Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

Document
D-01-01

FRODO: A Framework for Distributed Organizational Memories

Milestone M1:

Requirements and System Architecture

**A. Abecker, A. Bernardi, L. van Elst,
A. Lauer, H. Maus, S. Schwarz, M. Sintek**

March 2001

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH

Postfach 2080
67608 Kaiserslautern, FRG
Tel: +49 (631) 205-3211
Fax: +49 (631) 205-3210
E-Mail: info@dfki.uni-kl.de

Stuhlsatzenhausweg 3
66123 Saarbrücken, FRG
Tel: +49 (631) 302-5252
Fax: +49 (631) 302-5341
E-Mail: info@dfki.de

WWW: http://www.dfki.de

Deutsches Forschungszentrum für Künstliche Intelligenz

DFKI GmbH

German Research Centre for Artificial Intelligence

Founded in 1988, DFKI today is one of the largest non-profit contract research institutes in the field of innovative software technology based on Artificial Intelligence (AI) methods. DFKI is focusing on the complete cycle of innovation — from world-class basic research and technology development through leading-edge demonstrators and prototypes to product functions and commercialisation.

Based in Kaiserslautern and Saarbrücken, the German Research Centre for Artificial Intelligence ranks among the important "Centres of Excellence" world-wide.

An important element of DFKI's mission is to move innovations as quickly as possible from the lab into the marketplace. Only by maintaining research projects at the forefront of science can DFKI have the strength to meet its technology transfer goals.

DFKI has about 165 full-time employees, including 141 research scientists with advanced degrees. There are also around 95 part-time research assistants.

Revenues for DFKI were about 30 million DM in 2000, from government contract work and from commercial clients. The annual increase in contracts from commercial clients was greater than 20% during the last three years.

At DFKI, all work is organised in the form of clearly focused research or development projects with planned deliverables, various milestones, and a duration from several months up to three years.

DFKI benefits from interaction with the faculty of the Universities of Saarbrücken and Kaiserslautern and in turn provides opportunities for research and Ph.D. thesis supervision to students from these universities, which have an outstanding reputation in Computer Science.

The key directors of DFKI are Prof. Wolfgang Wahlster (CEO) and Dr. Walter Olthoff (CFO).

DFKI's five research departments are directed by internationally recognised research scientists:

- Knowledge Management (Director: Prof. A. Dengel)
- Intelligent Visualisation and Simulation Systems (Director: Prof. H. Hagen)
- Deduction and Multiagent Systems (Director: Prof. J. Siekmann)
- Language Technology (Director: Prof. H. Uszkoreit)
- Intelligent User Interfaces (Director: Prof. W. Wahlster)

In this series, DFKI publishes research reports, technical memos, documents (e.g. workshop proceedings), and final project reports. The aim is to make new results, ideas, and software available as quickly as possible.

Prof. Wolfgang Wahlster

Director

FRODO: A Framework for Distributed Organizational Memories

Milestone M1: Requirements Analysis and System Architecture

**A. Abecker, A. Bernardi, L. van Elst,
A. Lauer, H. Maus, S. Schwarz, M. Sintek**

DFKI-D-01-01

This work has been supported by a grant from The Federal Ministry of Education, Science, Research, and Technology (FKZ ITW-01 IW 901).

ISSN 0946-0098

Abstract

This document collects and presents the results of the requirements analysis and system design phase of the FRODO project. FRODO aims at developing a comprehensive technology for distributed organizational memories (OMs), comprising an agent-oriented framework as the software basis and a suitable methodology for its introduction in the enterprise. We analyze existing approaches both to OM methodology and to the IT support of distributed information organization and knowledge support for weakly-structured processes. We identify possible contributions and shortcomings and detail the expected functionality of distributed OMs, together with the requirements for an integrated OM methodology and for the envisioned software support. We examine existing agent-based approaches and frameworks and evaluate their usability in the FRODO setting. We elaborate on the problems encountered and the solutions envisioned in the area of information organization and ontology construction and maintenance. We conclude with the design specifications for the FRODO system which will cover the requirements identified above, and with a sketch of the application scenario which will serve as a testbed for the feasibility and usefulness of the FRODO achievements.

FRODO: A Framework for Distributed Organizational Memories

Milestone M1:

Requirements Analysis and System Architecture

A. Abecker, A. Bernardi, L. van Elst,
A. Lauer, H. Maus, S. Schwarz, M. Sintek

Contents

List of Figures

Kapitel 1

Einleitung

Effektives Wissensmanagement im betrieblichen Bereich ist allgemein als eine Aufgabe von zentraler Bedeutung erkannt und wird in vielfältiger Weise angestrebt. Das am DFKI durchgeführte Forschungsprojekt *KnowMore - Knowledge Management for Learning Organizations* hat in diesem Zusammenhang zu zentralen Themen Ergebnisse geliefert.

Die wichtigsten Ergebnisse und Begriffsbestimmungen von *KnowMore* können wie folgt zusammengefasst werden (vgl. [Abecker *et al.*, 1998c, Abecker *et al.*, 2000a]):

- *KnowMore* versteht IT-Beiträge zum Wissensmanagement im betrieblichen Umfeld als ein Mittel, Menschen bei der Bearbeitung wissensintensiver Aktivitäten informationstechnisch zu unterstützen.
- Das zu diesem Zweck relevante Wissen ist in einem *Organizational Memory* (OM) zusammengefasst, das vielfältige heterogene Informationsquellen kombiniert.
- Durch expliziten Anwendungsbezug wird Information zu Wissen. Das heißt, dass explizite Beziehungen zwischen Informationselementen und geeigneten Modellen und Strukturen repräsentiert werden müssen, um so eine Ableitung von Aktionen aufgrund der Information zu ermöglichen.
- Die wesentlichen Modelle, die zur Darstellung des Anwendungsbezugs von Informationen dienlich sind, fallen nach *KnowMore* in die folgenden Klassen (vgl. [Abecker *et al.*, 1998a, Abecker *et al.*, 1998b]):
  - Modelle der Dinge, Ontologien. Diese repräsentieren begriffliches Wissen über den Gegenstandsbereich, in welchem eine Anwendung bzw. das unternehmerische Handeln der betrachteten Organisation angesiedelt ist, und erlauben so, domänenbezogen den Inhalt von Informationsquellen explizit zu machen.
  - Modelle der statischen Unternehmensstrukturen. Rollen, Organisation, Entscheidungsstrukturen und ähnliches stellen die unternehmenstypischen Grundbausteine bzw. den organisationsspezifischen Rahmen aller Geschäftsprozesse dar. Auf ihrer Basis können Erzeuger und Nutzer bzw. Erzeugungs- und Nutzungskontext von Informationselementen explizit gemacht werden.
  - Modelle des dynamischen Verhaltens. Aktivitäten im Unternehmen können in Form von Prozessmodellen repräsentiert werden. Damit ist es möglich, den expliziten Bezug der Informationselemente zu Aufgaben, Tätigkeiten und Zielen herzustellen.

- Die statischen Modelle – Domänenontologie zur Strukturierung des Gegenstandsbereichs, Unternehmensontologie und -modelle zur Repräsentation von statischen und dynamischen Organisationsstrukturen, strukturelle Beschreibung von Informationsquellen durch eine Informationsontologie – sind die Grundlage von Attributierungen aller Informationselemente. Alle automatischen Operationen können sich auf diese Grundlagen beziehen.
- Zur Erstellung der Ontologien und zur Verknüpfung zwischen Informationselementen und Modellen wurden unterstützende Werkzeuge entwickelt (vgl. auch [Abecker *et al.*, 2000c]).
- *KnowMore* geht von einer durchgängigen Modellierung der unterstützten Geschäftsprozesse in Prozessmodellen aus. Diese Prozessmodelle können als Workflows verstanden und von entsprechenden Workflowsystemen operationalisiert werden (siehe [Abecker *et al.*, 2000b]).
- Der aktuelle Zustand eines Workflowsystems wird als dynamischer Teil des Kontexts der gerade bearbeiteten Aufgabe gesehen.

Auf dieser Basis realisierte *KnowMore* die aktive Unterstützung des Menschen bei der Bearbeitung wissensintensiver Aktivitäten: Wenn innerhalb des Workflowsystems eine als solche modellierte wissensintensive Aktivität gestartet wird, triggert das System gleichzeitig einen Informationsagenten. Unter Verwendung des dynamischen Kontexts aus dem Workflowsystem und des statischen Hintergrundwissens aus den verschiedenen Ontologien wird ein intelligentes Information Retrieval in den verschiedenen Informationsquellen durchgeführt. Die so ermittelte relevante Information wird dem Menschen automatisch präsentiert. Gleichermaßen wird neu entstehendes Material unter Berücksichtigung des Kontexts korrekt in das Organizational Memory eingestellt [Abecker *et al.*, 2001].

Die Architektur eines Organizational Memory umfasst folglich vier Ebenen (siehe Abbildung 1.1):

- *Application Level*: Die Anwendungsebene definiert den Kontext der aktuell relevanten Information und triggert die aktive Informationsbereitstellung. Nur der explizite Anwendungsbezug erlaubt es, von Wissensmangement zu sprechen.
- *Knowledge brokering level*: Geeignet implementierte Information Agents (im Bild als *knowledge access layer* bezeichnet) liefern und speichern die Information entsprechend den Parametern für Informationsverwendung (Speicherung, Zugriff) und Informationskontext aus der Anwendungsebene.
- *Knowledge Description Level*: Die Attributierung der Informationsquellen unter Bezugnahme auf die expliziten Modelle und Ontologien ist die Grundlage für die Arbeit der Information Agents.
- *Knowledge Object Level*: (in der Abbildung als *source layer* bezeichnet) enthält die verschiedenen Informationselemente bzw. -quellen, auf die die Information Agents zugreifen können.

FRODO behält die Ergebnisse aus *KnowMore* bei, baut darauf auf und erweitert sie in verschiedenen Bereichen. Ausschlaggebend ist dabei die Beobachtung, dass einige Festlegungen

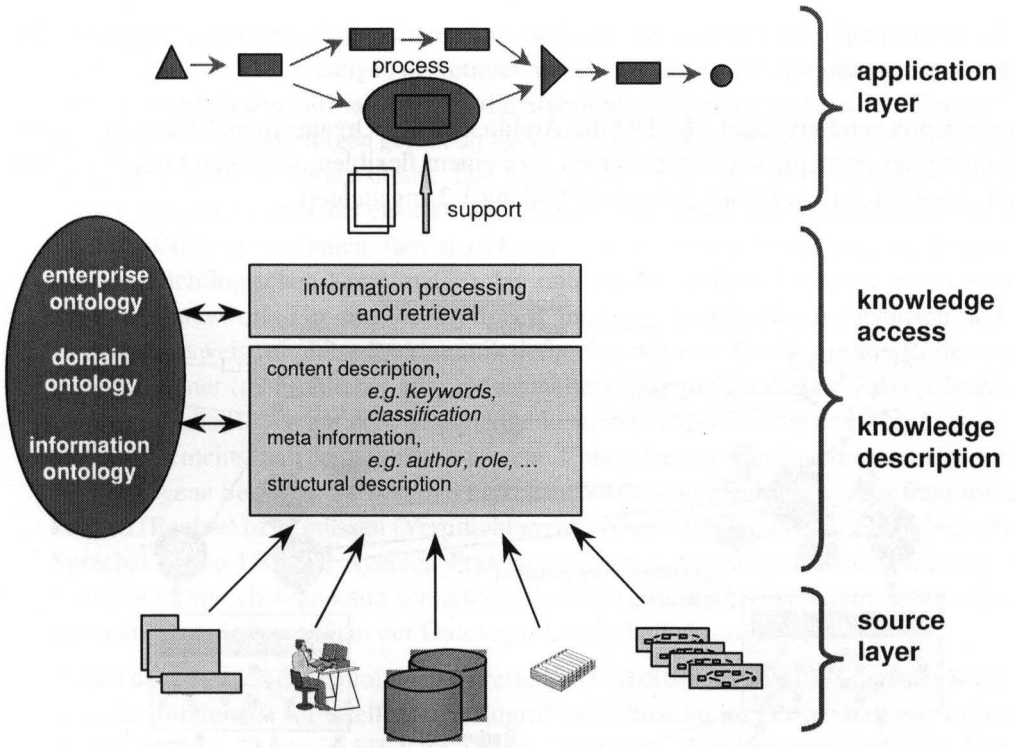

Abbildung 1.1: Die KnowMore Vier-Ebenen-Systemarchitektur für Organizational Memories.

des *KnowMore*-Projekts für interessante Wissensmanagement-Anwendungen zu eng gefasst sind. Insbesondere wird in *FRODO* berücksichtigt:

- Die Vision eines einzigen, zentralen Organizational Memory entspricht zumindest nicht den Gegebenheiten in der Aufwuchsphase eines unternehmensweiten Wissensmanagements. Vielmehr werden typischerweise an verschiedenen Stellen (Bereichen, Abteilungen) separate OMs aufgebaut und im Sinne des 'quick win' eingesetzt und genutzt. Damit ergibt sich die Aufgabe, die intendierte Unterstützung wissensbasierter Aufgaben auf der Basis mehrerer verteilter OMs zu realisieren.

- An die Stelle homogener, zentraler Ontologien tritt aus den gleichen Gründen eine Sammlung mehrerer, verteilter Modelle und Ontologien, die in ihrer Gesamtheit üblicherweise nicht völlig konsistent sind. Damit ist zu untersuchen, wie Aufbau, Wartung und Nutzung dieser Ontologien und Modelle unter den neuen Gegebenheiten machbar sind.

- Schließlich erweist sich die Vorstellung eines a priori durchmodellierten Geschäftsprozesses als für interessante wissensintensive Aktivitäten ungeeignet. Gerade die nicht durch Routine geprägten wissensintensiven Tätigkeiten sind vielmehr als ad-hoc-Folge von Einzelschritten zu verstehen, deren Reihenfolge und Auswahl bereits großes Wissen benötigt, nicht im voraus festgelegt werden kann, und erst bei der Erledigung der Aufgabe durch den Menschen realisiert wird. Damit ergeben sich neue Randbedingungen für

die Verwendung von Prozess- und Workflowinformation als Kontext und Trigger für den Informationssupport.

Dementsprechend erweitert FRODO die Architektur des Organizational Memory – unter Beibehaltung der prinzipiellen Komponenten – zu einem flexiblen, verteilten Framework mit Kommunikation auf allen Ebenen (wie in Abbildung 1.2 angedeutet):

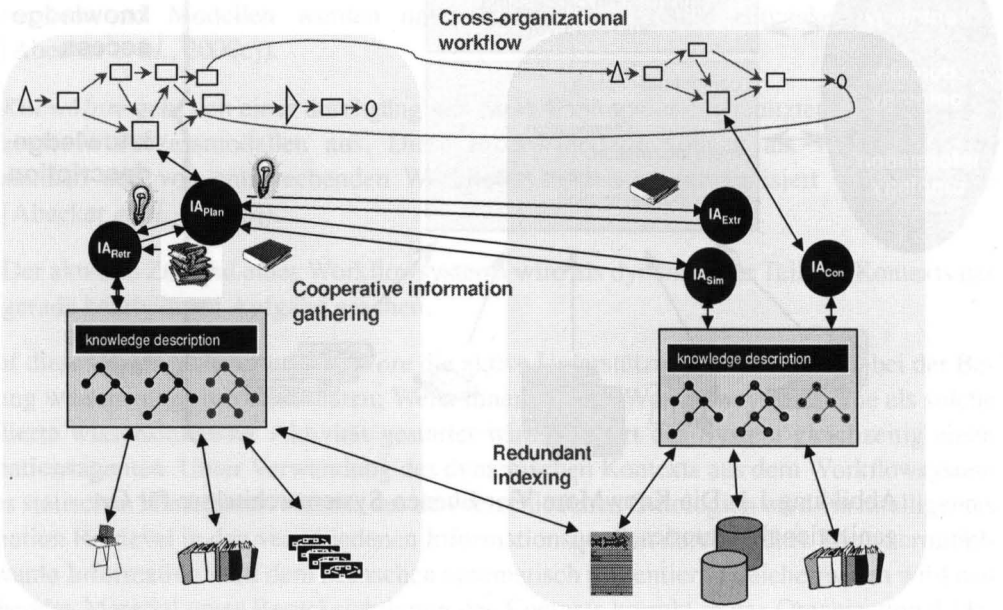

Abbildung 1.2: Beispiele für Kommunikationsströme in Multi-OM-Szenarien.

