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 5/2002

Anforderungen an den KOWIEN-Prototypen (V 1.1)

Dipl.-Kfm. Yilmaz Alan
Yilmaz.Alan@pim.uni-essen.de

Dipl.-Inform. Christof Bäumgen
Christof.Baeumgen@comma-soft.com

Das Drittmittelprojekt KOWIEN
("Kooperatives Wissensmanagement in Engineering-Netzwerken")
wird mit Mitteln des Bundesministeriums für Bildung und Forschung (BMBF)
(Förderkennzeichen Hauptband 02 PD 1060)
innerhalb des Rahmenkonzepts "Forschung für die Produktion von morgen"
gefördert und vom Projektträger Produktion und Fertigungstechnologien (PFT),
der Forschungszentrum Karlsruhe GmbH, betreut.
Die Mitglieder des Projektteams danken
für die großzügige Unterstützung ihrer Forschungs- und Transferarbeiten.

Juni 2002

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Motivation

1.1 Hintergrund

Innerhalb des Projektes KOWIEN werden durch die Projektpartner hauptsächlich *sechs* Ergebnisse angestrebt:

- *Konzeption für Wissensmanagementsysteme*: Es wird eine Konzeption für computergestützte Wissensmanagementsysteme entwickelt, die das Management von Kompetenzprofilen auf der Basis von Ontologien unterstützt.
- *Generisches Vorgehensmodell*: Um die praktische Anwendbarkeit der genannten Konzeption zu ermöglichen, wird am Beispiel der Service- und Produkt-Engineering-Szenarien ein Katalog entwickelt, der die praxisrelevanten Anforderungen an die Wissensakquisition, -strukturierung und -repräsentation enthält.
- *Customizing Instrumente*: Um das generische Vorgehensmodell situationsspezifisch anpassen zu können, werden Instrumente entwickelt, durch die eine Individualisierung des Wissensmanagementsystems ermöglicht wird.
- *Fallstudien*: Um die Projektergebnisse auf ihre Praxistauglichkeit hin überprüfen zu können, werden von den Praxispartnern Fallstudien entwickelt.
- *E-Learning Modul*: Die Praxispartner entwickeln gemeinsam ein Modul, das als „Learnware“ im Bereich der universitären Aus- und Weiterbildung eingesetzt werden kann.
- *Prototyp*: Durch die Comma Soft AG wird ein prototypisches IT-System entwickelt, in dem die Konzeption für Wissensmanagementsysteme und das generische Vorgehensmodell implementiert sind.

Innerhalb des Arbeitspaketes 2.3 wurde für die Konstruktion des *Prototypen* ein *Anforderungskatalog* angekündigt. Dieser Anforderungskatalog gilt als ein Teil der Anforderungen, die aus der Perspektive der betrieblichen Praxis von Instrumenten zur Unterstützung des Wissensmanagements auf der Basis von Kompetenzprofilen, Ontologien und Referenzmodellen erfüllt werden sollten. Als ein wesentlicher Bestandteil dieses Anforderungskatalogs wurden hierbei auch *Gütekriterien für Ontologien* angekündigt.

1.2 Problemstellung

Es kann angenommen werden, dass sich die Unterstützung des Wissensmanagements durch Softwaresysteme auf den Unternehmenserfolg effizienzsteigernd auswirkt. Aus betriebswirtschaftlicher Perspektive ist dabei zu beachten, dass lediglich unter der Bedingung einer anforderungsgestützten Entwicklung der Software ein positiver Beitrag zum Unternehmenserfolg gewährleistet werden kann. Wird die Entwicklung der Software nicht einer systematischen Anforderungsanalyse unterworfen, kann es sogar zu einer Minderung des Unternehmenserfolgs durch die Investitionen in die Software kommen. In diesem Fall würden die Kosten für die Implementierung der Software den Nutzen, der durch sie erreicht wird übersteigen[1)]. Eine „explosionsartige" Zunahme[2)] der (Grenz-)Kosten, die nötig wären, um fehlende oder falsch spezifizierte Anforderungen im Projektverlauf zu kompensieren, kann somit nur durch die systematische Entwicklung eines Anforderungskataloges verhindert werden.

1) Problematisch wird diese Aussage allerdings im Kontext der Entwicklung von Software zum Management von *Wissen über Kompetenzen*, da bisher keine Ergebnisse über die Höhe des Nutzens vorliegen, die durch IT-Unterstützung hierbei erreicht werden können.

2) Vgl. KAHLBRANDT (2001) S. 45; Es wird dabei allerdings eine empirische Studie vermisst, die diese Aussage rechtfertigen würde. Zudem wirkt jede „plakative" Darstellung - wie sie in Abbildung 1 erfolgt - künstlich und entbehrt einer betriebswirtschaftlichen Fundierung mit nachvollziehbaren Kennzahlen. Im Kontext der Problemstellung erscheint dies allerdings nebensächlich. Im Weiteren kann von einer intuitiven Zunahme der Grenzkosten ausgegangen werden.

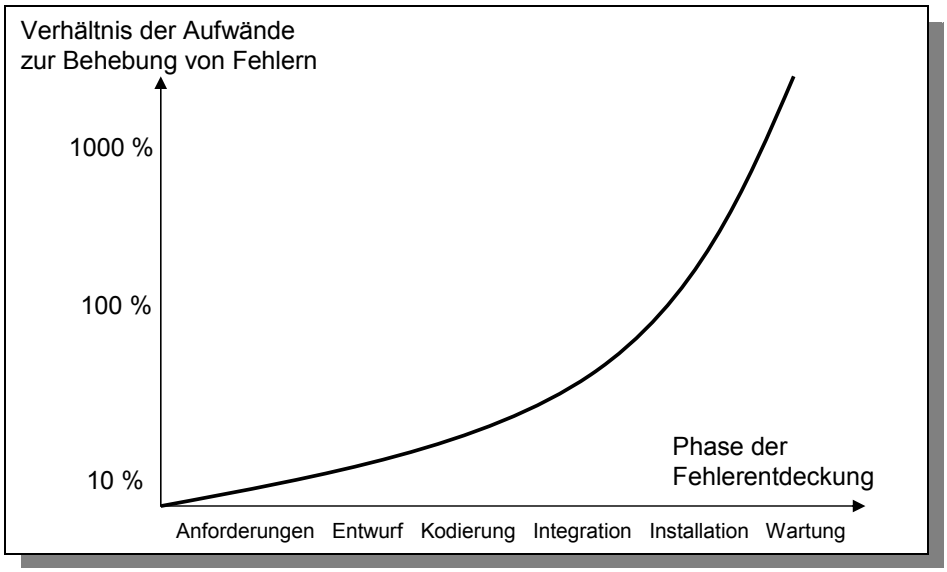

Abbildung 1: Aufwand der Fehlerbehebung[1)]

Um die Problemlösung zu erreichen, die von dem Einsatz von Software in der betrieblichen Praxis erhofft wird, ist es notwendig, sämtliche Faktoren zu benennen, die von der Software berücksichtigt werden sollen und die daraus abgeleiteten Funktionen zu implementieren. Somit gilt die Spezifikation von Anforderungen als kritischer Erfolgsfaktor bei der Konstruktion von Software. Bei einer fehler- oder mangelhaften[2)] Konstruktion eines Anforderungskataloges ist der Problemlösungscharakter des zu entwickelnden Tools nicht gewährleistet.

Über die Probleme einer anforderungsgestützten Softwarekonstruktion hinaus existieren hier spezifische Probleme, die sich aus dem Kontext des KOWIEN-Projektes ergeben. Die Konstruktion eines Anforderungskataloges für den KOWIEN-Prototypen ist dabei durch eine dreifache Komplexität gekennzeichnet:

1. Es ist geplant, den KOWIEN-Prototyp als Instrument im Rahmen des betrieblichen Wissensmanagements zu nutzen. Das Themengebiet Wissensmanagement stellt allerdings eine relativ „neuartige“ Forschungsrichtung dar. Da die bisherigen Untersuchungen in diesem Themengebiet größtenteils bei sozikulturellen oder organisationstheoretischen Aspekten verblieben sind, ist es schwierig zu bestimmen, welche Funktionen durch das System unterstützt werden sollen.

---

1) Quelle: modifiziert übernommen aus: KAHLBRANDT (2001) S. 45; Vgl. auch SNEED (1988) S. 47; RAASCH (1993) S. 6.

2) Es werden hier bereits implizit Anforderungen an eine Anforderungsspezifikation genannt, die im Rahmen der Analyse in Kapitel 3 vertieft werden. Um einen Zugang zu dem Themengebiet zu erhalten, wird bis dahin die Korrektheit und die Vollständigkeit der Anforderungsspezifikation genannt.

2. Die Neuartigkeit des KOWIEN-Projektes ergibt sich zudem aus der Konzentration auf das Wissensmanagement mit Kompetenzprofilen auf der Basis von Ontologien und Referenzmodellen. Während der Einsatz von Referenzmodellen bereits seit längerem in der Literatur untersucht wurde[1)], sind für ontologiebasierte Systeme nur wenige Gestaltungsempfehlungen zu finden.
3. Um die Allgemeingültigkeit der entwickelten Instrumente zu wahren, wurde innerhalb von KOWIEN eine Ausdehnung der Untersuchungen in das Service- und das Produkt-Engineering-Szenario vorgenommen. Der KOWIEN-Prototyp soll dabei aus einem gemeinsamen (generischen) Vorgehensmodell heraus die Unterstützung möglichst vieler Anwendungsfälle in den zwei unterschiedlichen Szenarien gewährleisten.

In dem vorliegenden Projektbericht werden die angesprochenen Problemfelder angegangen. Es werden hierzu bereits bestehende Ansätze aufgegriffen, weiterentwickelt und an KOWIEN-spezifische Umstände angepasst. Nachdem die Problemstellung erörtert wurde, werden im nächsten Kapitel die für die Arbeit erforderlichen Begriffsdefinitionen vorgenommen. Im Anschluss werden in Kapitel 3 Anforderungen aufgestellt, die durch eine Anforderungsspezifikation erfüllt werden sollten. Kapitel 4 bildet den Hauptteil der Untersuchung. Im ersten Teil werden hier Anforderungen vorgestellt, die sich aus dem Kontext der Aufgabenstellung des KOWIEN-Projektes ergeben. Im zweiten Teil werden dann Anforderungen vorgestellt, die - unabhängig von der zu bewältigenden Aufgabenstellung - von der Software erfüllt werden sollten. Der Bericht wird abgeschlossen mit einer Zusammenfassung der wesentlichen Ergebnisse und einem Ausblick auf weitere Aufgaben in KOWIEN.

