February 2005

Multiagentensystem zur Wissenskommunikation in der Produktentstehung - Rapid Product Development

Stavros Dalakakis
Universität Stuttgart

Michael Diederich
Fraunhofer Institut für Arbeitswirtschaft und Organisation (IAO)

Dieter Roller
Universität Stuttgart

Joachim Warschat
Fraunhofer Institut für Arbeitswirtschaft und Organisation (IAO)

In: Ferstl, Otto K, u.a. (Hg) 2005. *Wirtschaftsinformatik 2005: eEconomy, eGovernment, eSociety*; 7. Internationale Tagung Wirtschaftsinformatik 2005. Heidelberg: Physica-Verlag

ISBN: 3-7908-1574-8

Multiagentensystem zur Wissens-kommunikation in der Produktentstehung - Rapid Product Development

Stavros Dalakakis
Universität Stuttgart

Michael Diederich
Fraunhofer Institut für Arbeitswirtschaft und Organisation (IAO)

Dieter Roller
Universität Stuttgart

Joachim Warschat
Fraunhofer Institut für Arbeitswirtschaft und Organisation (IAO)

Zusammenfassung: Der Sonderforschungsbereich (Sfb) 374 „Entwicklung und Erprobung innovativer Produkte – Rapid Prototyping“ an der Universität Stuttgart thematisert den Entwicklungsprozess von der Idee bis zum Prototyp. Im Rahmen eines multidisziplinären Ansatzes wird untersucht, inwieweit frühzeitig unterschiedliche Einflüsse auf das zu entwickelnde Produkt angewandt werden können. Durch die Nutzung schneller Iterationszyklen und der situationsgerechten Verwendung von Prototypen kann der Ansatz einer evolutionären Produktentwicklung erreicht werden. Dabei werden die Informationen der am RPD-Prozess beteiligten Arbeitsbereiche wie Kostenrechnung, Projektplanung, Konstruktion, Prototypenbau, etc. semantisch verknüpft und in einem dafür konstruierten Aktiven Semantischen Netz (ASN) abgelegt. Bei der direkten Zusammenarbeit der einzelnen RPD-Domänen entstehen zum Beispiel Abstimmungsprobleme, die Mechanismen erfordern, die nicht direkt durch die einzelnen RPD-Anwendungen oder das ASN gelöst werden können. Dafür wurde eine Multiagenten-basierte Middleware entwickelt, die Gegenstand dieses Artikels ist.

Schlüsselworte: Multiagentensysteme, Agententechnologie, Information Retrieval, Assoziative Abfrage, Aggregierende Anfrage, Koordination, Monitoring, Multicastkommunikation.

1 Einleitung

1.1 Randbedingungen des Rapid Product Development

Die Ausgangssituation zeichnet sich dadurch aus, dass die Rapid Product Development (RPD)-Anwendungen aus den unterschiedlichsten Bereichen wie Prototypenbau, Visualisierung, Simulation, Kostenberechnung und Organisation ihre Informationen untereinander zur Verfügung stellen und austauschen müssen. Zu diesem Zweck wurde das Aktive Semantische Netz (ASN) entwickelt, das es ermöglicht, auf einer sehr abstrakten Ebene Wissen zu formulieren, semantisch zu verknüpfen und damit den anderen Domänen zur Verfügung zu stellen.

Neben der reinen Wissensablage und der rudimentären Manipulation von Wissenselementen wurde ein intuitiver, flexibler Zugriff auf das ASN benötigt, der zusätzlich Basis-Funktionalitäten zur Manipulation des Wissens anbietet. Diese Funktionalitäten sind, das Auffinden von Informationen aus dem ASN ohne dessen Struktur im Detail kennen zu müssen, das situationsgerechte Aufbereiten der gefundenen Informationen zur weiteren Verarbeitung innerhalb der RPD-Anwendung oder als Grundlage für weitere Suchanfragen an das ASN, das aktive Überwachen des ASNs zur Realisierung einfacher Synchronisationsmechanismen zwischen den RPD-Anwendungen, sowie Mechanismen zur flexiblen Koordination von RPD-Anwendungen entlang des RPD-Prozesses. Zur Erbringung dieser Dienste wurden vier auf das RPD abgestimmte Agententypen realisiert und prototypisch implementiert, die in Kapitel 3 näher beschrieben werden.

Nach einer kurzen Einführung in das ASN (siehe 1.2) und der Betrachtung der bereits existierenden Middlewaretechniken im Stand der Forschung und Technik (siehe 2) wird die im Rahmen des Sfb entwickelte Multiagenten-basierte Middleware detailliert vorgestellt (siehe 3) und in einem kurzen Anwendungsbeispiel veranschaulicht (siehe 4).

1.2 Grundlagen des Aktiven Semantischen Netzes

Das ASN besteht aus zwei Abstraktionsebenen. Die erste Abstraktionsebene, die Strukturebene umfaßt die Funktionalität, d.h. die Programmlogik des ASNs, sowie die Struktur der Datenablage, das Metamodell. Die zweite Abstraktionsebene ist die Instanzebene, die die eigentlichen Informationen beinhaltet (siehe Abbildung 1).

Das ASN ist auf der Strukturebene hierarchisch aufgebaut. Es besteht aus semantischen Netzen, die wiederum aus einer Vielzahl von Konzepten bestehen, die sich durch ihre Attribute auszeichnen und über Relationen semantisch verknüpft sind. Jedes Strukturelement des ASNs (Netz, Konzept, Attribut, Relation) besitzt einen in seinem Kontext eindeutigen Namen, um die eindeutige Identifizerung des Informationselements zu ermöglichen. Konzepte sind typisiert, so dass Konzepte

gleichen Typs gleiche Attribute besitzen. Durch die Flexibiltät des ASNs ist es möglich, Konzepte jederzeit um weitere Attribute zu erweitern.

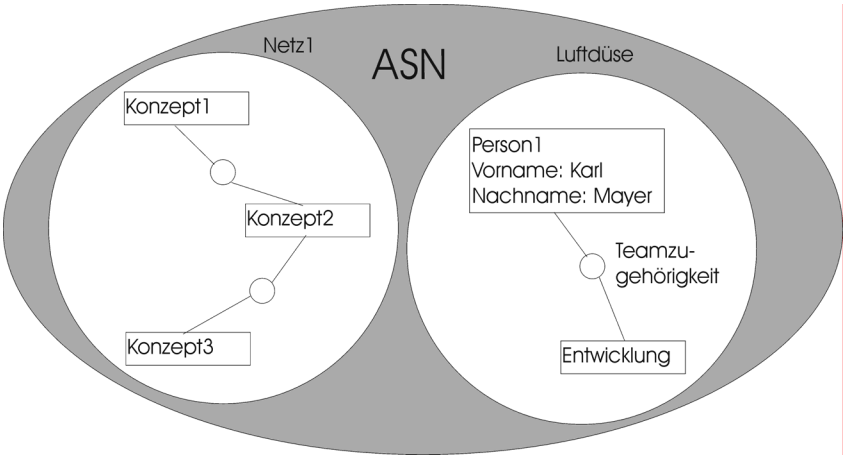

Abbildung 1: Struktur des ASNs auf Instanzebene

Auf der Instanzebene werden die Konzepte und Attribute mit Inhalten gefüllt und zwischen Konzepten die semantischen Verknüpfungen über Relationen hergestellt. So gehört ein instanziiertes Attribut genau einem Konzept an und dieses genau einem Netz. Die Angaben Netz, Konzept, Attribut sind notwendig um den Wert eines Attributs auszulesen.

