Wirkung beeinflussen. Einige der Kriterien verhalten sich komplementär zueinander; so führt zum Beispiel die (möglichst) vollständige Spezifikation eines Entwicklungsprozesses mit einer detaillierten Beschreibung der Phasenaktivitäten und Methoden in den meisten Fällen zu einer hohen Anwendungsbezogenheit. Auch die Einbeziehung von Möglichkeiten zur Prozessunterstützung durch geeignete Computerprogramme kann zu einer leichteren Realisierung und damit zur Anwendungsbezogenheit eines Vorgehensmodells beitragen.

Andererseits können Anforderungen auch gegensätzliche Zielsetzungen verfolgen, so dass eine gleichzeitige Umsetzung schwierig ist. Wenn die Klarheit des Modells durch eine formalsprachliche Beschreibung, wie zum Beispiel mit mathematischen Formeln, realisiert wird, kann sich dies nachteilig auf die Einfachheit und damit auf die Nach-

42) Eine besonders hoher Grad an Formalität geht allerdings oft zu Lasten der Nachvollziehbarkeit eines Vorgehensmodells. An dieser Stelle kann die Kombination informaler Ausführungen mit (semi-)formalen Beschreibungen wie Modellen dazu beitragen, dass gleichzeitig die Simplizität und Nachvollziehbarkeit sowie die Klarheit der Methodologie gewährleistet sind.

43) Zum Begriff der *Ereignisgesteuerten Prozessketten* vgl. SCHEER (1999).

vollziehbarkeit der Methodologie auswirken.[44] Oft lassen sich solche Zielkonflikte durch Hinzunahme von Hilfsmitteln auflösen oder mindern. Indem etwa verschiedene Abstraktionsniveaus sowohl eine generelle als auch eine anwendungsbezogene Sicht auf den Entwicklungsprozess ermöglichen oder für die Realisierung der Klarheit und Eindeutigkeit des Vorgehensmodells formale oder semi-formale Modelle, die zusätzlich zur textuellen Beschreibung erstellt, genutzt werden.

4.3 Ausgewählte Ansätze des Ontologies Engineerings

Es gab in den letzten Jahren zahlreiche Vorschläge für Vorgehensmodelle zur Konstruktion von Ontologien, die jedoch teilweise von sehr unterschiedlichen Zielsetzungen und Voraussetzungen ausgingen. Daher sollen an dieser Stelle nur diejenigen Ansätze vorgestellt werden, die für das Projekt KOWIEN, somit insbesondere für die Entwicklung von Kompetenz-Ontologien, von besonderer Relevanz sind.

Daneben konzentrieren sich viele existierende Ansätze in der Literatur bei der Ontologienentwicklung beispielsweise auf die Verknüpfung mehrerer bereits bestehender (Sub-)Ontologien, zum Beispiel SENSUS[45] oder auch KACTUS[46]. Das bedeutet, sie haben ihren Fokus nur auf eine Teilproblematik innerhalb der Ontologien-Entwicklung gelegt. Diese werden deshalb kurz der Vollständigkeit halber im anschließenden Kapitel vorgestellt, jedoch nicht einer Bewertung unterzogen.

Die IDEF5-Methode[47] wird ebenfalls keiner Bewertung unterzogen, da sie bislang (nach Wissen der Autoren) noch keine Anwendung in der Praxis erfahren hat, obwohl sie sich intensiv mit der Erstellung, Modifizierung und Wartung von Ontologien beschäftigt.

In den folgenden Abschnitten werden sechs der wichtigsten Vorgehensmodelle beschrieben, die wissenschaftlich anerkannt oder in der Praxis bewährt sind oder aufgrund

---

44) Hierüber lässt sich jedoch vortrefflich streiten, weil auch der Standpunkt vertreten werden kann, dass einem geübten Anwender eine formalsprachige Beschreibung die Nachvollziehbarkeit gerade erleichtert.

45) Vgl. SWARTOUT (1997). Diese Methodologie vereinigt u.a. die Ontologien Penman Upper Model, ONTOS sowie Wordnet, um aus einer breiten, allgemeinen SENSUS-Ontologie domänenspezifische Ontologien abzuleiten. Siehe hierzu auch Kapitel 4.6.1.

46) Vgl. zum Beispiel SCHREIBER (1995). Die Autoren von KACTUS, das im Rahmen des ESPRIT-Projekts entwickelt wurde, gehen von einer bereits bestehenden Wissensbasis aus, aus der durch Abstraktion eine generelle Ontologie entwickelt wird.

47) Vgl. KBSI (1994), S.25-60.

ihrer Zielsetzung und der vorgeschlagenen Methoden besondere Bedeutung für die Konstruktion von Kompetenz-Ontologien haben.

Die Auswahl von Ansätzen wurde also hinsichtlich ihrer Relevanz für das später zu entwickelnde generische Vorgehensmodell getroffen. Da die vorzustellenden Ansätze oft sowohl zeitlich als auch inhaltlich aufeinander aufbauen, werden die Ansätze weitgehend in der chronologischen Reihenfolge ihrer Veröffentlichung dargestellt.[48]

- Enterprise-Model-Ansatz
- TOVE-Methodologie
- METHONTOLOGY
- On-To-Knowledge-Methodologie
- Kollaborativer Ansatz
- IDEF5 Ontology Development Process

4.3.1 Enterprise-Model-Ansatz

USCHOLD und KING[49] erkennen als erste den Mangel an strukturierten Vorgehensweisen bei der Ontologienentwicklung. Basierend auf den Erfahrungen, die sie bei der Entwicklung der „Enterprise Ontology“ (Ontologien für Unternehmensmodellierungsprozesse[50]) gemacht haben, gehen USCHOLD und KING auf einige grundsätzliche Aspekte ein, die bei der Konstruktion von Ontologien beachtet werden sollen. So wird ein Grundgerüst für ein Vorgehensmodell vorgeschlagen, in dem die wichtigsten Schritte der Ontologienentwicklung aufgeführt sind und das als Ausgangspunkt für ein detaillierteres Vorgehensmodell dienen soll. Ihr Ansatz definiert die wichtigsten Schritte in Form einer Prozesskette. Nach der Bestimmung eines eindeutigen Einsatzzwecks werden die wich-

48) Die ersten beiden beschriebenen Vorgehensmodelle wurden etwa zeitgleich veröffentlicht; die „Enterprise Ontology“-Methodologie wird hier vorangestellt, da sie einen sehr grundlegenden Ansatz bildet. Die IDEF5-Methode wird an den Schluss gesetzt, weil sie, wie bereits erwähnt wurde, keiner Bewertung unterzogen wird.

49) Vgl. USCHOLD (1995), USCHOLD (1996a), USCHOLD (1996b).

50) Vgl. USCHOLD (1996c), S. 56. Für die Entwicklung eines „Enterprise Models“ werden unternehmensweite Daten-, Prozess- und Verhaltensmodelle zu einem konsistenten Modellsystem integriert, um ein Unternehmensabbild zu bilden.

tigsten Begriffe gemeinsam vereinbart, in eine formale Sprache überführt und durch die Integration von bereits bestehenden Ontologien erweitert. Nach mehrfacher Revision und kritischer Beleuchtung werden die Ontologien letztlich in ihre abschließende Form überführt und ausführlich dokumentiert.

Zusätzlich zu den Vorgehensschritten soll ein umfassendes Vorgehensmodell auch eine Reihe von Techniken, Prinzipien und Richtlinien für jede Phase sowie eine Beschreibung der Beziehungen zwischen den Aktivitäten[51] umfassen.

Die einzelnen Phasen des Basis-Vorgehensmodells werden im Folgenden kurz erläutert.[52]

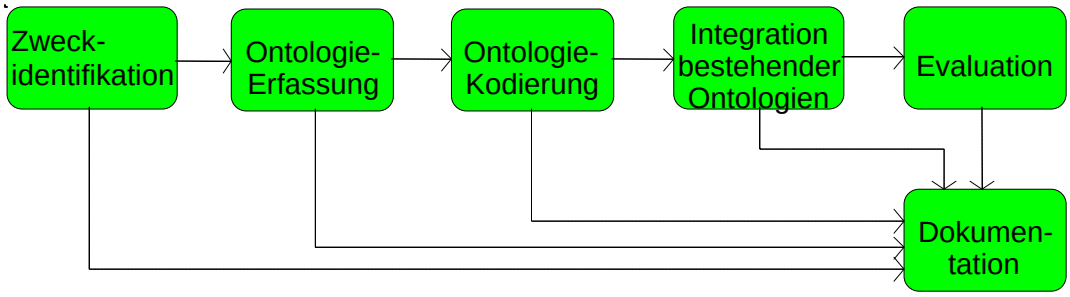

Abbildung 8: Vorgehensmodell von USCHOLD und KING

- **Zweckidentifikation:** Zu Beginn des Entwicklungsprozesses soll zunächst festgelegt werden, aus welchem Grund die Ontologien benötigt werden und welche Anforderungen sie erfüllen sollen. So besteht oftmals das oberste Ziel in der Wiederverwendung von Wissen. Aber auch die Art und Anzahl der Softwaresysteme, in denen die Ontologien eingesetzt werden sollen, sind wichtige Anforderungen, die vor der Konstruktion der Ontologien zu entscheiden sind. Eine weitere Aufgabe vor der eigentlichen Entwicklung ist die Identifizierung der Endanwender der Ontologien, ihrer Eigenschaften, Wünsche und Erwartungen.

51) Vgl. USCHOLD (1995), S. 2.

52) Ein gestrichelter Pfeil bedeutet hier: *„liefert Input für“*. Die grünen Felder bedeuten jeweils einen Prozessschritt in der Darstellung.

- **Ontologienentwicklung:** Diese Phase wird weiter unterteilt in die Schritte Ontologienerfassung, Kodierung und Integration. Die *Erfassung* umfasst die Identifikation der relevanten Konzepte und ihrer Beziehungen, die Darstellung der Konzepte durch präzise und eindeutige Textdefinitionen und die Festlegung von Begriffen für diese Konzepte und ihrer Beziehungen (Konzeptualisierung). Im Rahmen der *Kodierung* wird nach einer expliziten Repräsentation der Konzeptualisierung durch eine formale Sprache gesucht. Dafür schlagen die Autoren zunächst die Definition einer *Meta-Ontologie* vor, die die Basisausdrücke für die Ontologienspezifikation enthält, um anschließend eine Repräsentationssprache auszuwählen und damit den Kode zu generieren. Während dieser beiden vorangegangenen Aktivitäten stellt sich außerdem die Frage, ob und wie bereits existierende Ontologien integriert werden können. Für eine *Integration* bestehender Ontologien ist es wichtig, alle jeweils gemachten Annahmen explizit zu formulieren und eine Übereinstimmung zwischen den verschiedenen Benutzergruppen zu erzielen.
- **Evaluation:** Hierfür empfehlen die Autoren die Adaption von Methoden des Wissensmanagements. Voraussetzung für die Bewertung von Ontologien ist auch, dass zuvor ein Referenzrahmen festgelegt wurde, der als Grundlage für eine Überprüfung der entwickelten Ontologien dient, etwa eine Anforderungsspezifikation oder Kompetenzfragen (siehe Kapitel 4.3.2, Seite 37 f.).
- **Dokumentation:** Um eine effektive gemeinsame Nutzung und Wiederverwendung der Ontologien zu ermöglichen, sollen alle wichtigen Annahmen dokumentiert werden. Dazu zählen sowohl die Annahmen zu den Hauptkonzepten der Ontologien als auch die Annahmen für die Formulierung der Definitionen, also die Meta-Ontologie.