Das gemeinsame betriebliche Umfeld und die damit einhergehenden gemeinsamen Globalziele erlauben es, bei der Untersuchung der verschiedenen verteilten Szenarien und Kommunikationsaspekte von einem Grundkonsens aller Beteiligten auszugehen. Diese Beobachtung ist für den FRODO-Ansatz essentiell und wird im folgenden mehrfach betont werden.

Die nachfolgenden Kapitel dieses Dokuments untersuchen die Grundlagen und Anforderungen, die sich aus den hier skizzierten Erweiterungen der Grundannahmen ergeben. Aufgrund der so definierten Anforderungen wird eine umfassende Systemarchitektur präsentiert. Die zu ihrer Realisierung notwendigen technischen Ressourcen werden detailliert und in Bezug gesetzt zu existierenden Plattformen und Lösungen; notwendige Erweiterungen werden benannt.

Im einzelnen diskutieren wir die folgenden Themen:

- In Kapitel 2 **(Distributed Organizational Memories)** untersuchen wir grundsätzliche Typen von Skalierbarkeits- und Interoperabilitätsanforderungen, die sich aus der räumlich/logischen Verteiltheit und zeitlichen Evolution von Organizational Memories in betrieblichen Anwendungsszenarien ergeben. Vertikale Skalierbarkeit und horizontale Skalierbarkeit werden als Basisanforderungen verteilter OMs eingeführt und am Beispiel

erläutert.

Im Lichte dieser Zielsetzungen diskutieren wir Dimensionen der Informationscharakterisierung, nämlich den Formalitätsgrad, die Stabilität und das Ausmaß, in dem Formalisierungen und Strukturierungen geteilt und gemeinsam verwendet werden, sowie die Wechselwirkungen dieser Dimensionen. Die Erfordernisse, zur flexiblen und ökonomischen Umsetzung obiger Ziele bei einer konkreten Systemausgestaltung gut entlang dieser drei Achsen skalieren zu können, führen zu komplexen Szenarien hinsichtlich der zeitlichen und räumlich/logischen Koordination von ontologiebezogenen Diensten im angestrebten OM-Framework. Für eine konzeptionell, methodisch und softwaretechnisch saubere Herangehensweise an diese Problematik definieren wir eine Ontologie von Rollen ontologiebezogener (menschlicher oder maschineller) Akteure. Zu diesem Zweck charakterisieren wir die Rollen auf dem Knowledge Level und fügen Rechte und Verpflichtungen als eine Gemeinschaft konstituierende neue Charakterisierungsdimensionen ein. Ontologiebezogene Services, die von den einzelnen Rollen angeboten bzw. angefragt werden können (Rechte) bzw. müssen (Verpflichtungen) zeigen den Weg zu ontologiebezogenen Sprechakten im FRODO Agenten-Framework. Auf der Basis von ontologiebezogenen Rollen und Sprechakten kann schließlich die im weiteren Projektverlauf umzusetzende Agentenlandschaft aus Sicht der Ontologie-Landschaft skizziert werden.

Neben dem Bereich der Ontologien in verteilten OM-Szenarien ist die Frage der Nutzung formaler Inferenzen für intelligenten Zugriff und Nutzung im OM enthaltener Informationen zentrales Thema in FRODO. Dieses Thema hat zwei herausfordernde Teilaspekte: Einerseits muss man in realen Anwendungsszenarien davon ausgehen, dass häufig aus ökonomischen wie auch prinzipiellen Gründen es wünschenswert wäre, Informationen sehr unterschiedlichen Formalitätsgrades gemeinsam zu verarbeiten. Andererseits erfordert die dem Szenario in natürlicher Weise innewohnende hochgradige Verteiltheit neue Mechanismen der Inferenz. Wir zeigen, wie man beide Herausforderungen durch einen homogenen Lösungsansatz zur agentenbasierten formal-informalen Inferenz angehen kann. Im innersten Kern dieses Lösungsansatzes ist eine logische Sprache zur Darstellung und Verarbeitung von Agentenzuständen angesiedelt, die im darauffolgenden Kapitel 3 im Detail besprochen wird. Als Implementierungsbasis für solche Agenten und ihre Kommunikationsprotokolle wählen wir eine existierende Agentenplattform, deren konkrete Wahl auf der Grundlage des zuvor erstellten Anforderungsprofils in Kapitel 6 beschrieben wird.

- In Kapitel 3 **(TRIPLE – An RDF Query, Inference, and Transformation Language)** geben wir die Basisdefinitionen der in Zusammenarbeit mit der Stanford Database Group entwickelten Sprache TRIPLE, die speziell zur Anfrage und Transformation von RDF-Modellen gestaltet wurde und sich damit für das Semantic Web im allgemeinen, aber auch für Szenarien verteilter Organisationsgedächtnisse eignet, die in den relevanten Charakteristika (verteilte heterogene Informationsquellen, ontologische Annotation, ontologiebasierte Vermittlungs- und Übersetzungsdienste) hier als Spezialfälle des Semantic Web mit hoher ökonomischer Relevanz betrachtet werden können. Dabei führt die Fokussierung auf innerbetriebliche Anwendungsszenarien zu Einschränkungen (aber auch Ergänzungen) des allgemeinen Semantic-Web-Szenarios, die intelligente Dienste und wissensbasierte Lösungen sogar machbarer und potentiell nützlicher erscheinen lassen.

Wir stellen kurz die Syntax von TRIPLE vor, wobei a priori nicht eine einzige Semantik fest vorgegeben wird, sondern vielmehr ein Formalismus zur Verfügung gestellt wird, um vielfältige Semantiken unterliegender Repräsentationssprachen (Repräsentations- und Domänenontologien) einzubinden und zur adäquaten gemeinsamen Verarbeitung auf der Metaebene zu charakterisieren. Als Beispiele skizzieren wir die Einbindung von Dublin Core und von RDF Schema zur Charakterisierung von Informationsquellen. Der Weg zur in Arbeit befindlichen Implementierung und zur modelltheoretischen Fundierung der gemeinsamen Verarbeitung unterschiedlich semantisch fundierter Repräsentationssprachen ergibt sich durch die Rückführung von TRIPLE auf Hornlogik.

- In Kapitel 4 (**Applikations-Level**) beschreiben wir die Ebene der Aufgabenanbindung in FRODO, welche den in KnowMore gewählten Weg der Kopplung mit existierenden Workflow-Ansätzen erweitert zur Spezifikation und Ausführung schwach-strukturierter Workflows zur Umsetzung wissensintensiver Tätigkeiten.

  Hierzu fassen wir noch einmal (auf der Basis einer Analyse der existierenden Literatur, wie auch eigener Überlegungen) wesentliche Charakteristika wissensintensiver Tätigkeiten zusammen und leiten Anforderungen an die Systemunterstützung für solche Tätigkeiten ab. Diese Anforderungen werden in Entwurfsentscheidungen für das FRODO Workflow-Tool umgesetzt, deren konkrete Umsetzung dann im Detail erläutert wird. Wir führen einen alternativen, wissensorientierten Workflow-Begriff ein, dessen Lebenszyklusmodell im Gegensatz zum traditionellen Workflow stärker einzelfallorientiert ist, stärker auf kontinuierliche Prozessevolution abstellt, und den Power User zentraler im Mittelpunkt des Geschehens sieht. Primelemente dieses alternativen Workflow-Begriffs sind die in einer Task Konzept-Ontologie beschriebenen Aufgaben, deren semiformale Beschreibungen den Nutzer zur Laufzeit zu abgespeicherten möglichen Prozessbausteinen führen. Zur technischen Umsetzung eines solchen Workflow-Begriffs mit verschränkter Modellierung und Ausführung in Form einer dynamischen, schrittweisen, hierarchischen Task-Dekomposition mit eingebundener kontextabhängiger Informationslieferung, haben wir die erfoderlichen Datenstrukturen, Agententypen und Sprechakte identifiziert, sowie hieraus weitere Anforderungen an die unterliegende Agentenplattform abgeleitet. Eine kurze Diskussion verwandter Arbeiten zeigt schließlich, dass einige existierende Systeme interessante Beiträge zur konkreten Ausgestaltung des FRODO-Workflows liefern können, der von uns angestrebte Arbeitspunkt aber zur Zeit noch nicht besetzt ist.

- In Kapitel 5 (**Informal-Formal Transitions**) identifizieren wir noch einmal die wesentlichen Einsatzpunkte zur Verwendung von Verfahren zur Informal-Formal-Überführung im FRODO-Szenario. Zwei wesentliche Aufgaben sind hier zu nennen. Eine ist die Umsetzung (von Teilen) informeller Repräsentationen in formale, im einfachsten Falle die Indexierung oder Klassifikation, in komlizierteren die Informationsextraktion in ein formales Modell hinein bzw. die Teilformalisierung von Dokumentinhalten. Die andere ist die Analyse informeller Darstellungen (insbesondere Textdokumente) zur Gewinnung von Hinweisen für die Erstellung und Wartung formaler Modelle (d.h. insbesondere der Ontologien).

  Wir werden in FRODO keinen Schwerpunkt auf die Neuentwicklung solcher Verfahren legen, sondern primär den innovativen Einsatz existierender Algorithmen aus dem

Partnerprojekt ADAPTIVE READ und existierender Software-Tools des DFKI-Spin-Offs Insiders Information Management GmbH untersuchen. Da der Zweck des vorliegenden Dokuments primär die Erfassung von Anforderungen für Agentenplattform und Basisrepräsentation war und sich für diese Bereiche aus dem Informal-Formal-Übergang kaum Implikationen ergeben, spielt dieses Thema im vorliegenden Dokument keine besondere Rolle. Die konkreten Arbeiten in diesem Bereich umfassen zur Zeit überwiegend die Gestaltung geeigneter generischer Schnittstellen zur Einbindung geeigneter Moduln in das Gesamtkonzept. In weiteren Phasen der FRODO-Arbeiten wird ein wissenschaftliches Schwerpunktinteresse hier auf der Gestaltung lernender und textbasierter Ontologieerstellungs- und -wartungsverfahren liegen.

- In Kapitel 6 (**Agentenplattformen**) fassen wir zunächst die aus den vorangegangenen Betrachtungen resultierenden Anforderungen an die Agenten-Basisplattform zusammen. Diese umfassen die Bereiche Agentenverwaltung, Agentenkommunikation, Agentenverhalten, sowie Benutzerinteraktion. Einige weitere Kriterien, die üblicherweise zur Bewertung von Agentenplattformen herangezogen werden, können für FRODO als unerheblich betrachtet werden. Danach geben wir eine kurze Einführung in den FIPA-Standard, der der zu benutzenden Plattform unterliegen sollte. Auf dieser Basis diskutieren wir dann die Plattformen FIPA OS, JADE, Grashopper und ZEUS, die aus einer groben Vorauswahlphase als interessanteste Kandidaten hervorgegangen waren, im einzelnen. Es stellt sich heraus, dass die JADE Plattform die besten Voraussetzungen hat, als Basis der Software-Entwicklung in FRODO zu dienen, insbesondere wegen der großen Flexibilität des Systems.

- In Kapitel 7 (**Methodologie**) untersuchen wir verschiedene Methoden mit Hinblick auf das in FRODO anvisierte prozessorientierte Wissensmanagement. Dazu unterscheiden wir drei Phasen der Systemgestaltung (Systemdesign, Systemnutzung und Systemevolution) für die jeweils spezialisierte Methoden existieren. Wir stellen diese vor und diskutieren ihre Nützlichkeit im FRODO-Kontext. Aus dieser Betrachtung wird ersichtlich, dass durchgängige Methoden für ein prozessorientiertes Wissensmanagement im Sinne von FRODO zur Zeit fehlen. Daher präsentieren wir einen Ansatz für eine umfassende Methode, die – ausgehend von dem KnowNet-Framework – bereits etablierte Ansätze für bestimmte Teilbereiche kombiniert und durch FRODO-spezifische Aspekte erweitert. So wird etwa in der Designphase zur Modellierung der Workflows die aspekten-orientierte Analyse für Workflow-Anwendungen benutzt, jedoch um Methoden zur Bereitstellung eines dynamischen Prozesskontextes ergänzt.

- In Kapitel 8 (**Anwendung**) wird ein reales Anwendungsszenario präsentiert, das die in FRODO postulierten charakteristischen Eigenschaften besitzt und somit als ideales Umfeld zur Evaluation von FRODO-Ergebnissen dienen kann.

Kapitel 2

Distributed Organizational Memories

2.1 Framework Overview

The vision of knowledge management comprises the comprehensive use of an enterprise's knowledge, whoever acquired it, wherever it is stored and however it is formulated in particular. For a technical support of knowledge management, *centralized approaches* seem to be well–suited to guarantee that the *complete* information available is considered. Architectures for Organizational Memories often postulate a global view on the information sources [Bonifacio *et al.*, 2000]. In KnowMore for example, the problem of several heterogenous information sources is tackled by the introduction of a uniform *knowledge description level*. The various information items are annotated by knowledge descriptions which are based on an agreed upon vocabulary, namely the information, enterprise, and domain ontologies. Hence, a centralized view on a distributed information landscape is built.

Such centralized approaches have drawbacks with respect to two aspects: i) They neglect the advantages of the distributed nature of knowledge (e.g. with respect to development and use) in enterprises. It is very expensive or even impossible to obtain a globally negotiated vocabulary. OMs could benefit from balancing both *local expertise*—which may result in not globally shareable knowledge—and *overall views* on higher levels. ii) Centralized approaches are cumbersome in changing environments. An OM's environment may for example change due to reorganizations of an enterprise's structure. Furthermore, OM systems are typically not established at once for a whole company, but introduced separately (in terms of time and space) in various places (e.g. departments). To allow for a comprehensive management of knowledge these OMs have to cooperate or to be integrated.

To provide a flexible, scalable framework for Organizational Memory technology, two types of scalability should be supported:

- *Vertical scalability* describes the abilty of evolutionary growth within *one* Organizational Memory. New services may be integrated in order to meet additional requirements, or more legacy systems have to be incorporated into the Organizational Memory.
- *Horizontal scalability* means the cooperation between independently introduced Organizational Memories within one enterprise[1]. For example, there may exist separate Organizational Memories for different departments of an enterprise (design, production,

[1] Of course, the same techniques will be even more relevant when considering cross-organizational communication, services, and workflows. It is more than probable that such scenarios will become more and more important

costumer relationship management, etc.). In order to "globally optimize" a complete product lifecycle, information has to be used *across* the departments. Therefore, communication means *between* several Organizational Memories are needed.