Zum Beispiel beinhaltet auf Strukturebene das Netz für die Entwicklung der Luftdüse ein Konzept vom Typ Person, das neben anderen Attributen auch die Attribute Vor- und Nachname besitzt. Diese Personen gehören dann einem Team an, das ebenfalls durch ein Konzept repräsentiert wird und mindestens das Attribut Name mit dem Teamnamen besitzt.

Auf der Instanzebene kann dies zum Beispiel eine Instanz Netz mit dem Namen „Luftdüse“, eine Instanz Konzept mit dem Namen „Person1“ und den Attributen Vorname „Karl“ und Nachname „Mayer“ sowie eine Instanz Konzept Team mit dem Namen „Entwicklung“ sein. Des Weiteren wird eine Relation zwischen dem Team und der Person hergestellt (siehe Abbildung 1).

In einer ähnlichen Vorgehensweise versucht man verteilte Daten im Internet wie eine globale Wissensbasis zu behandeln. Diese Form der abstrakten Wissensrepräsentation nennt man Semantic Web [Sema04]. Dabei stützt sich das Semantic Web auf Resource Description Framework (RDF) Standards[Rdf04]. RDF und ASN sind parallele Entwicklungen. Das ASN stützt sich am eigenentwickeltem Metamodell wobei die RDF-Standards sich, in einer Form, auf der Beschreibungs-Logik sich stützen. So gesehen ist das Semantic Web eine für den Zweck des Internet bereitgestellte Lösung des Semantischen Netzes.

2 Stand der Forschung und Technik

2.1 Agentenarchitekturen und -modelle

Die Kommunikation ist das wesentliche Merkmal innerhalb eines Multiagentensystems und per Definition ein Mindestkriterium, das erfüllt werden muss [GeKe94]. Das setzt eine gemeinsame Kommunikationssprache innerhalb des Multiagentensystems, eine so genannte "Agent Communication Language" (ACL), voraus. Die FIPA (Foundation of Intelligent Physical Agents) stellt Spezifikationen für die gesamte Breite der Agententechnologie bereit [FIPA02a] und hat bereits als ACL für die Kommunikation eine "message structure" standardisiert [FIPA02b]. Auch die FIPA-ACL Semantik ist durch die FIPA-"Communicative Act Library" (CAL) spezifiziert worden [FIPA02c]. Den Mittelpunkt der gesamten FIPA-Aktivitäten bilden die Beziehungen zwischen den Agenten, mit dem Ziel, die Interoperabilität in komplexen Systemen zu erhöhen. Durch die Standardisierungen von FIPA ist eine weit verbreitete sowie allgemein akzeptierte Agenten-Architektur geschaffen worden. So stimmen wichtige Projekte, wie ABLE [Able03] [Bigu+02] und JADE [Jade03] (siehe 2.3), mit den Spezifikationen der FIPA überein.

In [Matu95] wird ein Verfahren für einen Koordinationsmechanismus vorgestellt, dabei wird die Verhandlung durch den Einsatz einer zentralen vermittelnden Instanz gesteuert. Der Einsatz einer zentralen Koordinationsinstanz, z.B. durch einen Mediator, ist in der gewählten Architektur nicht praktikabel, da sich die Anforderungen an die Koordination laufend ändern und damit vorausgesetzt wird, dass das Koordinationsprotokoll frei definierbar ist.

2.2 Agenten-Kommunikationsprotokolle

Im Bereich der Interaktionsprotokolle hat sich das sichere Protokoll „Light-weight Reliable Multicast Protocol“ (LRMP) durchgesetzt [Liao00]. Die Multicast-Technologie hat gegenüber der Broadcast-Technologie den Vorteil, dass nur die Instanzen mit Informationen versorgt werden, die diese auch benötigen. Des Weiteren berücksichtigt das LRMP die Unterstützung einer Fehler überwachten (Reliable) Verbindung ähnlich dem TCP. Damit werden die Schwächen des Standard-Multicast-Protokolls ausgeglichen. LRMP ist für heterogene Netzwerke aufgebaut mit Unterstützung mehrerer Sender. Bei der Koordination von RPD-Aufgaben, beispielsweise bei der Verhandlung von Ressourcenkonflikten, wie sie in der teamorientierten Projektplanung (TOPP) entstehen, findet eine Kommunikation, vermittelt durch den Koordinationsagenten, zwischen mehreren TOPP-Instanzen statt. Das setzt eine zuverlässige Kommunikationsplattform voraus. LRMP gewährleistet diese Zuverlässigkeit und bietet mit Hilfe des Multicast-Protokolls die Möglichkeit über verteilte Standorte Informationen auszutauschen. Im Allgemeinen wird das LRMP für die Implementierung von „Java Message Service“, dies ist

eine wichtige Komponente für Verteilte Systeme, eingesetzt [Java00] [Swif01]. Auch für Realisierungen von Diensten der Peer-to-Peer-Spezifikation JXTA wurde LRMP benutzt [Sun03].

2.3 Multiagentensystem in der Forschung

Ein richtungsweisendes Projekt ist das "Agent Building and Learning Environment" (ABLE) von IBM. Dies ist eine prototypische Entwicklung von Agenten mit starkem Einfluss aus dem Bereich der Computational Intelligence [Able03] [Bigu+02]. ABLE besteht aus einem Java-Framework, einer Komponenten-Library und aus einem Werkzeugsatz, der durch die Nutzung von Techniken aus dem Maschinellen Lernen und Reasoning das Bauen und Benutzen von hybriden intelligenten Agenten ermöglicht und berücksichtigt die FIPA-Spezifikation. Es werden neben Lernmethoden auch Optimierungs-Algorithmen eingesetzt, die ähnlich aufgebaut sind wie JavaBeans, die AbleBeans genannt werden, wodurch eine sehr flexible Agenten-Plattform angeboten wird.

Auch das Projekt „Java Agent Development Framework“ (JADE) baut auf die von FIPA entwickelten Spezifikationen auf [Jade03]. Die gesamte Plattform bietet runtime Systemservices für Agenten. Komponenten, wie das „Agent Management System“ und der „Agent Communication Channel“ regeln den Transport von ACL-Nachrichten. Die internen Interaktionen finden durch „light-weight“ Kommunikation auf der Ebene der Java Objekte statt. Analog zu einem Applikationsserver, der den Container von Objekten verwaltet, ist der Agenten-Container innerhalb der Agenten-Plattform organisiert. Dabei wird von einem Server-Objekt der gesamte Lebenszyklus über das Entstehen bis zum Entfernen aller Agenten koordiniert. Das Autonomie-Merkmal der Agenten wird über ein Modell dargestellt, das sich auf eine Verhaltensabstraktion für alle möglichen Agentenzustände stützt. Im Normalfall werden verschiedene, vorbereitete Verhaltensmuster, abhängig von dem Agentenbefehl, abgearbeitet.