4.3.2 TOVE-Methodologie

Ziel des Projekts TOVE (Toronto Virtual Enterprise), das u.a. von GRÜNINGER und FOX durchgeführt wurde, war die Entwicklung eines „common sense“ Unternehmensmodells, das auf Ontologien basieren sollte.[53] Die Autoren sprechen gar vom „Knowledge

53) Vgl. GRÜNINGER (1995), S. 1.

Engineering zweiter Generation“ im Hinblick auf das zu entwickelnde ontologienbasierte Unternehmensmodell.

Aus den während des Projekts gewonnenen Erkenntnissen wurde anschließend ein umfassendes Vorgehensmodell formuliert, das Richtlinien für die Konstruktion von Ontologien sowie für die Evaluation ihrer Angemessenheit vorsieht. Kern des Vorgehensmodells ist die Erstellung der folgenden sechs Artefakte:

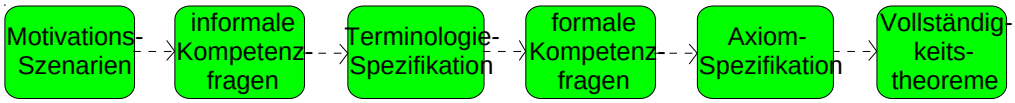

Abbildung 9: Vorgehensmodell von GRÜNINGER und FOX

- **Motivations-Szenarien:** Den Ausgangspunkt für die Ontologienentwicklung bilden die unterschiedlichen Probleme, die z.B. von Unternehmen identifiziert werden und durch Ontologien gelöst werden sollen. Sie werden in Form von Motivations-Szenarien beschrieben, wobei zusätzlich auch erste intuitive Lösungsansätze zu präsentieren sind. Jeder Vorschlag für neue Ontologien oder die Erweiterung bestehender Ontologien soll ein solches Motivations-Szenario beinhalten, um die Begründbarkeit für die Objekte der Ontologien zu gewährleisten.
- **Informale Kompetenzfragen:** Aus den zuvor erfassten Szenarien wird eine Reihe von Kompetenzfragen abgeleitet. Diese Kompetenzfragen müssen die Ontologien später beantworten können. Sie sind also auch für die Evaluation der Ontologie von Bedeutung. Die Autoren empfehlen eine hierarchische Strukturierung der informalen Kompetenzfragen und schlagen als erste Gliederung die Unterteilung in eine „Aktivitäten-Ontologie“ (Fragen zu den Aktivitäten in der Organisation und zur Planung der Aktivitäten) und eine „Organisations-Ontologie“ (Fragen bezüglich möglicher Einschränkungen dieser Aktivitäten) vor.
- **Terminologie-Spezifikation:** In dieser Phase wird die Terminologie der Ontologien definiert, indem für jede Kompetenzfrage alle zu ihrer Beantwortung notwendigen Objekte, Objektattribute und Objektbeziehungen identifiziert und spezifiziert werden. Dafür sollen die Beteiligten zunächst alle relevanten Konzepte oder Objekte

festlegen und diese dann als Konstanten und Variablen in einer logisch-formalen Sprache (etwa KIF[54]) darstellen. Die Attribute der Objekte werden anschließend in Form einstelliger Prädikate, die Relationen zwischen den Objekten durch mehrstellige Prädikate repräsentiert.

- **Formale Kompetenzfragen:** Die zuvor nur informal formulierten Kompetenzfragen werden an dieser Stelle in eine formale Darstellung, also in die für die Ontologienspezifikation gewählte Sprache, transformiert. Die Spezifizierung von formalen Kompetenzfragen ist eine notwendige Voraussetzung für die spätere Evaluation der Ontologien. Außerdem schränken sie die Menge der Axiome ein, die später in die Ontologien aufzunehmen sind.
- **Axiom-Spezifikation:** Um die Definition der Begriffe der Ontologien und die Bedingungen ihrer Interpretation festzulegen, wird eine Menge von Axiomen definiert. Sie spezifizieren somit die Semantik der Begriffe und werden formalsprachlich formuliert unter Anwendung der Prädikate der Ontologien. Sie müssen notwendig und hinreichend sein für die Formulierung der formalen Kompetenzfragen und ihrer Antworten. Wenn die vorgeschlagenen Axiome in dieser Hinsicht unzureichend sind, müssen in einem iterativen Prozess zusätzliche Axiome in die Ontologien aufgenommen werden.
- **Vollständigkeitstheoreme:** Für die Evaluation der Ontologien werden zunächst die Bedingungen spezifiziert, unter denen die Antworten auf die Kompetenzfragen vollständig sind. Mit Hilfe dieser ebenfalls formalsprachlich formulierten Vollständigkeitstheoreme können die Entwickler dann die Ontologien und die enthaltenen Axiome auf ihre Vollständigkeit in Bezug auf ihre Kompetenzfragen prüfen.

4.3.3 METHONTOLOGY

Der METHONTOLOGY-Ansatz, den FERNÁNDEZ, GÓMEZ-PÉREZ und JURISTO 1996 am Zentrum für Künstliche Intelligenz des Polytechnikums von Madrid erstmals veröffentlichten, sollte eine umfassende Hilfestellung für die Konstruktion von Ontologien

54) Das Akronym KIF steht für Knowledge Interchange Format, vgl. GENESERETH (1992). KIF wurde entwickelt, um den Austausch von Wissen zwischen verschiedenartigen Computerprogrammen zu erleichtern.

von Grund auf bieten („from the scratch“).[55] METHONTOLOGY orientiert sich stark an der IEEE Norm 1074-1995.[56] Von GÓMEZ-PÉREZ[57] und FERNÁNDEZ ET AL.[58] wird der Ontologienentwicklung vor ihrem Ansatz eher das Prädikat eines Handwerks als das einer wissenschaftlichen Disziplin zugewiesen. Jedes Entwicklungsteam verwende nämlich zuerst seine eigenen Prinzipien, Entwurfskriterien und –phasen. Der dadurch entstehende Mangel an strukturierten, vereinheitlichten Vorgehensweisen beschränkt die Zusammenarbeit zwischen verschiedenen Entwicklungsteams, woraus die Notwendigkeit eines „expliziten und vollständig dokumentierten Konzeptualisierungsmodells“[59] abgeleitet wird. Dazu präsentieren die Autoren eine Reihe von Aktivitäten, die Bestandteile des Ontologienentwicklungsprozesses sein sollen, sowie die Darstellung eines prototyp-basierten Lebenszyklus für Ontologien. Teil dieses Rahmenwerks ist auch das METHONTOLOGY-Vorgehensmodell - eine detaillierte Beschreibung der Entwicklungsaktivitäten, wie sie durchzuführen sind, welche Techniken sich dafür eignen und welche Artefakte am Ende jeder Phase produziert werden. Dabei wird unterschieden zwischen den „technischen“ Aktivitäten der Ontologienentwicklung (die im oberen Teil der folgenden Abbildung aufgeführten Phasen Anforderungs-Spezifizierung, Konzeptualisierung und Implementierung) sowie den zusätzlich auszuführenden Unterstützungsaktivitäten (Wissensakquisition, Integration, Evaluation und Dokumentation).[60]

55) Vgl. FERNÁNDEZ (1997), S. 1.

56) Vgl. IEEE (1996).

57) Vgl. GÓMEZ-PÉREZ (1996), S. 41.

58) Vgl. FERNÁNDEZ (1999), S. 37.

59) FERNANDEZ (1999), S. 37.

60) Vgl. GOMEZ-PEREZ (2001), S. 9.

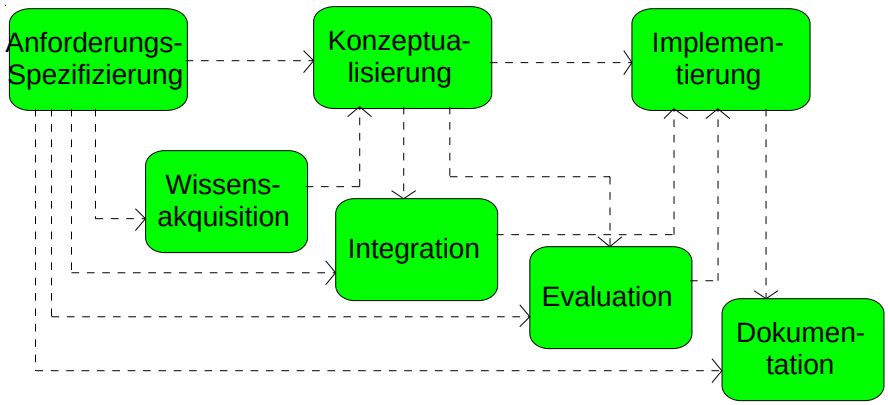

Abbildung 10: Vorgehensmodell von FERNÁNDEZ ET AL.

- **Anforderungsspezifizierung:** Ziel dieser Phase ist die Produktion eines natürlichsprachlich formulierten Ontologienspezifikationsdokuments, das die folgenden Informationen enthält: die intendierten Anwendungen und Benutzer der Ontologien, den Formalitätsgrad, in dem die Ontologien implementiert werden soll, sowie den Umfang und die Granularität der Begriffsrepräsentationen. Mehrere Alternativen für das Vorgehen bei der Spezifikation sind möglich, z.B. Kompetenzfragen oder auch natürlichsprachige Texte.
- **Wissensakquisition:** Die Wissensidentifikation und -erfassung wird zumindest teilweise parallel zur Spezifikation vorgenommen. In diesem Ansatz wird hierfür keine bestimmte Vorgehensweise vorgeschrieben. Statt dessen stellen die Autoren eine Reihe von möglichen Techniken vor, wie etwa Experteninterviews und Textanalysen. Anschließend sollen die relevanten Wissensquellen aufgelistet und die angewandten Erhebungstechniken beschrieben werden.
- **Konzeptualisierung:** Um die Problemstellung und ihre Lösung durch ein konzeptuelles Modell strukturiert darstellen zu können, wird zunächst eine vollständige Begriffssammlung erstellt, die die Konzepte, Instanzen, Verben und Eigenschaften der Domäne umfasst. Daraufhin werden die Begriffe in die beiden Kategorien *Konzepte* und *Verben* klassifiziert; diese werden dann in Baumhierarchien weiter verfeinert, um andere verwandte Konzepte oder Verben zu identifizieren. Zur Darstellung der Begriffe empfehlen die Autoren an dieser Stelle informale Repräsentationsarten wie Verzeichnislisten und Tabellen.

- **Integration:** Da eines der wichtigsten Ziele der Ontologienentwicklung die Wiederverwendung von Wissen ist, sollen die Verantwortlichen bereits existierende Ontologien auf ihre Möglichkeiten zur Integration überprüfen. Durch die Untersuchung von Meta-Ontologien und die Analyse von Ontologien hinsichtlich ihrer Kohärenz zu der im eigenen Projekt erarbeiteten Konzeptualisierung kann beurteilt werden, ob und wie andere Ontologien integriert werden können. Als Resultat dieser Phase sieht METHONTOLOGY ein Integrationsdokument vor, das alle eventuell benutzten Meta-Ontologien, Ontologien und die daraus verwendeten Begriffe zusammenfasst.
- **Implementierung:** In diesem Schritt soll die informale Darstellung der Ontologien in eine formale Repräsentation transformiert werden. Die Wahl der Sprache ist dabei in diesem Ansatz offen. Es wird jedoch die Nutzung einer Entwicklungsumgebung für die Kodierung der Ontologien empfohlen, um die Fehlerwahrscheinlichkeit zu verringern und die Implementierung zu erleichtern.
- **Evaluierung:** Vor ihrer Anwendung sollen die Ontologien anhand eines Referenzrahmens, in diesem Falle der Anforderungsspezifikation, untersucht und bewertet werden. Dazu zählt einerseits die Verifikation, also die Überprüfung der Korrektheit der Ontologien sowie ihrer Software-Umgebung und ihrer Dokumentation, andererseits die Validation, das heißt die Eignung der Ontologien im Hinblick auf ihren Zweck und die gestellten Anforderungen.
- **Dokumentation:** Während des gesamten Ontologienentwicklungsprozesses sollen die Beteiligten alle wichtigen Annahmen, Entscheidungen und Ergebnisse dokumentieren, damit die gemeinsame Nutzung und auch die Wiederverwendung der Ontologien erleichtert werden. Nach dem Vorgehensmodell dieses Ansatzes sollen die in jeder Phase entstehenden Beschreibungen und Schriftstücke (Anforderungsspezifikation, Wissensakquisitionsdokument, konzeptuelles Modell sowie Formalisierungs-, Integrations- und Implementierungsdokument) eine umfassende und durchgehende Dokumentation gewährleisten.