2 Begriffliche Grundlagen

Untersuchungen aus dem Themengebiet der Erforschung von Methoden zur Erstellung von Anforderungskatalogen an Softwaresysteme sind durch eine Vielfalt der Schwerpunkte bei den benutzten Definitionen gekennzeichnet. Ohne den Versuch zu unternehmen, eine Zusammenfassung dieser Untersuchungen zu geben, werden im Folgenden die für diese Arbeit relevanten Begriffe definiert.

---

1) vgl. z.B. KNACKSTEDT (2001).

Im Fokus der Betrachtung liegt der Begriff *Anforderung*. Unter einer Anforderung (*Requirement*) wird hier eine Funktion oder eine Eigenschaft der Anwendung verstanden, die von Personen verlangt wird, die von der Einführung des Softwaresystems entweder mittelbar oder unmittelbar betroffen sind[1)]. Die Hervorhebung in der benutzten Arbeitsdefinition von *Eigenschaften* einerseits und *Funktionen* andererseits spiegelt eine mögliche Systematisierung wider. Demnach können Anforderungen unterteilt werden in funktionale und nicht-funktionale Anforderungen[2)]. Als funktionale Anforderung gelten alle Anforderung an ein Softwaresystem, die von dessen spezifischem Einsatz abhängen. Nicht-funktionale Anforderungen sind hingegen solche, die unabhängig von Anwendungsfällen des Systems aufgestellt werden können[3)].

Die Erarbeitung von Anforderungen an ein Software-System wird mit dem Begriff *Requirements Engineering* gekennzeichnet. Das Verständnis für Requirements Engineering reicht in der Literatur von der Konstruktion eines Anforderungskataloges bis zu einem Teilgebiet der Informatik[4)]. Innerhalb dieses Berichts wird das Verständnis begrenzt auf einen systematischen Prozess zur Entwicklung von Anforderungen[5)]. Entsprechend dem Prozessmodell von POHL werden die Phasen des Requirements Engineerings in 4 Phasen unterteilt[6)]. In Abbildung 2 wird der Zusammenhang zwischen den Phasen verdeutlicht.

---

1) Es wird hier bewusst keine Konzentration auf die Gruppe der *Nutzer* vorgenommen, da diese lediglich *eine* Perspektive auf den Anforderungskatalog widergeben. Neben den potenziellen Nutzern eines Software-Systems müssen auch (indirekt) betroffene Personen berücksichtigt werden. Bei einer Fokussierung lediglich auf die Nutzer würde bei der Evaluation des Systems der *Akzeptanz* eine Gewichtung zukommen, die vielleicht in solch einer Größe nicht gerechtfertigt wäre. Zudem können Personen in verschiedenen *Rollen* Anforderungen an ein System stellen. In der Rolle der Verantwortlichen könnte z.B. das Management andere Anforderungen an das System haben als in der Rolle der Nutzer.

2) Vgl. HOFMANN (2000) S. 6 ff.; KULAK/GUINEY (2001) S. 8 ff.; SOMMERVILLE (2001) S. 100 ff.; SOMMERVILLE/SAWYER (2000) S. 7 f.;

3) Obwohl sich diese Unterteilung zu einem Großteil in der Literatur durchgesetzt hat, sind damit auch verschiedene Probleme verbunden. Die Zugehörigkeit einer Anforderung zu einer der beiden genannten Kategorien ist oft abhängig von der *Perspektive*, die eingenommen wird (vgl. POHL (1996) S. 4). Wird beispielsweise die *Sicherheit* eines Systems gefordert, so könnte dies als nicht-funktional aufgenommen werden, da diese Eigenschaft unabhängig von den intendierten Anwendungen gegeben sein sollte. Im Fall der Entwicklung eines Systems zum Management von Kompetenzprofilen weist die Sicherheit allerdings hohe funktionale Aspekte auf. Es muss beispielsweise bei der Modellierung von Anwendungsfällen (vgl. Kapitel 4.1.2) berücksichtigt werden, welchen Nutzerprofilen welche Rechte zugewiesen werden.

4) Vgl. PARTSCH (1998) S. 17.

5) Vgl. POHL (1996) S. 3.

6) Vgl. POHL (1996) S. 16 ff.

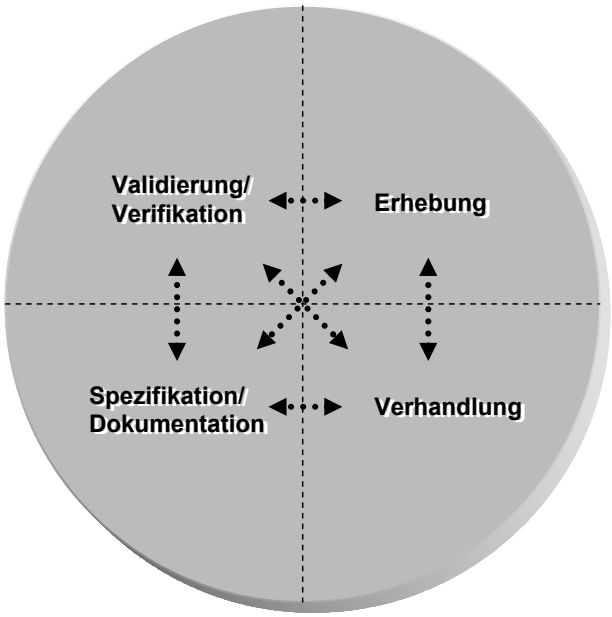

Abbildung 2: Phasenmodell des Requirements Engineering[1)]

- Die erste Phase entspricht der *Erhebung von Anforderungen*. Um in den späteren Phasen über die Anforderungen diskutieren zu können, muss das Wissen der Personen über ihre Anforderungen an das System akquiriert[2)] werden. Es existieren unterschiedliche Methoden, die hierzu verwendet werden können[3)].
- In der zweiten Phase des Requirements Engineering wird zwischen den Personen über die erhobenen Anforderungen *verhandelt*. Die Verhandlung wird dann notwendig, wenn Konflikte zwischen den erhobenen Anforderungen existieren. Um eine Entscheidung treffen zu können, welche Anforderung realisiert werden soll, muss eine Bewertung der Anforderungen erfolgen. Hierfür stehen unterschiedliche Methoden zur Verfügung, durch die eine Priorisierung von Anforderungen erreicht werden kann.
- In der Phase der *Spezifikation* müssen die Anforderungen, deren Erfüllung zuvor beschlossen wurde, in einer Form notiert werden, die für alle betroffenen Personen verständlich ist. Verfolgt werden hiermit zwei Ziele: Zum

1) Quelle: modifiziert übernommen aus: POHL (1996) S. 18.

2) Die Begriffe Wissens*akquisition* und Wissens*erhebung* werden im Folgenden synonym verwendet.

3) Vgl. *Projektbericht 3/2001* für eine Analyse unterschiedlicher Methoden, die für die Wissensakquisition in Frage kommen. Die dort vorgestellten Methoden wurden innerhalb des KOWIEN-Projektes unter anderem hinsichtlich ihrer Eignung für die Akquisition von *Wissen über Kompetenzen* bewertet. Aus der Analyse lassen sich aber auch Gestaltungsempfehlungen für die Akquisition von Wissen über Anforderungen im Rahmen des Requirements Engineering geben.

einen wird versucht, mittels einer einheitlichen Notation, die für alle Personen verständlich ist, weitere Anforderungen herauszuarbeiten. Begünstigt wird dies durch Assoziationskontexte, die bei der Lektüre der Spezifikation geschaffen werden können. Zum anderen ist eine einheitliche Spezifikation notwendig, um in der letzten Phase der Validierung/ Verifikation die Erfüllung der dokumentierten Anforderungen zu überprüfen. Um die Güte einer (Anforderungs-)Spezifikation beurteilen zu können, muss sie selbst Anforderungen genügen, die in Kapitel 3 behandelt werden.

- In der vierten Phase muss der Anforderungskatalog auf seine *Validität* überprüft und mit der beabsichtigten Implementierung *verifiziert* werden. Die Validität des Anforderungskatalogs gibt Aufschluss über die Frage, ob die beabsichtigte Implementierung einen Beitrag zu der Problemlösung leistet, die ursprünglich gefordert wurde. Die Verifikation dient wiederum dazu, zu überprüfen, ob die Anforderungsspezifikation sich mit der gewünschten Implementierung deckt.

Wie bereits in Abbildung 2 verdeutlicht, können die Phasen des Requirements Engineering nicht unabhängig voneinander betrachtet werden. Sie sind gekennzeichnet durch ein gegenseitiges Beziehungsgeflecht. Teilweise kann sich dabei die Durchführung der Phasen überschneiden.

Im Kontext der Konstruktion von *Anwendungsfällen* werden unterschiedliche Begriffsdefinitionen notwendig. Ein Anwendungsfall (*use case*) ist die Beschreibung des Verhaltens eines Systems vom Standpunkt des Anwenders aus[1)]. Er wird stets von einem *Akteur* in Gang gesetzt. Ein Anwendungsfall hat einen Namen, einen Basisablauf und möglicherweise mehrere alternative Abläufe (Sequenz von Schritten), d.h., dass man ihn auch als eine Menge von Abläufen definieren kann, die durch ein gemeinsames Benutzerziel miteinander verbunden sind.[2] Anwendungsfälle können z.B. mittels der *Unified Modeling Language* (UML) dargestellt werden[3)]. Durch die Spezifizierung von Anwendungsfällen wird nicht vorgeschrieben, *wie* die Anforderungen an das System umgesetzt werden sollen.

---

1) Vgl. KULAK/GUINEY (2001) S. 35 ff.;

2) Einige Autoren verwenden an Stelle des Begriffs *Ablauf* den Begriff *Szenario* (vgl. KAHLBRANDT (2001) S. 106).

3) Vgl. Kapitel 3.2.