FIPA-OS ist ein mit den FIPA Spezifikationen übereinstimmendes „Open Source“ Projekt [FIPA03]. Durch das komponentenorientierte Agentenwerkzeug werden die zwingend erforderlichen FIPA-gerechten Komponenten bereit gestellt. Der Entwurf eines Multiagentensystems startet mit der Spezifikation der Architektur und der Komponenten sowie mit der Serviceschnittstelle und der Implementierung. Später werden die Transport Verschlüsselung und Content Sprache festgelegt.

3 RPD-Multiagentensystem

Wie bereits in 1.1 angerissen, ist eine flexible zentrale Datenhaltung für den RPD-Prozess nicht ausreichend, die funktionale Lücke wird durch eine Multiagenten-

basierte Middleware geschlossen. Das Multiagentensystem des RPDs ordnet sich als Middleware zwischen den RPD-Anwendungen und der zentralen Datenhaltung, dem ASN ein (siehe Abbildung 2). Die Middleware besteht zum einen aus Agentenframework, d. h. dem Umfeld, in dem sich die einzelnen Agenten befinden und den Agenten selbst.

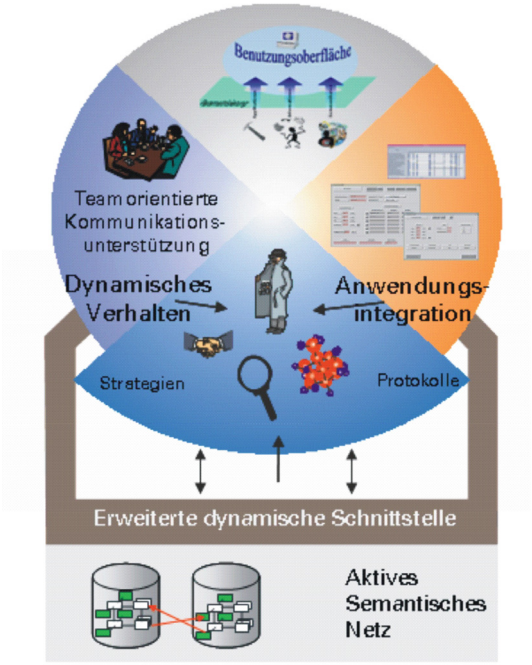

Abbildung 2: Architektur des Rapid Product Development

3.1 Agentenframework

Das Agentenframework hat die funktionalen Anforderungen und die strukturellen Anforderungen der RPD-Anwendungen zu berücksichtigen. Die funktionalen Anforderungen umfassen die Aufgaben, die die Agenten der Middleware zu übernehmen haben:

- Informationsbeschaffung auf einem höheren und abstrakteren Niveau als mit einer einfachen Programmierschnittstelle zum Abfragen von Informationen, wie es derzeit mit dem ASN möglich ist (siehe 3.2 Retrieval-Agent).
- Aufbereitung des abgefragten und gefundenen Wissens durch die von der RPD-Anwendung vorgegebenen Richtlinien. Damit soll ein und dasselbe Wissen jeweils im Kontext der RPD-Anwendung aufbereitet werden (siehe 3.3 Aggregations-Agent).

- Aktives Überwachen des ASNs mit dem Ziel, auf strukturelle und inhaltliche Veränderungen schnellstmöglich reagieren zu können ohne dabei innerhalb der RPD-Anwendung aufwendige Polling-Mechanismen implementieren zu müssen (siehe 3.4 Monitor-Agent).
- Koordination von Informationsflüssen zwischen RPD-Anwendungen, um interdisziplinäre Kommunikation zu strukturieren und zu vereinfachen (siehe 3.5 Koordinations-Agent).

Die strukturellen Anforderungen berücksichtigen die Gegebenheiten, in denen sich die RPD-Anwendungen und das ASN befinden und müssen vom Agentensystem berücksichtigt werden. Diese Anforderungen sind:

- Die netzwerkweite Verfügbarkeit der Dienstleistungen aufgrund der Verteiltheit der RPD-Anwendungen.
- Einfache, standardisierte, programmiersprachenunabhängige Schnittstellen zwischen RPD-Anwendung und Middleware, um den verschiedensten programmiertechnischen Anforderungen der RPD-Anwendungen, wie z.B. einfache Anwendungen, webbasierte Anwendungen, Maschinensteuerungen, gerecht zu werden.
- Eine leicht verständliche, intuitive Schnittstelle zum Agentensystem, mit dem Ziel, ohne große Kenntnisse der inneren Struktur der Middleware, optimale Ergebnisse zu erreichen.
- Eine sichere und fehlertolerante Kommunikation zwischen RPD-Anwendung und Middleware, damit im Fehlerfalle die RPD-Anwendung in der Lage ist, Gegenmaßnahmen ergreifen bzw. Meldungen an den Benutzer aussenden zu können.

3.1.1 Schnittstelle RPD-Anwendung / Middleware ASN

Um dem Ziel einer möglichst einfachen und programmiersprachenunabhängigen Schnittstelle gerecht zu werden, wurde auf eine nachrichtenbasierte Schnittstelle zurückgegriffen. Als Nachrichtenformat wird XML benutzt. Die RPD-Anwendungen müssen zur Kommunikation mit der Middleware lediglich XML-Nachrichten erzeugen und empfangene Nachrichten auswerten, unabhängig von ihrer internen Struktur und ihrer Implementierungssprache. Für die Middleware hat es den Vorteil, dass ein einheitliches Empfangs- und Aufbereitungsmodul für die Agenten entwickelt werden konnte. Die möglichen Nachrichten und ihr exaktes Format hängen stark von der Funktionalität der einzelnen Agententypen ab. Um den Zugriff auf die Middleware für die RPD-Anwendungen einfach zu halten, repräsentiert sich die Middleware mit einer einheitlichen Multicast-Adresse, die die Nachrichten empfängt (siehe 3.1.2).

Die Schnittstelle zwischen Middleware und ASN ist geprägt durch die Vorgaben des ASNs. Das ASN baut hierbei auf einer Applikationsserverlösung auf, die in

der Programmiersprache JAVA implementiert ist. Für eine einfache und effiziente Anbindung der Middleware an das ASN wurde für die Middleware auf JAVA zurückgegriffen.

3.1.2 Multicast-Umgebung

Bei der Kommunikation zwischen RPD-Anwendung und Middleware entsteht folgendes Problem. Die RPD-Anwendung hat ein Problem zu lösen und benötigt dabei die Unterstützung der Middleware.