4.3.4 On-To-Knowledge-Methodologie

Die On-To-Knowledge-Methodologie ist Bestandteil eines Konzepts zur Entwicklung eines ontologienbasierten Wissensmanagementsystems, das erstmals 2000 von SCHNURR, STUDER, SURE und AKKERMANNS vorgestellt wurde[61] und mittlerweile im deutschsprachigen Raum eine große Bekanntheit erlangt hat. In diesem Ansatz wird zwischen zwei Arten von Prozessen unterschieden. Einerseits wird in den Prozess der Einführung und Instandhaltung von Wissensmanagementsystemen (*Wissens-Metaprozess*) unterschieden, andererseits wird der Prozess der Generierung, Erfassung und Nutzung des Wissens, der als *Wissensprozess* bezeichnet wird, unterschieden. Der Wissens-Metaprozess umfasst gleichzeitig auch ein Vorgehensmodell zur Konstruktion von Ontologien und verfolgt dabei das Ziel, durch diese Einbettung der Ontologienentwicklung in die übergeordnete Wissensmanagementsystementwicklung die Anwendungsorientierung des Systems zu fördern. Das im Folgenden dargestellte Vorgehensmodell beschreibt daher den gesamten Wissens-Metaprozess, der an die Vorgehensweise des „CommonKADS“-Ansatzes[62] angelehnt ist.

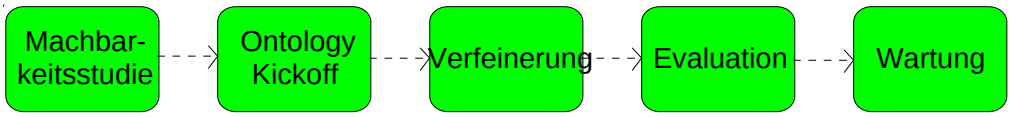

Abbildung 11: Vorgehensmodell von SCHNURR ET AL.

- **Machbarkeitsstudie:** Bevor die Mitglieder des Projektteams mit dem „eigentlichen“ Entwicklungsprozess beginnen, soll eine Machbarkeitsstudie durchgeführt werden, um die organisatorischen Rahmenbedingungen der Wissensmanagementapplikation zu erfassen. Hauptbestandteil dieser Studie sind eine Markt- und Wettbewerbsanalyse sowie eine Untersuchung der unternehmensinternen Voraussetzungen. Hierbei sollen einerseits Chancen und Risiken aufgedeckt und dabei mögliche Lö-

61) Vgl. SCHNURR (2000), S. 24 ff. Ferner vgl. auch SURE (2002a) und LAU (2002).

62) Zu CommonKADS vgl. SCHREIBER (2001) und Kapitel 3.2.3.

sungsansätze skizziert werden. Andererseits sollen die identifizierten Faktoren in eine breitere organisatorische Perspektive eingebettet werden. Durch diese Informationen sollen Probleme und Möglichkeiten aufgezeigt und eine Entscheidung getroffen werden, inwiefern ein ontologienbasiertes Wissensmanagementsystem für die Lösung der betrachteten Probleme geeignet ist, d.h. sie soll als Entscheidungsgrundlage für die ökonomische und technische Machbarkeit des Projektes dienen.

- **Ontology Kickoff:** Wenn die Machbarkeitsstudie mit einer Entscheidung für das Projekt abgeschlossen wurde, müssen im nächsten Schritt die Anforderungen an die Ontologien erhoben werden. Dazu wird mit Hilfe strukturierter Interviews eine Anforderungsspezifikation erstellt, die das Ziel der Ontologien, die Anwendungsdomäne und den Umfang, die potenziellen Anwender und die Benutzeranforderungen beinhaltet. Es soll beschrieben werden,
    - welches Ziel, welcher Anwendungsbereich und welche Anwendungen von den Ontologien unterstützt werden sollen,
    - welche Wissensquellen (Domänenexperten, Organisationsdiagramme, Geschäftspläne, Wörterbücher, Indexlisten, Datenbank-Schemata etc.) vorliegen,
    - welche Benutzer auf das ontologienbasierte System zugreifen und
    - welche Fragen von dem System beantwortet werden sollen.

  Dieser Anforderungskatalog soll die Entwickler der Ontologien bei der hierarchischen Strukturierung der Begriffe unterstützen. Außerdem soll diese Phase eine Analyse der relevanten Wissensquellen, die Formulierung von Kompetenzfragen sowie eine erste informale (oder semi-formale) Beschreibung der Ontologien umfassen.
- **Verfeinerung:** In Zusammenarbeit mit Domänenexperten entwerfen die Ontologienentwickler eine Basis-Ontologie, die dann schrittweise erweitert und verfeinert wird. Dafür müssen alle relevanten Konzepte, Relationen zwischen den Konzepten und Regeln identifiziert und dargestellt werden und hinsichtlich ihrer Konsistenz und Vollständigkeit überprüft werden. Anschließend wählen die Entwickler eine Sprache zur formalen Repräsentation der Ontologien aus (die Autoren schlagen

Sprachen wie RDF oder DAML+OIL[63] vor) und transformieren die Elemente der Basis-Ontologie in formalsprachige Konzepte, Relationen und Axiome der „Ziel-Ontologie“. Es lassen sich somit die folgenden Teilphasen unterschieden:

- Erstellung einer ersten informalen Taxonomie, die alle relevanten Begriffe aus der Kickoff-Phase enthält,
- Akquisition von Wissen von Domänen-Experten mit dem Ziel der Erstellung einer ersten *Basis-Ontologie*, die relevante Konzepte, Relationen zwischen den Konzepten und darauf aufbauende Axiome enthält,
- Formalisierung der Basis-Ontologie in eine erste *Ziel-Ontologie*, die formalsprachig repräsentiert wird.

- **Evaluation:** In dieser Phase werden die entwickelten Ontologien im Hinblick auf ihren Nutzen und ihre Anwendbarkeit in der vorgesehenen Umgebung bewertet. Zunächst überprüfen die Entwickler, ob die Ziel-Ontologien alle Kriterien der Anforderungsspezifikation erfüllen und ob sie die festgelegten Kompetenzfragen „beantworten“ können. Darüber hinaus werden die Ontologien anhand eines Prototyps des Wissensmanagementsystems in ihrer späteren Anwendungsumgebung getestet, um durch Rückmeldungen der Tester und der Benutzer Fehler zu identifizieren. Falls sich durch die Evaluierung oder den Systemtest ein Revisionsbedarf ergibt, müssen die Fehler durch eine Rückkehr in die Verfeinerungsphase behoben werden.

  In der Evaluationsphase soll die Nützlichkeit der entwickelten Ontologien und der darauf aufbauenden Softwareumgebung bewertet werden. Zuerst sollen die Ziel-Ontologien hinsichtlich der Anforderungen überprüft werden. Dabei soll insbesondere analysiert werden, ob diese Ontologien alle Fragen des Kompetenz-Fragebogens unterstützen oder beantworten. In einem folgenden Schritt soll die Ontologie in der jeweiligen Anwendungsumgebung getestet werden und durch Feedback von Beta-Testern verbessert werden.

  Die Evaluationsphase ist eng mit der Verfeinerungsphase gekoppelt. Bei der Entwicklung der Ontologien müssen solange Zyklen durchlaufen werden, bis die Ziel-Ontologien den gewünschten Detaillierungsgrad erreicht haben.

---

63) Aktuelle Informationen zu den Sprachen DAML+OIL (DARPA Annotated Markup Language + Ontology Inference Layer) und RDF (Resource Description Framework) sind unter den URLs http://www.w3.org/TR/daml+oil-walkthru/ bzw. http://www.w3.org/RDF/ (Zugriff am 10.11.2002) zu finden.

- **Wartung:** SCHNURR ET AL.[64] empfehlen eine regelmäßige Erweiterung und Anpassung der Ontologien, um Änderungen in der Umgebung Rechnung zu tragen. Da die Instandhaltung von Ontologien nach Meinung der Autoren in erster Linie ein organisationaler Prozess ist, sollten Regeln für die Aktualisierung formuliert und die Verantwortlichkeiten zur Durchführung der Modifikationen festgelegt werden (bspw. neue Versionsstände der Ontologien freigeben). Für größere Änderungen wie z.B. Umstrukturierungen ist oft ein erneuter Durchlauf der Verfeinerungs- und der Evaluierungsphase erforderlich. Analog der ersten Verfeinerungsphase soll durch Feedback von den Benutzern wichtige Hinweise für erforderliche Änderungen an den Ziel-Ontologien liefern. Nach Inbetriebnahme der ontologienbasierten Wissensmanagementsystems müssen nicht nur die Ontologien, sondern auch das darauf aufbauende System instand gehalten werden. Sowohl Modifikationen der Ontologien, als auch Änderungen am Layout der Anwendung sollen hierbei im organisatorischen Kontext berücksichtigt und strukturiert werden.

4.3.5 Kollaborativer Ansatz

HOLSAPPLE und JOSHI verfolgen in ihrem Ansatz zur Ontologienentwicklung das Ziel, den Aspekt der Bindung der unterschiedlichen Beteiligten an die Ontologien möglichst früh in den Entwicklungsprozess zu integrieren, um so die Akzeptanz und die gemeinsame Nutzung der Ontologien zu fördern.[65] Die Autoren unterscheiden fünf verschiedene Ansätze mit unterschiedlichen Ausgangspunkten für das Design von Ontologien[66]: *Inspiration* (ausgehend von einer individuellen Sicht auf die Domäne), *Induktion* (mit einem konkreten Fall als Grundlage für eine Verallgemeinerung), *Deduktion* (basierend auf generellen Prinzipien, die angepasst und erweitert werden), *Synthese* (die Zusammenfassung mehrerer bestehender Ontologien zu einer umfassenden) und *Kollaboration* (Einbringung der Ansichten und Erfahrungen mehrerer Personen).[67] Jede dieser Vorge-

64) Vgl. SCHNURR (2000).

65) Vgl. HOLSAPPLE (2002), S. 43.

66) Vgl. HOLSAPPLE (2002), S. 44; diese fünf Klassen von Ansätzen können auch miteinander kombiniert werden.