Leider wird der Begriff „Anwendungsfall“ nicht nur im Rahmen der Anwendungsfallanalyse verwendet, sondern auch zur Bezeichnung von *Geschäftsvorfällen*, was oft zu Verwirrungen führt. Hier wird unter einem Geschäftsvorfall die natürlichsprachliche Beschreibung der Situationen, in denen die zu erstellende Anwendung durch einen potenziellen Nutzer benötigt werden könnte, verstanden. Das, was der Kunde beschreiben soll, um dem Systemanalytiker seine Anforderungen mitzuteilen, sind keine formalen Anwendungsfälle, sondern die Geschäftsvorfälle, die von dem zu erstellenden Softwaresystem unterstützt werden sollen. Damit ist gemeint, dass der Kunde die Aufgaben seines Geschäftsalltags beschreiben soll, für die das System erstellt werden soll. Eine verbreitete Methode zur Beschreibung von Geschäftsvorfällen ist die Formulierung von *Anwenderfragen* (das sind Fragen der Anwender, die das System beantworten können soll). Darüberhinaus können aber auch *Geschichten* verwendet werden. Eine Geschichte (engl. *Story*) ist im Zusammenhang mit der Anforderungsbeschreibung des Kunden die Beschreibung einer Alltagssituation des Kunden, die vom System unterstützt werden soll, bzw. die Beschreibung von Problemen des Kunden, die vom System gelöst werden sollen.

Entitäten, die einen Ablauf in Gang bringen oder die von einem Ablauf betroffen sind, werden *Akteure* genannt. Neben menschlichen Benutzern können auch externe Systeme oder eine Systemuhr Akteure sein. Ein Akteur initiiert einen Anwendungsfall (d.h. führt den ersten Schritt einer Sequenz von Schritten aus) und ein Akteur (eventuell, aber nicht notwendigerweise der initiierende) empfängt von dem Anwendungsfall etwas von Wert. Im *Anwendungsfalldiagramm*, einem der Diagrammtypen der UML (s. 3.2), werden die initiierenden Akteure links und die empfangenden Akteure rechts dargestellt.

3 Spezifikation der Anforderungen

3.1 Anforderungen an den Anforderungskatalog

Das Resultat der Anforderungsspezifikation ist der *Anforderungskatalog* (Pflichtenheft, Lastenheft, Produktdefinition). Er dient als Kommunikationsmedium zwischen dem Entwickler und dem zukünftigen Nutzer des Systems. Damit die durch

den Anforderungskatalog angestrebten Funktionen in höchstmöglicher Form realisiert werden können, muss er selber einer Anzahl von Anforderungen genügen[1)].

In der Phase der Spezifikation des Anforderungskataloges gilt es, sämtliche als relevant erachteten Anforderungen an das Software-System in einer Form zu dokumentieren, die von allen Betroffenen verstanden werden kann. Mit der Nutzung der *natürlichen Sprache* als Grundlage für diese Spezifikation sind verschiedene Probleme verbunden. Beispielsweise sind Aussagen, die in natürlicher Sprache getroffen werden, oft durch *Vagheit* und *Mehrdeutigkeit* gekennzeichnet. Als vage gelten Aussagen dann, wenn ihr Wahrheitsgehalt nicht eindeutig bestimmt werden kann. Mehrdeutige Aussagen sind teilweise dann vorhanden, wenn Begrifflichkeiten nicht eindeutig definiert wurden. Der synonyme oder homonyme Gebrauch von Begriffen ist hiervon ein Spezialfall. Um die Eignung einer Anforderungsspezifikation für unterschiedliche Zwecke zu gewährleisten, werden deswegen im Folgenden Anforderungen untersucht, die von einer Anforderungsspezifikation erfüllt werden sollten.

Die *Korrektheit* der Anforderungsspezifikation wird gewahrt, indem nur Anforderungen berücksichtigt werden, die mindestens von einer entweder mittelbar oder unmittelbar von der Anwendung betroffenen Person geäußert oder akzeptiert wurden. Hierzu werden auch Anforderungen gezählt, die sich *logisch* aus den bereits vorhandenen Anforderungen ableiten lassen.

Die *Relevanz* des Anforderungskataloges ist dann hoch, wenn die enthaltenen Anforderungen sinnvoll zu der Problemlösung beitragen. Durch die unnötige Gewichtung von Anforderungen, die keinen wesentlichen Beitrag zu der Problemlösung leisten, kann es zu einer *Verzerrung* des Anforderungskataloges kommen.

Damit die *Konsistenz* der Anforderungsspezifikation erhalten bleibt, muss in der Phase vor der Spezifikation darüber verhandelt werden, wie eventuell vorhandene Widersprüche zwischen einzelnen Anforderungen behoben werden können. Zu unterscheiden bleibt hier zwischen der *internen* und der *externen* Konsistenz einer Anforderungsspezifikation. Während die interne Konsistenz auf die Widerspruchsfreiheit innerhalb der Anforderungsspezifikation ausgerichtet ist, ist die

1) Vgl. HOFMANN (2000) S. 9 ff.; MELCHISEDECH (2000) S. 22 ff..

externe Konsistenz abhängig von Dokumenten, die auch für das zu erstellende System von Relevanz sind.

Die *Verständlichkeit* der Anforderungsspezifikation ist dann hoch, wenn Personen, die von letzterer betroffen sind, ihre Bedeutung schnell erfassen können. Die Verständlichkeit der Anforderungsspezifikation korreliert zudem positiv mit der *Überprüfbarkeit* der Anforderungsspezifikation.

3.2 Spezifikation von Anforderungen mittels der UML

Der Teilprozess der Spezifikation von Anforderungen hat im KOWIEN-Projekt eine zweifache Bedeutung und wird deshalb im Folgenden explizit angesprochen. Zum einen ergibt sich aus der Aufgabenstellung im KOWIEN-Projekt eine besondere Konzentration für diesen Punkt aus einem „Selbstzweck". Um einen Anforderungskatalog zu erstellen, der ein exaktes Abbild der Anforderungen ergibt, ist es notwendig, einen Formalismus einzuführen, der die oben genannten Anforderungen an den Anforderungskatalog unterstützt. Zum andern ergibt sich ein Forschungsinteresse bei diesem Thema, da die Überwindung von *Mehrdeutigkeiten* bei der Kommunikation einem Ziel entspricht, dass durch Ontologien zu erreichen versucht wird. Zwar ist der Einsatz von Ontologien im Rahmen des Requirements Engineerings noch nicht ausführlich in der Literatur behandelt worden, dennoch wird seitens der Autoren hierin ein hohes Potenzial gesehen[1)].

Die Formalisierung von Anforderungskatalogen ist mit einem Trade-off verbunden, der sich nicht auf triviale Weise überwinden lässt: Während die Formalisierung einen hohen Beitrag dazu leistet, Eigenschaften wie die Konsistenz und die Eindeutigkeit der Spezifikation zu überprüfen, leidet darunter ihre Verständlichkeit. Um die Vorteile einer präzisen Formalisierung mit der Nachvollziehbarkeit für Dritte zu verbinden, wird in dieser Arbeit eine semi-formale Darstellung der Anforderungen anhand der *Unified Modeling Language* (*UML*) vorgestellt[2)]. Zu-

1) Erste Anzeichen für die Verwendung von UML in Verbindung mit Ontologien haben sich im Rahmen der Repräsentation von Ontologien ergeben. Beispielsweise wird in CRANEFIELD (2001) die Möglichkeit diskutiert, die im Folgenden vorgestellte Sprache UML (insbesondere Klassendiagramme) zur Repräsentation von Ontologien zu verwenden.

2) Vgl. JACOBSON ET AL. (1992)
Ein Überblick über verschiedene Sprachen, die sich zur Repräsentation von (Anforderungs-)Wissen eignen, wurde im KOWIEN-Projekt bereits im Projektbericht 1/2002 geleistet. Aus diesem Grund wird hier nicht mehr auf weitere Formalismen eingegangen.

dem zeichnet sich UML als De-facto-Standard für die Modellierung von - zumindest objektorientierten - Systemen ab[1)].

Die UML ist ein Formalismus zur graphischen Beschreibung von Informationssystemen. Hierfür stellt die UML vier Diagrammtypen zur Verfügung. Sie werden im Folgenden mit absteigender Priorität für das KOWIEN-Projekt vorgestellt:

- *Anwendungsfalldiagramme*

  Anwendungsfalldiagramme dienen dazu, die verschiedenen Anwendungsfälle einer Anwendung, ihre Beziehungen untereinander und ihre Beziehungen zu den Akteuren des Systems darzustellen. Die initiierenden Akteure werden links und die empfangenden Akteure rechts dargestellt. Die Abläufe der Anwendungsfälle werden in diesem Diagramm nicht aufgeführt. Abbildung 3 gibt die exemplarische Visualisierung von zwei Anwendungsfällen wider, die in Kapitel 4.1.2.2 weiter erläutert werden.

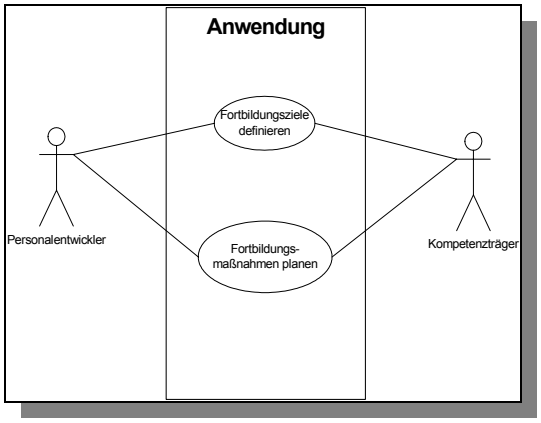

Abbildung 3: Exemplarisches Anwendungsfalldiagramm

- *Klassendiagramme*

  Klassendiagramme werden verwendet, um den Analytikern ein ganzheitliches Bild über die Struktur der Anwendung zu verschaffen. Um Methoden, Eigenschaften und Operationen für eine größere Menge von Elementen des modellierten Realitätsausschnitts zu bestimmen, werden gleichartige Objekte zu *Klassen* zusammengefasst. Anstatt Klasse wird auch der Begriff *Typ* verwendet[2)]. Die statische Beziehung zwischen Klassen wird in Klas-

1) Vgl. FOWLER/SCOTT (2000) S. 2 ff.; OESTEREICH (1998) S. 19 ff.; PARTSCH (1998) S. 249.