Vertical Scalability

When establishing Organizational Memory technology in a company, often not a full solution is implemented at once, but partial functionalities are introduced and then extended. For example, only one element in a special business process is supported by a proactive information delivery service. Hence, an ontology for the relevant aspects of the domain is developed and a few information sources are wrapped or annotated in terms of this ontology. A dedicated information agent is designed that realizes the information service (e.g. document retrieval and ranking). After a while this system may be extended: i) Additional processes have to be supported. ii) Additional internal or external sources shall be used. iii) The domain ontology has to be extended. iv) New services are needed, e.g. not only document retrieval but also summarization.

Levels for Horizontal Integration

Integration should be possible on all levels:

- *Object Level*: Here, elements of information sources are shared between different Organizational Memories. If these information sources are disparate and OM_1 just passively uses a source of OM_2 this source can be added similarly to vertical integration: The information items just have to be described with the vocabulary (KDL_1) of OM_1. On the other hand, if OM_1 also needs writable access to a information source of OM_2, a more complex synchronisation procedure is needed because changes in OM_2's sources may be propagated to its application level. For the same reason, there is always a complex synchronisation mechanism[2] needed when two Organizational Memories share *the same* information source: Changes performed by one OM may affect information support for the other OM's application level.

- *Knowledge Description Level*: Cooperation between Organizational Memories on the knowledge description level is typically provided by ontology integration or ontology mapping services. Often this is the basis to allow for cooperation on higher levels, i.e. to allow for communication between intelligent information agents. Of course ontology mapping may also be used to directly access "the right/intended" objects on a different OM's object level (if direct access is allowed).

- *Knowledge Brokering Level ("Information Agents")*: If two Organizational Memories cooperate on the knowledge brokering level, they provide each other complex services e.g. in form of information agents. Mechanisms for cooperative information gathering

in the networked economy of the near future. Two trends into this direction are the idea of virtual enterprises, and the concept of the extended enterprise. Though being a very specific example, the FRODO application example presented in chapter 8 exhibits characteristics of both concepts.

[2]Research on database technology has developed synchronisation and locking mechanisms to achieve consistency for distributed access to information objects. However, these mechanisms do not ensure consistency on a semantic level so that further coordination has to be added.

or intelligent information integration are examples for a horizontal OM–integration on the knowledge brokering level. Normally, such services presuppose ontology integration services.

- *Application Level*: There are (at least) two situations where integration on the application level is needed:
  i) There is one process that crosses the boundaries of an Organizational Memory. Typical examples are globally operating companies or virtual enterprises where each site deploys one Organizational Memory. Here cross-organizational workflows are an approach for facilitating integration on the application level.
  ii) Second–order processes are considered. For example, dedicated knowledge management activities can be realized as second–order processes. Such processes would complement the particular local optimizations of the different Organizational Memories by a superordinate view that allows for *global* optimization.

These types of cooperation between different Organizational Memories allows to see a "snapshot of an actual use case" as an *ad hoc configuration of a virtual Organizational Memory*, similar to a view in database systems.

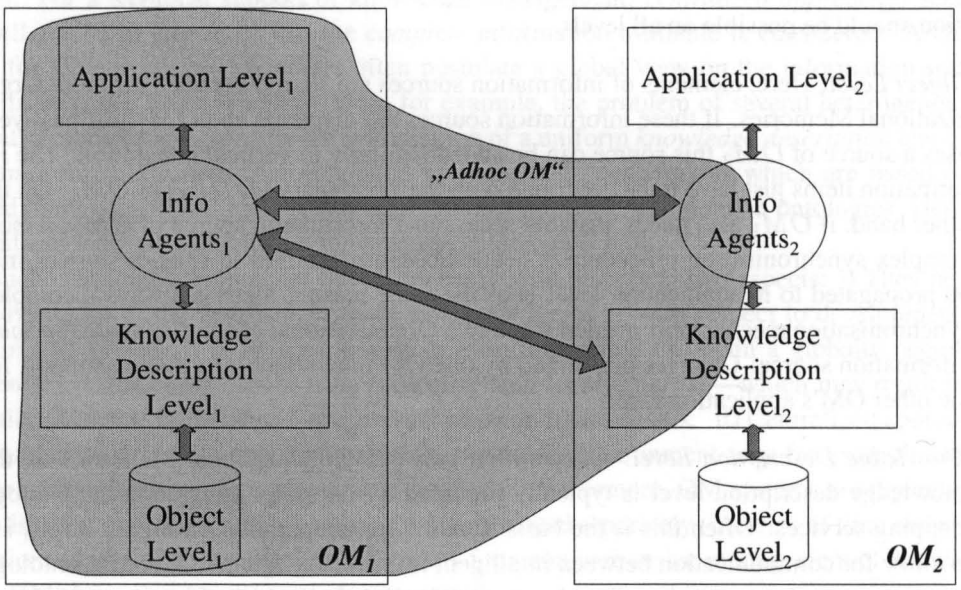

Figure 2.1: Cooperation on the Knowledge Description and knowledge brokering levels: Configuration of an ad hoc OM from two OMs.

Figure 2.1 shows the information landscape for two Organizational Memories. OM_2 supports OM_1 on the knowledge description level as well as on the knowledge brokering level. OM_1 might require a special inferencing service that can be provided by OM_2. One of OM_1's information agents therefore utilizes the knowledge description level of OM_2 to formulate its request. Then, an information agent in OM_2 performs the inferences and delivers its answer to OM_1's agent. Now this agent can interpret the answer with the help of the knowledge description level in OM_2. Of course, one of the central questions is how the mappings between

different ontologies on the knowledge description levels are realized. This issue will be discussed in section 2.2.

Horizontal OM Integration: Complexity of Communication vs. Quality of Service

In order to select a specific form of cooperation between OMs a *tradeoff* has to be balanced between *complexity of communication* and the *quality of service.* While the direct exchange of information objects on a "syntactical basis"—e.g. referenced by their unique identifiers—is quite cheap in terms of communication effort, the lack of semantic information holds down the quality of service; e.g. it is not clear whether the object really provides the information desired. On the other hand, high quality services at the knowledge brokering level typically need more communication, e.g. to negotiate the intended meaning of a request, its costs etc.

The following simple example illustrates this tradeoff. Imagine two "simple OMs". OM_1 consists of

- a set of documents, e.g. books (information object level)
- that are arranged on bookshelfs with category labels (knowledge description level),
- an intelligent information assistant who can select relevant books with respect to a query (knowledge brokering level), and
- a researcher working in a project (application level).

OM_2 is a library and organized quite similarly: Documents are arranged on bookshelfs according to the "ACM classification". There are assistants to the "chief librarian", helping to organize the library, select new books, write recommendations for literature etc.
The standard process within OM_1 can be described as follows: The researcher working on a special work package of the project has a specific information need (e.g. about agent–oriented software architectures, AOS). He therefore asks his assistant to fetch the relevant documents with respect this topic from the shelf. As the assistant is intelligent he not only fetches the documents on the shelf labelled "AOS", but also documents from other shelfs labelled with related topics (e.g. "FIPA specifications"). Furthermore, he sorts these documents with respect to specific criteria (e.g. software platforms, text books etc.), briefly summarizes them, and leaves documents in the shelf that are definitely outdated. The result is presented to the researcher.
Perhaps the documents in his own bookshelf do not satisfy the researcher's information need about AOS. Hence, he asks his assistant to fetch more information from the library (OM_2).
Now the information assistant (OM_1) can cooperate with the library (OM_2) on various levels:

- object level: The assistant knows about a specific book and just fetches it from the library.
- knowledge description level: The assistant searches for documents on AOS. He uses the catalog of keywords to find out which categories of the ACM classification are appropriate, and then fetches the books.
- knowledge brokering level: He asks one of the library assistants to suggest and fetch him a set of relevant books. In order to get a good result, he has to explain the term "AOS" to the library assistant and tell him other criteria for "relevant books". Then the library assistant performs an extensive search for literature, compiles the documents and delivers them to the researcher's information assistant.

Furthermore, cooperation between the library and the researchers can happen on even higher levels:

- application level: The workplan for the researcher's project includes a task "ordering of project literature". Therefore, he has to cooperate with the "chief librarian" who manages a global budget for literature.
- second order processes: In order to establish a really useful libary and not to buy all the relevant literature locally in the projects, every six month the library commission—consisting of some researchers and the chief librarian—meet and define a strategy for the purchase of books and magazines.

While the lower–level cooperations only require very little communication effort, the higher levels rely on communication between the various agents (e.g. "What are the goals for a department library?", "What does the term AOS mean?"). On the other hand the latter services typically are of better quality and higher impact: Suggestions by good technical librarian are more precise and relevant than books found by a pure keyword search or the set of all books labelled with a particular ACM classification code.

As a consequence of the considerations so far, we see that we examine an information landscape with various actors on the information provider and the information consumer side, as well as mediating information agents. Thus we have a strong need to establish a shared understanding between these actors. This is where ontologies show their particular importance. Coming from the definition of ontologies, three dimensions for characterizing information in an OM can be identified. These are presented in more detail in the next section.

2.2 Domain Ontology Agents for Distributed OMs.

In knowledge management (KM), it is widely accepted that ontologies as explicit specifications of conceptualizations [Gruber, 1995] provide a useful means to facilitate access and reuse of knowledge. Typical utilization scenarios comprise discussion groups, search engines, information filtering, access to non-textual information objects, and expert-user communication. In the context of information support ontologies provide a vocabulary for specifying information needs as well as information resources. The origin and composition of the information needs may be quite complex and heterogenous. Likewise it is often difficult to establish a relation between the information items and their descriptions. Therefore in the FRODO information landscape with its various actors on the information provider side, the information consumer side, and the mediating information agents, we have a strong need to establish a shared understanding between these actors.

In the first section, we encourage a comprehensive view on three dimensions of information that have fundamental impact on the usefulness of ontology-based KM systems, namely formality, stability, and sharing scope. On this basis we characterize the way domain ontologies for information support are embedded in the FRODO framework.

2.2.1 Three Dimensions of Information in KM

Information in KM information systems can be described with respect to various dimensions, e.g., granularity, trustworthiness, or explicitness. When designing such a system, three dimensions are especially relevant:

- *Stability*: Information can have different levels of stability. For example, the contact person for a specific concern in an enterprise will normally be quite permanent. However, if the enterprise sources out some functions to a call center, the knowledge about a contact person might become rather momentary, because each time one calls a new contact person is assigned.
- *Sharing Scope*: Information be can individual or shared within a group: individual because it has not yet been published or disseminated, or because it is aimed to be individual. Some notes on a post-it are individual, whereas a design team should develop a more shared understanding of the product to be developed. Furthermore, knowledge and information can be shared within even larger structures, like a group of groups (e.g., across the whole enterprise with its various divisions).
- *Degree of Formality*: Information can be highly formalized (e.g., formal business process models or rules in an expert system), or it can be more informal (e.g., text documents). Formal information is meant to be machine-readable and machine-interpretable.

It is desirable to identify information at the high end of these dimensions: Stable information can be reused over time; widely shared information can be reused across the organizational structure; formal information is a basis for powerful automation services. Chapter 5 discusses techniques that enable transitions along these dimensions. In order to utilize them optimally in a comprehensive KM system, a detailed understanding of their interactions is useful.

Interactions between the Three Dimensions

Below we describe how either two of the dimensions - stability, formality, sharing scope - mutually influence one another (cf. 2.2):

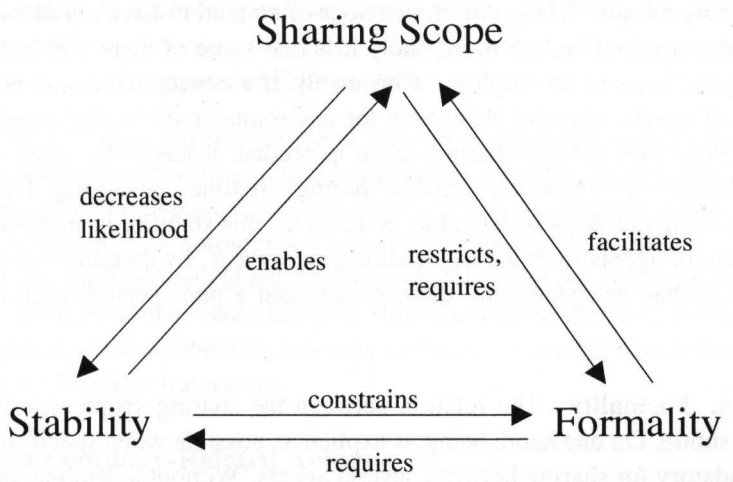

Figure 2.2: Sharing scope, stability, and formality of information in KM systems

Formality vs. Stability Reaching a high level of formality requires high effort: Specially trained personnel is needed; the process of formalization itself is time consuming, ambiguous and error-prone. This leads to high costs which normally only charge off if the application period of the formalization is orders of magnitude longer than the creation duration. However, the stability of a domain to be formalized is prevalently an externally determined factor. Therefore the degree of formalization must be chosen carefully, according to the expected stability. In the KnowMore information delivery architecture, for instance, three ontologies are used for information modeling, namely: (i) the information ontology, a metamodel for structural description of information items (formats, types etc.), (ii) the enterprise ontology to specify creation and intended application context, and (iii) the domain ontology for content descriptions. Apart from the principal/ontological distinction, these ontologies differ especially in stability. While the vocabulary to structurally describe information (What essentially constitutes a book or a paper?) is quite stable, or at least expanding relatively monotonously (in contrast to the eighties, we now have web-documents), the domain ontology is often more a living organism: New topics become important, knowledge is acquired permanently, and respective conceptualizations are added or changed; old conceptualizations may be no longer valid, etc. The stability of an enterprise ontology resides between these two extremes. Enterprises change their organization from time to time. These changes may be slightly, within a fixed top-level ontology (What is a department, an employee, a process?), or they may be more rigorous, with a deep restructuring, e.g., as result of a merger. Consequently, for domain ontologies predominantly relatively weak formalizations (thesauri, concept lists, or hierarchies) are applied which can be maintained easily, often with automatic support. On the other hand, enterprise and information ontologies can be specified in a logical language with rich expressive power (e.g., TOVE [Fox *et al.*, 1993]). In general, a detailed analysis of an expected ontology life cycle can be a powerful guide to achieve an optimal level of formalization in terms of costs and benefits.

Stability vs. Sharing Scope These two dimensions often tend to have a tradeoff. The more agents share a conceptualization, the more likely it is that some of them will break the commitments forming the basis of an ontology. Conversely, if a conceptualization is shared only between a couple of agents, stronger changes in the environment are needed to enforce a shift in the ontology. Whenever a large sharing scope is needed, it has to be taken into account that most negotiation processes (which facilitate sharing) are time-consuming. Therefore, high stability of the resulting ontology is desirable. Sometimes this stability is artificially achieved by excluding groups of agents from the negotiation process, e.g., by dictating ontologies. Normally, this results in bad acceptance by these groups and a poor performance of the entire system.

Sharing Scope vs. Formality The relation between the sharing scope and the degree of formality is quite subtle. On one hand, being as explicit as possible when specifying a conceptualization is mandatory for sharing between several agents. Without a detailed understanding of a proposed ontology no commitments can be made by the agents, because probably there is no common interpretation of the proposal, and misunderstandings during application are predetermined. On the other hand, using a formal specification can be a hurdle for potential agents to participate in the sharing process. However, if formally less trained people are main addressees of an ontology, these users can not be excluded from negotiation and commitment. Often the acceptance of information systems fails, because the ontology is specified by highly trained

designers in some formal language without taking into account that a less trained user may not comprehend the implications and therefore may not use the system in a proper way. For example, information representation in WWW search engines has a low degree of formality, but a large sharing scope, whilst powerful retrieval mechanisms are less common. In summary, being formal is a prerequisite to allow for sharing, but it inhibits a wide scope as it needs highly trained agents.