67) Diese Kategorien können auch für die Klassifizierung von Vorgehensmodellen verwendet werden. So baut TOVE beispielsweise auf der *Inspiration* auf. Das Konzept der *Kollaboration* ist in den meisten Ansätzen zu finden. Auf einer Synthese basierende Vorgehensmodelle werden in dieser Arbeit nicht betrachtet, weil im Rahmen des KOWIEN-Projekts noch keine Ontologien für Kompetenzmanagementsysteme vorliegen.

hensweisen hat Vor- und Nachteile und ist jeweils für eine bestimmte Situation besser geeignet als eine andere. HOLSAPPLE und JOSHI haben sich für die Durchführung einer Fallstudie für den kollaborativen Ansatz entschieden, da durch die Beteiligung einer größeren Menge von Personen die Chancen für die spätere Akzeptanz der Ontologien steigen und auch die Risiken einer lückenhaften Ontologienspezifikation verringert werden können.

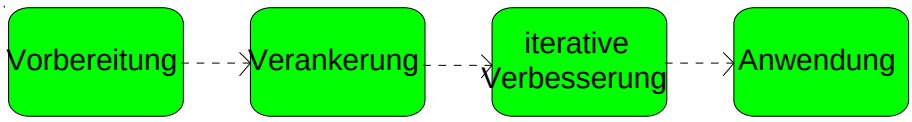

Abbildung 12: Vorgehensmodell von HOLSAPPLE und JOSHI

- **Vorbereitung:** In dieser ersten Phase werden Design-Kriterien wie z.B. Verständlichkeit, Korrektheit, Klarheit oder Nutzen definiert, die einerseits als Richtlinien während der Entwicklung und andererseits für die Bewertung der Qualität der Ontologien im Nachhinein dienen sollen. Anschließend werden Grenzbedingungen festgelegt, um die Anwendung der Design-Kriterien auf bestimmte Bereiche einzuschränken. Die vorgestellte Fallstudie konzentriert sich auf betriebliches Wissensmanagement in Unternehmen, auf die Beschreibung von Wissensmanagement-Phänomenen (deskriptiver Gestaltungsanspruch) und auf ein „Top-Down"-Vorgehen. Außerdem sollen in der Vorbereitungsphase Evaluationsstandards bestimmt werden, beispielsweise existierende Ontologien, bewährte Wissensmanagementkonzepte und Forschungsergebnisse, zu denen der Entwicklungsprozess und die resultierenden Ontologien konsistent sein müssen.
- **Verankerung:** Jeder der angesprochenen fünf Entwicklungsansätze kann für die Formulierung einer Basis-Ontologie angewendet werden, die einen „Anker" zur Orientierung bei der eigentlichen Entwicklung darstellt. Die Autoren entschieden sich für eine Synthese, indem sie die Sprachen, Konzepte und Beziehungen aus den Evaluationsstandards konsolidierten und neu organisierten und dann in mehreren Iterationen die „Anker-Ontologie" durch einen Abgleich mit den Design-Kriterien weiterentwickelten (siehe auch *iterative Verbesserung)*.

- **Iterative Verbesserung:** In dieser Phase wird eine Adaption der Delphi-Methode[68] eingesetzt, um die Basis-Ontologie schrittweise zu verfeinern und zu vervollständigen. Zunächst werden dafür bestimmte Teilnehmer (vor allem die Endanwender) identifiziert und in Ausschüssen gruppiert. Die Ontologienentwickler senden die Basis-Ontologie an die Mitglieder der Ausschüsse, damit diese den ersten Entwurf kritisieren und kommentieren. Nachdem die Bewertungen der Ontologien zurückgeschickt worden sind, findet eine Revision statt, um die Kritik und Verbesserungsvorschläge in die Ontologien einzuarbeiten, und anschließend wird die modifizierte Version erneut an die Teilnehmer gesendet. Dieser Prozess wird iterativ so lange durchgeführt, bis die Ontologien von allen Beteiligten übereinstimmend als vollständig und den Anforderungen (also den Design-Kriterien) entsprechend angesehen werden. Um die verschiedenen Kommentare und Vorschläge zu verwalten und umzusetzen, teilen die Entwickler die Antworten in Kategorien ein (z. B.: begründet und häufig, gelegentlich, selten und zufällig) und fertigen ein Dokument zur Zusammenfassung der Kritik an, die Modifikationen der nächsten Iteration sollen sich hieran orientieren.
- **Anwendung:** Abschließend werden die Qualität und der Nutzen der Ontologien durch ihre Anwendung in konkreten Projekten überprüft. Die Autoren setzten ihre Ontologien z.B. als Rahmenwerk bei der Untersuchung von Wissensselektionsprozessen und Wissensselektionstechnologien und bei der Generierung eines „Wissenskettenmodells“ ein.

4.3.6 IDEF5 Ontology Development Process

IDEF5 wurde von KBSI (Knowledge Based Systems Inc.) entwickelt. Die IDEF5-Methode wurde speziell entworfen um die Entwicklung, die Modifizierung und die Instandhaltung von Ontologien zu unterstützen. IDEF steht hierbei als Akronym für **I**ntegrated Computer Aided Manufacturing **Def**inition und ist Teil einer Familie von Definitionen.

68) Vgl. LINSTONE (1975). Die Delphi-Methode ist eine strukturierte Vorgehensweise für die Befragung von Experten und die Sammlung und die Integration der verschiedenen Sichten auf eine bestimmte Problemstellung.

IDEF5 ist eine “ontology (description) capture method”[69], die einen strukturierten Ansatz zur Entwicklung von Domänen-Ontologien beinhaltet. Die Methodologie beinhaltet ein generelles Vorgehen mit folgenden Richtlinien:[70]

- **Organizing and Scoping.** The organizing and scoping activity establishes the purpose, viewpoint, and context for the ontology development project, and assigns roles to the team members.
- **Data Collection.** During data collection, raw data needed for ontology development is acquired.
- **Data Analysis.** Data analysis involves analyzing the data to facilitate ontology extraction.
- **Initial Ontology Development.** The initial ontology development activity develops a preliminary ontology from the data gathered.
- **Ontology Refinement and Validation.** The ontology is refined and validated the ontology to complete the development process.

Des Weiteren beinhaltet IDEF5 Ontologien-Sprachen zur Unterstützung des Entwicklungsprozesses. Zum einen eine *schematic language*, die speziell als graphische Sprache (vergleichbar etwa mit UML) entwickelt wurde, um übersichtlich und nachvollziehbar das Wissen einer Domäne abbilden zu können. Der durchschnittliche Benutzer kann somit die erstellten Ontologien leicht nachvollziehen. Zum anderen gibt es eine Ausarbeitungssprache (*elaboration language)*, diese stellt eine strukturierte textuelle Sprache dar, die es erlaubt, detailliert Elemente der Ontologien zu charakterisieren.

Die IDEF5 Methode enthält im zitierten Report-Anhang eine strukturierte Bibliothek von üblicherweise benutzten Relationen, wie beispielsweise zeitliche Relationen, Abhängigkeitsrelationen und Einflussrelationen, auf die zweckmäßiger Weise zurückgegriffen werden kann, um beispielsweise eine neue Relationen zu klassifizieren.

---

69) KBSI (1994), S. 1.

70) Http://www.idef.com/idef5.html. Zugriff am 4.9.2002. Auf eine Übersetzung wird hier verzichtet um nicht die Aussagen zu verfälschen.

Wie bereits erwähnt, findet die Methodologie keinen Eingang in die Bewertung, weil von Seiten der Autoren keine Anwendung der Methodologie ermittelt werden konnte. Hieraus folgend wird auf eine Abbildung verzichtet.

4.4 Bewertung der Ansätze

Die im vorigen Unterkapitel dargestellten Vorgehensmodelle setzen verschiedene Schwerpunkte und lassen sich für einzelne Entwicklungsprojekte unterschiedlich gut einsetzen. Um eine Bewertung der Ansätze hinsichtlich ihrer Eignung für die Konstruktion von Kompetenz-Ontologien wie im KOWIEN-Projekt vornehmen zu können, werden im Folgenden die in Kapitel 4.2 aufgestellten Anforderungen zur Evaluation herangezogen. Für jedes Kriterium wird untersucht, ob und inwieweit es in den einzelnen Ansätzen erfüllt wird. Anschließend werden die Ergebnisse dieser Analyse, also der Grad der Erfüllung der jeweiligen Anforderung durch die verschiedenen Methodologien, in einer Tabelle am Ende dieses Kapitels dargestellt.

4.4.1 Generizität

Dem Kriterium der Generizität wird in dem von USCHOLD und KING vorgestellten Ansatz besondere Aufmerksamkeit geschenkt. Bevor sie detaillierter auf ihre eigenen Erfahrungen mit der Ontologienentwicklung eingehen, beschreiben die Autoren ein allgemeines, grob formuliertes Vorgehensmodell[71], das für ein sehr breites Spektrum von Projekten zur Ontologienentwicklung angewendet werden kann.

Die TOVE-Methodologie von GRÜNINGER und FOX ist weniger generisch konzipiert. Zwar lässt sich auch dieses Modell auf viele Ontologienentwicklungsvorhaben übertragen, die Art der Durchführung ist aber zum Teil schon durch die Strukturierung der Phasen vorgegeben. So werden zum Beispiel bei TOVE die Spezifikation der Anforderungen durch Kompetenzfragen und die Implementierung der Ontologien durch Formulierung in Prädikatenlogik der ersten Stufe realisiert.

Die Autoren von METHONTOLOGY dagegen lassen sowohl die Wahl der formalen Sprache als auch die Art der Anforderungsspezifikation offen. Sogar die Reihenfolge der Aktivitäten, deren Durchführung sie empfehlen, kann für jedes Projekt neu entschieden werden. Die Generizität des Ansatzes wird besonders deutlich durch die all-

71) Vgl. USCHOLD (1995), S. 2. Dieses Modell bildet jedoch nur ein Grundgerüst und sollte nach Aussage der Autoren durch entsprechende Erweiterungen sowie durch die Festlegung von Techniken und Methoden an konkrete Projekte angepasst werden.

gemeine Formulierung der Phasen und durch die große Anzahl an Alternativen, die für ihre Umsetzung vorgeschlagen werden.

Auch die On-To-Knowledge-Methodologie umfasst eine eher grobe Beschreibung der Aktivitäten, die durch Fallstudien instanziiert und detailliert werden. Das Vorgehensmodell ist auf andere Projekte übertragbar, jedoch nur auf solche, bei denen die Ontologien den Kern eines Wissensmanagementsystems bilden und in deren Rahmen die Ontologienentwicklung Teil der Wissens-managementsystementwicklung ist.

Bei der Arbeit von HOLSAPPLE und JOSHI liegt der Fokus ausschließlich auf der Ontologienentwicklung (unabhängig von ihrem Umfeld), und der vorgestellte kollaborative Ansatz erfüllt das Kriterium der Generizität in der Hinsicht, dass bei Projekten zur Ontologienentwicklung grundsätzlich mehrere Personen mit verschiedenen Ansichten beteiligt sind. Aber weil der Einsatz der Delphi-Methode in diesem Kontext einen hohen Aufwand und umfassende Kenntnisse der Beteiligten über die Domäne sowie über Ontologien erfordert, ist die Methode hauptsächlich für größere und komplexere Projekte anwendbar.

4.4.2 Anwendungsbezogenheit

Die meisten der vorgestellten Ansätze lassen sich als relativ detailliert und anwendungsnah einstufen.

Die „Enterprise Ontology“-Methodologie hingegen erhebt nur den Anspruch, ein Grundgerüst für das Vorgehen bei der Ontologienentwicklung bereitzustellen. Die einzelnen Phasen werden genannt und erläutert. Eine genaue Beschreibung und Hinweise zur Umsetzung werden jedoch nicht gegeben. Allerdings gilt dies nur für das Vorgehensmodell selbst, denn in ihrem Artikel von 1995 gehen die Autoren anschließend auf ihre Erfahrungen aus einer Fallstudie ein, wobei sie konkrete Methoden zur Umsetzung ihrer Vorgehensmodells darstellen.[72] Teilweise wird dadurch eine Erweiterung und Verfeinerung des Modells gegeben, doch für viele Vorgehensschritte nutzen USCHOLD und KING die Vorschläge und Erkenntnisse aus anderen Arbeiten und verweisen auf die jeweiligen Literaturquellen.