Eine mögliche Problemlösung ist die Überprüfung der Agenten durch die anfragende RPD-Anwendung auf deren Fähigkeit, die entsprechende Aufgabe zu lösen. Das hat aber den Nachteil, dass allen RPD-Anwendungen die Adressdaten aller Agenten bekannt sein müssen und zum anderen dauert das sequentielle Abfragen der Agenten, ob sie in der Lage sind, die Aufgabe zu übernehmen, sehr lange. Eine weitere Möglichkeit dieses Problem zu lösen, ist das Implementieren einer zentralen Serverinstanz, die alle Anfragen entgegen nimmt und an den Agenten weiterreicht. Ist die Serverinstanz im Fehlerfall nicht verfügbar, bedeutet dies den Zusammenbruch des Systems. Des Weiteren muss die Serverinstanz zyklisch alle Agenten auf Verfügbarkeit und Fehlerfreiheit sequentiell überprüfen.

Auf Grund dieses Problems wurde eine Mischung aus beiden Ansätzen gewählt, nämlich eine Multicast-Umgebung (Abbildung 3). Mit Hilfe der Multicast-Umgebung müssen die RPD-Anwendungen nur eine Adresse, die Multicast-Adresse des Agentensystems, kennen. Dies löst das Problem des Wissens der Adressdaten aller Agenten. Das Problem des Auffindens eines verfügbaren Agenten wird auf das Multiagentensystem verlagert. Innerhalb des Multiagentensystems wird eine zentrale Instanz implementiert, der so genannte Master-Agent [DiLe01], [Wars+02]. Dieser Agent übernimmt die Aufgabe, die anderen Agenten auf ihre Verfügbarkeit hin zu überwachen und zu überprüfen. Zusätzlich beendet der Master-Agent alle Agenten, die durch nicht korrekt abgeschlossene Aufträge, verursacht durch die RPD-Anwendungen, verwaist sind.

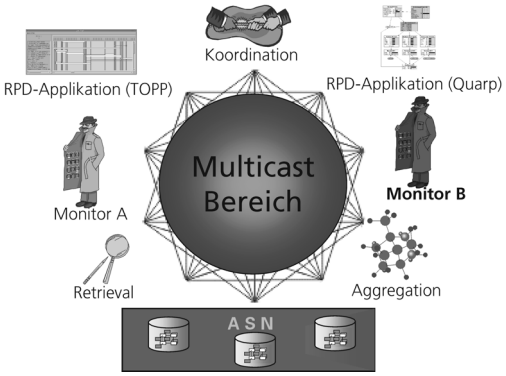

Abbildung 3: Multicast-Umgebung der Middleware

Dem „Single-Point-of-Failure“-Problem, bedingt durch die Verfügbarkeit des Master-Agenten, wird dadurch begegnet, dass der Master-Agent nicht als einzelner, besonderer Agent realisiert ist, sondern als Bestandteil aller Agenten. Dadurch ist es möglich, dass alle Agenten Master-Agent werden können. Um den Master-Agent zu bestimmen, bedient man sich dem Netzwerkprotokoll Tokenring [IEEE802.5].

3.2 Retrieval-Agent

Das Ziel des Retrieval-Agenten ist es, ein effektives Wissens-Retrieval zu realisieren. Dabei werden Methoden erstellt, die den semantischen Gehalt der Relationen so nutzen, dass das bereits konzeptualisierte Wissen zusätzlich durch das relationsbasierte Wissen bereichert wird.

Der Lebenszyklus einer Anfrage ist in Abbildung 4 als Aktivitätsdiagramm zu sehen. Dabei ist für den RPD-Anwender nicht relevant, ob die gestellte Anfrage durch ein scharfes oder ein unscharfes Retrieval realisiert wird. Interessant ist nur die Findung des relevanten Wissens aus dem ASN. Nach der Anfrageformulierung wird erst mit scharfen Methoden gesucht und wenn keine zufriedenstellenden Ergebnisse geliefert werden, werden unscharfe Methoden angewendet.

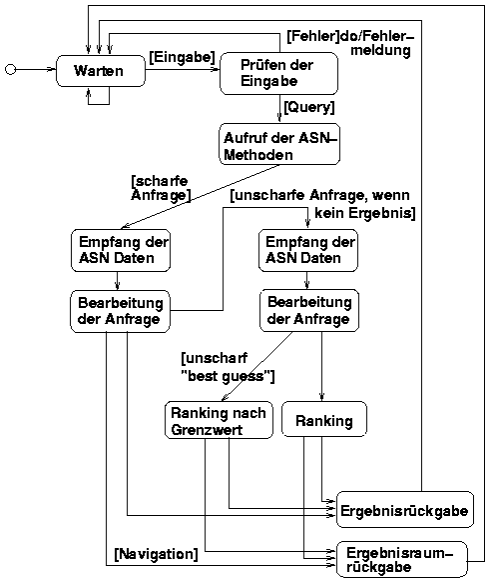

Abbildung 4: Zustände der Anfrage

3.2.1 Anfragesprache

Die für die Anfragesprache gestellten Anforderungen sind einerseits Anforderungen, die an das darunterliegende Repräsentationsmodell gestellt werden, aber auch

spezielle Anforderungen, die aus dem RPD-Prozess entstehen. Die bereits festgelegte Repräsentation des Wissens als ASN bildet das Repräsentationsmodell, auf das sich die zu entwickelnde Anfragesprache stützt. Die Navigation durch Pfadausdrücke, Operationen auf Relationen und die Möglichkeit zu unscharfem Retrieval sind Anforderungen aus dem RPD-Umfeld.

Die Active Semantic Network Query Language (ASN-QL) ist eine Anfragesprache, die ausschließlich für den Zugriff auf das Aktive Semantische Netz entworfen worden ist [Dala+04]. Typische Fragewörter, die im Ingenieurbereich vorkommen sind z.B. „Wieviel", „Wer" oder „Wie oft". Dabei wird bereits durch die Anfrageformulierung die Form der Ergebnisse beschrieben. ASN-QL ermöglicht zusätzlich ein rekursives Verhalten der Anfrage sowie Navigation in dem Ergebnisraum.

3.2.2 Scharfes Retrieval

Beim scharfen Retrieval werden die Konzepte hinreichend beschrieben, z.B. durch eine ID, sodass genügend Informationen für eine erfolgreiche Suche vorliegen.