2) Vgl. OESTEREICH (1998) S. 223.

sendiagrammen repräsentiert. Zur Repräsentation der statischen Struktur stehen in UML zwei *Relationstypen* zur Verfügung: Die *taxonomische Relation* gibt für eine Klasse alle Subklassen an. Damit kann beispielsweise die Klassenbeziehung zwischen der Klasse aller Kompetenzen und der Klasse der Sozialkompetenzen modelliert werden. Die *nicht-taxonomischen* Beziehungen entsprechen sonstigen Beziehungen zwischen Klassen. Darunter fällt beispielsweise die Relation „arbeitet_für", die die Klasse aller Mitarbeiter mit der Klasse aller Projekte verbindet.

- *Verhaltensdiagramme*
  - *Sequenzdiagramme*

    Sequenzdiagramme repräsentieren die Menge aller Nachrichten, die eine ausgewählte Menge von Objekten oder Klassen miteinander austauschen. Dabei steht der zeitliche Verlauf der Nachrichten, die ausgetauscht werden, im Vordergrund.
  - *Kollaborationsdiagramme*

    Das Kollaborationsdiagramm kann für die gleichen Verwendungszwecke wie das Sequenzdiagramm verwendet werden. Der repräsentierte Nachrichtenaustausch wird hierbei allerdings aus der Perspektive der Objekte modelliert. Sequenzdiagramme und Kollaborationsdiagramme werden gemeinsam zu den *Interaktionsdiagrammen* gezählt[1)].
  - *Zustandsdiagramme*

    Die Folge von Zuständen, die Objekte in der Anwendung einnehmen können, werden durch Zustandsdiagramme repräsentiert. Sie können eingesetzt werden, um das Verhalten von Objekten über mehrere Anwendungsfälle hinweg zu beschreiben.
  - *Aktivitätsdiagramme*

    Aktivitätsdiagramme sind eine spezielle Form von Zustandsdiagrammen. Sie sind beschränkt auf solche Zustände, die durch die Ausführung einer Aktivität gekennzeichnet sind.

1) vgl. FOWLER/SCOTT (2000) S. 64; KAHLBRANDT (2001) S. 106.

- *Implementierungsdiagramme*
  - *Komponentendiagramme*

    Im Komponentendiagramm werden die Komponenten des Systems und ihre Abhängigkeiten untereinander repräsentiert. Unter Komponenten sind physische Quelltextmodule zu verstehen.
  - *Einsatzdiagramme*

    Einsatzdiagramme repräsentieren die Einsatzumgebung (Prozessoren, Computer etc.) der Anwendung. Sie beschreiben unter anderem die Konfiguration und die Kommunikationsbeziehungen zwischen den Objekten der Einsatzumgebung.

Bei der Spezifizierung von Anforderungen an die Anwendung mit der UML, empfiehlt sich ein Vorgehen, bei dem die einzelnen Schritte einen steigenden Formalisierungsgrad aufweisen[1)]. So kann in einem ersten Schritt die natürlichsprachliche Beschreibungen von Arbeitsabläufen bzw. Geschäftsvorfällen, die von der Anwendung unterstützt werden sollen, dazu verwendet werden, um daraus Anwendungsfälle abzuleiten. Die Repräsentation dieser Anwendungsfälle kann mittels Anwendungsfalldiagrammen erfolgen. Im Anschluss daran können Klassen-, Verhaltens- und Implementierungsdiagramme erstellt werden, die die operationale Funktionsweise der Anwendung näher beschreiben.

4 Anforderungen an den Prototypen

Im Folgenden werden die Anforderungen vorgestellt, die von den Partnern innerhalb des KOWIEN-Projektes erarbeitet wurden. Unterschieden wird hier - entsprechend der Systematisierung in Kapitel 2 - zwischen funktionalen und nichtfunktionalen Anforderungen.

4.1 Funktionale Anforderungen an den KOWIEN-Prototypen

Die Praxispartner wurden dazu aufgefordert, die Geschäftsvorfälle zu beschreiben, die vom KOWIEN-Prototypen unterstützt werden sollen. Dabei wurde es Ihnen freigestellt, in welcher Form sie diese *Geschäftsvorfälle* beschreiben, ob durch die Angabe von *Anwenderfragen*, die das System beantworten können soll, oder durch das Aufschreiben von *Geschichten* (s. S. 11). Im ersten Unterabschnitt

1) Vgl. PARTSCH (1998) S. 275 ff.

dieses Abschnitts werden die von den Praxispartnern beschriebenen Geschäftsvorfälle aufgeführt. Im darauf folgenden Unterabschnitt werden dann aus den Geschäftsvorfällen Anwendungsfälle abgeleitet, die vom KOWIEN-Prototypen zu unterstützen sind. Daneben werden auch einige weitere Anwendungsfälle beschrieben, die sich zwar nicht direkt aus den Geschäftsvorfällen der Praxispartner ergeben, die aber bei einem Arbeitstreffen der an der Anforderungsanalyse beteiligten Personen des Instituts PIM und der Comma Soft AG unter Berücksichtigung des Projektzieles (ontologiebasiertes Management von Kompetenzen) erarbeitet worden sind.

4.1.1 Beschreibung von Geschäftsvorfällen

Die Praxispartner haben bei der Leistung Ihres Beitrags zur Erstellung eines Anforderungskatalogs für den KOWIEN-Prototypen größtenteils auf die Methode der Formulierung von Fragen zurückgegriffen, die für die Anwendung potenziell von Relevanz sein könnten. Diese Methode, die in der Literatur unter dem Namen *Goal-Question-Metric Ansatz*[1)] bekannt ist - führt zu Fragen, die später verschiedenen Kategorien zugeordnet werden können.

Die Anwenderfragen erlangen auch im Kontext der Konstruktion der KOWIEN-Ontologie(n) eine besondere Bedeutung. GRÜNINGER/FOX empfehlen vor der Erstellung einer Ontologie so genannte *competency questions* zu formulieren, deren Beantwortung auf der Basis der zu erstellenden Ontologie möglich sein soll[2)]. Die Erstellung eines solchen Fragenkataloges soll dazu beitragen, die *funktionale Vollständigkeit* der Ontologie zu erreichen[3)].

Competency Questions leisten in zweifacher Weise einen Beitrag zur Entwicklung einer Ontologie, die die intendierten Anwendungen unterstützt: Zum einen kann aus der *syntaktischen Struktur* der Fragen abgeleitet werden, welche Begrifflichkeiten in der Ontologie enthalten sein sollten. Beispielsweise lässt sich aus der potenziellen Frage „Welche Weiterbildungsmaßnahmen hat der Mitarbeiter besucht?“ ableiten, dass ein Bedarf nach einer Relation *besuchte_Weiterbildung* e-

---

1) Vgl. BALZERT (1998) S. 263 ff. Es wurde in KOWIEN von einer Quantifizierung der Kennzahlen, die für die Fragestellungen von Relevanz sein können, abgesehen, da dies in der noch frühen Projektphase nicht angebracht schien. In späteren Versionen dieses Projektberichtes wird das u.U. nachgeholt.

2) Vgl. GRÜNINGER/FOX (1994).

3) Vgl. S. 32.

xistiert, dessen Typ die Menge aller Mitarbeiter mit der Menge aller Weiterbildungsmaßnahmen verbindet. Zum anderen kann aus dem *semantischen Inhalt* der Frage auf potenzielle Anwendungsfälle geschlossen werden. Bei der Gestaltung der Anwendungsfälle kann sich erneut ein Bedarf nach Begriffsstrukturierung ergeben.

Die folgende Liste von Anwenderfragen ist eine Zusammenstellung, bei der von den einzelnen Unternehmen abstrahiert worden ist. Das heißt, dass z.B. in der Frage „Welcher externe Projektpartner besitzt komplementäre Kompetenzen zur DMT für eine Projektbearbeitung“ das Wort „DMT“ durch „Unternehmen“ ersetzt worden ist, weil dieser Geschäftsvorfall auch für die anderen Unternehmen von Bedeutung ist.

- Welcher externe Projektpartner besitzt komplementäre Kompetenzen zum Unternehmen für eine Projektbearbeitung?
- Welcher Mitarbeiter hat das geforderte Kompetenzprofil und steht er zur Verfügung?
- Wie sieht der detaillierte Lebenslauf / berufliche Werdegang des Mitarbeiters aus?
- Wer hat erforderliches Experten-/Spezialistenwissen auf dem notwendigen Level?
- Wo sind im Hinblick auf ein bestimmtes Projekt Skill-Gaps bei Mitarbeitern? Wie groß sind diese?
- Welcher Mitarbeiter hat in einem ähnlichen Projekt früher erfolgreich mitgearbeitet? Welche Funktion / Rolle hat er eingenommen?
- Welche Referenzen hat das Unternehmen zu bestimmten Themenfeldern?
- Welcher Mitarbeiter hat mit einer bestimmten Firma Kontakt?
- Welche Referenzen/Kompetenzen hat eine Division/Unit/Tochterfirma/Beteiligung des Unternehmens?
- Welchen Service/Zuarbeit kann ich von den Zentralbereichen bei der Vorbereitung und Bearbeitung meiner Projekte (von der Akquisition bis zur Schlussabrechnung) erwarten und einfordern?

Diese zehn Anwenderfragen sind ein repräsentativer Ausschnitt der eingegangen Fragen. Die KSM GmbH hat daneben auch noch eine Geschichte eingebracht, die ein häufig auftretendes Problem beschreibt, nämlich die Schwierigkeit, einen kompetenten Ansprechpartner in einem Partnerunternehmen zu finden und das Wissen über die Kompetenzen festzuhalten und zu verteilen. Aus Platzgründen wird diese Geschichte im Anhang zitiert.

4.1.2 Auflistung von Akteuren und Anwendungsfällen

Um die funktionalen Anforderungen an die Anwendung zu beschreiben, wenden wir die Anwendungsfallanalyse an, die als Bestandteil der UML entwickelt worden ist (s. 3.2). Im Rahmen dieser Analyse werden zunächst die Akteure des zu entwickelnden Systems bestimmt. Daher beginnt dieser Unterabschnitt mit einer Beschreibung der identifizierten Akteure. Im Anschluss daran werden die Anwendungsfälle aufgeführt, in denen diese Akteure vorkommen.