---

72) Vgl. USCHOLD (1995), S. 4 ff.

Bei TOVE ist stattdessen die Beschreibung der Implementierungsmöglichkeiten und der Aktivitäten Bestandteil des Vorgehensmodells selbst. So werden beispielsweise konkrete Empfehlungen bezüglich der Anforderungsspezifizierung (Kompetenzfragen) und der Wahl der formalen Sprache (Prädikatenlogik) gegeben und die Vorgehensschritte bei der Formalisierung relativ detailliert erläutert.

Auch der Ansatz METHONTOLOGY beinhaltet bereits einige Beschreibungen von Methoden für die Durchführung einer Ontologienentwicklung. Obwohl die Autoren den Anspruch erheben, einen hohen Detaillierungsgrad zu erreichen[73], sind die Phasenaktivitäten in den vorliegenden Quellen nur teilweise durch Realisierungshinweise konkretisiert. Dabei ist nicht festgelegt, auf welche Art und Weise das Vorgehensmodell umzusetzen ist, aber es werden für jede Phase mehrere Alternativen zur Realisierung der Aufgaben vorgestellt. So könnten die Projektverantwortlichen sich zum Beispiel bei der Wissensakquisition für strukturierte oder unstrukturierte Interviews, für Brainstorming, Textanalyse oder für computerunterstützte Wissenserhebung entscheiden.

Die On-To-Knowledge-Methodologie kann ebenfalls als anwendungsbezogen angesehen werden. Dabei konzentrieren sich die Autoren besonders darauf, die Ergebnisse der einzelnen Phasen, also etwa das Anforderungsspezifikationsdokument oder das Kompetenzfragen-Formular, genau zu beschreiben und ihren Aufbau durch Beispiel-Grafiken zu verdeutlichen.

Im Kollaborativen Ansatz sind die Phasen des Vorgehensmodells und die Möglichkeiten zu ihrer Umsetzung auch sehr detailliert dargestellt. Damit erfüllt die Arbeit in dieser Hinsicht die Anforderung der Anwendungsbezogenheit. Allerdings liegt der Schwerpunkt dieses Ansatzes auf der Verfeinerung von Ontologien durch mehrere beteiligte Personen, während die Erstellung und Formalisierung einer ersten Basis-Ontologie nicht ausreichend genau behandelt wird, um die Methodologie in einem konkreten Projekt ohne weitere methodische Fundierung anwenden zu können.

---

73) Vgl. FERNÁNDEZ (1999), S. 4-9.

4.4.3 Dokumentation

Obwohl in den meisten Fällen Übereinstimmung darin besteht, dass eine umfassende und stetige Dokumentation für Projekte jeder Art von essentieller Bedeutung ist und eine Selbstverständlichkeit sein sollte, erscheint es sinnvoll, diese Tätigkeit als Bestandteil eines Vorgehensmodells explizit aufzunehmen.

USCHOLD und KING etwa definieren das Dokumentieren und Begründen von Entscheidungen und Annahmen als eigene Phase ihres Ansatzes und stellen damit die Beachtung dieser wichtigen Aufgabe bei einer Anwendung der „Enterprise Ontology"-Methodologie sicher. Jedoch ist an dieser Stelle zu erwähnen, dass in den eigentlichen Entwicklungsphasen sowie bei der Darstellung der Fallstudie nicht mehr näher auf diese Tätigkeit eingegangen wird. In dem Vorgehensmodell von TOVE ist die Dokumentation der relevanten Entscheidungen und Ergebnisse eher implizit enthalten. So wird beispielsweise durch die Formulierung von Motivations-Szenarien und Kompetenzfragen das Resultat der vorangegangenen Aktivitäten festgehalten, die dabei vorhandenen personen- und situationsabhängigen Annahmen und Einschränkungen werden jedoch in diesem Ansatz nicht beachtet.

Die Autoren von METHONTOLOGY hingegen präsentieren die Aufgabe der Dokumentation einerseits als eigenständige Phase in ihrem Vorgehensmodell und gleichzeitig auch als Bestandteil jeder einzelnen Ontologienentwicklungsphase. Die Dokumentation wird dabei als entscheidende Aktivität während des gesamten Entwicklungsprozesses dargestellt und durch die verschiedenen Artefakte realisiert, die im Laufe des Projekts entstehen, wie zum Beispiel Anforderungsspezifikation, Wissensakquisitionsdokument, oder konzeptuelles Modell.

Im Gegensatz dazu wird dem Grundsatz der Dokumentation bei der Beschreibung der On-To-Knowledge-Methodologie wenig Aufmerksamkeit geschenkt. Er findet durch die Struktur des Vorgehensmodells keine Erwähnung, und auch in den wenigen während des Entwicklungsprozesses produzierten Artefakten werden Entscheidungen und ihre Begründungen allenfalls implizit dokumentiert.

Der Kollaborative Ansatz von HOLSAPPLE und JOSHI enthält die Dokumentation auch nicht als eigenen Bestandteil des Phasenmodells, doch in jedem Vorgehensschritt werden die wichtigsten Ergebnisse in Form von schriftlichen Ausführungen als Entschei-

dungsgrundlage an die nächsten Phasen weitergegeben. So soll etwa beim Einsatz der Expertenbefragung in der Verfeinerungsphase in jeder Iteration eine Zusammenfassung der eingegangenen Kritikpunkte und Kommentare angefertigt werden.

4.4.4 Einfachheit

Da die Einfachheit eines Vorgehensmodell zu einem großen Teil durch die Komplexität seines Aufbaus beeinflusst wird, kann ein so generelles und grob strukturiertes Grundgerüst wie etwa der „Enterprise Model"-Ansatz dieses Kriterium leichter erfüllen als eine umfassendere, ausführlich beschriebene Methodologie.

Das von USCHOLD und KING entwickelte Vorgehensmodell besteht aus nur vier Phasen, von denen die umfassendste (die Ontologienentwicklung selbst) durch die Unterteilung in drei Vorgehensschritte weiter detailliert wird. Auch die Phasenbezeichnungen und ihre Beschreibung der Aktivitäten sind wenig technisch, und für weitere Einzelheiten wird oft auf andere Quellen verwiesen, so dass das Modell relativ simpel gehalten und für die meisten Personen mit geringen Vorkenntnissen verständlich ist.

Beim Einsatz der TOVE-Methodologie dagegen ist spätestens bei der formalen Spezifikation der Kompetenzfragen und anschließend der Axiome und der Vollständigkeitstheoreme logisch-mathematisches Vorstellungsvermögen erforderlich, um die einzelnen Schritte sowie die Erläuterungen und Definitionen nachvollziehen zu können. Natürlich wird die tatsächliche Formalisierung meist von Personen mit Informatik- oder Mathematik-Kenntnissen vorgenommen, doch es könnte bei TOVE schon bei der Planung zu Verständnis- oder sogar Akzeptanzschwierigkeiten auf der Seite des Projektleiters und anderer Beteiligter mit nicht formalem Hintergrund kommen.

Die Phasenbeschreibungen, die bei METHONTOLOGY gegeben werden, und die Erläuterungen der in diesem Vorgehensmodell vorgeschlagenen Techniken sind nicht sehr komplex aufgebaut. Auf der anderen Seite kann die Struktur des Modells selbst dadurch unüberschaubar wirken, dass der Ansatz sehr umfassend konzipiert ist und eine große Anzahl von Phasen aufweist.

Die Menge der Vorgehensschritte, die in der On-To-Knowledge-Methodologie dargelegt werden, ist dagegen noch übersichtlich und verständlich. Dennoch erfüllt dieser Ansatz das Kriterium der Einfachheit nur teilweise, weil er viele verschiedene Einflüsse

aufnimmt (aus den Bereichen Software Engineering, Wissensmanagement und Ontologienentwicklung) und daher für die Realisierung der Aktivitäten sehr unterschiedliche Techniken vorgeschlagen werden.[74]

Die Methodologie des Kollaborativen Ansatzes ist sowohl in Hinsicht auf ihre äußere Struktur als auch bezüglich der Erläuterungen zu den einzelnen Phasen vergleichsweise simpel gestaltet. Da eventuelle „technische“ Details, die etwa die Formulierung und Darstellung der Basis-Ontologie oder auch später die formalsprachige Implementierung betreffen, außer acht gelassen werden, ist die Beschreibung der Phasen und ihrer Aktivitäten wenig komplex und durchaus leicht zu erfassen.

4.4.5 Klarheit

Die dargestellten Ansätze beinhalten kaum formale bzw. semi-formale Beschreibungen der jeweiligen Methodologie, dennoch sind die Modelle meist klar und verständlich formuliert und oft durch eine graphische Darstellung des Phasenablaufs verdeutlicht.

Das Vorgehensmodell des „Enterprise Model“-Ansatzes etwa besitzt eine klare Zielsetzung und eine übersichtliche Struktur, und auch die einzelnen Aktivitäten sind allgemein verständlich beschrieben, obwohl die Autoren für detailliertere Ausführungen oft auf andere Quellen verweisen.

Die Anforderung der Klarheit wird bei dem TOVE-Konzept von GRÜNINGER und FOX vor allem durch das sehr systematisch aufgebaute Phasenmodell umgesetzt, doch auch die Phasen selbst sind eindeutig formuliert. Besonders bei der Darstellung der Aktivitäten zur Formalisierung der Ontologien verwenden die Autoren zusätzlich formalsprachliche Notationen, um die Präzision ihres Ansatzes zu erhöhen.

Auch das METHONTOLOGY-Vorgehensmodell präsentiert ein strukturiertes Konzept zur Ontologienentwicklung. Das Grundgerüst dieses Ansatzes ist an den IEEE-Standard für die Entwicklung von Software-Lebenszyklus-Prozessen angelehnt[75] und wird durch

---

74) So wird beispielsweise für die Erhebung der Anforderungen die Erstellung eines Anforderungsspezifikationsdokuments (aus dem Bereich des Software Engineerings) sowie zusätzlich die Formulierung von Kompetenzfragen (ein eher ontologieentwicklungsspezifisches Konzept) empfohlen und auch beide Dokumente später als Referenzrahmen bei der Evaluation herangezogen.

75) Vgl. FERNÁNDEZ LOPÉZ (1999), S. 4-9.

eine detaillierte und klar formulierte Beschreibung der Methoden und Techniken erweitert.

Der Grundsatz der Klarheit wird in der On-To-Knowledge-Methodologie ebenfalls beachtet: Sie beinhaltet ein strukturiertes Phasenmodell mit festgelegtem Umfeld und Zielvorgabe. Allerdings ist die Zuordnung der Aktivitäten zu den Phasen (in verschiedenen Veröffentlichungen) mehrdeutig oder sogar inkonsistent, so wird zum Beispiel die Erstellung von Basis-Ontologien einmal der Phase „Kickoff“ und einmal der „Verfeinerung“ zugeordnet.

Die Vorgehensweise im Kollaborativen Ansatz ist ausschließlich informal, aber eindeutig und verständlich dargestellt und durch ein strukturiertes Phasenschema unterstützt. Dennoch können dadurch Unklarheiten entstehen, dass die Autoren auf einige Aktivitäten, die Bestandteil der Methodologie sind, wie z.B das Vorgehen bei der Entwicklung der Basis-Ontologien, gar nicht oder nur oberflächlich eingehen.