Die Methoden sind:

- *get-concept*: diese Funktion nimmt Eigenschaften von Konzepten entgegen und liefert ein oder mehrere im ASN existierende Konzepte zurück.
- *is-associated-with*: diese Funktion verwendet eine oder zwei Konzeptbeschreibungen für das Start- und Zielkonzept und liefert alle ausschließlich assoziierten Konzepte zurück, die im Umkreis der Range liegen.
- *is-aggregated-of*: diese Funktion gibt eine Liste der Konzepte, die ausschließlich in einer Aggregations-Beziehung zu dem angegebenen Konzept stehen, zurück. Man kann die Aggregationsbeziehung auch für ein Aggregat-Konzept und sein Teil-Konzept erfragen, insbesondere wenn die Konzept-Id's bekannt sind.
- *is-composed-of*: diese Funktion gibt eine Liste der Konzepte, die Bestandteil einer Kompositionsbeziehung sind zurück. Werden zwei Konzepte im Umkreis der Range vermutet, das Start- sowie das Zielkonzept, dann wird die Kompositionsbeziehung aller zwischenliegender Konzepte ermittelt.
- *is-generalization-of*: diese Funktion verwendet eine Konzeptbeschreibung oder zwei Konzeptbeschreibungen für das Start- und Zielkonzept, und gibt alle generalisierten Konzepte zurück, die im Umkreis der Range liegen.

3.2.3 Unscharfes Retrieval

Nicht vollständige oder nicht exakte Abfragen liefern durch unscharfe Methoden Ergebnisse. Eine teilweise Überprüfung ist ausreichend, um Ergebnisse zu liefern. Es können auch Konzepte berücksichtigt werden, die nur eine Teilmenge der gesuchten Attribute aufweisen. Führt das unscharfe Retrieval zu keinen Ergebnissen,

dann wird durch das Keyword eine Menge von Suchbegriffen eingegeben, die sich sowohl auf ein Attribut als auch auf ein Konzept beziehen können.

Die Methoden sind:

- *get-concept-unsharp*: nimmt Teilinformationen über die Konzepte auf und liefert eine Anzahl von Konzepten.
- *is-related-with*: nimmt Teilinformationen über die Konzepte und bewertet Pfade mit dem gegebenen Range, wobei die verschiedenen Relationen nach der Non-Terminal-semantic gewichtet sind, um eine semantische Nähe ermittelt, die für die Liste der zurückgegebenen Konzepte ausschlaggebend ist.
- *consists-of*: diese Funktion erlaubt den semantischen Unterschied zwischen den Relationsformen Aggregation und Komposition zu abstrahieren. Bei der unscharfen Suche nach vermuteter Aggregation wird auch die Komposition gesucht, um eine eventuelle Unschärfe der Abfrage zu ermöglichen.
- *is-similar-to*: findet zu einem übergebenen Konzept ähnliche Konzepte. Eine Grobauswahl findet anhand der semantischen Beziehungen von Konzepten statt. Anschließend werden die ausgewählten Konzepte und das zu vergleichende Konzept in einem Vektorraum mit der Dimension der Anzahl der Attribute eingetragen. Die Gewichtung der Attribute ergibt sich aus ihrem Wert, falls dieser quantifizierbar ist. Mit dem Cosinus-Maß wird die Ähnlichkeit für alle Konzepte bestimmt.

3.2.4 Beispiel

In Abbildung 5 ist eine Beispielanfrage zu sehen, formuliert als Befehl an den Retrieval-Agenten. Dabei wird folgende Frage gestellt: „Wie viele Motoren mit dem Namen „MotorA“ stehen in Beziehung zu dem Konzept namens „Auto“?“. Die Suche soll auf fünf Konzepte beschränkt werden. Die Aufbereitung der Ergebnisse wird durch das Terminal „wieviele“ generiert.

```
<REQUEST sender="me" destination="searcher1/RetrievalAgent/01" reqcontrol="" >
<DATA type="ASNQLRequest">
wieviele hat-Beziehung-zu( CONCEPT NAME MotorA, CONCEPT NAME Auto, 5)
</DATA>
```

Abbildung 5: Beispiel eines Befehls für den Retrieval-Agenten

3.3 Aggregations-Agent

Aggregation als Repräsentationsform, die die Beziehung zwischen zwei Konzepten beschreibt, wird durch das Aggregat und seine Bestandteile definiert. Analog dazu können komplexe Problemstellungen als Problemteile P1, P2, ... Px erfasst werden. Die vorliegenden Aggregationsteile werden nach einer Abarbeitung in Teilergebnisse E1, E2, ... Ex transformiert und anschließend als Ergebnis inner-

halb des Aggregations-Agenten zusammengeführt. Die aus dem RPD-Prozess gestellten Anforderungen an den Aggregations-Agenten sind zum einen die Umformung von komplexen Anfragen, so dass diese von dem nachgelagerten Agenten in einer adäquaten Form weiterverarbeitet werden können, und zum anderen die Auswahl einer anwendungsspezifischen Sicht bzw. der userspezifischen Sicht. Zusätzliche Anforderungen betreffen das Performance-Problem, das im Zusammenhang mit der gesamten Systemperformance steht, z.B. die schnelle Abwicklung der Anfragen.

3.3.1 Anfragesprache

Die von dem Aggregations-Agenten ausgehende Anfrage wird an den Retrieval-Agenten weitergeleitet (siehe Abbildung 6) und nach dem Abwickeln der Anfrage wird das Ergebnis an den Aggregations-Agenten geliefert, der die entsprechende Aufbereitung des Aggregationsergebnisses übernimmt.

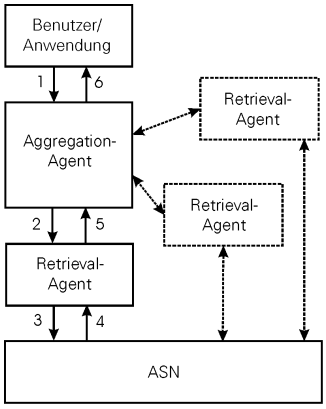

Abbildung 6: Kooperationsablauf zwischen dem Aggregations- und dem Retrieval-Agenten

Die zusätzliche Intelligenz des Aggregations-Agenten wird nicht nur über die Interpretation der gewünschten Sicht des Retrieval-Agenten-Ergebnisses widergespiegelt, sondern durch die Möglichkeit der expliziten Formulierung einer Anfrage und das daraus entstehende Wissen. Das neue Wissen entsteht aus der aggregierenden Kombination von verschiedenen Konzepten und Attributen.

3.3.2 Beispiel

In der Abbildung 7 ist ein Beispiel für das Finden einer Spezifikation eines Leiters für eine bereits bekannte Arbeitsgruppe. Dabei gilt es den Altersmittelwert aller Gruppenmitglieder und die maximale Anzahl an durchgeführten Projekten zu vereinen.

```
<Aggregation name="Leader" type="result" context="">
  <Element name="AverageAge" type="age">
    <Query id="0" name="avage" resultType="int">
      <QuerySource queryType="xquery">
        <!-- EXAMPLE:  find the average age of the people in team
             TEAM_WITHOUT_LEADER    -->
      </QuerySource> </Query>   </Element>
  <Element name="ProjectsNumber" type="int">
    <Query id="1" name="projnum" resultType="int">
      <QuerySource queryType="xquery">
        <!-- EXAMPLE:  find the max number of projects in which the
          people in the team TEAM_WITHOUT_LEADER participated -->
      </QuerySource> </Query>   </Element>
</Aggregation>
```