4.1.2.1 Akteure

Ein *Akteur* entspricht der Rolle, die ein Benutzer in Bezug auf die Anwendung einnimmt[1)]. Es kann sich dabei sowohl um Personen als auch um andere Anwendungen handeln. Akteure können in unterschiedlichen *Anwendungsfällen*[2)] auftauchen. Andererseits kann ein Anwendungsfall von mehreren Akteuren durchgeführt werden. Die Beschreibung eines Akteurs beinhaltet neben dem Namen des Akteurs hauptsächlich die Aufgaben, die der Akteur in dem System übernimmt bzw. die Tätigkeiten, die der Akteur mit dem System ausführen möchte. Darüber hinaus kann die Beschreibung auch die Verantwortlichkeit und die Rechte des Akteurs im System umfassen.

Selbsteinschätzer
Ein Selbsteinschätzer ist ein Mitarbeiter des Unternehmens, der seine Kompetenzen (aus Eigeninitiative oder auf Nachfrage) selbst einschätzt.

1) Vgl. FOWLER/SCOTT (2000) S. 37; KAHLBRANDT (2001) S. 192.

2) Vgl. S. 10f.

Fremdeinschätzer
Während der Selbsteinschätzer seine eigenen Kompetenzen beurteilt, werden durch den Fremdeinschätzer die Kompetenzen anderer eingeschätzt. Es kann sich dabei sowohl um Mitarbeiter des Beurteilten als auch um seine Vorgesetzten handeln.

Kompetenzträger
Der Begriff des Kompetenzträgers umfasst sowohl die Mitarbeiter des Unternehmens, als auch das Unternehmen selbst. Während das erste Begriffsverständnis auf personale Kompetenzen ausgerichtet ist, umfasst zweiteres organisationale Kompetenzen.

Kompetenzfragensteller
Ein Kompetenzfragensteller stellt zu einem bestimmten Zweck Fragen an das System bezüglich der Kompetenz der Mitarbeiter des Unternehmens. Dabei kann es sich z.B. um einen Mitarbeiter handeln, der zur Lösung eines konkreten Problems nach einem Kollegen sucht, der über Kenntnisse verfügt, die für die Problemlösung wichtig sind.

Personalentwickler
Ein Personalentwickler ist ein Mitarbeiter der Personalabteilung, der u.a. die Fortbildungsziele definiert. Er übernimmt daher u.a. Funktionen, die üblicherweise vom Leiter der Personalabteilung ausgeübt werden.

Projektteamkonfigurator
Dieser Akteur erfüllt die Aufgabe, basierend auf den Kompetenzanforderungen eines konkreten Projektes und den Kompetenzen der verfügbaren Mitarbeiter ein Projektteam zusammenzustellen. Das Projektteam kann entweder von jemandem zusammengestellt werden, der den Überblick über das verfügbare Personal hat (z.B. jemand aus der Personalabteilung), aber selbst nicht beim Projekt mitarbeiten wird, oder von dem zukünftigen Projektleiter.

Kooperationsexperte

Der Kooperationsexperte kümmert sich um die Vorbereitung und Betreuung der Kooperation mit externen Unternehmenspartnern. Eine seiner Aufgaben ist z.B. die Erfassung der verschiedenen Ansprechpartner der Partner zu bestimmten Themen.

Ontologie-Administrator

Der Ontologie-Administrator ist für den Inhalt der Ontologie verantwortlich und entscheidet über die Aufnahme neuer Begriffe, die von ihm selbst oder von anderen vorgeschlagen werden, sowie über das Löschen von Begriffen aus der Ontologie. Es ist dabei zu beachten, dass es als konstitutiv für Ontologien angesehen wird, dass sie das gemeinsame Verständnis mehrerer Personen der enthaltenen Begriffe widerspiegeln. Die Funktion des Ontologie-Administrators ist vergleichbar mit der des *Wissensingenieurs* im Knowledge Engineering. Um nicht den Anschein zu erwecken, die Tätigkeiten des Ontologie-Administrators seien rein technischer Natur, wird er hier vom Wissensingenieur abgegrenzt. Die Funktionen des Ontologie-Adminstrators erfordern neben technischen auch linguistische Fähigkeiten.

Kompetenzenrechercheur

Ein Kompetenzenrechercheur hat die Aufgabe, neue Kompetenzen zu recherchieren und dem Ontologie-Administrator zur Aufnahme in die Kompetenzontologie vorzuschlagen, d.h. er sorgt dafür, dass neue Kompetenzen (z.B. eine neue Programmiersprache) in das System gelangen (und evtl. auch, dass veraltete Kompetenzen aus dem System gelöscht werden).

KOWIEN-Anwendungsbetreiber
Der KOWIEN-Anwendungsbetreiber ist für den reibungslosen Betrieb und das Controlling der KOWIEN-Anwendung zuständig. Seine Kompetenzen liegen im Bereich der Informationstechnik. Welche zusätzlichen Eigenschaften von dem KOWIEN-Anwendungsbetreiber noch eingefordert werden sollten, wird bestimmbar sein, wenn sämtliche Funktionalitäten der KOWIEN-Anwendung spezifiziert wurden.

Die Liste der Akteure ist ebenso wie die nachfolgende Liste der Anwendungsfälle als vorläufig zu betrachten, da es möglich ist, dass sich im Laufe des zweiten Projektjahres herausstellt, dass noch weitere Akteure mit dem System interagieren sollen oder dass es sinnvoll ist, zwei Akteure mit sehr speziellen Aufgaben/ Tätigkeiten zu einem Akteur mit weniger speziellen Aufgaben/ Tätigkeiten zusammenzufassen.

4.1.2.2 Anwendungsfälle

Die Beschreibung eines Anwendungsfalls beinhaltet neben einer Auflistung der Akteure, die den Anwendungsfall in Gang setzen (initiierende Akteure) oder die in irgendeiner Weise von dem Anwendungsfall betroffen sind (empfangende Akteure), eine Kurzbeschreibung und u.U. einen Basisablauf[1]. Optional kann sie auch noch mehrere alternative Abläufe, sowie Vorbedingungen und Nachbedingungen umfassen. Letztere sind vor allen Dingen in der Phase der Qualitätssicherung von Bedeutung, wenn es darum geht, aus den Anwendungsfällen Testfälle abzuleiten.

1) In späteren Versionen des Projektberichts werden die Basisabläufe der Anwendungsfälle ergänzt.

Eigene Kompetenzen beschreiben
Akteur: Selbsteinschätzer
Ein neuer Mitarbeiter soll dazu aufgefordert werden, mit Hilfe der Begriffe der Kompetenzontologie seine Kompetenzen zu beschreiben (Art und Ausprägung). Außerdem soll jeder Mitarbeiter die Möglichkeit haben, in Eigeninitiative die Aussagen über seine Kompetenzen auf dem neuesten Stand zu halten bzw. entsprechende Änderungsvorschläge (bezüglich neu zuzuweisender Kompetenzen) zu machen.

Feedback zu Kompetenzfortschritt geben
Akteur: Selbsteinschätzer
Wenn ein Mitarbeiter an einem Projekt oder an einer Fortbildungsmaßnahme teilgenommen hat, soll er danach vom System um eine Einschätzung gebeten werden, inwieweit sich das Projekt bzw. die Fortbildungsmaßnahme auf seine Kompetenzen ausgewirkt hat. Das System könnte z.B. basierend auf der Beschreibung des Projekts bzw. der Fortbildungsmaßnahme Kompetenzbegriffe vorschlagen, denen der Mitarbeiter eine bestimmte Ausprägung zuweist.

Kompetenzen anderer Mitarbeiter beschreiben
Akteur: Fremdeinschätzer
Ein entsprechend befugter Mitarbeiter (z.B. der direkte Vorgesetzte oder der Projektleiter) kann, wenn die Realisierung dieses Anwendungsfalls gewünscht ist, die Kompetenzen anderer Mitarbeiter mit Hilfe der Begriffe der Kompetenzontologie beschreiben.

Kompetenzträger suchen
Akteure: Kompetenzfragensteller, Kompetenzträger
Ein Mitarbeiter, der ein konkretes Problem hat, soll die Möglichkeit haben, nach Personen zu suchen, die ihm bei seinem Problem mit ihren Kenntnissen helfen können. Dieser Anwendungsfall wird wahrscheinlich in verschiedenen anderen Anwendungsfällen auftauchen, wie z.B. in dem Anwendungsfall der Bildung eines Projektteams.

Details zu einem Mitarbeiter anzeigen lassen
Akteure: Kompetenzfragensteller, Projektteamkonfigurator, Personalentwickler
Die verschiedenen Akteure sollen entsprechend ihren Rechten die Details zu einem bestimmten Mitarbeiter (wie z.B. seinen Lebeslauf / beruflichen Werdegang) einsehen können.

Kompetenzen des Unternehmens bzw. einer Abteilung ermitteln
Akteure: Kompetenzfragensteller
Es soll möglich sein, die Kompetenzen eines Unternehmens bzw. einer Abteilung zu ermitteln, welche sich u.a. aus den Kompetenzen der zugehörigen Mitarbeiter ableiten lassen.

Ansprechpartner von externen Unternehmenspartnern erfassen
Akteure: Kooperationsexperte
Die Ansprechpartner von externen Unternehmenspartnern zu bestimmten Themen sollen vom System erfasst werden können.

Kompetenzen von potentiellen externen Unternehmenspartnern ermitteln
Akteure: Kooperationsexperte, Projektteamkonfigurator
Es soll ermittelt werden können, welche externen Unternehmenspartner komplementäre Kompetenzen zum Unternehmen für eine Projektbearbeitung besitzen.

Fortbildungsziele definieren
Akteur: Personalentwickler
Die Definition von Fortbildungszielen ist die Grundlage für die Planung von Fortbildungsmaßnahmen.

Fortbildungsmaßnahmen planen
Akteure: Personalentwickler, Kompetenzträger
Das System könnte die Planung von Fortbildungsmaßnahmen basierend auf den Fortbildungszielen und den Kompetenzen der Mitarbeiter unterstützen.