Assessment of Techniques

Several modules can be used to balance a concrete information system within the design space spanned by the described dimensions:

- Monitoring Services: Analyzing the utilization of an ontology (e.g., feedback from a search machine) as well as monitoring the outer world can provide hints when the ontology should be reengineered.
- Responsibility Concepts: In order to organize complex negotiation processes in large groups, it makes sense to think about specific roles and responsibilities (thematic area managers, publishers, ...). These roles can be enacted by human as well as by machine agents.
- Communication Support: To achieve mutual understanding and generally agreed upon commitments, powerful discussion and negotiation services are required. Systems like Tadzebao [Domingue, 1998] for example serve as a discussion space for human experts when developing a common ontology.
- Formalization Services: Informal–formal as well as tacit–explicit transitions are subject to different fields of research. Classical knowledge engineering provides human-centered methods and tools for both transitions; contributions to informal-formal transitions on the basis of text documents come from information extraction, document analysis and understanding, and computational linguistics.

While the first (monitoring services) and the last (formalization services) item refers to the stability and the formality dimension of information, respectively, the middle two concepts (responsibilities and communication) address the sharing ascpect of ontologies. They will lead later on to our approach of an ontology society of ontology-related agents.

Comprehensive methodological and tool support for designing and maintaining enterprise KM ontologies throughout the whole lifecycle should comprise elements from all these areas. In following sections we describe how ontology services—especially for domain ontologies—are provided in the FRODO framework.

2.2.2 Roles of Ontology-Related Actors

In the FRODO distributed OM scenario there are various actors dealing with domain ontologies. In order to describe these actors we use the following dimensions:

- Goals: The actors operate in a regularly changing environment. In doing so, they not only *react* to such changes but also have their own goals and objectives which they try to achieve.

- Knowledge: Actor's have knowledge with respect to the relevant realms of their environment, e.g. objects and other actors, as well as with respect to their own goals.
- Competencies: An actors abilities to perceive and manipulate its environment and its own internal state define its abilities. In a multi-actor environment, the abilities to communicate with other actors are particularly important. Through communication, knowledge about facts, goals, competencies, etc. can be exchanged. This allows for negotiation and aggreements which may lead to a distribution of tasks between actors or to changes of an actor's knowledge and goals.
- Rights: Rights are a subset of an actor's competencies. They describe what an actor is allowed to do, e.g. read or manipulate an information item, or grant rights to other actors.
- Obligations: Obligations are also a subset of an actor's competencies. They describe what an actor is expected to do, e.g. due to a commitment in consequence of a complex negotiation procedure or because of an actor's intrinsic role[3]. Fairness guarantees that an actor virtually meets its obligations.

The first three dimensions are the standard knowledge level descriptions as proposed by Newell [Newell, 1982]. The latter two reflect that the various actors in distributed OMs form a *society*, not just an accumulation (cf. [Wooldridge *et al.*, 2000]). Rights and obligations are the basis for coordinating the negotiation processes that are needed to create a *shared* understanding.

Figure 2.3 shows a taxonomy[4] of possible roles[5] which ontology actors in a distributed organizational memory can take. The set of actors in an Organizational Memory taking one of these roles with respect to a specific ontology form an ontology society.
First, we distinguish between *ontology providers* and *consumers*. Ontology providers attend to the provision of ontology services (e.g. *experts* can answer queries about the relationship between two concepts) as well as to the acquisition and maintenance of a domain ontology (*editors*). Consumers, on the other hand, utilize a domain ontology in order to execute a specific application, e.g. find some knowledge items, annotate documents, etc. These groups of actors typically have different goals with respect to an ontology. While the consumers are only interested in completeness and soundness of an ontology with regard to their specific application, maintenance services take a more global view and claim these properties for the whole ontology. Within the group of ontology consumers we distinguish between *active* and *passive* users. The passive users neither help to improve the ontology nor do they have any claims with respect to the ontology service. *Associates* also do not neccessarily contribute to the ontology evolution, but have special quality requirements. Therefore, they are notified whenever the ontology changes. *Partners* commit to support the improvement of the domain ontology, hence they are both ontology consumers and providers. For the editor of an domain ontology, partners are of special importance as they are the main source for information about the utility of an ontology. However, the final responsibility for the ontology is in the editor's hand.

[3] An inferencing agent, for example, may be obliged only to apply "correct" inferencing rules. A document manager should reliably store documents.

[4] The dimensions of an actor's knowledge level description (cf. section 2.2.2) provide the basis for the ontological distinctions.

[5] [Wooldridge *et al.*, 2000] propose such a role–oriented analysis as a natural step in their Gaia methodology for agent–oriented design, especially when it is manifest to take an organizational view on the application scenario.

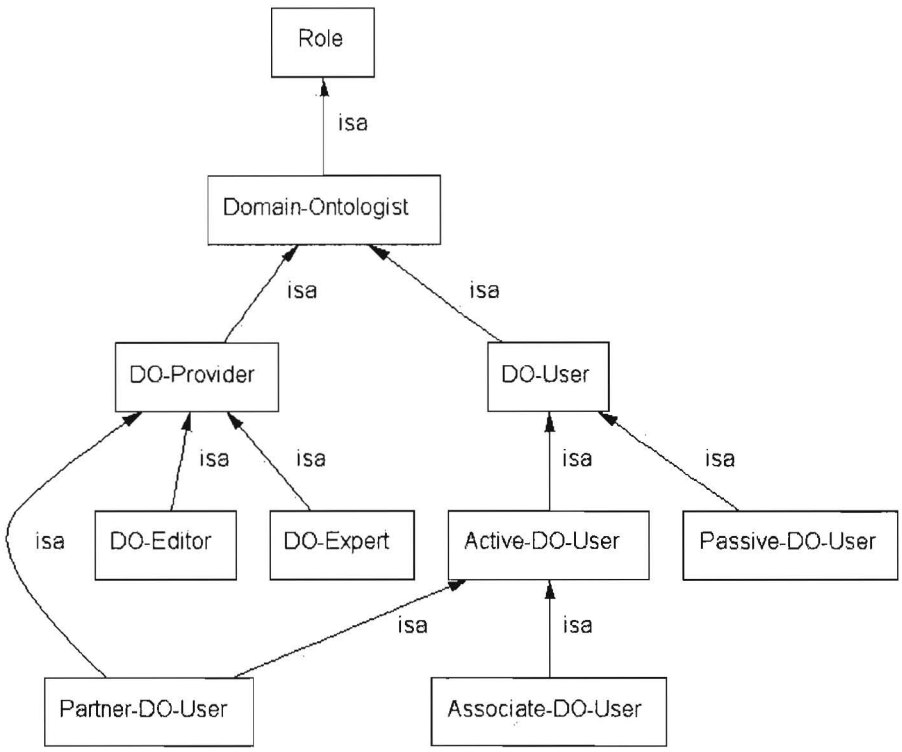

Figure 2.3: Taxonomy of Actors Dealing with a Domain Ontology (DO).

Table 2.1 summarizes rights and obligations of the various user groups of a domain ontology regarding some typical ontology operations. In the following we briefly sketch the competencies which are stated in table 2.1:

- Query: All actors have the right to query an ontology service about properties of the domain. There may be different types of queries, e.g. about
  - concepts: "Is a concept in the ontology?", "Give a natural language description of a concept.", ...
  - concepts and relationships: "Holds the relationship R between concept A and concept B?", ...
  - ontologies: "Is ontology A equivalent to ontology B?", "Is ontology A a sub-ontology of B?", ...
  - copy: "Give me a copy of ontology A and guarantee validity until revocation."
- Receive Update: All actors but the passive ones have the right to be notified whenever a guarantee about the validity of an ontological information doesn't hold any longer.
- Suggest Update: Clearly, any member of an "ontology community" can contribute to an improvement of the ontology. Partners and editors, in addition, commit to actively push ontology evolution.

	Non User	Passive User	Associate User	Partner User	Expert	Editor
Query		R	R	R	R	R
Receive Update			R	R	R	R
Suggest Update		R	R	R/O	R	R/O
Answer Queries					R/O	R
Edit						R
Send Update Notification						R/O
Apply for Role	R	R	R	R		
Grant Guarantees						R
Guarantee Quality						O

Table 2.1: Rights (**R**) and Obligations (**O**) of Ontology Actors.

- Answer Queries: To answer queries like the ones described above is one of the central tasks of an ontology service. The actor that attends this task is called *ontology expert*. An editor of an ontology is also able to answer these queries. However, he is not obliged to.

- Edit: Only editors can assert, modify and retract ontological propositions. As they have responsibilty for the quality of an ontology, they are not forced to follow other actor's suggestions. However, in order to obtain high acceptance and use of an ontology an editor will take all suggestions into consideration. Potentially, an editor has to coordinate a complex negotiation procedure between the actors to conceive his decision.

- Send Update Notification: An editor has the right and obligation to keep all given guarantee (e.g. with respect to an ontology's validity) and notify the active users in the case of changes.

- Apply for Role: This is a basic competence for joining an ontology society or changing an actor's role within the society. The application is sent to an editor. This editor can then grant guarantees. Thereby the respective rights and obligations are negotiated.

- Grant Guarantees: E.g. validity for a certain time or until a certain event (cancellation), also the rights a user has when entering a ontology society.

- Guarantee Quality: Editors try to obtain a high quality of the domain ontology. Aspects of quality may be formal properties like soundness and completeness, as well as "soft

factors" like a good ratio between acquisition costs and use benefits. Guarantees about quality may be framed by a time interval or other constrains.

In summary, the previously described competencies can be grouped into three categories:

- *Ontology Utilization*: Competencies like *Query* and *Answer Queries* are needed in the use phase of an ontology. Typical actors will be settled on the knowledge brokering level. A retrieval agents for example might exploit ontological knowledge to achieve higher recall and precision or to better present his results to the information consumer. Therefore it asks an ontology expert about the relation between two concepts.

- *Ontology Evolution*: These competencies are necessary to negotiate ontology updates. E.g., if a retrieval agent takes the role of a partner user in an ontology society it might realize that information consumers often ask for information using a term that is not defined in the ontology. Hence, the retrieval agent would suggest the ontology editor to introduce a new concept. The ontology editor would thereupon coordinate a negotiation procedure between the active ontology users.

- *Ontology Socialization*: Actors can join or leave an ontology society or they may change their role (e.g. from passive user to partner). In order to make a decision which role an actor wants to take it might need information about the content of an ontology and about the rights and obligations it has. Thus the affiliation in an ontology society might presuppose a complex negotiation procedure between the potential ontology user and the editor that grants guarantees.

As we deal with a multiple-OM scenario, we will have several domain ontologies. So, actors in this scenario can take one of the roles described above for each available ontology service. For example, the editor of domain ontology A might be an associate with respect to ontology B (e.g., the ontology in a different department). The *weakest* role "passive consumer" allows for a straightforward integration of external ontologies, because no severe commitments about rights and obligations are made.

2.2.3 Domain Ontologies in the FRODO Framework

In section 2.1 it was argued that the FRODO should allow for *vertical* scaling as well as for *horizontal* scaling. With respect to *domain ontology services* this requires facilities for both adding domain ontologies to an OM and accessing ontology services from other OMs. Therefore we propose two types of ontology services: Domain Ontology Agents (DOA) and Distributed Domain Ontology Agents (D^{2}OA). Domain Ontology Agents are responsible for ontologies *within one Organizational Memory*, Distributed Domain Ontology Agents are located *between several Organizational Memories* and facilitate cross-OM communication. So, the task of D^{2}OAs is quite similar to "standard information integration ontologies" (e.g. mapping services), but much easier as the sources are already formal ontologies, not just "any information provider". Typical questions to DOAs are "What are the subconcepts of concept A?". D^{2}OAs on the other hand answer questions like "Which OM contains concepts like A and B?" or "What does A mean in OM_y?".
This structure better embraces the inherently distributed nature of (ontological) knowledge. Not *all conceptualizations* are shared between *all actors* of the system, but *ontology societies*

are formed with respect to relevant domains. Additional infrastructure enables communication between these ontology societies. Imagine for example two groups of experts, one for domain A, one for domain B. Each group negotiates its own domain ontology managed by DOA_A and DOA_B, respectively. D^{2}OA has knowledge what these ontologies are about and tries to identify points of contact or overlaps between them. Then, D^{2}OA initiates a negotiation procedure between DOA_A and DOA_B. The result might be a common upper level ontology or a mapping for some parts of the ontologies. In summary, this concept of ontolgy societies tries to find a *reasonable sharing scope* for portions of knowledge so that a common understanding is possible at all.
DOAs as well as D^{2}OAs can be described in terms of the roles that are outlined in section 2.2.2. For their own ontologies they have the rights and obligations of *Ontology Experts* and *Ontology Editors*. DOAs are *Associate* or *Partner Users* of the D^{2}OA ontologies and vice versa.

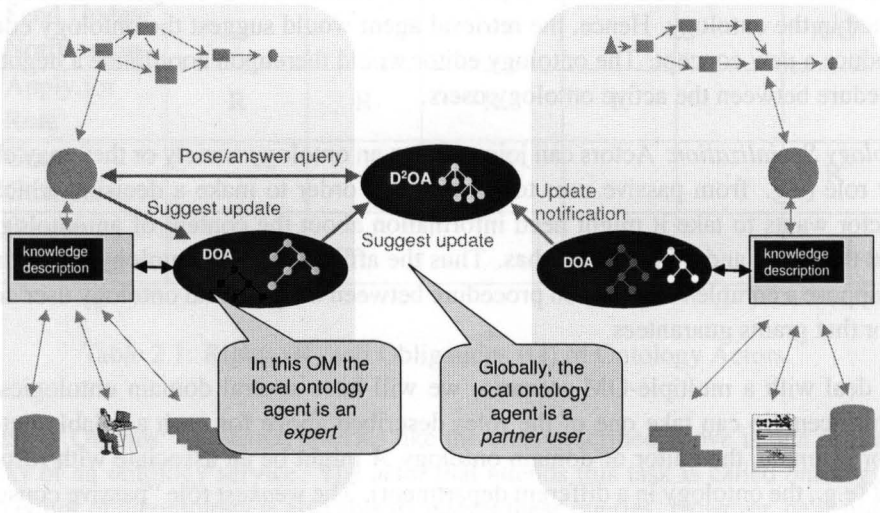

Figure 2.4: Sample Framework Instantiation with a D^{2}OA coordinating two OMs with local domain ontology agents (DOA).

Figure 2.4 illustrates these ideas a bit: Here, we have two OM instances with their respective ontologies. In each of these OMs, there is an agent maintaining the local ontology, being an editor with respect to this local ontology. Information retrieval or information extraction agents within the two OMs may be partner users exploiting the ontological knowledge to perform their own services, and maybe sometimes suggesting ontology updates because they come too often to wrong answers or bad performance because of a mismatch between formalized ontologies and the evolution of the real world. If the local ontology agent decides to accept such an update suggestion and change the local ontology, all other agents actively using the ontology must be notified. Further, the global ontology agent should be notified in order to adapt mapping rules accordingly. It could also be the case that the local ontology agent, playing the role of a partner user with respect to the global ontology, might suggest to change the global ontology because specific local changes are so radical that this should be reflected in an update of the overall structures.