Abbildung 7: Beispiel einer Aggregationsanfrage

Das Ergebnis (siehe Abbildung 8) ist die Zusammensetzung der gewünschten Eigenschaften der geeigneten Person und nicht die durch Filtersuche passende Person. Es werden keine Suchkriterien definiert, sondern es wird neues Wissen aus bereits vorhandenem Wissen aggregiert.

```
<Aggregation name="Leader" type="result" context="">
  <Element name="AverageAge" type="age">
    <Result queryId="0">28</Result>   </Element>
  <Element name="ProjectsNumber" type="int">
    <Result queryId="1">5</Result>   </Element>
</Aggregation>
```

Abbildung 8: Das Aggregationsergebnis des Beispiels

3.4 Monitor-Agent

Der Monitor-Agent hat zum Ziel, das ASN auf bestimmte Veränderungen zu überwachen und diese der RPD-Anwendung zu melden. Der Schwerpunkt wurde hierbei auf die Entwicklung einer Überwachungsmethode gelegt, die es erlaubt, nicht nur einzelne Attribute im ASN auf Veränderungen zu überwachen, sondern die auch komplexe Strukturen und Abhängigkeiten im ASN berücksichtigt, bis hin zu einfachen Funktionen zur Aufbereitung der ASN-Inhalte, aber auch der ASN-Zugriffe.

3.4.1 Anfragesprache

Um dieses Ziel zu erreichen wurde eine einfache Anfragesprache entwickelt, die den folgenden, durch die RPD-Anwendungen gestellten Anforderungen, genügt.

1. Vergleich von konkreten Attributen des ASNs mit anderen Attributen oder Fixwerten.
2. Aufbau von komplexen Vergleichen, die mehrere Vergleiche berücksichtigen.
3. Aufbau von Funktionen zur Überwachung einfacher Vorgänge im ASN.

Die Anfragesprache orientiert sich an den in der Programmierung üblichen Bedingungsanweisungen (IF-Statements). Die erste Bedingung, nämlich der Vergleich von konkreten Inhalten, wird in der Anfragesprache durch einfache logische Vergleiche wie =, <>, <=, >=, <, > erreicht. Die zweite Bedingung, der Aufbau von komplexen Vergleichen, wird dadurch erfüllt, dass die einfachen logischen Vergleiche um aussagenlogische Operatoren wie AND, OR, NAND, NOR, XOR sowie die Negation NOT erweitert wurden.

Bei der dritten Bedingung, der Überwachung einfacher Abläufe im ASN, werden drei Gruppen von Funktionen unterschieden. Die erste Gruppe, die Gruppe der Strukturfunktionen, bietet Funktionen zur Überwachung von Struktur und Inhaltsänderungen des ASNs an. Die zweite Gruppe überwacht den Zugriff auf das ASN (Zugriffsfunktionen). Bei der dritten Gruppe von Funktionen handelt es sich um mathematische Funktionen zur Berechnung einfacher Werte. Die Berechnungsergebnisse der Funktionen sind dann Bestandteil der Vergleichsoperationen.

Die Strukturfunktionen sind:

- *ChangeVersion*: Test auf Veränderung der Version eines Konzepts, eines Attributs oder einer Relation.
- *ChangeAttributeValue*: Test auf Veränderung eines Attributwerts.
- *DeleteAttribute* / *DeleteKonzept* / *DeleteNet* / *DeleteRelation*: Test, ob ein Attribut / Konzept / Netz / Relation gelöscht wurde.
- *IncludeConcept*: Test, ob eine Relation ein bestimmtes Konzept enthält.

Die Strukturfunktionen liefern als Berechnungsergebnisse aussagenlogische Werte, die in komplexen Vergleichen zum Einsatz kommen.

Die Zugriffsfunktionen sind:

- *CountRead* / *CountWrite*: Es werden die Lese-Zugriffe / Schreib-Zugriffe auf ein Attribut gezählt.
- *Version*: Es wird die Versionsnummer eines Attributs bestimmt.

Die mathematischen Funktionen sind:

- *Add / Sub / Mul / Div*: Addition / Subtraktion / Multiplikation / Division zweier Werte.

Als Parameter für die mathematischen Funktionen können zum einen Attributwerte aus dem ASN und zum anderen Ergebnisse der Zugriffsfunktionen und mathematische Funktionen verwendet werden.

3.4.2 Beispiel

Im Beispiel Abbildung 9 ist die Anfrage dargestellt, die überprüft, ob ein Mitarbeiter „Person1“ Mitglied des Teams „Entwicklung“ und dessen Status „anwesend“ ist, wobei die Konzepte „Person1“ und „Entwicklung“ dem Netz „Luftdüse“ angehören, das Attribut mit dem Namen „Status“ den Status des Mitarbeiters widerspiegelt und die Relation zwischen Team und Mitarbeiter den Namen „Teamzugehörigkeit“ hat.

```
(IncludeConcept (Luftdüse, Entwicklung, Teamzugehörigkeit) (Luftdüse, Person1)) AND
((Luftdüse, Person1, Status) = String (anwesend))
```

Abbildung 9: Beispiel einer Monitoranfrage

3.5 Koordinations-Agent

Die Zielsetzung des Koordinations-Agenten ist es, Abläufe innerhalb einer Anwendung zwischen Instanzen einer RPD-Anwendung, aber auch zwischen mehreren verschiedenen RPD-Anwendungen zu koordinieren. Da es sich hierbei um Anwendungen unterschiedlicher Domänen handelt und die Koordinationsaufgaben sich ständig ändern, ist ein starr vorgegebenes Koordinationsprotokoll nicht sinnvoll. Daraus folgt, dass der Koordinations-Agent in der Lage sein muss, jedes beliebige Koordinationsprotokoll abarbeiten zu können.

Als flexible Beschreibungsform eines Koordinationsprotokolls wurde das Zustandsübergangsdiagramm gewählt. Die Zustände repräsentieren Situationen im Koordinationsprotokoll, die Übergänge werden mit Hilfe von bestimmten durch die RPD-Anwendung vorgegebenen Nachrichten initiiert und es werden an die RPD-Anwendungen bestimmte, vordefinierte Ausgabenachrichten gegeben. Die Ein- und Ausgabenachrichten bestehen aus einem Textbaustein und im Falle der Eingabenachricht des Senders bzw. Empfängers im Falle der Ausgabenachricht. Dadurch ist es möglich, den Zustandsübergang abhängig vom Sender zu vollziehen und nur bestimmte Koordinationspartner (RPD-Anwendungen) über den Zustandswechsel zu informieren. Natürlich ist es möglich, die Ausgabenachrichten an alle Koordinationspartner zu schicken und die Eingabenachrichten von allen Koordinationspartnern zu akzeptieren. Das Zustandübergangsdiagramm wird im Koordinations-Agenten durch einen endlichen Automaten realisiert.