Projektteam zusammenstellen
Akteure: Projektteamkonfigurator, Kompetenzträger
Für ein konkretes Projekt soll basierend auf den Projekt-Anforderungen und den Kompetenzen und der Verfügbarkeit (evtl. auch unter Berücksichtigung der Fortbildungsziele) der Mitarbeiter ein Projektteam zusammengestellt werden.

Skill-Gaps von Mitarbeitern in Bezug auf eine Projektbearbeitung ermitteln
Akteure: Projektteamkonfigurator, Personalentwickler, Kompetenzträger
Im Hinblick auf ein bestimmtes Projekt sollen die Skill-Gaps der für eine Projektbeteiligung in Frage kommenden Mitarbeiter ermittelt werden.

Externe Partner für eine Projektbearbeitung suchen
Akteure: Kooperationsexperte, Projektteamkonfigurator
Für eine gemeinsame Projektbearbeitung sollen externe Kooperationspartner gesucht werden.

Zu einem gegebenen Projekt ähnliche Projekte finden
Akteure: Projektteamkonfigurator
Ausgehend von der Beschreibung eines bestimmten Projektes sollen ähnliche Projekte gefunden werden, um von früher gemachten Erfahrungen zu profitieren und auf erfahrene Mitarbeiter aufmerksam zu werden.

Details zu einem Projekt anzeigen lassen
Akteure: Projektteamkonfigurator
Entsprechend befugte Akteure sollen die Möglichkeit haben, die Details zu einem Projekt einzusehen, z.B. um Mitarbeiter zu ermitteln, die in früheren Projekten eine bestimmte Rolle/ Aufgabe erfolgreich ausgeübt/ ausgeführt haben.

Ontologie importieren
Akteur: Ontologie-Administrator
Es wäre vorstellbar, dem Ontologie-Administrator die Möglichkeit zu geben, eine in einer bestimmten Ontologiebeschreibungssprache (wie z.B. F-Logic oder DAML+OIL) vorliegende Ontologie in das System zu importieren.

Ontologie erstellen und überarbeiten
Akteur: Kompetenzenrechercheur, Ontologie-Administrator
Die Kompetenzenontologie soll vom Ontologie-Administrator von Grund auf neu erstellt und überarbeitet (Begriffe hinzufügen, löschen oder anders zueinander in Beziehung setzen) werden können. Bei diesem Prozess arbeiten die Kompetenzenrechercheure mit, indem sie dem Ontologie-Administrator Vorschläge machen.

Wie schon im Anschluss an die Liste der Akteure gesagt wurde, handelt es sich auch bei der Liste der Anwendungsfälle nicht um ein fertig gestelltes Pflichtenheft, welches nur noch implementiert werden muss. Einige Anwendungsfälle werden sich erst dann präzise beschreiben lassen, wenn das generische Vorgehensmodell erarbeitet worden ist, was Gegenstand des zweiten Projektjahres ist.

4.2 Nicht-funktionale Anforderungen an den KOWIEN-Prototypen

4.2.1 Nicht-funktionale Anforderungen an die Anwendung

Innerhalb der Untersuchung zu den nicht-funktionalen Anwendungen erscheint eine weitere Unterteilung in Abhängigkeit von der *Perspektive* des Anforderungsstellers als nützlich. Eine mögliche Unterteilung kann in Form von Anforderungen aus Anwender- und Entwicklersicht erfolgen[1)]. In Abbildung 4 sind die nicht-funktionalen Anforderungen in Abhängigkeit von der Perspektive des Anforderungsstellers dokumentiert.

1) Vgl. zu ähnlichen Systematisierungen: KAHLBRANDT (2001) S. 45 ff.; RAASCH (1993)S. 21 ff..

Da die Anforderungen nicht unabhängig voneinander betrachtet werden können, werden in Kapitel 4.2.1.3 mögliche Wechselwirkungen zwischen den funktionalen Anforderungen untersucht.

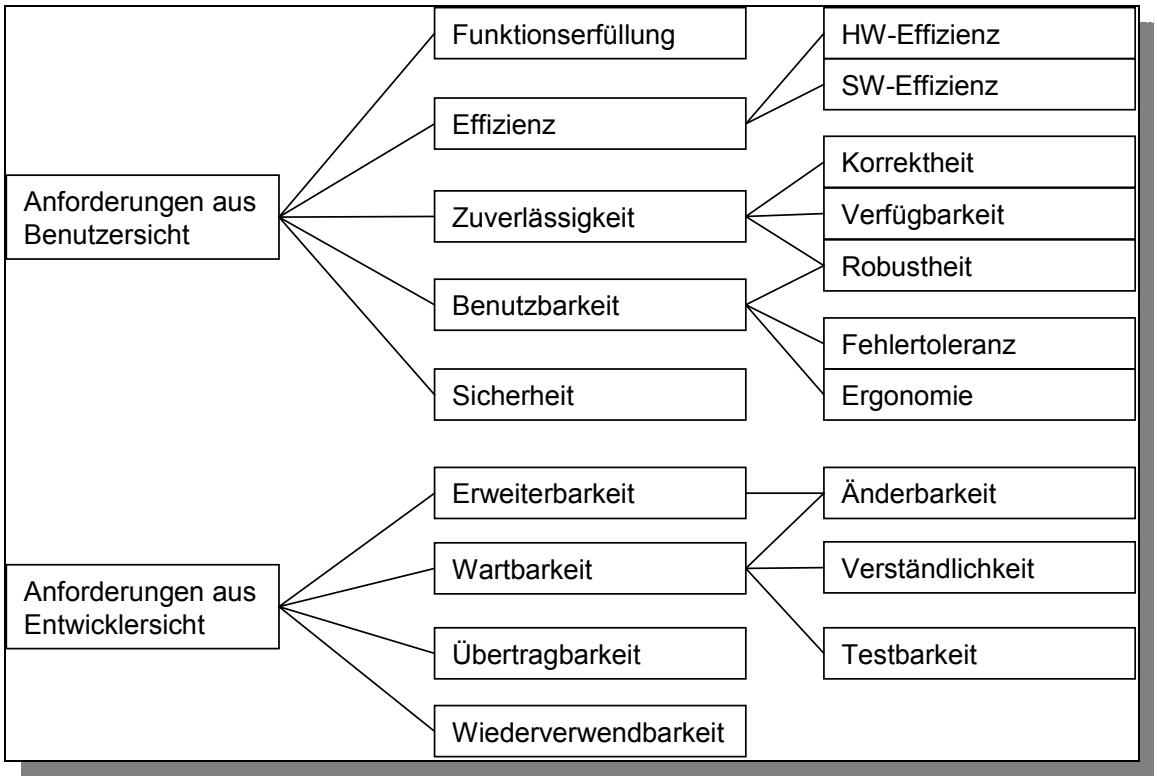

Abbildung 4: Systematisierung nicht-funktionaler Anforderungen[1)]

4.2.1.1 Nicht-funktionale Anforderungen aus Anwendersicht

Die primäre Anforderung der potenziellen Anwender an den KOWIEN-Prototypen ist die *Funktionserfüllung*. Unter der Funktionserfüllung wird der Grad verstanden, in dem die realisierten mit den geplanten Funktionen übereinstimmen[2)]. Die geplanten Funktionen für den KOWIEN-Prototypen wurden bereits im Rahmen der funktionalen Anforderungen in Kapitel 4.1.1 behandelt. Ihre Berechnung erfolgt durch das Verhältnis zwischen der Menge aller aktuellen (Ist-)Funktionen und der Menge aller intendierten (Soll-)Funktionen.

Die *Effizienz* der Anwendung entspricht dem Verhältnis zwischen dem Grad der Funktionserfüllung und dem dabei verursachten Ressourcenverbrauch. Während die *Hardware-Effizienz* durch immer leistungsfähigere Systeme in den Hinter-

1) Quelle: modifiziert und erweitert aus: RAASCH (1993) S. 21.

2) Vgl. RAASCH (1993) S. 23.

grund gerückt ist, hat die *Software-Effizienz*[1] an Bedeutung gewonnen. Wesentliches Merkmal für die Software-Effizienz ist die Antwort- oder Durchlaufzeit, die das System für eine Ausgabe braucht.

Die *Zuverlässigkeit* ist der Grad, in dem die Anwendung die erwarteten Funktionen erbringt, ohne in Zustände zu gelangen, die nicht erwünscht sind. Somit sind die *Korrektheit, Robustheit* und *Verfügbarkeit* Unterfälle der Zuverlässigkeit. Die Korrektheit entspricht dem Verhalten des Systems entsprechend der Anforderungsspezifikation. Sie kann berechnet werden durch das Verhältnis zwischen den fehlerfreien Transaktionen und den gesamten Transaktionen des Systems. Die Robustheit entspricht der Fähigkeit des Systems nach einer fehlerhaften Eingabe[2] einen definierten Zustand überzugehen. Ihr Maß kann berechnet werden durch das Verhältnis zwischen der Anzahl der korrekten Behandlung aller möglichen fehlerhaften Eingaben und der Anzahl aller möglichen fehlerhaften Eingaben. Die Zuverlässigkeit ist die Wahrscheinlichkeit, die Anwendung in einem Zustand anzutreffen, der dessen korrekte Benutzung erlaubt. Ihre Berechnung wird durch das Verhältnis zwischen den durchschnittlichen Zeit zwischen Ausfällen und der Summe der durchschnittlichen Zeit zwischen Ausfällen und der durchschnittlichen Zeit bis zur Wiederinbetriebnahme der Anwendung wiedergegeben.

Unter der *Benutzbarkeit* werden alle Anforderungen aufgezählt, die dem Benutzer ein angenehmes und fehlerfreies Arbeiten mit der Anwendung ermöglichen. Dazu gehören die *Ergonomie*, die *Fehlertoleranz* und die *Robustheit* der Anwendung. Die Ergonomie betrifft die Anpassung zum einen der Hardware und zum anderen der Software an die physiologischen Bedürfnisse des Benutzers. Die Software-Ergonomie kann z.B. durch die Verwendung von *Farben* und die Implementierung von *Fehlertoleranz* des Systems erhöht werden. Die Anwendung gilt dann als fehlertolerant, wenn sie auf fehlerhafte Eingaben mit Meldungen reagiert, die den Benutzer darauf hinweisen und eventuell Hilfestellungen anbieten. Um *Schwellenängste* der Nutzer zu überwinden ist die Fehlertoleranz von hoher Bedeutung. Die Nutzer könnten möglicherweise Angst haben, sie könnten bei der Nutzung des Systems etwas „kaputt“ machen. Entsprechend sollte die Anwen-

---

1) Die Software-Effizienz wird in der Literatur teilweise auch als *Performanz* gekennzeichnet. Vgl. RAASCH (1993) S. 24.