2.3 Distributed Inferences

In section 2.2.1, the *degree of formality* was identified as one of the highly relevant dimensions of information in KM. While it is in general desirable to have information as highly formalized as possible, we are often confronted with situations where information remains informal. Reasons for this include:

- *the ontology is incomplete:* e.g., some of the topics a new document talks about have not yet been incorporated in the domain ontology (and the user adding the document to the repository is not allowed or able to extend the domain ontology)
- *a mediating ontology has not yet been established:* information from several Organizational Memories is involved in a single reasoning process, but the ontologies underlying their knowledge descriptions are not compatible and no mediating ontology that links the separate ontologies has been defined yet
- *extraction of metadata is too expensive:* especially in the case of legacy systems containing a huge amount of informal data, the extraction of metadata might be too expensive

In order to let informal sources participate in query answering, it is necessary to integrate informal, semi-formal, and formal reasoning techniques. Figure 2.5 shows the spectrum from informal to formal reasoning techniques used for information retrieval and query answering in Organizational Memories.

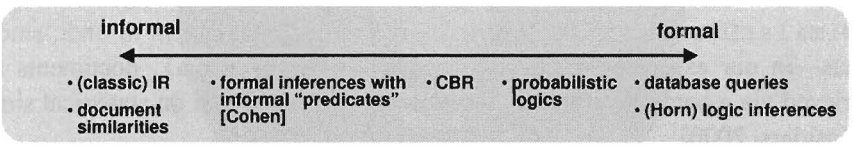

Figure 2.5: The informal-formal reasoning spectrum.

Interaction of these informal and formal reasoning techniques with each other and with other Organizational Memory components like ontologies, knowledge descriptions, workflow tasks, etc. is enabled by an *agent framework* (see chapter 6).

Figure 2.6 shows how a query containing formal and informal "predicates" is evaluated by delegating to the various formal and informal reasoning services.

- in the current task, the user fills in a form with formal and informal input, in this case the term "Sintek" from the enterprise ontology and "RDF" (as a natural language expression)
- a query identified to deliver relevant information for the current task is instantiated with the user input: `author(m1, Doc, ''Sintek'') ∧ containsConcept(om1, Doc, ''RDF'') ∧ similarDoc(om2, Doc, Doc1)`
- an information agent analyzes this query and delegates separate conjuncts to other agents depending on their location and degree of formality:
  - `author(m1, Doc, ''Sintek'')` is handled by an agent in `om1` that wraps a document collection and its metadata containing knowledge descriptions like author, date, etc., and performs a lookup on the metadata

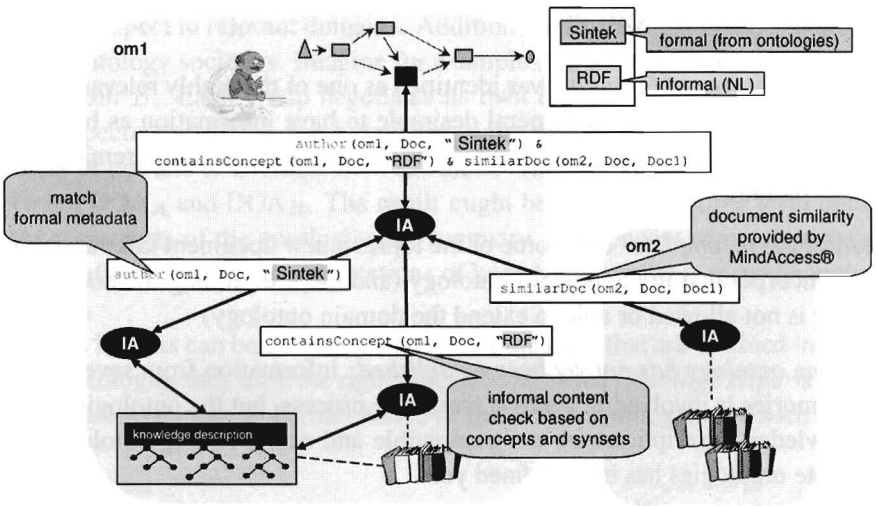

Figure 2.6: An example for informal-formal reasoning.

- `containsConcept(om1, Doc, ''RDF'')` is handled by an `om1` agent that maps natural language expression to ontological concepts (e.g., by using synsets) and looks up documents annotated with these concpets
- `similarDoc(om2, Doc, Doc1)` is handled by an agent in `om2`; since `om2` has—in our example—no access to the ontologies in `om1`, documents are retrieved via a component that can retrieve documents based on statistical similarity [Insiders, 2000]

- the documents found in `om1` and `om2` (i.e., the bindings for `Doc` and `Doc1`) are presented to the user

Consequently, we envision a query language which integrates the formal and informal components. The interpretation of the language is given by the specification of the behavior of the various agents.

In [Wagner, 1998], reaction rules as a declarative formalism for specifiying the behavior of agents were proposed, which are quite useful for distributed query answering and updates in multi-database and multi-knowledgebase systems.

In FRODO, we extend this formalism to also support informal and semi-formal reasoning agents. Furthermore, the dimensions identified in section 2.2.2, i.e., goals, knowledge, and competencies, become explicit agent states. The newly developed logic programming language TRIPLE (see chapter 3) is used as the central (formal and semi-formal) reasoning component. Informal services are wrapped by agents that use reaction rules and TRIPLE for communication and handling of internal states; the informal services themselves are made accessible by extending TRIPLE with builtins (similar to approaches for text classification with ILP techniques as proposed by [Cohen, 1995]).

Figure 2.7 shows the resulting agent architecture:

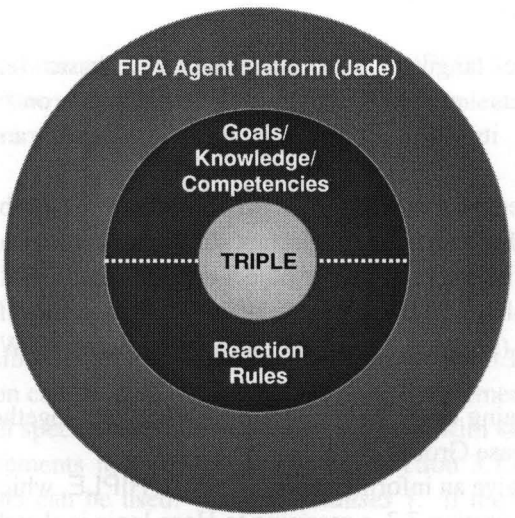

Figure 2.7: The FRODO agent architecture

Chapter 3

TRIPLE–An RDF Query, Inference, and Transformation Language

TRIPLE is a new logic language that borrows many basic features from F-Logic [Kifer *et al.*, 1995] but is especially designed to work on the semantic web [Berners-Lee, 1999], i.e., its main purpose is to query (and transform) RDF models [W3C, 2001b].

TRIPLE can be viewed as a successor of SiLRI (Simple Logic-based RDF Interpreter [Decker *et al.*, 1998]). One of the most important differences to F-Logic and SiLRI is that TRIPLE does not have a fixed semantics for object-oriented features like classes and inheritance. It was designed to allow such features to be easily defined for different object-oriented and other RDF extensions (see, e.g., the definition of RDF Schema [W3C, 2001a] in section 3.2.2).

TRIPLE is currently being developed in the FRODO project together with Stefan Decker (Stanford University Database Group).

In section 3.1, we first give an informal description of TRIPLE, which is followed by some examples in section 3.2. In section 3.3, a mapping to Horn logic is sketched, which serves as a first basis for defining the semantics of TRIPLE.

3.1 Introduction to TRIPLE

In the following sections, the main features of TRIPLE will be informally described. In section 3.2, some examples are provided for clarification.

3.1.1 Namespaces and Resources

Since namespaces and resources are essential for RDF, TRIPLE has special support for them.

Namespaces are declared via clause-like[1] constructs of the form "*nsabbrev* := *namespace*.", e.g.

rdf := "http://www.w3.org/1999/02/22-rdf-syntax-ns#".

[1] A logical foundation of abbreviations will be given in section 3.3 (equation 3.9).

Resources are written as *nsabbrev:name*, where *nsabbrev* is a namespace abbreviation and *name* is the local name of the resource.

Resource abbreviations can be introduced analogously to namespace abbreviations, e.g.

$$\text{subClassOf} := \text{rdfs:subClassOf}.$$

3.1.2 Statements and Molecules

An RDF statement (triple) is—inspired by F-Logic object syntax—written as

$$subject[predicate \rightarrow object]$$

Several statements with the same subject can be abbreviated as "molecules":

$$s[p_1 \rightarrow o_1; p_2 \rightarrow o_2; \ldots]$$

3.1.3 Models

RDF models, i.e., sets of statements, are made explicit in TRIPLE. Statements from a specific model are written as *stmt@model*, where *stmt* is a statement (or molecule) and *model* is a model (i.e., a resource denoting a model), e.g.

$$O[P \rightarrow Q]@m1$$

TRIPLE also allows skolem functions as model specifications (which is useful for parameterized models), e.g.

$$O[P \rightarrow Q]@sf(m1,X,Y)$$

If all (or many) statements/molecules in a formula (see section 3.1.5) are from one model, the following abbreviation can be used: *@model formula.* All statements/molecules in *formula* without an explicit model specification are implicitely suffixed with *@model.*

If all (or many) statements in a set of clauses (see section 3.1.6) are from one model, a block-building construct can be used: *@model {clauses }*. If the model specification is a skolem function containing variables, all variables must be introduced by a $\forall$ quantor (see section 3.1.5).

Model specifications may also be expressions like $(model_1 \cap model_2)$ and $(model_1 \setminus model_2)$, e.g.

$$O[P \rightarrow Q]@(m1 \cap m2)$$

3.1.4 Reified Statements

Reified statements are written as: *<statement>*.

3.1.5 Logical Formulae

TRIPLE uses the usual set of connectives and quantifiers for building formulae from RDF molecules and normal logic predicates, i.e., $\wedge$, $\vee$, $\neg$, $\forall$, $\exists$, etc.[2] All variables must be introduced via quantors, therefore marking them syntactically is not necessary (i.e., TRIPLE does not require variables to start with an uppercase letter as in PROLOG).

[2]For TRIPLE programs in plain ASCII syntax, the symbols AND, OR, NOT, FORALL, EXISTS, etc. are used; see the example in section 3.2.2.

3.1.6 Clauses

A TRIPLE clause is either a fact or a rule, e.g.:
persons:jdow[lastname → Doe; firstname → John; mother → persons:mdow].
∀ Person, Parent Person[parent → Parent] ←—
    Person[father → Parent] ∨ Person[mother → Parent].
Rule heads may only contain conjunctions and must not contain (explicitly or implicitly) any disjunctive or negated model expressions.

3.2 Examples

3.2.1 Dublin Core Metadata

The Dublin Core Metadata Initiative [DCMI, 2001] defines a set of elements for marking up documents with metadata like title, creator, date, subject, etc. An encoding of Dublin Core metadata in RDF is straightforward. The following example adds some simple metadata to a document and defines a rule that searches for documents with a specified subject.

```
rdf := "http://www.w3.org/1999/02/22-rdf-syntax-ns#".
dc := "http://purl.org/dc/elements/1.0/".
dfki := "http://www.dfki.de/".

@dfki:documents {

  dfki:d_01_01[
    dc:title → "FRODO: A Framework for Distributed Organizational Memories";
    dc:creator → "Andreas Dengel"; ... ; dc:creator → "Michael Sintek";
    dc:subject → FRODO; dc:subject → OM; ... ].

  ∀ S, D search(S, D) ←—
      D[dc:subject → S].
}
```

3.2.2 RDF Schema

As mentioned earlier, TRIPLE does not have a predefined semantics for RDF extensions like RDF Schema [W3C, 2001a]. In order to show that the features of TRIPLE are adequate for specifying the semantics of such extensions, the rules defining the semantics of the RDF Schema model for a given model are given:

```
rdf := "http://www.w3.org/1999/02/22-rdf-syntax-ns#".
rdfs := "http://www.w3.org/TR/1999/PR-rdf-schema-19990303#".
type := rdf:type.
subPropertyOf := rdfs:subPropertyOf.
subClassOf := rdfs:subClassOf.
∀ Mdl @rdfschema(Mdl) {
```

$$
\begin{aligned}
&\text{transitive(subPropertyOf).}\\
&\text{transitive(subClassOf).}\\
&\forall O,P,V\ \ O[P\rightarrow V] \longleftarrow\\
&\qquad O[P\rightarrow V]@\text{Mdl}.\\
&\forall O,P,V\ \ O[P\rightarrow V] \longleftarrow\\
&\qquad \exists S\ \ S[\text{subPropertyOf}\rightarrow P] \wedge O[S\rightarrow V].\\
&\forall O,P,V\ \ O[P\rightarrow V] \longleftarrow\\
&\qquad \text{transitive}(P) \wedge \exists W\ \ (O[P\rightarrow W] \wedge W[P\rightarrow V]).\\
&\forall O,T\ \ O[\text{type}\rightarrow T] \longleftarrow\\
&\qquad \exists S\ \ (S[\text{subClassOf}\rightarrow T] \wedge O[\text{type}\rightarrow S]).\\
\}&
\end{aligned}
$$

The equivalent in plain ASCII syntax is:

```
rdf := 'http://www.w3.org/1999/02/22-rdf-syntax-ns#'.
rdfs := 'http://www.w3.org/TR/1999/PR-rdf-schema-19990303#'.

type := rdf:type.
subPropertyOf := rdfs:subPropertyOf.
subClassOf := rdfs:subClassOf.

FORALL Mdl @rdfschema(Mdl) {

  transitive(subPropertyOf).
  transitive(subClassOf).

  FORALL O,P,V   O[P->V] <-
      O[P->V]@Mdl.

  FORALL O,P,V   O[P->V] <-
    EXISTS S   S[subPropertyOf->P] AND O[S->V].

  FORALL O,P,V   O[P->V] <-
    transitive(P) AND
    EXISTS W   (O[P->W] AND W[P->V]).

  FORALL O,T   O[type->T] <-
    EXISTS S   (S[subClassOf->T] AND O[type->S]).
}
```

3.3 Mapping to Horn Logic

In this document, the semantics of TRIPLE is given—as a first approximation—by a mapping to Horn logic. This allows TRIPLE to be implemented on top of XSB [SUNY, 2000] (i.e., PROLOG with tabled resolution), analogously to the F-Logic Flora [Ludäscher *et al.*, 1999]. In a future document, a model-theoretic sematics will be provided.

The following rewrite rules only show how to map RDF-specific features like resources and statements. All other mappings are wellknown (Lloyd-Topor transformations

for handling of quantors [Lloyd and Topor, 1984]) or straightforward (see the SiLRI system [Decker *et al.*, 1998]).

$$
\begin{array}{rcll}
X & \longrightarrow & X \quad \text{for literals and symbols } X & (3.1)\\
A : N & \longrightarrow & \text{resource}(A, N) & (3.2)\\
O[P \rightarrow V] & \longrightarrow & \text{statement}(O, P, V) & (3.3)\\
S@M & \longrightarrow & \text{true}(S, M) \quad \text{for statements } S & (3.4)\\
<S> & \longrightarrow & S \quad \text{for statements } S & (3.5)\\
O[P_1 \rightarrow V_1; P_2 \rightarrow V_2; \ldots]@M & \longrightarrow & O[P_1 \rightarrow V_1]@M \ \wedge & (3.6)\\
 & & O[P_2 \rightarrow V_2]@M \ \wedge \ \ldots & \\
\text{true}(S, M_1 \cap M_2) & \longrightarrow & \text{true}(S, M_1) \ \wedge \ \text{true}(S, M_2) & (3.7)\\
\text{true}(S, M_1 \backslash M_2) & \longrightarrow & \text{true}(S, M_1) \ \wedge \ \neg \ \text{true}(S, M_2) & (3.8)\\
X := Y.\, S(X) & \longrightarrow & \forall X \ (X = Y \ \wedge \ S(X)) & (3.9)\\
 & & \text{for clause sequences } S(X) &
\end{array}
$$

Example:
p:jdow[p:lastname → doe]@$m1$. ⟶

```
true(statement(resource(p, jdow), resource(p, lastname), doe),
m1).
```

Kapitel 4

Applikations-Level

Wie Eingangs erwähnt, nutzt *KnowMore* die Geschäftsprozesse eines Unternehmens als ein Mittel für das betriebliche Wissensmanagement. Diese Geschäftsprozesse werden in Workflow-Modellen abgebildet und durch Workflow-Management-Systeme (WfMS) operationalisiert. Daher bildeten WfMS einen zentralen Ansatzpunkt, um das in *KnowMore* angestrebte Wissensmanagement zu verwirklichen.