4.4.6 Werkzeugunterstützung

Da das Gebiet der Ontologienentwicklung für Informatik- und betriebswirtschaftliche Zwecke ein vergleichsweise junges Forschungsfeld ist, ist die Auswahl an Werkzeugen für die computerunterstützte Entwicklung von Ontologien begrenzt. Dennoch versuchen einige der Autoren der existierenden Ansätze, Hinweise und Empfehlungen bezüglich einer eventuell sinnvollen Software-Umgebung für die Umsetzung ihres Vorgehensmodells zu geben.

USCHOLD und KING etwa verweisen bei ihrem Enterprise Model Ansatz für die Kodierung der Ontologien und für die Dokumentation der Entwicklung auf die Nutzung von Ontolingua[76], obwohl die Aktivitäten der Methodologie selbst oft nur auf abstrakter Ebene beschrieben werden.

Für das TOVE-Vorgehensmodell dagegen wäre es wegen seines strukturierten Phasenaufbaus und der vorgeschlagenen formalsprachlichen Notationen nahe liegend, den

76) Ontolingua ist eine Software-Umgebung zur Konstruktion von Ontologien durch verteilt arbeitende Gruppen, die neben einer Bibliothek bestehender Ontologien auch einen Editor für die Generierung und Bearbeitung von Ontologien bereitstellt und diese in verschiedene Sprachen übersetzen kann; vgl. dazu GRUBER (1993), S. 203ff. und FARQUHAR (1996), S. 44.3ff.

Entwicklungsprozess durch Computerwerkzeuge zu unterstützen. Diese Hilfsmittel werden aber bei der Darstellung des Ansatzes nicht erwähnt.

Die Autoren von METHONTOLOGY unterstreichen jedoch den Nutzen des Einsatzes einer Software-Umgebung für die Ontologienentwicklung[77] und empfehlen für den Import, Export und die Erstellung von Ontologien die Arbeitsplattform ODE bzw. WebODE[78].

Im Rahmen der On-To-Knowledge-Methodologie wird eine umfassende Architektur für die Realisierung einer Entwicklungsumgebung vorgestellt[79], deren Kernbestandteil „OntoEdit" ist, eine Anwendung zur Erstellung, Bearbeitung, Formalisierung und Visualisierung von Ontologien.

Der Kollaborative Ansatz von HOLSAPPLE und JOSHI hingegen fokussiert die zwischenmenschliche Diskussion zur gemeinsamen Ontologienentwicklung. Auf mögliche Unterstützung durch spezielle Computerwerkzeuge gehen die Autoren nicht ein. Nur für die Sammlung und Speicherung der Rückmeldungen der verschiedenen Teilnehmer wird der Einsatz einer Datenbank als sinnvolle Computerunterstützung erwähnt.

Insgesamt fällt auf, dass die Vorschläge, die hinsichtlich der Nutzung von Software-Werkzeugen gemacht werden, sich in fast allen Fällen auf die Anwendung in einzelnen Projektphasen beziehen statt auf den gesamten Entwicklungsprozess, wie es zum Beispiel im Software Engineering bereits Praxis ist.

---

77) Vgl. FERNÁNDEZ (1997), S. 39.

78) Vgl. zu WebODE ARPÍREZ (2001).

79) Vgl. SURE (2002b), S. 17ff.

4.5 Zusammenfassung der Analyseergebnisse

Die folgende Tabelle fasst die Ergebnisse zur Anforderungserfüllung durch die verschiedenen Ansätze zusammen.

Ansatz / Anforderung	Enterprise Model Ansatz	TOVE Methodologie	METHONTOLOGY	On-To-Knowledge Methodologie	Kollaborativer Ansatz
Generizität	+	o	+	o	o
Anwendungsbezogenheit	- / o	+	o	+	o / +
Begründung, Dokumentation	+	o	+	-	o
Einfachheit	+	-	o	o	+
Klarheit	+	+	+	o	o
Werkzeugunterstützung	o	-	o	+	-

Tabelle 2: Ergebnisse der Überprüfung der Anforderungen[80]

Eine Untersuchung von FERNÁNDEZ LÓPEZ[81] kommt zu dem abweichenden Urteil, dass METHONTOLOGY die derzeit am weitesten entwickelte Vorgehensmodell zur Entwicklung von Ontologien darstellt. Nichts desto trotz erscheint es jedoch notwendig für das Projekt KOWIEN ein spezifisches Vorgehensmodell zu entwickeln um die Besonderheiten eines Kompetenzmanagementsystems ausreichend zu berücksichtigen, insbesondere weil Wissen über Wissen erhoben werden soll und nicht auf der Objekt-Ebene verharrt wird.[82]

---

80) Legende: (-) entspricht *negativ*, (o) entspricht *durchschnittlich*, (+) entspricht *gut*. Die scheinbare Doppelbewertung soll den Zwischenfall darstellen.

81) Vgl. FERNÁNDEZ LOPÉZ (1999). Die Untersuchung vergleicht Enterprise Model Approach, TOVE, METHODOLOGY, KACTUS und SENSUS. Die Autoren folgen hierbei nicht der Auffassung, dass es sich bei SENSUS auch um ein allgemeines Vorgehensmodell zur Erstellung von Ontologien handelt, sondern vielmehr um eine Anwendung, aus der auch Ontologien entwickelt werden können. Weil SENSUS jedoch auf eine bestehende Ontologie aufsetzt, sehen die Autoren den Einfluss als zu hoch an, um zu sagen, dass es sich hierbei um die Erstellung „neuer" Ontologien handelt. Ferner berücksichtigen die Autoren IDEF5 und Ontokick als „vollständige" Vorgehensmodelle.

82) Siehe hierzu auch den KOWIEN-Projektbericht „Generisches Vorgehensmodell Version 1" 2/2003, in Vorbereitung.

4.6 Vorgehensmodelle mit spezifischem Teilfokus Ontologien

An dieser Stelle werden Vorgehensmodelle berücksichtigt, die sich nur mit Teilen eines *Ontologies-Life-Cycle* dezidiert auseinandergesetzt haben. Sie besitzen somit nicht den umfassenden Anspruch wie die vorher genannten Ansätze an die Entwicklung von ontologienbasierten Systemen über den gesamten Life-Cycle, jedoch erscheinen einige Erkenntnisse als so wertvoll, dass sie an dieser Stelle kurz vorgestellt werden sollen.

4.6.1 SENSUS

Vorgestellt von SWARTOUT[83], unterscheidet der Ansatz zuerst einmal in *domain ontology* und *theory ontology*. Während die theory ontology ein Set von Konzepten bereithält, welches einen Aspekt der Welt abbildet (z.B. Zeit, Raum oder Pläne), werden bei der domain ontology Begriffe abgebildet, die eine bestimmte Domäne beschreiben. Theory ontologies sind erwartungsgemäß weniger umfangreich als domain ontologies, die leicht tausende Konzepte berücksichtigen können. Der SENSUS-Ansatz berücksichtigt lediglich domain ontologies.

SENSUS stellt eine natürlichsprachliche („natural language based“) Ontologie dar, die in ihrer Endversion über 50.000 Konzepte beinhaltet. Sie ist das Ergebnis des Mergers von *Penman Upper Model, Ontos, WordNet* und der Begriffssammlungen aus elektronischen Wörterbüchern.

Bei der Entwicklung von Domänen-Ontologien soll SENSUS als Ausgangsbasis genutzt werden. Zur Erstellung der spezifischen Domänen-Ontologien werden „Saat“-Begriffe („*seed“ terms*) als repräsentativ selektiert und von Hand mit SENSUS verlinkt. Alle Konzepte ausgehend von den Saat-Begriffen hin zu den Root-Begriffen werden in den zu erstellenden Domänen-Ontologien berücksichtigt. Hinzu kommen relevante semantische Kategorien und ganze Unterklassenbäume, die eine besondere Bedeutung für die Verbindung von Saat- und Root-Begriffen darstellen. Die verbleibenden SENSUS-Begriffe werden als irrelevant verworfen, um Speicherplatz zu sparen und die Effizienz zu erhöhen.

---

83) Vgl. SWARTOUT (1996).

4.6.2 ONIONS

Die ONIONS (**ON**tologic **I**ntegration **O**f **N**aïve **S**ources) Methode resultiert aus dem Interesse, bereits existierende Ontologien zu integrieren und wurde ab 1992 entwickelt.[84]

Die Ziele waren ein gut abgestimmtes Set von generischen Ontologien, die Integration relevanter Domänen-Ontologien in eine formale, konzeptionell befriedigende Ontologien und ein explizites Nachverfolgen der Entwicklung von Konzepten, Constraints und Variantenwahrnehmung in der Ontologienentwicklung zu ermöglichen.[85]

Die Methode beinhaltet ein allgemeines Vorgehen zur Interpretation von fachsprachigen Definitionen der spezifischen Domänen-Ontologien und der anschließenden Erstellung neuerer und offener Domänen-Ontologien, die die zu integrierenden Domänen-Ontologien zusammenbringen. Das Modell gliedert sich in die folgenden 6 Schritte:

1. Extrahieren von Quell-Begriffen,
2. Feststellen der lokalen Definition der Begriffen,
3. Multi-lokale Definition der Begriffen: Die lokalen Definitionen werde in Bezug auf generelle Theorien angereichert,
4. Multi-lokale Definition der Begriffen: Die Theorien werden zur Bildung von Top-Level-Kategorien weiterentwickelt,
5. Multi-lokale Definition der Begriffen: Lokale Definitionen und Top-Level-Kategorien werden zusammengeführt,
6. Das integrierte Modell wird formalisiert.

ONIONS fokussiert hierbei vor allem auf Probleme der Ontologie-Erstellung, wie Abbruch-Kriterium und Relevanz-Kriterium. Das Abbruch-Kriterium beschäftigt sich mit dem den Umständen entsprechenden sinnvollsten Detaillierungsgrad einer Konzeptualisierung. Das Relevanz-Kriterium beschäftigt sich mit der Frage, wie festgestellt werden kann, was als konzeptionell relevant anerkannt wird. Wie JONES feststellt, bleiben die Entwickler jedoch hier ausreichende Antworten schuldig.[86]

---

84) Vgl. GANGEMI (1996), GANGEMI (1998).

85) Vgl. GANGEMI (1998), S. 163.

86) Vgl. JONES (1998), S. 9.

4.6.3 ONTOCLEAN

Die OntoClean Methode wurde von GUARINO ET AL. entwickelt. Sie dient insbesondere der Validierung von Taxonomien, indem sie unangemessene und inkonsistente Modellierungen aufdeckt.[87] Hierbei bedient sie sich formaler Begriffe, die so allgemeinen Charakter besitzen, dass sie in jeder Ontologie verwendet werden können. Mit diesen Begriffen wird ein Satz von Meta-Eigenschaften definiert, der genutzt wird um relevante Teile von Attributen, Klassen und Relationen mit ihrer intendierten Bedeutung wiederum abzubilden. Aus diesem Vorgehen ergeben sich Erkenntnisse hinsichtlich der Bedeutung einzelner Begriffe.

4.6.4 MENELAS

Die MENELAS Ontologie wurde von BOUAUD ET AL.[88] vorgestellt. Während ihrer Entwicklung wurden insbesondere vier Grundsätze ermittelt, die wichtig für die taxonomische Abbildung von Wissen innerhalb von Ontologien sind. Das Ziel von MENELAS war die Entwicklung eines „natural language understanding system based on Conceptual Graphs“[89]. Nachfolgend werden die vier Grundsätze kurz erläutert:[90]

1. Similarity Principle:

   Eine Unterklasse muss vom selben Typ der Oberklasse sein.