Bei der Instanziierung des Koordinationsprotokolls entstehen zwei Probleme. Zum einen müssen alle beteiligten RPD-Anwendungen Kenntnis über das Zustands-

übergangsdiagramm und damit über das Koordinationsprotokoll haben und zum anderen muss der benutzte Koordinations-Agent allen Koordinationspartnern und umgekehrt bekannt sein.

Das erste Problem lässt sich nur auf Seiten der RPD-Anwendung lösen. So muss allen an der Koordination beteiligten RPD-Anwendungen das Koordinationsprotokoll und damit das Zustandsübergangsdiagramm und damit die möglichen Ein- und Ausgabenachrichten bekannt sein. Dieses Problem ist angesichts der Tatsache, dass das Koordinationsziel von den RPD-Anwendungen vorgegeben wurde, als gering zu betrachten.

Das zweite Problem, nämlich den richtigen Koordinations-Agenten aufzufinden, wird dadurch gelöst, dass das Koordinationsprotokoll mit einem eindeutigen, allen Koordinationspartnern bekannten Namen versehen wird. Durch Senden einer Anmeldenachricht an das Agentensystem, die den Namen des Koordinationsprotokolls beinhaltet, wird der korrekte Koordinations-Agent gefunden.

3.5.1 Automatendefinition

Das Koordinationsprotokoll wird im Koordinations-Agenten durch einen Endlichen Automaten realisiert. Das dafür benötigte Zustandsübergangsdiagramm wird durch die RPD-Anwendung in Form einer Sprache definiert. Dabei werden zum einen die Zustände mit Namen angegeben und zum anderen alle möglichen Zustandsübergänge mit deren Ein- und Ausgabenachrichten. Es muss genau ein Startzustand und mindestens ein Endzustand vorhanden sein und es müssen alle Zustände vom Startzustand aus erreichbar sein. Des Weiteren muss der Startzustand ungleich dem Endzustand sein, da sonst keine Koordination stattfinden würde. Diese Automatensprache wird benutzt, um den Agenten das Koordinationsprotokoll zu übermitteln.

3.5.2 Ausgetauschte Nachrichten

Neben der Automatendefinition werden zum Betrieb folgende Nachrichten zwischen RPD-Anwendung und Koordinations-Agent ausgetauscht.

- *Input / Output*: Diese Nachricht erzwingt einen Zustandsübergang und verschickt als Ausgabe die „Output“-Nachricht inklusive des neuen Zustandsnamens.
- *Input-Ack*: Diese Nachricht wird an den Sender einer Eingabenachricht als Empfangsbestätigung verschickt. Das ist wichtig, da der Sender nicht zwingend in der Liste der zu empfangenden Ausgabenachrichten enthalten ist.
- *Wrong-Input*: Diese Nachricht wird verschickt, wenn die Eingabe im aktuellen Zustand unzulässig ist.

- *Request-State / Current-State*: Mit dieser Nachricht kann die RPD-Anwendung den aktuellen Zustand beim Koordinations-Agenten abfragen und erhält diesen mit der Nachricht „Current-State“.
- *FSM-Definition / FSM-Definition-Ack*: Mit dieser Nachricht wird der Automat durch die RPD-Anwendung definiert und der Emfpang durch den Agenten mit „FSM-Definition-Ack“ bestätigt.
- *Request-FSM /FSM*: Diese Nachricht dient zur Abfrage des Automaten. Die Anwort ist das Zustandsübergangsdiagramm in der „FSM“-Nachricht.
- *Request-Coordination-Agent / Reqeust-Coordination-Agent-Ack*: Diese Nachricht wird von den RPD-Anwendungen an das Multiagentensystem geschickt, die an der Koordination beteiligt sind, aber nicht das Koordinationsprotokoll festgelegt haben und erhalten vom Agenten die „Request-Coordination-Agent-Ack“-Nachricht als Bestätigung.
- *Initial-Coordination-Agent / Initial-Coordination-Agent-Ack*: Mit dieser Nachricht wird ein neuer Koordinations-Agent erzeugt und mit Initial-Coordination-Agent-Ack quittiert.

3.5.3 Beispiel

Das Beispiel zeigt den Abstimmungsprozess (Abbildung 10), wie er bei der Problemlösung zwischen zwei Partnern auftritt.

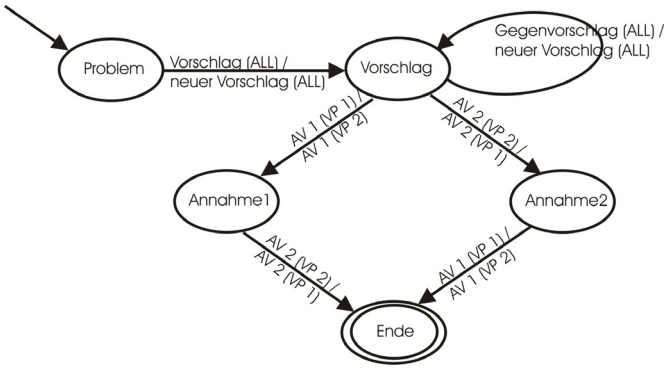

Abbildung 10: Beispiel Koordinationsprotokoll als Zustandsübergangsdiagramm

Vom Startzustand „Problem“ kann einer der beiden Verhandlungspartner (VP) einen Vorschlag machen. Der Eingang des Vorschlags wird beiden Verhandlungspartnern durch die Ausgabenachricht „neuer Vorschlag“ angezeigt und in den Zustand „Vorschlag“ gewechselt. In diesem Zustand können beliebig viele Gegenvorschläge durch einen der beiden Verhandlungspartner vorgenommen werden. Entschließt sich ein Verhandlungspartner, den aktuellen Vorschlag bzw. Gegenvorschlag anzunehmen, wird der Abstimmungsprozess angestoßen. Hierzu wird je

nach Verhandlungspartner in die Zustände „Annahme1“ oder „Annahme2“ verzweigt und dem jeweils anderen Verhandlungspartner die Annahme des Vorschlags (AV) durch eine entsprechende Ausgabenachricht angezeigt. Die Zustände „Annahme1“ und „Annahme2“ können nur noch in Richtung Endzustand, durch die Annahme des Vorschlags durch den jeweils anderen Verhandlungspartner, verlassen werden. Dadurch werden beide Verhandlungspartner zur Akzeptanz des Vorschlags gezwungen bzw. beide Verhandlungspartner haben sich auf diesem Wege abgestimmt.