Schwerpunkt des Geschäftsprozess- und Workflow-Management-Marktes bilden jedoch Unternehmensprozesse, die geprägt sind durch Routineaufgaben und administrative Tätigkeiten (siehe auch [Leymann and Roller, 2000]), also i.d.R. wohlverstandene Prozesse mit klarer Aufgabenbeschreibung, Datenflüssen und Zuständigkeiten. Diese Systeme wurden von *KnowMore* als Umgebung genutzt zur Informationsunterstützung wissensintensiver Tätigkeiten in den Geschäftsprozessen. Demzufolge mussten die dort betrachteten wissensintensiven Tätigkeiten als Aktivitäten eines klassischen Workflows modelliert werden, den der Wissensarbeiter durchführt. Hierbei bedeutet klassisch, dass

- der Workflow vorab modelliert ist,
- Kontroll- und Datenfluss damit festgelegt sind,
- zur Laufzeit keine Änderungen vorgenommen werden können,
- und kein Modellierungskonzept für Kontext existiert (s.a. [Wenzel and Maus, 2001]).

Deswegen fügte *KnowMore* der Workflow-Aktivität eine Beschreibung des Informationsbedarfs und dazu benötigte Kontextvariablen hinzu, auf die ein Informationsagent zur Laufzeit zugreifen kann. Die Ausführung des Workflows hatte dann die Operationalisierung der Unterstützungsbeschreibung zur Folge, mit der der Informationssupport gesteuert wurde. Damit lieferte der Workflow den dynamischen Teil des Kontextes zur Informationsunterstützung. Somit musste aber nun die Unterstützung für eine wissensintensive Tätigkeit vorab modelliert sein. Das bedeutet, dass der Inhalt einer wissensintensiven Tätigkeit, deren Informationsbedarf und deren Kontext bereits zur Modellierungszeit bekannt sein müssen. Damit zeigte *KnowMore* einerseits, dass das Workflow-Management eine geeignete Ausgangsbasis zur Unterstützung solcher Aufgaben bietet, andererseits stellte sich heraus, dass heutige WfMS den Anforderungen sowohl der Informationsunterstützung als auch der Wissensarbeit nicht gerecht werden (siehe hierzu auch [Abecker *et al.*, 2000b]).

4.1 Knowledge-intensive Tasks

Betrachtet man nun Wissensarbeiter und deren Aufgaben genauer, kann man Eigenschaften und Anforderungen beobachten, die durch klassische WfMS nicht oder nur ungenügend unterstützt werden. Wissensintensive Tätigkeiten (knowledge-intensive tasks, KiT) zeichnen sich durch ein hohes Maß an Informationsbedarf und Unsicherheit aus. Genauer: mehrere Arbeitschritte innerhalb eines solchen Prozesses können nicht bereits im voraus vollständig modelliert werden, da die benötigte Information zur Modellierungszeit nur vage oder sogar überhaupt nicht bekannt ist [Deiters, 1997]. Weiterhin werden solche Prozesse geprägt durch eine Vielzahl von verschiedenen Lösungsmöglichkeiten, Ausnahmen und situationsabhängigen Informationen, was durch die klassische Workflow Modellierung nicht abgedeckt wird, da dort jede Aktivität mit ihrem Datenfluss im voraus modelliert sein muss. Erste Lösungsansätze in der Workflow-Management Forschung sprechen von "flexiblen", "ad-hoc" oder "adaptiven" Workflows (ein Überblick findet sich in [Han *et al.*, 1998]). Weiterhin ist eine aufwendige a priori Analyse von solchen Prozessen teuer und oft nicht kosteneffektiv, wenn die Prozesse wenig wiederholt werden bzw. einzigartig sind.

Dennoch ist eine Unterstützung solcher Prozesse sinnvoll und gewünscht. Da also die klassischen Workflow-Management-Systeme wissensintensive Aufgaben nur sehr eingeschränkt unterstützen können, wird in *FRODO* ein neuer, adäquater Ansatz benötigt, der sowohl eine Workflow-Unterstützung der Wissensarbeit liefert, als auch die Informationsunterstützung durch dynamischen Prozesskontext aus den Workflows gewährleistet. Der in *FRODO* vorgesehene Ansatz soll in diesem Kapitel erarbeitet und präsentiert werden. Zuerst wird dazu ein Beispiel für Wissensarbeit diskutiert, anhand dessen wissensintensive Tätigkeiten in *FRODO* charakterisiert werden. Daraus werden Anforderungen an ein WfMS expliziert. Anschließend werden die gewünschte Unterstützung und die sich daraus ergebenden Anforderungen für den Software-Support spezifiziert. Ein abschließender Überblick stellt verwandte Systeme vor und bewertet diese.

4.1.1 Beispiel für typische Wissensarbeit

Ein typisches Beispiel für die Arbeit von Wissensarbeitern stellt das Durchführen eines Forschungsprojekts dar, dessen Teilaufgaben in der Regel etwa folgendermaßen strukturiert sind:

- Projektvorbereitungphase
  - Proposal lesen
  - Projektstruktur planen:
    - Meilensteine festlegen
    - Termine, Deadlines identifizieren und festhalten
    - Arbeitspakete (AP) festlegen und Zuständigkeiten an Mitarbeiter verteilen
  - Mitarbeiter einweisen
- Projektdurchführungphase
  - AP bearbeiten
    - Dokumente festlegen (in-/output), zur Planungszeit i.d.R. nur grob bekannt
    - Programmmodule spezifizieren und erstellen

- Meilensteine vorbereiten:
  - Bericht (planen, schreiben, abgeben)
  - Präsentation vorbereiten

- Projektabschlußphase
  - Abschlußbericht
  - Abschlußpräsentation
  - Aufräumarbeiten

- Zusätzliche Aktionen:
  - Publikation schreiben
  - Konferenz durchführen/besuchen
  - Projekt-Demonstration
  - etc.

Diese Strukturierung ist in dieser Form nicht nur relativ grob, es fehlen beispielsweise auch zeitliche Abhängigkeiten, die bereits bekannt sind. So sind sicherlich die obersten Aufgaben in einer Reihenfolge angeordnet: `Projektvorbereitungphase` → `Projektdurchführungphase` → `Projektabschlußphase`.

Außerdem können die `Zusätzliche[n] Aktionen` *jederzeit* und *mehrfach* ausgeführt werden. Man wird sie aber zu gegebener Zeit planen.

4.1.2 Charakterisierung wissensintensiver Aufgaben

Schon in den 70er Jahren charakterisierte Rittel [Rittel, 1972, Rittel and Webber, 1973] (siehe auch [Conklin and Weil, 1997]) "Wissensarbeit" (knowledge work, KW) als primär durch den Umgang mit *wicked problems* ("harten Problemstellungen", im Gegensatz zu *tame problems*, einfachen Problemen) geprägt. Aufbauend auf diesen und weiteren Darstellungen von Kidd [Kidd, 1994], Buckingham Shum [Buckingham Shum, 1997] und Davenport [Davenport *et al.*, 1996] fassen wir typische Merkmale bei der Bearbeitung wissensintensiver Aufgaben (knowledge intensive tasks, KiTs) zusammen:

- Modellierung
  - KiTs sind stark spontan, veränderungsanfällig, schwer planbar und schwierig a priori zu analysieren und zu modellieren (s.a. [Remus and Lehner, 2000]).
  - Oft ist ein Nachmodellieren während der Durchführung einer Task vonnöten, um anfangs freigelassene Modellierungsstellen zu füllen.
  - Unvorhersehbare Ereignisse und Änderungen sind keine Seltenheit und erfordern ständig Modellierungsmodifikationen.
  - Überdenken, Optimieren und Modifizieren von Aufgabenmodellierungen sind erforderlich, um in dem variablen Umfeld heutiger Unternehmungen konkurrenzfähig zu bleiben (Reengineering).

- Die einzelnen Aufgaben können selten flach und sequentiell angeordnet werden. Hinzu kommt, dass die Anzahl der Aufgaben sehr hoch werden kann. Dies gilt schon alleine, weil ein Wissensarbeiter typischerweise in vielen verschiedenen Projekten tätig ist, die, je nach Komplexität, sehr viele Teilaufgaben besitzen können. Die hierarchische Strukturierung von Aufgaben und Teilaufgaben fördert, aufgrund der intuitiven Benutzbarkeit, eher mehr als weniger Teilaufgaben. Eine effiziente (manuelle) Handhabung aller zur Zeit anstehenden Einzelaufgaben – alle mit nicht-trivialen Ablaufanforderungen gespickt – gestaltet sich daher in der Praxis oft schwierig. Ohne Unterstützung verliert man leicht den Überblick.
- Wissensarbeit ist oft auf mehrere Mitarbeiter verteilt. Jedem Bearbeiter einer Teilaufgabe müssen folgende Abhängigkeiten bekannt sein:
  * Informationsfluss (woher kommt/ wohin geht welche Information)
  * Kontrollfluss (Reihenfolge der Teilaufgaben)
  * Organisationsmodell (wer bearbeitet welche Aufgabe)
- KiTs sind typischerweise Einzelaufgaben, die oft derart individuell sind, dass eine (exakte) Wiederholung dieser Aufgabe nicht möglich ist. Verursacht die Modellierung einer Aufgabe hohe Kosten, wird die Aufgabenmodellierung aber nur ein einziges Mal zur Durchführung eben dieser einen Aufgabe benutzt, lohnt sich der Modellierungsaufwand nicht. Bei derart individuellen Aufgaben ist eine einfache, grobe Modellierung (viele unter-/unspezifizierte Teile) effektiver.

- Ausführung
  - Gerade bei Wissensarbeit ist dem Benutzer ein möglichst großer Entscheidungsfreiraum zu geben, um die ihm gestellten Aufgaben zu bearbeiten. Dazu benötigt er jedoch ständig passende, aktuelle Informationen zur Modellierung oder zum Fällen von Entscheidungen. Anstatt dem Benutzer eine präskriptive Abfolge von Aktionen abzuverlangen, ist eine informationsunterstützende Assistenzdienstleistung für den Benutzer adäquater. Konkret bedeutet dies, dass man dem Benutzer volle Entscheidungsfreiheit gewährt, bei gleichzeitiger Bereitstellung notwendiger Informationen zum Fällen dieser Entscheidungen.
  - Außerdem werden Entscheidungen, während der Wissensarbeit, selten aufgrund modellierbarer, statischer Regeln getroffen. Oft müssen Entscheidungen zwischen Personen *ausgehandelt* werden, was Kommunikation und Verhandlung zwischen Personen erfordert.

- Info-Support (siehe hierzu auch [Abecker *et al.*, 2000a, Abecker *et al.*, 2000b])
  - Wissensarbeit erfordert viel Wissen aus vielen verschiedenen Informationsquellen.
  - Dieses Wissen muss für den Wissensarbeiter zur Verfügung stehen, ohne dass von diesem zu viel Aufwand abverlangt wird. Dies kann am besten durch eine pro-aktive, kontext-bezogene Informationsbereitstellung geleistet und im Sinne der erwähnten Assistenzfunktionalität geliefert werden.
  - Dabei ist besonders zu berücksichtigen, dass die genannten Eigenschaften von Wissensarbeit die Informationsunterstützung beeinträchtigen. So stellt sich die Frage, was man einer unterspezifizierten Aufgabe an Informationen bereitstellen kann.

4.2 Unterstützung der KiTs

4.2.1 Aufgaben-Spektrum

Generell kann man die in einem Unternehmen auftretenden Aufgaben in ein Spektrum einordnen, wie es von Noël Craven und Dirk Mahling in [Craven and Mahling, 1995] vorgestellt wurde. Abbildung 4.1 zeigt das Aufgaben-Spektrum zusammen mit den Charaktereigenschaften der Endpunkte.

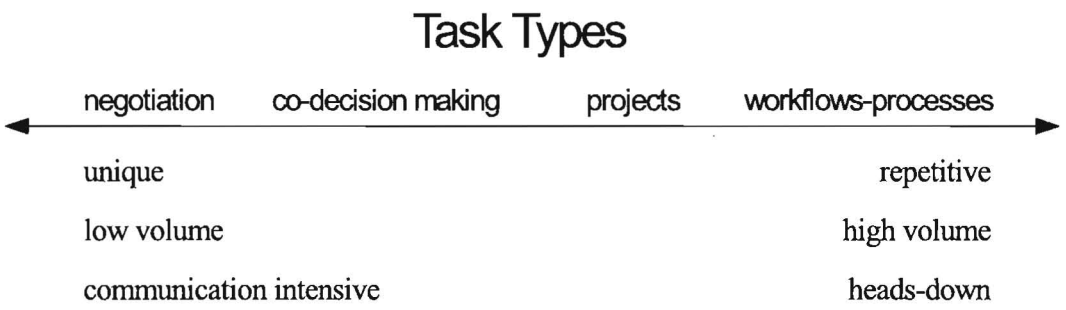

Abbildung 4.1: Das Aufgaben-Spektrum – vorgestellt von Noël Craven und Dirk Mahling in [Craven and Mahling, 1995].

Die Abbildung beschreibt die verschiedenen Aufgabentypen mit ihren typischen Eigenschaften. Vergleicht man diese mit den oben angeführten Eigenschaften von KiTs, so fällt auf, dass sich die wissensintensiven Tätigkeiten *zwischen* den Extrema befinden (mit deutlicher Tendenz zur linken Seite).

Für alle in dieser Abbildung dargestellten Aufgabenbereiche existieren entsprechende Unterstützungswerkzeuge bzw. sind solche denkbar. Man kann diese Unterstützungswerkzeuge analog zu den entsprechenden Aufgaben ebenfalls in dieses Diagramm einordnen, wobei auch dabei zwei extreme Lager entstehen: Klassische Workflows werden sich dabei auf der rechten Seite als Koordinationsunterstützung wiederfinden, während sich Kommunikations- und Verhandlungstools auf der linken Seiten (Kollaboration) ansiedeln würden.

Entsprechend der Einordnung wissensintensiver Tätigkeiten in das Diagramm wird sich eine Unterstützung wissensintensiver Tätigkeiten (KiTs) ebenfalls *zwischen* den Extrema befinden. Tendenziell muss sich diese Unterstützung in Richtung Kollaboration bewegen.

In der Workflow-Forschung wurden bereits verschiedene Ausprägungen für Workflows vorgeschlagen, die Elemente von Wissensarbeit berücksichtigen [Nutt, 1996, COI GmbH,] (Ausnahmen, Unplanbarkeit, etc.). Der derzeitige Trend, einige der vorgestellten Charaktereigenschaften von Wissensarbeit zu unterstützen, hat in der Workflow Community zu "ad-hoc Workflows", "flexiblen Workflows" etc., geführt. Dabei ist der Ad-hoc Workflow auf der linken Seite des Spektrums (Kollaboration) anzusiedeln und der flexible Workflow zwischen `projects` und `workflows-processes`.