2. Specificity Principle:
   Eine Unterklasse muss von der Oberklasse unterscheidbar sein. Der Unterschied zusammen mit der Definition der Oberklasse bildet dann die notwendigen und hinreichenden Bedingungen für die Definition der Unterklasse.

3. Opposition Principle:
   Die Unterklassen eines Konzepts sollen disjunkt sein. Alle Unterklassen müssen paarweise inkompatibel sein.

---

87) Vgl. GUARINO (2002), S. 61.

88) Vgl. BOUAUD (1994), BOUAUD (1995).

89) BOUAUD (1994), S. 2.

90) Vgl. BOUAUD (1995), S. 5.

4. Unique Semantic Axis Principle:
   Die Subklassen eines Konzepts sollen eingeengt werden, anhand einer gemeinsamen Dimension oder Achse, um sich von der Oberklasse zu unterscheiden.
   Der vierte Grundsatz ergibt sich, um die beiden Grundsätze zwei und drei zu berücksichtigen.

JONES ET AL. merken an, dass das hier kurz umrissene Vorgehensmodell einen idealisierten Blick auf Taxonomien als Möglichkeit zur Repräsentation von Domänen wirft. Sie zweifeln an der Anwendbarkeit taxonomischer Repräsentationen bei vielen Domänen.[91]

91) Vgl. JONES (1998), S. 9.

Literaturverzeichnis

ALAN (2002)
ALAN, Y.: Anforderungen an den KOWIEN-Prototypen. Projektbericht 6/2002, Projekt KOWIEN, Institut für Produktion und Industrielles Informationsmanagement, Universität Essen, Essen 2002.

ALPARSLAN (2002)
ALPARSLAN, A.; DITTMANN, L.; ZELEWSKI, S.; ILGEN, A.: Wissensmanagement im Anlagenbau – Computergestütztes Management von Wissen über Mitarbeiterkompetenzen. In: Industrie Management, 18 (2002) 6, S. 45-48.

ARPÍREZ (2001)
ARPÍREZ, J.; CORCHO, O.; FERNÁNDEZ LÓPEZ, M.; GÓMEZ-PÉREZ, A.: WebODE: a Scalable Workbench for Ontological Engineering. In: GIL, Y.; MUSEN, M.; SHAVLIK, J. (HRSG.): Proceedings of the International Conference on Knowledge Capture – KCAP-01, Victoria 2001, S. 6-13.

BIETHAHN (2000)
BIETHAHN, J.; MUKSCH, H.; RUF, W.: Ganzheitliches Informationsmanagement, Band 1: Grundlagen, 5. Auflage, München 2000.

BOEHM (1988)
BOEHM, B. W.: A Spiral Model of Software Development and Enhancement. In: IEEE Computer, 21 (1998) 3, S. 61-72.

BOUAUD (1994)
BOUAUD, J.; BACHIMONT, B.; CHARLET, J.; ZWEIGENBAUM, P.: Acquisition and Structuring of an Ontology within Conceptual Graphs. In: Proceedings of ICCS'94 Workshop on Knowledge Acquisition using Conceptual Graph Theory, University of Maryland, College Park 1994, S. 1-25.

BOUAUD (1995)
BOUAUD, J.; BACHIMONT, B.; CHARLET, J.; ZWEIGENBAUM, P.: Methodological Principles for Structuring an Ontology. Presented at the Workshop on Basic Ontological Issues in Knowledge Sharing (in conduction with IJCAI-95), Montreal 1995. Auch erschienen als DIAM Rapport Interne RI-95-148, http://www.biomath.jussieu.fr, Zugriff am 18.2.2003, S. 1-7.

BRÖHL (1995)
BRÖHL, A.-P.; DRÖSCHEL, W.: Einführung in das V-Modell. In: BRÖHL, A.-P.; DRÖSCHEL, W. (HRSG.): Das V-Modell – Der Standard für die Softwareentwicklung mit Praxisleitfaden, 2. Auflage, München 1995.

BULLINGER (1997)
BULLINGER, H.-J.; FÄHNRICH, K.-P.: Betriebliche Informationssysteme - Grundlagen und Werkzeuge der methodischen Softwareentwicklung, Berlin 1997.

FARQUHAR (1996)
FARQUHAR, A.; FIKES, R.; RICE, J.: The Ontolingua Server: a Tool for Collaborative Ontology Construction. In: Proceedings of the 10th Knowledge Acquisition for Knowledge-Based Systems Workshop (1996), Banff 1996, S. 44.1-44.19.

FERNÁNDEZ (1997)
FERNÁNDEZ, M.; GÓMEZ-PÉREZ, A.; JURISTO, N.: METHONTOLOGY: From Ontological Art Towards Ontological Engineering. In: FARQUAR, A.; GRUNINGER, M. (HRSG.): Ontological Engineering - Papers from the 1997 AAAI Spring Symposium, Stanford/California 1997, S. 33-40.

FERNÁNDEZ LOPÉZ (1999a)
FERNÁNDEZ LÒPEZ, M.; GOMEZ-PEREZ, A.; PAZOS SIERRA, J.; PAZOS SIERRA, A.: Building a Chemical Ontology using METHONTOLOGY and the Ontology Design Environment. In: IEEE Intelligent Systems: Special Issue on Uses of Ontologies. 14 (1999) 1, S. 37-46.

FERNÁNDEZ LOPÉZ (1999b)
FERNÁNDEZ LÒPEZ, M.: Overview Of Methodologies For Building Ontologies. In: BENJAMINS, V.R.; CHANDRASEKARAN, B.; GOMEZ-PEREZ, A.; GUARINO, N.; USCHOLD, M. (HRSG.): Proceedings of the IJCAI-99 Workshop on Ontologies and Problem-Solving Methods, Stockholm 1999, S. 4.1-4.13.

GANGEMI (1996)
GANGEMI, A.; STEVE, G.; GIACOMELLI, F.: ONIONS: An Ontological Methodology for Taxonomic Knowledge Integration. In VAN DER VET, P. (HRSG.): Proceedings of the Workshop on Ontological Engineering (in conjunction with ECAI´96), Budapest 1996. URL: http://wwwkbs.cs.utwente.nl/EcaiWorkshop/fullpapers.html, Zugriff am 18.2.2003, S. 1-11.

GANGEMI (1998)
GANGEMI, A.; PISANELLI, D. M.; STEVE, G.: Ontology Integration: Experiences with Medical Terminologies. In: GUARINO, N. (HRSG.): Formal Ontology in Information Systems, Amsterdam 1998, S. 163-178.

GENESERETH (1992)
GENESERETH, M. R.; FIKES, R. E.: Knowledge Interchange Format – Version 3.0, Reference Manual. Technical Report, Computer Science Department, Stanford 1992. URL: http://www-ksl.stanford.edu/knowledge-sharing/papers/kif.ps, Zugriff am 18.2. 2003, S. 1-68.

GOMEZ-PEREZ (1996)
GOMEZ-PEREZ, A. ; FERNANDEZ, M.; DE VINCENTE, A.J.: Towards a Method to Conceptualize Domain Ontologies. In: Proceedings of the Workshop Ontological Engineering ECAI-96, Budapest 1996, S. 41-52.

GÓMEZ-PÉREZ (2001)
GÓMEZ-PÉREZ, A.: Methodologies, Tools and Languages. Where is the meeting point? Präsentation anlässlich: 5th International Protégé Workshop, Newcastle 2001.
URL: http://www.schin.ncl.ac.uk/protege2001/presentations/GOMEZ-~1.PDF, Zugriff am 18.2.2003, S. 1-38.

GRUBER (1993)
GRUBER, R.: A Translation Approach to Portable Ontology Specifications. In: Knowledge Acquisition, 5 (1993) 2, S. 199-220.

GRÜNINGER (1995)
GRÜNINGER, M.; FOX, M. S.: Methodology for the Design and Evaluation of Ontologies. In: Workshop on Basic Ontological Issues in Knowledge Sharing (IJCAI-95), Montreal/Canada 1995.
URL: http://www.eil.utoronto.ca/EIL/public/method.ps, Zugriff am 19.2.2003, S. 1-10.

GUARINO (2002)
GUARINO, N.; WELTY, C.: Evaluating Ontological Decisions with OntoClean. In: Communications of the ACM 45 (2002) 2, S. 61-65.

HAUN (2000)
HAUN, M.: Wissensbasierte Systeme, Renningen 2000.

HOLSAPPLE (2002)
HOLSAPPLE, C. W.; JOSHI, K. D.: A Collaborative Approach to Ontology Design. In: Communications of the ACM, 45 (2002) 2, S. 42-47.

IEEE (1996)
IEEE 1074-1995, IEEE Standard for Developing Software Life Cycle Processes.

JONES (1998)
JONES, D.; BENCH-CAPON, T.; VISSER, P.: Methodologies for Ontology Development, In: Proceedings IT&KNOWS Conference, XV. IFIP World Computer Congress, Budapest 1998, S. 62-75.

KBSI (1994)
O. V.: KBSI (KNOWLEDGE BASED SYSTEMS INC.): The IDEF5 Ontology Description Capture Method Overview, Texas 1994. URL: http://www.idef.com, Zugriff am 4.9.2002, S. 1-187.

LAU (2002)
LAU, T. ; SURE, Y.: Introducing Ontology-based Skills Management at a large Insurance Company. In: GLINZ, MÜLLER-LUSCHNAT (HRSG.): Modellierung 2002, Modellierung in der Praxis – Modellierung für die Praxis, Bonn 2002, S. 123-134.

LINSTONE (1975)
LINSTONE, H.; TUROFF, M.: The Delphi Method: Techniques and Applications, Reading 1975.

MOODY (1994)
MOODY, D.; SHANKS, G.: What Makes a Good Data Model? Evaluating the Quality of Entity Relationship Models. In: LOUCOPOULOS, P. (HRSG.): Entity-Relationship Approach – ER'94 (13th International Conference on the Entity-Relationship Approach, Manchester), Heidelberg 1994, S. 94-111.

POMBERGER (1993)
POMBERGER, G.; BLASCHEK, G.: Software-Engineering - Grundlagen des Software Engineering: „Prototyping und objektorientierte Software-Entwicklung", München 1993.

ROSEMANN (1999)
ROSEMANN, M.; SCHÜTTE, R.: Multiperspektivische Referenzmodellierung. In: BECKER, J.; ROSEMANN, M.; SCHÜTTE, R. (HRSG.): Referenzmodellierung - State-of-the-Art und Entwicklungsperspektiven, Heidelberg 1999, S. 22-44.

ROYCE (1970)
ROYCE, W. W.: Managing the Development of Large Software Systems, Concepts and Techniques. Proceedings of the IEEE WESCON, Los Angeles 1970, S. 1-9.

SCACCHI (2001)
SCACCHI, W.: Process Models in Software Engineering. In: MARCINIAK, J. (HRSG.): Encyclopedia of Software Engineering, 2. Auflage, Wiley 2002, S. 993-1005.

SCHEER (1999)
SCHEER, A.-W.: „ARIS - House of Business Engineering" - Konzept zur Beschreibung und Ausführung von Referenzmodellen. In: BECKER, J.; ROSEMANN, M.; SCHÜTTE, R. (HRSG.): Referenzmodellierung - State-of-the-Art und Entwicklungsperspektiven, Heidelberg 1999, S. 1-21.