4 Anwendungsbeispiel

Anhand eines Anwendungsbeispiels soll ein kurzer Überblick über den Einsatz des Agentensystems gegeben werden. Hierzu wird die Arbeitsweise des Projektplaners an einem beliebigen Arbeitstag verdeutlicht. Der erste Schritt des Projektplaners ist das Einloggen am RPD-Portal, dabei wird die Teaminformation und die Stati seiner Kollegen mit Hilfe des Retrieval- und Aggregations-Agenten in einem Schritt aus dem ASN beschafft, anstatt wie bisher viele Anfragen an eine Datenbank zu stellen. Der Projektplaner hat in diesem Beispiel die Aufgabe übernommen zu überwachen, ob der Projektplan für die Erstellung des realen Prototyps für eine Fahrzeug-Luftdüse eingehalten wird. Ist dies nicht der Fall wird eine Koordination der beteiligten RPD-Nutzer notwendig. Ohne den Einsatz der RPD-Middleware wird der aktuelle Projektplan aus dem ASN beschafft. Der Status der Aktivität *Simulation_Prototyp_Luftdüse* wird dann zyklisch überwacht. Ist die Fertigstellung der Simulation termingerecht erfolgt, so wird die Überwachung beendet. Dieses zyklische Überwachen wird durch den Einsatz des Monitor-Agenten vermieden, wodurch die Performance der RPD-Anwendung gesteigert wird. Ist der Termin überschritten wird eine Abstimmung aller beteiligten notwendig, die durch die Koordinations-Agenten informationstechnisch unterstützt wird.

5 Zusammenfassung

Die Middleware wurde als Agentensystem konzipiert und mit dem Ziel realisiert, dass jederzeit durch weitere Agententypen neu auftretende RPD-Problemstellungen angegangen werden können. Des Weiteren wurde bei den Arbeiten darauf geachtet, dass der Zugang zu der Middleware über standardisierte Schnittstellen erfolgt. Durch den Einsatz der Agenten als Werkzeuge wird die Zusammenarbeit entlang des RPD-Prozesses unterstützt und vereinfacht. So realisiert der Koordinations-Agent strukturierte Abstimmungsverfahren zwischen RPD-Nutzern, ohne dass diese sich treffen bzw. sich gleichzeitig über ein beliebiges Medium austauschen müssen. Mit Hilfe des Monitor-Agenten ist eine flexible und proaktive Re-

aktion auf vollzogene Tätigkeiten anderer RPD-Nutzer möglich. Durch den Einsatz des Retrieval- und Aggregations-Agenten ist die RPD-Anwendung in die Lage versetzt worden, neben den eigenen Informationen, Informationen anderer aufzufinden und aufzubereiten ohne dabei detaillierte Kenntnisse über die Struktur der Informationen besitzen zu müssen. Darüber hinaus ist es nicht nur möglich einzelne Informationen aus dem ASN zu beschaffen, sondern weiterführende Informationen zu einer gesuchten Information zu erhalten. Die kontextabhängige Aufbereitung von Wissen erwies sich als schwierig, da bei der Vielzahl und den großen Unterschieden der RPD-Domänen die Darstellungsformen stark variieren. Durch die Gruppierung und Zuordnung von Ergebnissen zu ihren Suchanfragen wurde ein erster Schritt in Richtung Wissensaufbereitung vollzogen. Die Nutzung der RPD-Middleware stellt einen entscheidenden Mehrwert dar, so müssen nicht mehr die Informationen über eine Vielzahl von Interaktionsschritten aus dem ASN beschafft werden. Ebenso ist ein zyklisches Abfragen des ASNs auf Veränderungen genauso überflüssig geworden, wie das Aushandeln und Setzen von Parametern im ASN um eine einfache Synchronisationskoordination zu erreichen.

Literatur

[Able03] Agent Building and Learning Environment, : URL: http://www.alphaWorks.ibm.com/tech/able, Abruf am 2004-06-29.

[Dala+04] Dalakakis, S., Stoyanov, E., Roller, D.: A Retrieval Agent Architecture for Rapid Product Development, in: Perspectives from Europe and Asia on Engineering Design and Manufacture, EASED 2004, X.-T. Yan, Ch-Y. Jiang, N. P. Juster, (eds.), Kluwer Academic Publishers, 2004, S. 41-58.

[DiLe01] Diederich, M. K.; Leyh, J.: Agent Based Middleware to Coordinate Distributed Development Teams in the Rapid Product Development. In: Proceedings for the Conference on Product Development, 1st Automotive and Transportation Technology Congress and Exhibition (1st ATTCE), 2001, Barcelona, Spain.

[FIPA02a] FIPA Abstract Architecture Specification : Foundation for Intelligent Physical Agents, 2002. URL: http://www.fipa.org/specs/fipa00001, Abruf am 2004-09-29.

[FIPA02b] FIPA ACL Message Structure Specification : Foundation for Intelligent Physical Agents, 2002. URL: http://www.fipa.org/specs/fipa00061, Abruf am 2004-09-29.

[FIPA02c] FIPA Communicative Act Library Specification : Foundation for Intelligent Physical Agents, 2002. URL: http://www.fipa.org/specs/fipa00037, Abruf am 2004-09-29.

[FIPA03] FIPA-OS : Foundation for Intelligent Physical Agents Open Source from Nortel Networks. URL: http://fipa-os.sourceforge.net/summary.htm, Abruf am 2004-09-29.

[GeKe94] Genesereth, M. R.; Ketchpel, S. P. : Software Agents. In: Communications of the ACM, (1994), Vol. 37 (7), S. 48-53.

[Bigu+02] Bigus, J. P., Schlosnagle, D.A., Pilgrim, J.R., Mills, W.N., Diao, Y. : IBM Systems Journal, (2002), Vol. 41, No. 3.

[IEEE802.5] IEEE 802.5: Tokenring.

[Jade03] JADE - Das Java Agent Development Framework : Telecom Italia Lab. URL: http://sharon.cselt.it/projects/jade/, Abruf am 2004-06-29.

[Java00] Java Shared Data Toolkit : Sun Microsystems Inc. URL: http://java.sun.com/products/java-media/jsdt/, Abruf am 2004-06-29.

[Liao00] Liao, T.: Light-weight reliable Multicast Protocol. INRIA, Rocquencourt, BP 105 / 8153 Le Chesnay Cedex, France URL: http://webcanal.inria.fr/lrmp/, Abruf am 2004-06-29.

[Matu95] Maturana, F. P.; Norrie, D. H.: A Multi-Agent Coordination Architecture for Distributed Organizational Systems. In: The Mediator Research Program. 1995.

[Rdf04] RDF Primer,W3C Recommendation 10 February 2004 : URL: http://www.w3.org/TR/rdf-primer/, Abruf am 2004-09-29.

[Sema04] Semantic Web, : URL: http://www.w3.org/2001/sw/, Abruf am 2004-09-29.

[Sun03] JXTA:Sun Microsystems Inc.: URL: http://services.jxta.org/servlets/ProjectHome, Abruf am 2004-06-29.

[Swif01] SwiftMQ : URL: http://www.swiftmq.com/products/index.html, Abruf am 2004-06-29.

[Wars+02] Warschat, J.; Cebulla, T.; Dangelmaier, M.; Diederich, M.; Aslanidis, S.; Tippmann, V.:Systemumgebung einer multidisziplinär wissensbasierten Produktentwicklung. In: Nagl, M.; Westfechtel, B. (Hrsg.) Modelle, Werkzeuge und Infrastrukturen zur Unterstützung von Entwicklungsprozessen, 20.-22. März 2002 in Aachen. Aachen, Wiley-VCH, S. 293-307.