Bei Ad-hoc Workflows handelt es sich mehr oder weniger um eine Möglichkeit, einen Workflow einmalig für eine bestimmte Aufgabe zu entwerfen. Dies wird i.d.R. durch eine Routing-Spezifikation umgesetzt, in der Art einer Umlaufmappe, deren Teilnehmer per e-Mail benachrichtigt werden. Eine Prozesslogik ist dadurch ebenso wenig existent wie ein Kontext, der eine *kontext-sensitive* Informationsbereitstellung erlauben könnte.

Der flexible Workflow entspricht dagegen quasi dem klassischen Workflow, mit dem Unterschied, dass zur Laufzeit Änderungen an der Modellierung möglich sind. Eine einheitliche Verwendung des Begriffs "flexibler" Workflow existiert allerdings nicht. So wird bereits die Möglichkeit, Modellierungsänderungen – wenn möglich – in laufende Instanzen zu übernehmen, als Flexibilität definiert.

4.2.2 Anforderungen an eine Unterstützung von KiTs

Aufgrund der hohen Spontanität, Veränderungsanfälligkeit und Unvorhersehbarkeit von wissensintensiver Arbeit darf eine adäquate Unterstützung von Wissensarbeit keine Vollständigkeit in Bezug auf die Modellierung erzwingen. Das ständige Nachmodellieren hat zur Folge, dass die Modellierung eines Wissensprozesses keine statische Information, sondern eben nur eine *aktuelle* widerspiegelt. Es handelt sich hierbei um ein extrem dynamisches System, das aber im Gegenzug stets mit aktuellen Daten aufwarten kann.

Weiterhin soll das Anlegen von Instanzen "from-scratch" (Instanziieren eines "leeren" Modells) und das Verwenden von unterspezifizierten Teilaufgaben erlaubt werden. Im Extremfall soll es möglich sein, Teilaufgaben nur als Black-Boxes zu modellieren, die jenseits einer von Benutzer gespeicherten informellen Beschreibung keinerlei Information oder Automatismus bieten. Damit soll der geringen Planbarkeit bzw. Spezifizierbarkeit von Wissensarbeit Rechnung getragen werden.

Dies bieten flexible Workflow-Systeme typischerweise genauso wenig, wie die hierarchische Strukturierung von Aufgaben in Teilaufgaben. Zur adäquaten Unterstützung von wissensintensiven Tätigkeiten müssen wir dies allerdings explizit verlangen. Daraus folgt, dass die angestrebte Unterstützung von Wissensarbeit für *FRODO* zwischen ad-hoc und flexiblen Workflows einzureihen ist, genauer gesagt: in der Mitte des Diagramms in Abbildung 4.1, also zwischen `co-decision making` und `projects`.

Die für Wissensarbeit typische Hauptenscheidungsgrundlage, aktuelle, kontextspezifische Information, fordert eine dynamische Informationsunterstützung, die sich den Kontext zum Zeitpunkt des Informationsbedarfes erst ableitet (im Gegensatz zur Vorabmodellierung in *KnowMore*).

Wissensarbeit erfordert ein derart dynamisches Verhalten, dass eine statische, präskriptive (vom System diktierte) Abarbeitung von Arbeitseinheiten von vorneherein auszuschließen ist. Eine solche Unterstützung würde den Wissensarbeiter in seiner Arbeit mehr behindern als nützen, was im Extremfall dazu führen würde, dass er die Unterstützung gar nicht benutzen und stattdessen Workarounds beschreiten würde. Es muss sich daher vielmehr um eine Unterstützung in Form eines *Assistenten* handeln, die ihn bei der Planung, Organisation, Kommunikation und Verhandlung *unterstützt* und ihn dabei *fortwährend, automatisch* mit *kontextrelevanten* Informationen versorgt. Geforderte Aktionen werden dem Benutzer lediglich als *Vorschläge* präsentiert. Der Benutzer kann dann von Fall zu Fall entscheiden, ob er diese Vorschläge akzeptiert oder gänzlich andere Wege geht.

Dies bedeutet aber nun keineswegs, dass die Unterstützung an Zweckmäßigkeit verliert – im Gegenteil: Oftmals ist ein Wissensarbeiter mit der *manuellen* Organisation einer Vielzahl von einzelnen Teilaufgaben völlig überlastet. Hier kann eine sinnvolle Unterstützung die Übersicht und Effektivität der Wissensarbeit erhöhen und somit letztendlich seine Arbeit verbessern.

Entscheidungen werden auch durch Kommunikation und Verhandlungen getroffen, was eine Unterstützung von Wissensarbeit auch im Bereich der Kollaboration erfordert. Daher strebt

Amit Sheth [Sheth, 1997] eine *einheitliche* Integration von Koordination, Kollaboration und Informations-Management an. In *KnowMore* wurde bereits die Integration von Informations-Management (pro-aktiver, kontext-sensitiver Informationssupport) und Koordination (klassischer Workflow) angegangen. Dazu sollten nun auch relevant erscheinende Elemente der Kollaboration berücksichtigt werden (etwa Newsgroup-Zugriff aus dem Prozesskontext).

Weiterhin erhält man durch die Repräsentation des Wissensprozesses im Workflow-System wertvollen Prozesskontext, der ohne eine solche Repräsentation für eine Informationsunterstützung nicht vorhanden bzw. schwer zu inferieren ist. Dies kann z.B. bei dem Watson-System [Budzik and Hammond, 1999, Budzik and Hammond, 2000] beobachtet werden, das den Benutzer observiert, um relevante Informationen aus dem Netz bereitzustellen. Dabei wird jedoch nur die aktuelle Aktion eines Benutzers mit einem Aufgaben-Modell verglichen, was dem System die Möglichkeit gibt, den Informationsbedarf abzuschätzen. Watson verwendet also keinen Prozesskontext, da nur isolierte Aktionen betrachtet und keine Prozessmodelle herangezogen werden.

4.2.3 Leitmotive der Unterstützung

In [Schwarz, 2000, Schwarz *et al.*, 2001] wurden bereits erste Anforderungen an eine adäquate Unterstützung wissensintensiver Tätigkeiten vorgestellt. In Anlehnung daran und unter Beachtung der vorangegangenen Abschnitte wurden die Anforderungen für die in FRODO angestrebte Unterstützung von Wissensarbeit hergeleitet, welche im folgenden hier aufgelistet werden:

- **Assistenz-Funktionalität**
  Das System soll den Benutzer bei seiner Wissensarbeit adäquat unterstützen, d.h. bei Planungs- und Organisations-, Kommunikations- und Verhandlungstätigkeiten. Das System soll dabei Vorschläge statt Befehle präsentieren.

- **Verzahnung von Modellierung und Ausführung**
  Die Modellierung sollte auch parallel zur Prozessausführung stattfinden können. (vgl. [Petrie *et al.*, 1999]).

- **Lazy/Late Modeling**
  Modellierung soll unvollständig begonnen und sukzessive verfeinert werden können. (vgl. [Petrie *et al.*, 1999]).

- **Hierarchische Dekomposition**
  Jede Aufgabe soll rekursiv in Unteraufgaben aufgeteilt werden können. (vgl. [Craven and Mahling, 1995]).

- **Ausdrucksmächtige Prozesslogik**
  Zur Spezifikation von Ablaufconstraints sollen komfortable Mittel zur Verfügung gestellt werden.

- **kontext-sensitive Wissensmanagement-Anbindung**
  Eine solche Anbindung wurde (für *klassische* Workflows) bereits in *KnowMore* realisiert. In *FRODO* benötigt die teilweise unterspezifizierte und sich entwickelnde Modellierung einer KiT besondere Aufmerksamkeit, da aus dieser nur wenig Hinweise auf relevante Informationen gezogen werden können bzw. sich Hinweise erst zur Laufzeit ergeben.

4.2.4 Fazit

Abschnitt 4.5 beleuchtet und bewertet Workflow-Systeme, welche die obigen Anforderungen zumindest teilweise oder in ähnlicher Weise erfüllen. Wie dort noch näher aufgeführt, realisiert allerdings keines dieser Workflow-Systeme Lösungen zu *allen* geforderten Anforderungen.

Daher werden wir im weiteren Verlauf des FRODO-Projekts am DFKI ein eigenständiges Workflow-System realisieren, das allen Anforderungen gerecht wird und leicht zukünftige Ausbaumöglichkeiten erlaubt.

4.3 Vision einer Unterstützung von KiTs

Die in [Schwarz *et al.*, 2001] prognostizierte Vision einer Unterstützung von Wissensarbeit wurde auf FRODO-spezifische Anforderungen angepasst und daraus ein adäquater Ansatz hergeleitet. Dieser wird im folgenden nun dargestellt, beginnend mit den Konzepten zur Modellierung und Ausführung des Workflows. Schließlich wird die gewünschte Informationsunterstützung präsentiert. Ein Beispiel, wie ein Wissensarbeiter diesen Ansatz nutzt, rundet dieses Kapitel ab.

4.3.1 Modellierung

In klassischen Workflow-Systemen, wie durch [WfMC, 1999] spezifiziert, werden Geschäftsprozesse durch "Workflows" repräsentiert. Aufgaben oder Arbeitsschritte innerhalb eines Geschäftsprozesses werden durch "Aktivitäten" innerhalb des entsprechenden Workflows repräsentiert. Diese können einzelne "Tasks" besitzen, die jedoch nicht mehr unter der Kontrolle des WfMS stehen, sondern eher eine Art Agenda für den Benutzer darstellen. Also besitzen die Aktivitäten innerhalb eines Workflows keine Unteraktivitäten – sie bilden damit eine flache Schicht unterhalb des Workflows.

Erste Aufweichungen dieser flachen Struktur bieten Workflow-Systeme (z.B. FlowMark [Leymann and Roller, 1994] und WorkParty [Ruppietta and Wernke, 1994]), die statt Aufgaben an manchen Stellen auch Sub-Workflows erlauben. Die Intention dabei ist, andere Workflows einzubetten und damit eine Modularisierung zu erreichen. Der Sub-Workflow ist jedoch eine eigenständige Einheit, eine Black-Box für den Workflow, die lediglich einen Datenaustausch während Start und Ende des Sub-Workflows erlaubt.

Im Gegensatz dazu soll in FRODO die hierarchische Dekomposition ein Normalfall der Modellierung sein. Damit bietet sich einerseits die Möglichkeit, die Aufgaben hierarchisch zu strukturieren, aber dennoch die Kontrolle im Workflow zu behalten, und andererseits erlaubt es auch eine *modulare* Modellierung von Aufgaben. Dazu können einzelne Standardvorgehensweisen bei Bedarf als Unteraufgaben einer Aufgabe eingebaut werden.

Abgesehen von der entsprechend modularen Prozesslogik (Oberaufgaben spezifizieren grobe Ziele, Unteraufgaben feinere Teilziele), bietet diese Vorgehensweise dem Bearbeiter einer Aufgabe eine übersichtliche, strukturierte Aufgabenbeschreibung, die bei einer großen Anzahl von Teilaufgaben leichter zu erfassen ist als eine entsprechende, flache, Flussdiagramm-ähnliche Darstellung (typische Darstellung eines klassischen Workflows).

Daher wird in FRODO zu jedem Zeitpunkt, in jeder beliebigen Detaillierungsstufe einer Aufgabenmodellierung, eine weitere *Verfeinerung* erlaubt.

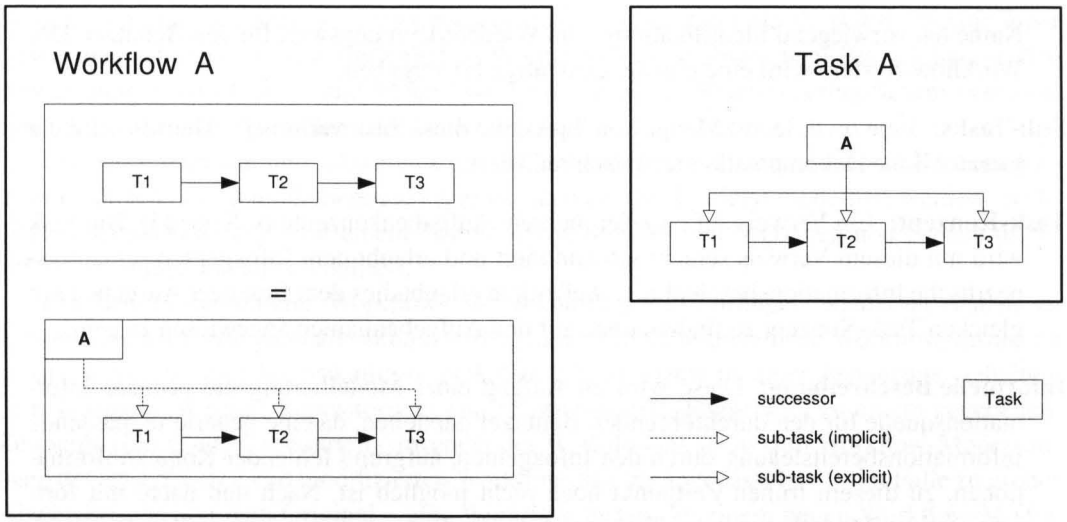

Abbildung 4.2: Links sehen wir einen typischen traditionellen Workflow, der drei aufeinanderfolgende Tasks enthält. Das Enthalten-sein von Tasks in einem Workflow entspricht quasi einer impliziten Sub-Task-Relation. Die Task-Struktur auf der rechten Seite dagegen realisiert eine explizite Sub-Task-Relation.

Alternativer Workflow-Begriff

Das Konzept der Verfeinerung hat zur Folge, dass erstens *jede* Aufgabe Unteraufgaben besitzen kann. Zweitens spezifiziert eine Aufgabenbeschreibung indirekt die Ziele, die mit dieser Aufgabe verfolgt werden. Unteraufgaben dieser Aufgaben sollten demnach Teilziele verfolgen.[1]

Mit dieser Modellierung eines Workflows verschwimmt aber nun (bewusst) die semantische Grenze zwischen einem Workflow und einer Teilaufgabe innerhalb dieses Workflows. Da eine Teilaufgabe stets auch *eigenständige* Ziele verfolgt, liegt es nahe, auch diese Teilaufgabe als einen Workflow aufzufassen bzw. auffassen zu dürfen (siehe dazu Abbildung 4.2). Gleiches gilt auch umgekehrt: Ein Workflow kann auch als Teilaufgabe eines übergeordneten Workflows aufgefasst werden.

Eine einzelne Aufgabe enthält somit im FRODO Workflow-System also die Mächtigkeit eines gesamten Workflows (und umgekehrt).[2]

Aufgrund der resultierenden, *bewussten* semantischen Verschmelzung von Workflows und Aufgaben liegt es nahe, sowohl Workflows als auch Aufgaben durch ein und dasselbe formale Objekt zu repräsentieren.

Dies wird in FRODO durch das Konzept der **Task** erreicht, die folgende Elemente besitzt:

Name: Vom Benutzer gegebener, beschreibender Name der Aufgabe. Gegenüber Schwester-Tasks (Tasks auf gleichem Level) muss dieser Name eindeutig sein. Dieser

[1] Die maschinelle Überprüfbarkeit dieser Tatsache wird zu diesem Zeitpunkt nicht angestrebt.

[2] Dennoch unterscheiden sich Workflows und Aufgaben prinzipiell durch die Tatsache, dass Workflows – im Gegensatz zu Tasks – weder Vorgänger noch Nachfolger kennen und sich dementsprechend abgekapselt in einer closed world befinden.