SCHNURR (2000)
SCHNURR, H.; SURE, Y.; STUDER, R.; AKKERMANNS, H.: On-To-Knowledge Methodology – Baseline Version. On-To-Knowledge Deliverable D-15, Institut AIFB, Universität Karlsruhe, Karlsruhe 2000.
URL: http://www.ontoknowledge.org/countd/countdown.cgi?del15.pdf, Zugriff am 19.2.2003.

SCHREIBER (1995)
SCHREIBER, G.; WIELINGA, B.; JANSWEIJER, W.: The KACTUS View on the 'O' Word. In: Workshop on Basic Ontological Issues in Knowledge Sharing (IJCAI'95), Montreal/Canada, 1995.
URL: http://www.swi.psy.uva.nl/usr/Schreiber/papers/Schreiber95a.pdf, Zugriff am 19.2.2003, S. 1-10.
SCHREIBER (2001)
SCHREIBER, G.; AKKERMANS, H.; ANJEWIERDEN, A.; DE HOOG, R.; SHADBOLT, N.; VAN DE VELDE, W.; WIELINGA, B.: Knowledge engineering and management: the CommonKADS methodology, 2. Aufl., Massachusetts 2001.

SCHÜTTE (1997)
SCHÜTTE, R.: Die neuen Grundsätze ordnungsmäßiger Modellierung. Paper zum Forschungsforum ’97, Leipzig 1997.

SCHÜTTE (1998)
SCHÜTTE, R.: Grundsätze ordnungsmäßiger Referenzmodellierung - Konstruktion konfigurations- und anpassungsorientierter Modelle, Dissertation Universität Münster, Wiesbaden 1998.

SOMMERVILLE (2001)
SOMMERVILLE, I.: Software Engineering, 6. Auflage, Edinburgh 2001.

STAAB (2000)
STAAB, S.; MAEDCHE, A.: Ontology engineering beyond the modeling of concepts and relations. In: BENJAMINS, R.V.; GOMEZ-PEREZ, A.; GUARINO, N.; USCHOLD, M. (HRSG.): Proceedings of the ECAI'2000 Workshop on Application of Ontologies and Problem-Solving Methods, Amsterdam 2000.
URL: http://www.aifb.uni-karlsruhe.de/WBS/ama/publications/ontoengineer_staabmaedche.pdf, Zugriff am 10.2.2003, S. 1-6.

STUDER (1998)
STUDER, R.; BENJAMINS, V. R.; FENSEL, D.: Knowledge Engineering: Principles and Methods. In: IEEE Transactions on Data and Knowledge Engineering, 25 (1998) 1-2, S. 161-197.

SURE (2002a)
SURE, Y.: On-To-Knowledge - Ontology based Knowledge Management Tools and their Application. In: KI - Künstliche Intelligenz, 14 (2002) 1, S. 35-38.

SURE (2002b)
SURE, Y.; STUDER, R.: On-To-Knowledge Methodology – final version. On-To-Knowledge deliverable D-18, Institut AIFB, Universität Karlsruhe, Karlsruhe 2002.
URL: http://www.ontoknowledge.org/downl/del18.pdf, Zugriff am 9.1.2003.

SWARTOUT (1996)
SWARTOUT, B.; PATIL, R.; KNIGHT, K.: Toward distributed use of large-scale ontologies. In: GAINES B.; MUSEN, M. (HRSG.): Proceedings of the 10th Knowledge Acquisition for Knowledge-Based Systems Workshop (KAW '96), Alberta 1996, S. 32.1-32.19.

USCHOLD (1995)
USCHOLD, M.; KING, M.: Towards a Methodology for Building Ontologies. In: Workshop on Basic Ontological Issues in Knowledge Sharing (in conjunction with IJCAI-95), Montreal 1995. Auch erschienen als AIAI Technical Report 183, S. 1-13.

USCHOLD (1996a)
USCHOLD, M.: Converting an Informal Ontology into Ontolingua: Some Experiences. In: Proceedings of the Workshop on Ontological Engineering (in conjunction with ECAI´96), Budapest 1996. Auch erschienen als AIAI Technical Report 192, S. 1-15.

USCHOLD (1996b)
USCHOLD, M.: Building Ontologies: Towards A Unified Methodology. In: 16th Annual Conference of the British Computer Society Specialist Group on Expert Systems, Cambridge 1996. Auch erschienen als AIAI Technical Report 197, S. 1-18.

USCHOLD (1996c)
USCHOLD, M.; GRÜNINGER, M.: Ontologies: Principles, Methods and Applications. In: Knowledge Engineering Review, 11 (1996) 2, S. 93-155.

VERLAGE (1998)
VERLAGE, M.: Vorgehensmodelle und ihre Formalisierung. In: KNEUPER; MÜLLER-LUSCHNAT; OBERWEIS, A. (HRSG.): Vorgehensmodelle für die betriebliche Anwendungsentwicklung, Leipzig 1998, S. 60-76.

VOSSEN (1994)
VOSSEN, G.: Datenmodelle, Datenbanksprachen und Datenbank-Management-Systeme, 2. Auflage, Bonn 1994.

WANG (2002)
WANG, Y.: Software Engineering Standards: Review and Perspectives. In: CHANG, S. K. (HRSG.): Handbook of Software Engineering und Knowledge Engineering, Vol. 1: Fundamentals, Singapore 2002, S. 277-304.

ZEHNDER (1998)
ZEHNDER, C. A.: Informationssysteme und Datenbanken, 6. Auflage, Stuttgart 1998.

ZELEWSKI (2002)
ZELEWSKI, S.: Wissensmanagement mit Ontologien. In: Essener Unikate 11 (2002) 18, S. 62-73.

Institut für Produktion und Industrielles Informationsmanagement Universität Duisburg-Essen / Campus Essen

---

Verzeichnis der KOWIEN-Projektberichte

Nr. 1: ALPARSLAN, A.: Ablauforganisation des Wissensmanagements. Projektbericht 1/2002, Projekt KOWIEN, Institut für Produktion und Industrielles Informationsmanagement, Universität Essen, Essen 2002.

Nr. 2: ALAN, Y.: Methoden zur Akquisition von Wissen über Kompetenzen. Projektbericht 2/2002, Projekt KOWIEN, Institut für Produktion und Industrielles Informationsmanagement, Universität Essen, Essen 2002.

Nr. 3: DITTMANN, L.: Sprachen zur Repräsentation von Wissen - eine untersuchende Darstellung. Projektbericht 3/2002, Projekt KOWIEN, Institut für Produktion und Industrielles Informationsmanagement, Universität Essen, Essen 2002.

Nr. 4: DITTMANN, L.: Zwecke und Sprachen des Wissensmanagements zum Managen von Kompetenzen. Projektbericht 4/2002, Projekt KOWIEN, Institut für Produktion und Industrielles Informationsmanagement, Universität Essen, Essen 2002.

Nr. 5: ALAN, Y.; BÄUMGEN, C.: Anforderungen an den KOWIEN-Prototypen. Projektbericht 5/2002, Projekt KOWIEN, Institut für Produktion und Industrielles Informationsmanagement, Universität Essen, Essen 2002.

Nr. 6: ALPARSLAN, A.: Wissensanalyse und Wissensstrukturierung. Projektbericht 6/2002, Projekt KOWIEN, Institut für Produktion und Industrielles Informationsmanagement, Universität Essen, Essen 2002.

Nr. 7: ALAN, Y.: Evaluation der KOWIEN-Zwischenergebnisse. Projektbericht 7/2002, Projekt KOWIEN, Institut für Produktion und Industrielles Informationsmanagement, Universität Essen, Essen 2002.

Nr. 8: ZUG, S.; KLUMPP, M.; KROL, B.: Wissensmanagement im Gesundheitswesen, Arbeitsbericht Nr. 16, Institut für Produktion und Industrielles Informationsmanagement, Universität Duisburg-Essen (Campus Essen), Essen 2003.

Nr. 9: APKE, S.; DITTMANN, L.: Analyse von Vorgehensmodellen aus dem Software, Knowledge und Ontologies Engineering. Projektbericht 1/2003, Projekt KOWIEN, Institut für Produktion und Industrielles Informationsmanagement, Universität Duisburg-Essen (Campus Essen), Essen 2003.

Nr. 10: ALAN, Y.: Konstruktion der KOWIEN-Ontologie. Projektbericht 2/2003, Projekt KOWIEN, Institut für Produktion und Industrielles Informationsmanagement, Universität Duisburg-Essen (Campus Essen), Essen 2003.

Nr. 11: ALAN, Y.: Ontologiebasierte Wissensräume. Projektbericht 3/2003, Projekt KOWIEN, Institut für Produktion und Industrielles Informationsmanagement, Universität Duisburg-Essen (Campus Essen), Essen 2003.

Nr. 12: APKE, S.; DITTMANN, L.: Generisches Vorgehensmodell KOWIEN Version 1.0. Projektbericht 4/2003, Projekt KOWIEN, Institut für Produktion und Industrielles Informationsmanagement, Universität Duisburg-Essen (Campus Essen), Essen 2003.

Nr. 13: ALAN, Y.: Modifikation der KOWIEN-Ontologie. Projektbericht 5/2003, Projekt KOWIEN, Institut für Produktion und Industrielles Informationsmanagement, Universität Duisburg-Essen (Campus Essen), Essen 2003.

Nr. 14: ALAN, Y.; ALPARSLAN, A.; DITTMANN, L.: Werkzeuge zur Sicherstellung der Adaptibilität des KOWIEN-Vorgehensmodells. Projektbericht 6/2003, Projekt KOWIEN, Institut für Produktion und Industrielles Informationsmanagement, Universität Duisburg-Essen (Campus Essen), Essen 2003.

Nr. 15: ENGELMANN, K.; ALAN, Y.: KOWIEN Fallstudie - Gebert GmbH. Projektbericht 7/2003, Projekt KOWIEN, Institut für Produktion und Industrielles Informationsmanagement, Universität Duisburg-Essen (Campus Essen), Essen 2003.

Nr. 16: DITTMANN, L.: Towards Ontology-based Skills Management. Projektbericht 8/2003, Projekt KOWIEN, Institut für Produktion und Industrielles Informationsmanagement, Universität Duisburg-Essen (Campus Essen), Essen 2003.

Nr. 17: ALPARSLAN, A.: Evaluation des KOWIEN-Vorgehensmodells, Projektbericht 1/2004, Projekt KOWIEN, Institut für Produktion und Industrielles Informationsmanagement, Universität Duisburg-Essen (Campus Essen), Essen 2004.

Nr. 18: APKE, S.; BÄUMGEN, C.; BREMER, A.; DITTMANN, L.: Anforderungsspezifikation für die Entwicklung einer Kompetenz-Ontologie für die Deutsche Montan Technologie GmbH. Projektbericht 2/2004, Projekt KOWIEN, Universität Duisburg-Essen (Campus Essen), Essen 2004.

Nr. 19: HÜGENS, T.: Inferenzregeln des „plausiblen Schließens“ zur Explizierung von implizitem Wissen über Kompetenzen. Projektbericht 3/2004, Projekt KOWIEN, Universität Duisburg-Essen (Campus Essen), Essen 2004.

Nr. 20: ALAN, Y.: Erweiterung von Ontologien um dynamische Aspekte. Projektbericht 4/2004, Projekt KOWIEN, Institut für Produktion und Industrielles Informationsmanagement, Universität Duisburg-Essen (Campus Essen), Essen 2004.

Nr. 21: WEICHELT, T.: Entwicklung einer E-Learning-Anwendung zum kompetenzprofil- und ontologiebasierten Wissensmanagement – Modul 1: Grundlagen. Projektbericht 5/2004, Projekt KOWIEN, Universität Duisburg-Essen (Campus Essen), Essen 2004.