2) Die Eingabe wird hier nicht beschränkt auf Eingaben, die durch die Benutzer getätigt werden. Es kann sich dabei auch um *Nachrichten* handeln, die Module des Systems untereinander austauschen.

dung eine Ausgabe liefern, die den Nutzer auf eine möglicherweise falsche Eingabe hinweist und die Gründe dafür liefert.

Grafische Benutzeroberflächen sind ein weiteres Beispiel dafür, wie die Arbeit mit der Anwendung für den Nutzer angenehmer gestaltet werden kann. Um die Akzeptanz des Nutzers durch die Benutzeroberfläche zu erhöhen, muss sie verschiedene Faktoren aufweisen. Sie sollte zunächst einheitlich gestaltet sein, damit der Nutzer sich nicht bei unterschiedlichen Modulen des Systems auf eine neue Umgebung einstellen muss. Die Gestaltung der Benutzerschnittstellen sollte entsprechend durchgängig sein. Es wäre im Weiteren wünschenswert, die Anwendung auf Nutzerprofile umstellen zu können. In den Nutzerprofilen sollten neben Sicherheitsbestimmungen auch Anpassungen der Anwendung an den Kenntnisstand des Nutzers möglich sein. Die Erfüllung der genannten Anforderungen zur Umsetzung der Software-Ergonomie kann größtenteils durch den Rückgriff auf bereits bewährte *Standardfunktionen* realisiert werden. Darüber hinaus sollte ein umfangreiches Hilfesystem dem Anwender bei der Nutzung des Systems zur Seite stehen. Es haben sich dabei Hilfesysteme bewährt, die nicht nur in Papierform vorliegen sondern auch in die Anwendung integriert werden.

Die *Sicherheit* hat bei der Konstruktion des KOWIEN-Prototypen eine besondere Bedeutung, da hierbei Transaktionen mit teilweise sensitivem Wissen über Kompetenzen durchgeführt werden. Durch die Einbindung von Zugriffsrechten in die Rollenspezifikation sollten Kontrollmechanismen ermöglicht werden. Die Gewährleistung der Sicherheit ist dabei sowohl unternehmensintern als auch -extern zu beachten. Unternehmensintern ist eine Einschränkung einzuführen, dass nicht jeder das Kompetenzprofil abrufen kann. Das lässt sich am besten realisieren, wenn die Verweise auf gesuchte Informationen bei fehlender Befugnis gar nicht erst angezeigt werden. Entsprechend dem Benutzerprofil und den darin angegebenen Rechten des Benutzers hat das System lediglich Verweise auf Informationen anzuzeigen, über die der Benutzer auch verfügen darf. Eine Staffelung der Zugriffsrechte kann in einer kaskadenförmigen Unterteilung nach ihren Zugriffsrechtearten erfolgen, die für jedes Benutzerprofil bestimmt werden müssen[1)].

1) Vgl. GEBERT (2001) S. 15.

4.2.1.2 Nicht-funktionale Anforderungen aus Entwicklersicht

Aus der Perspektive der Entwickler der KOWIEN-Anwendung ergeben sich die Anforderungen *Erweiterbarkeit*, *Wartbarkeit*, *Übertragbarkeit* und *Wiederverwendbarkeit*.

Unter *Erweiterbarkeit* wird die Eigenschaft der Anwendung verstanden, die Implementierung neuer Funktionalitäten zuzulassen, ohne dabei wesentliche Eingriffe in die bereits bestehende Struktur durchführen zu müssen. Wenn zum Beispiel neue Funktionen aus dem Kontext des Managements von Wissen über Kompetenzen erforderlich werden, so sollte das Datenmodell der Anwendung die Implementierung der Funktionen zulassen, ohne eine radikale Umstrukturierung zu erfordern. Das objektorientierte Paradigma bei der Entwicklung komplexer Software hat sich dafür als besonders geeignet erwiesen.

Durch die Modularisierung des Codes wird ebenso die *Wartbarkeit* der Anwendung erhöht. Sie setzt sich zusammen aus der Änderbarkeit und der Korrigierbarkeit. Während die Änderbarkeit auf das Maß für den Aufwand zur Anpassung der Anwendung auf neue Anforderungen ausgerichtet ist, beinhaltet die Korrigierbarkeit das Maß für den Aufwand zur Korrektur von Abweichungen zwischen Ist- und Soll-Funktionen.

Unter der *Übertragbarkeit* (Portabilität) wird das Maß für den Aufwand verstanden, der nötig ist, um die Anwendung von einer technischen Umgebung in eine andere zu überführen. Die *Wiederverwendbarkeit* von Modulen ist dann hoch wenn, sie in anderen Anwendungsumgebungen eingesetzt werden können als der derzeitigen.

Für die Umsetzung der Anforderungen aus Entwicklersicht wurde innerhalb des KOWIEN-Projektes beschlossen, auf das bereits bewährte Produkt der Comma soft infonea zurückzugreifen.

4.2.1.3 Wechselwirkungen zwischen nicht-funktionalen Anforderungen

Die zuvor aufgezeigten nicht-funktionalen Anforderungen stehen oft in einem Verhältnis zueinander. Manche der Anforderungen weisen untereinander ein positives Verhältnis auf, während andere in einem negativen Verhältnis zueinander stehen. In Tabelle 1 ist ein Überblick über mögliche Korrelationen zwischen dem

Erfüllungsgrad der einzelnen Anforderungen aus den Kapiteln 4.2.1.1 und 4.2.1.2 wiedergegeben.

	Funktionserfüllung	Effizienz	Zuverlässigkeit	Benutzbarkeit	Sicherheit	Erweiterbarkeit	Wartbarkeit	Übertragbarkeit	Wiederverwendbarkeit
Funktionserfüllung		+	+	+	o	o	o	o	+
Effizienz	+		o	-	-	+	+	-	+
Zuverlässigkeit	+	o		+	-	o	o	o	+
Benutzbarkeit	+	-	+		-	o	o	o	o
Sicherheit	o	-	-	-		-	-	-	-
Erweiterbarkeit	o	+	o	o	-		+	+	+
Wartbarkeit	o	+	o	o	-	+		+	+
Übertragbarkeit	o	-	o	o	-	+	+		+
Wiederverwendbarkeit	+	+	+	o	-	+	+	+	

Tabelle 1: Wechselwirkungen zwischen nicht-funktionalen Anforderungen[1)]

4.2.2 Nicht-funktionale Anforderungen an die Ontologie

Unter den nicht-funktionalen Anforderungen für die Ontologie(n) die innerhalb des KOWIEN-Prototypen eingesetzt werden sollen, werden die aus der Literatur bekannten *Gütekriterien für Ontologien* herangezogen.

Einen ersten Versuch, Gütekriterien für Ontologien vorzugeben wurde von GRUBER vorgenommen[2)]. Sein Anforderungskatalog beinhaltet *clarity, coherence, extendibility, minimal encoding bias* und *minimal ontological commitment*. Die *Klarheit* (oder Objektivität) einer Ontologie entspricht ihrer interpersonellen Nachvollziehbarkeit. Während die Entscheidung zur Aufnahme eines Begriffs in die Ontologie durch den spezifischen Verwendungszusammenhang gegeben sein kann, sollte die Definition des Begriffs auch in anderen Kontexten nachvollziehbar sein. Formale Spezifikationen erweisen sich dabei aufgrund ihrer eindeutigen

1) Die Zeichen in der Tabelle lassen sich wie folgt interpretieren: „+“ bedeutet, dass die Anforderungen in einem sich gegenseitig begünstigenden Verhältnis stehen. „- “ bedeutet, dass die Anforderungen in einem sich gegenseitig beeinträchtigenden Verhältnis zueinander stehen. Das Zeichen „o“ ist an Stellen angebracht, bei denen zwei Anforderungen keine Korrelation miteinander aufweisen.

2) Vgl. GRUBER (1995).

Bedeutungsbelegung als vorteilhaft. Unterschiede existieren bei einzelnen Formalismen in den Möglichkeiten zur Unterscheidung zwischen notwendigen und hinreichenden Bedingungen für Prädikate. Die *Kohärenz* einer Ontologie kann in erster Linie durch Integritätsregeln gewahrt werden. Wenn zum Beispiel durch eine entsprechende Inferenzregel auf eine spezielle Kompetenz eines Mitarbeiters aufgrund einer Weiterbildungsmaßnahme geschlossen werden soll, so ist dies unzulässig, wenn zuvor die „Inkompetenz" des Mitarbeiters in diesem Bereich explizit modelliert wurde. Um die *Erweiterbarkeit* der Ontologie zu ermöglichen, müssen zukünftige Bedürfnisse nach Begriffsstrukturierung antizipiert werden. Dabei sollte es nicht erforderlich sein, die bereits bestehende Struktur zu verändern. Die Minimierung der Anzahl programmierter systematischer Fehler wird durch die Konstruktion der Ontologie auf einer konzeptuellen Ebene (*Knowledge Level*) gewährleistet. Die Transformation der Ontologie zwischen unterschiedlichen Repräsentationsformalismen sollte demnach ohne Reibungsverluste möglich sein. Die Minimierung ontologischer Übereinkünfte verlangt, das die Nutzer der Ontologie so wenige Verbindlichkeiten wie möglich eingehen müssen, um die Ontologie in der ursprünglich intendierten Anwendung noch nutzen zu können.

Ein weiterer Anforderungskatalog wird von FOX/GRÜNINGER vorgestellt[1)]. Die hierin enthaltenen Anforderungen lauten: *functional completeness, generality, efficiency, perspicuity, precision/granularity, minimality.* Die *funktionale Vollständigkeit* ist gegeben, wenn sämtliche der intendierten Anwendungen der Ontologie unterstützt werden. Die Allgemeingültigkeit der Ontologie ist gegeben, wenn die Ontologie in unterschiedlichen Domänen eingesetzt werden kann. Durch die Einsetzbarkeit der Ontologie in unterschiedlichen Anwendungsdomänen wird ihre *Allgemeingültigkeit* gesichert. Die Effizienz der Ontologie ist in erster Linie auf ihre Axiomatik ausgerichtet: Die Inferenzregeln der Ontologie sollten so konstruiert werden, dass aus den bereits bestehenden Fakten eine Vielzahl „neuer" Fakten abgeleitet werden können. Die Präzision der Ontologie wird durch die Verringerung der Konzept-Mengen gesichert, die sich „überlappen". Die Granularität verlangt, dass durch die Einbindung einer taxonomischen Hierarchie unterschiedliche Abstraktionsebenen ermöglicht werden.

1) Vgl. FOX/GRÜNINGER (1997) S. 195.

Ein dritter Anforderungskatalog stammt von GÓMEZ-PÉREZ[1)]. Ihre Gütekriterien lauten: *consistency*, *completeness*, *conciseness*, *expandability* und *sensitiveness*. Die *Konsistenz* bezieht sich auf die Integrität der Ontologie. Die Ontologie gilt dann als konsistent, wenn keine widersprüchlichen Fakten aus der Ontologie abgeleitet werden können. Die *Vollständigkeit* ist dann gegeben, wenn der Bedarf an Begriffsdefintionen durch die Ontologie gedeckt wird. Durch die *Kürze (Prägnanz)* der Ontologie wird zum einen gefordert, dass in der Ontologie keine unnötigen Begriffsdefinitionen existieren sollen. Dies impliziert die Forderung nach Relevanz der verwendeten Begrifflichkeiten. Zum anderen sollen keine Redundanzen in der Ontologie existieren. Die Erweiterbarkeit entspricht auch hierbei - wie bei GRUBER - dem Aufwand, der notwendig ist, um neue Begriffe in die Ontologie aufzunehmen. Die Sensitivität betrifft das Ausmaß, in dem eine Änderung in einem Prädikat, auch die Änderung anderer Prädikate erfordert.

5 Zusammenfassung und Ausblick

In dem vorliegenden Projektbericht wurden Anforderungen untersucht, die durch den KOWIEN-Prototypen umgesetzt werden sollen. Es wurde dabei unterschieden zwischen funktionalen und nicht-funktionalen Anforderungen an den Prototypen. Innerhalb der Analyse der funktionalen Anforderungen wurden einige Anwendungsfälle beschrieben, die entsprechend einer noch vorzunehmenden Priorisierung bei der Implementierung des Prototypen in der entsprechenden Reihenfolge berücksichtigt werden sollten. Zum anderen wurde darauf hingewiesen, dass die Ontologien, welche die Grundlage für einige der Anwendungsfälle darstellen, die dafür relevanten Begrifflichkeiten enthalten sollten. Im zweiten Teil der Anforderungsanalyse wurden nicht-funktionale Anforderungen an den Prototypen sowohl aus der Perspektive der potenziellen Nutzer als auch aus der Perspektive der Entwickler vorgestellt.

Die Ergebnisse dieser Analyse werden im folgenden Arbeitspaket 2.4 „Erprobung und Evaluierung / Analyse“ durch die Unternehmenspartner evaluiert werden. Sollten sich hierbei Modifikationen gegenüber der gegenwärtigen Anforderungsspezifikation ergeben, so wird dies in einer weiteren Version dieses Projektberichtes dokumentiert werden.

---

1) Vgl. GOMEZ-PÉREZ (2001) S. 394 ff.

Für das weitere Vorgehen innerhalb der Anforderungsanalyse wird es notwendig sein, weitere Anwendungsfälle von den Praxispartnern zu erheben. Um die Erfassung von Anwendungsfällen in strukturierter Form zu unterstützen, wurde von Herrn Bäumgen eine spezielle infonea[1] Anwendung entwickelt, mit der u.a. die Geschäftsvorfälle der Praxispartner in Beziehung zu den daraus abgeleiteten Anwendungsfällen gesetzt werden können, so dass die Praxispartner später nachvollziehen können, in welcher Weise die von ihnen beschriebenen Geschäftsvorfälle vom KOWIEN-Prototypen unterstützt werden.

1) http://www.infonea.com

Anhang A: Geschäftsvorfall der Karl Schumacher GmbH

Das Ganze begann mit einer technischen Problemstellung, bei der die Unterstützung eines großen Komponentenlieferanten bzw. -hersteller notwendig war. Dieser ist ein weltweit tätiger Marktführer im Bereich von Pneumatikkomponenten.

Im ersten Telefonat teilte der Mitarbeiter aus dem technischen Bereich mit, das er nicht wisse, ob man in der Lage ist, die gestellten Anforderungen zu erfüllen. Er werde die Frage aber an seinen Kollegen weitergeben, der sich dann in der nächsten Stunde melden würde. Wie zu erwarten kam kein Rückruf.

Am nächsten Tag nochmaliger Anruf beim Hersteller. Dieses Mal war ein scheinbar kompetenter Technik an der Leitung, der im Brustton der Überzeugung mitteilte, dass man die Anforderungen nicht erfüllen könne. Es gäbe auch keinerlei Sonderlösungen.

Daraufhin wurden von uns Konkurrenzprodukte angefragt, die dann alle in der Lage waren, die Anforderungen zu erfüllen.

Am Nachmittag desselben Tages dann der mittlerweile schon in Vergessenheit geratene Rückruf. Diesmal war jedoch in unserem Hause niemand erreichbar (war ja auch Arbeitstreffen für KOWIEN).

Als dann mittlerweile zwei Tage nach dem ersten Telefonat der Rückruf des Herstellers erfolgreich war, teilte dieser uns mit, dass es sicherlich kein Problem wäre, die Anforderungen zu erfüllen. Hierfür gäbe es schließlich Sonderlösungen, auf die der Anrufer persönlich spezialisiert wäre. Er machte in der Tat einen sehr kompetenten Eindruck und gab etliche Tipps am Rande, die sehr hilfreich waren. Diese Tipps wie auch die Durchwahl des Anrufers wurden sofort notiert und bei uns in den Umlauf gegeben.

Literaturverzeichnis

BALZERT (1998)
Balzert, H.: Lehrbuch der Software Technik: Software - Management, Software Qualitätssicherung, Unternehmensmodellierung. Heidelberg - Berlin 1998.

CRANEFIELD (2001)
Cranefield, S.: UML and the Semantic Web. In: Cruz; I.F.; Decker, S.; Euzenat, J.; McGuiness, D. (Hrsg.): Proceeding of SWWS´01, The First Semantic Web Working Symposium, 30. Juli - 1.August, Stanford Universität, Kalifornien USA, S. 113-130.

FOWLER/SCOTT (2000)
Fowler, M.; Scott, K.: UML - konzentriert. 2. Aufl., München et al. 2000.

FOX/GRÜNINGER (1997)
Fox, M.S.; Grüninger, M.: Ontologies for Enterprise Modeling. In: Kosanke, K.; Nell, J.G. (Hrsg.): Enterprise Engineering and Integration: Building International Consensus. Proceedings of ICEIMT 1997, International Conference on Enterprise Integration and Modeling Technology. Berlin et al. 1997, S. 190-200.

GEBERT (2001)
Gebert, H.: Kompetenz-Management – Bewirtschaftung von implizitem Wissen in Unternehmen. Kolloquium für Doktoranden der Wirtschaftsinformatik. 18.9.2001. http://wi.oec.uni-bayreuth.de/doctoral/Beitraege/gebert.pdf (eigene Paginierung). Zugriff: 18.9.2002.

GOMEZ-PÉREZ (2001)
Gomez-Pérez, A.: Evaluation of Ontologies. In: International Journal of Intelligent Systems, 16. Jg. (2001), S. 391-409.

GRUBER (1995)
Gruber, T.R.: Towards Principles for the Design of Ontologies Used for Knowledge Sharing. International Journal of Human-Computer Studies, 43. Jg. (1995), S. 907-928.

GRÜNINGER/FOX (1994)
Grüninger, M.; Fox, M.S.: The Role of Competency Questions in Enterprise Engineering. Workshop on Benchmarking - Theory and Practice, Trondheim - Norwegen 1994. URL: http://www.eil.utoronto.ca/enterprise-modelling/papers/benchIFIP94.pdf.
Zugriff: 18.9.2002

HOFMANN (2000)
Hofmann, H.F.: Requirements Engineering - A Situated Discovery Process. Wiesbaden 2000.

JACOBSON ET AL. (1992)
Jacobson, I.; Booch, G.; Rumbaugh, J.: The Unified Software Development Process. Biston et al. 1992.

KAHLBRANDT (2001)
Kahlbrandt, B.: Software-Engineering mit der Unified Modeling Language. Berlin et al. 2001.

KNACKSTEDT (2001)
Knackstedt, R.: Konfigurative Referenzmodelle als Instrumente des Wissensmanagements bei der Data-Warehouse-Entwicklung. In: Schnurr, H.-P., Staab, S., Studer, R., Stumme, G., Sure, Y. (Hrsg.): Professionelles Wissensmanagement. Erfahrungen und Visionen. Aachen 2001, S. 113-128.

KULAK/GUINEY (2001)
Kulak, D.; Guiney, E.: Use Cases - Requirements in Context. 3. Aufl., Boston et al. 2001.

MELCHISEDECH (2000)
Melchisedech, R.: Verwaltung und Prüfung natürlichsprachlicher Spezifikationen. Diss., Aachen 2000.

OESTEREICH (1998)
Oesterreich, B.: Objektorientierte Systementwicklung: Analyse und Design mit der Unified modeling language. 4. Aufl., München et al. 1998.

PARTSCH (1998)
Partsch, H.: Requirements - Engineering systematisch. Berlin et al.1998.

POHL (1996)
Pohl, K.: Process-Centered Requirements Engineering. New York et al. 1996.

RAASCH (1993)
Raasch, J.: Systementwicklung mit strukturierten Methoden. 3. Aufl., München - Wien 1993.

SNEED (1988)
Sneed, H.M.: Software Qualitätssicherung. Köln 1988.

SOMMERVILLE (2001)
Sommerville, I.: Software Engineering. 6. Aufl., Harlow et al. 2001.

SOMMERVILLE/SAWYER (2000)
Sommerville, I.; Sawyer, P.: Requirements Engineering - A Good Practice Guide. Chichester et al. 2000.

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.