Name hat vorwiegend Identifikations- und Wiedererkennungswert für den Benutzer. Das Workflow-System wird eine eigene, eindeutige ID vergeben.

Sub-Tasks: Eine (evtl. leere) Menge von Tasks, die diese Task verfeinern. Hiermit wird die hierarchische Dekomposition technisch realisiert.

Task-Konzept: Ein Verweis auf ein oder mehrere Aufgabenkonzepte (s. Seite 45). Die Task wird mit diesem Verweis semantisch annotiert und erlaubt dem Infoagenten semantikspezifische Informationsbeschaffung. Außerdem erlaubt dies dem Benutzer Aufgaben mit gleichen Task-Konzept zu finden, ohne auf den Aufgabennamen angewiesen zu sein.

Informelle Beschreibung: Diese wird zu Anfang einer Modellierung die primäre Informationsquelle für den durchführenden Benutzer darstellen, da eine generierte, passende Informationsbereitstellung durch den Infoagenten, aufgrund fehlender Kontextinformationen, zu diesem frühen Zeitpunkt noch nicht möglich ist. Nach und nach, mit fortschreitender Modellierung der Task, wird die Bedeutung dieser informellen Beschreibung mehr und mehr dem Informationssupport des Infoagenten weichen.

Ablaufconstraints: And-/or-/xor-Split/Join u.a. zur deskriptiven Spezifikation des Ablaufs (Ablaufreihenfolge) verschiedener Tasks. Diese Ablaufconstraints werden mittels Vorbedingungen realisiert werden, was bedeutet, das die Ablaufreihenfolge dynamisch, also erst während der Laufzeit inferiert wird.

Attribute, Input- und Output-Container: Eine Task besitzt Attribute, die dem Benutzer angezeigt werden und die er editieren kann. Damit wird der Datenaustausch zwischen Benutzer und/oder externen Programmen realisiert. Die Input- und Output-Container beschreiben den Datenfluss in die Task und aus der Task heraus. Sie bilden damit den Datenfluss innerhalb des Task-Modells.

Wir betrachten das Task-Modell als einen Vorteil gegenüber der klassischen Workflow-Betrachtung, da der Benutzer zur Laufzeit ein besseres Verständnis der aktuellen Aufgabe erhält, weil er sich durch das Task-Modell über die Einordnung (z.B. bzgl. der Ziele) seiner Aufgaben klar werden und die Aufgabe mit anderen in Beziehung setzen kann.

Wenn im folgenden von einer "Aufgabe" die Rede ist, handelt es sich um eine Aufgabe, die in der wirklichen Welt erledigt werden muss. Wird dagegen von einer "Task" gesprochen, ist damit ein technischer Repräsentant für eine "Aufgabe" gemeint.

Modelle und Instanzen

Aufgrund der hohen Flexibilitätsansprüche erscheint es wenig sinnvoll, den Lebenslauf eines Workflows (bzw. einer Workflow-Instanz) in traditioneller Weise zu übernehmen. Statt der strengen Unterteilung in Modellierungs- und Laufzeit eines Workflows, d.h. zuerst Workflow-Modelle zu erstellen und diese dann für die durchzuführenden Workflows (Instanzen) zu instanziieren, soll ein leicht modifizierter Weg beschritten werden:

Die Workflows werden zwar weiterhin durch Instanziieren von Workflow-Modellen gebildet. Jedoch bleibt die Bindung zum ursprünglichen Workflow-Modell nur sehr schwach. Genauer bedeutet dies, eine Workflow-Instanz erhält eine Kopie des Modells, auf dem nun alle Änderungen vorgenommen werden können, ohne das Original zu ändern. Die Bindung zum

Modell bleibt nur in Form eines Verweises bestehen. Daher kann die Workflow-Instanz unter Umständen sehr stark vom ursprünglichen Workflow-Modell abweichen. Bei solchen großen Abweichungen von Instanz und Modell wird man in einem Reengineering-Schritt (s.u.) ein neues Modell auf der Grundlage der Instanz erzeugen.

Um der Spontanität und schweren a priori Planbarkeit von Wissensprozessen Rechnung zu tragen, können neu zu modellierende Prozesse auch erst zur Laufzeit modelliert werden. Technisch wird dabei anfangs mit einem "leeren" Workflow-Modell begonnen und danach direkt auf der erzeugten Instanz modelliert.

Um bereits modellierte Workflows für zukünftige Ausführungen wiederverwenden zu können, werden konsequent alle aktuell vorliegenden Modellierungen von Workflow-Instanzen nebst entsprechenden Modifikationsprotokollen (Audit-Data) in einer Datenbank gehalten, dem sogenannten Audit-Repository. In einer Reengineering-Phase wird zu einem gegebenem Zeitpunkt (durch einen versierten Anwender des Workflow-Systems) ein Workflow-Modell anhand der ausgeführten und modifizierten Workflow-Instanz angepasst bzw. – im Falle zu großer Abweichungen von dem Original – eine Workflow-Instanz zu einem neuen Workflow-Modell konvertiert.

Durch diese Vorgehensweise werden die Modelle sukzessive mehr und mehr den Gegebenheiten der Wissensarbeit angepasst (vergleiche 4.3). Es liegt nahe, dass dieser Vorgang einiges an Fachwissen voraussetzt, da hier Generalisierungen und geeignete Parametrisierungen vonnöten sind.

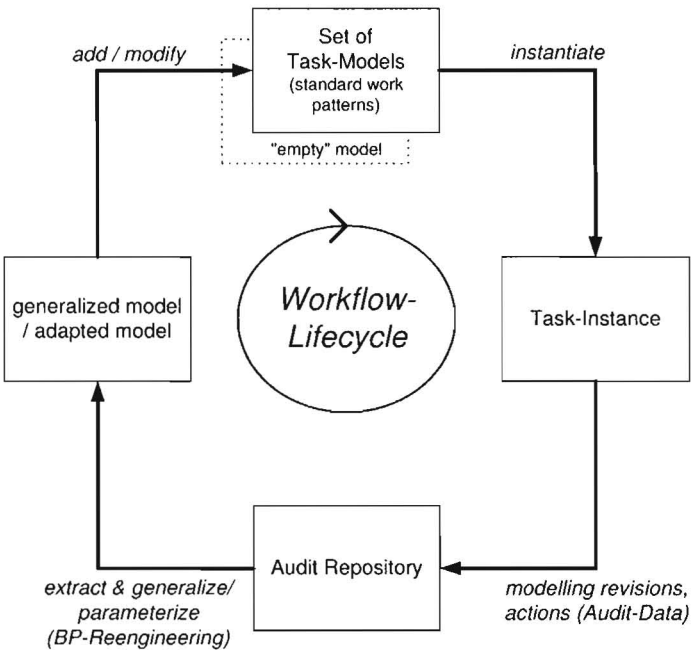

Abbildung 4.3: Der Workflow-Lifecycle.

Das so erhaltene Workflow-Modell beschreibt eine (generische) Standardvorgehensweise zu einer bestimmten Aufgabe und kann nun wiederum dem Workflow-System als neues Modell zur Verfügung gestellt werden. Bei Bedarf wird dieses Modell instanziiert, gegebenenfalls auf neue Bedürfnisse angepasst, eventuell wieder parametrisiert und erneut in modifizierter Form dem System zur Verfügung gestellt, so dass nach vielfachem Durchlaufen dieses Zykels

nach und nach eine Menge von Modellen entsteht, die die Wissensprozesse des Unternehmens abbildet.

Dieser angesprochene Zyklus findet sich in Abbildung 4.3 wieder und beschreibt optisch exakt den gleichen Workflow-Lebenszyklus, wie man es vom traditionellen Workflow-Paradigma her kennt. Ein grundlegender Unterschied besteht jedoch darin, dass die Modellierung (i) auch *zur Laufzeit* und (ii) auf den Workflow-*Instanzen* geschieht statt nur *a priori* und/oder auf *Modellen.*

Durch unser Verständnis eines Tasks liegt es nahe, dass wir nicht bloß ganze Task-Modelle generalisieren und parametrisiert als Modelle für zukünftige Task-Instanzen zur Verfügung stellen, sondern auch die Teile davon (so etwa eine Task mit samt ihren Sub-Tasks, die eine Aufgabe realisieren, die auch in anderen Modellen benötigt werden könnte). Dies ermöglicht eine Modellierung eines Prozesses durch Zusammenbau verschiedener Task-Modelle, die jeweils Teilaufgaben umsetzen.

Aufgabenkonzepte (Task-Concepts)

Aufgaben besitzen einen Namen, der vom Benutzer während der Modellierung vergeben wird. Da der Benutzer frei in der Wahl des Namens ist, können unterschiedlichste Bezeichnungen für ein und denselben Typ von Aufgabe auftreten. So etwa für die Zusammenstellung von Literatur für eine Publikation: `LitarturSuche`, `LitSuche`, `Suche Literatur`, `Sichte Related Work`, etc. Für einen externen Beobachter, wie etwa einen Informationsagenten, ist damit die Erfassung der Aufgabe erschwert bzw. unmöglich. Er müsste zusätzlich andere Quellen betrachten, wie die textuelle Beschreibung der Aufgabe.

Daher soll zur zusätzlichen Beschreibung einer Task ein sogenanntes *Aufgabenkonzept* (Task-Konzept) benutzt werden, welches die Aufgabenart repräsentieren soll. So könnte man der Suche nach einer Adresse oder nach Literatur im Internet das Task-Konzept *InternetSuche* zuordnen. Eine informelle Beschreibung könnte sein: "Tätigkeit mit dem Ziel einen Informationsbedarf zu befriedigen unter Zuhilfenahme eines Internet-Dienstes". Die Verwendung dieses Konzeptes für eine Task gibt dann einem Agent den Hinweis, dass ein vorhandener Informationsbedarf durch einen Internet-Dienst befriedigt werden kann.

Die Semantik eines Konzeptes wird zu Beginn durch eine solche informelle Beschreibung der Aufgabe gegeben sein. Im Laufe der Systemnutzung kann einem Task-Konzept ein Task-Modell, das diese Task umsetzt, zur Beschreibung zugeordnet werden. D.h. ein Task-Konzept erhält so auch eine Beschreibung durch ein Modell. Der Umfang des Modells kann von einer einzelnen Task bis hin zu einer Sequenz von Tasks reichen.

Die Task-Konzepte werden in einer Ontologie repräsentiert, welche verschiedene Relationen enthält wie Spezialisierung, Aggregation und Ähnlichkeit. Damit steht ein ausdrucksstarkes Mittel zur Verfügung, semantisch reichhaltigere Task-Modelle zu erstellen. Abbildung 4.4 zeigt ein Beispiel für eine solche Task-Konzept-Ontologie.

Da man durch die Relationen verwandte Task-Konzepte findet, kann die Task-Konzept-Ontologie den Benutzern auch als Modellierungshilfe dienen. So könnte z.B. ein Informationsagent bei einer Task `LiteraturRecherche` mit zugewiesenem Task-Konzept `C:Lit-Rech`, über die Generalisierung `C:Recherche`, das speziellere Konzept `C:WWW-Rech` finden und das eventuell vorhandene Task-Modell als Verfeinerung vorschlagen. Somit dient diese Ontologie als eine zusätzliche Sicht auf das Modell-Repository des Workflow-Systems.

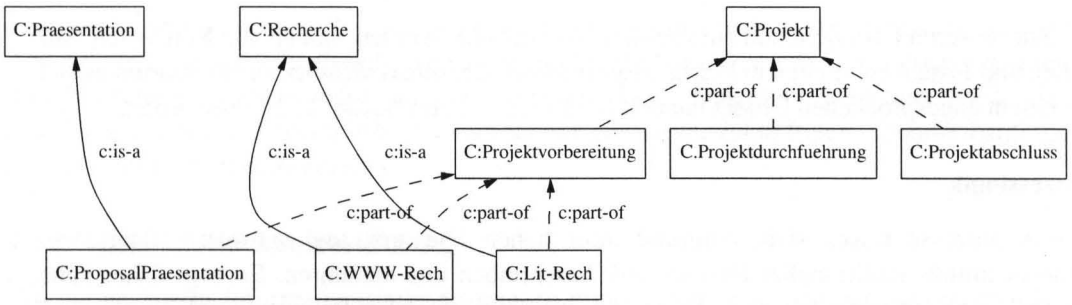

Abbildung 4.4: Beispiel für eine Aufgaben-Konzept-Ontologie.

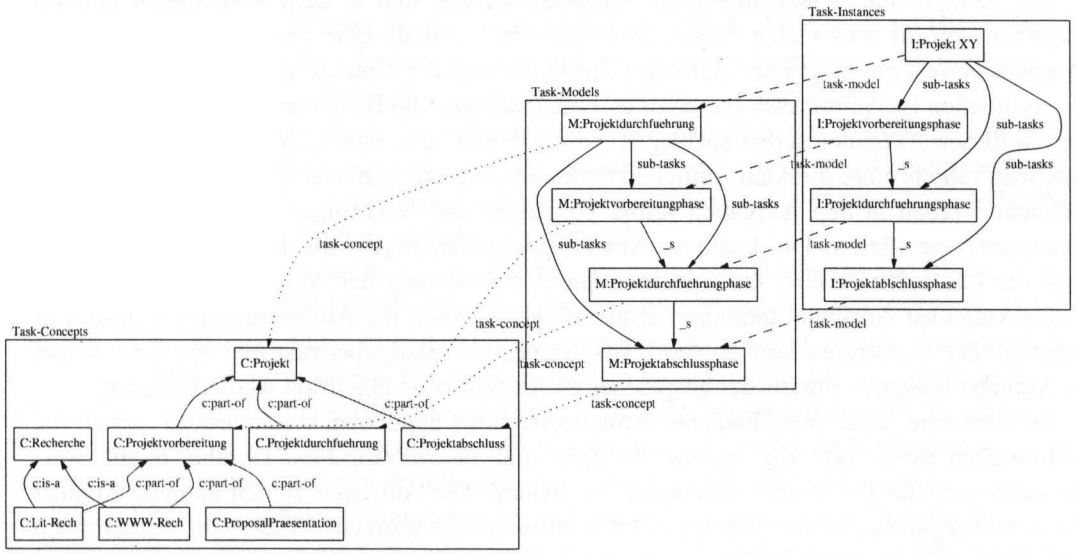

Abbildung 4.5: Konzepte, Modelle und Instanzen[3].

Beziehung zwischen Task-Konzepten, -Modellen und -Instanzen

Die Abbildung 4.5 verdeutlicht das Zusammenwirken von Task-Konzepten, -Modellen und -Instanzen. Der Übersicht wegen wurden hier die Konzeptannotationen (`task-concept`) bei den Instanzen weggelassen. Tatsächlich besitzen jedoch sowohl Modelle als auch Instanzen diese Konzeptannotationen als Möglichkeit, die Funktion der entsprechenden Aufgabe semantisch zu beschreiben.

Die Abbildung zeigt außerdem nicht, dass die Konzepte von den realisierenden Modellen und Instanzen wissen. Tatsächlich ist dies aber der Fall. Genaugenommen bieten die Task-Konzepte in der Task-Konzept-Ontologie die Möglichkeit, semantisch nach Modellen und Instanzen zu suchen, wie bereits im vorigen Abschnitt beschrieben.

Die Relation `_s`, die ebenfalls in der Abbildung dargestellt ist, liegt in Wirklichkeit nicht in dieser expliziten Form vor, sondern muss erst aus der Modellierung (Prozesslogik) inferiert werden. Sie stellt hier direkt die Vorgänger-Nachfolgerbeziehungen zwischen den einzelnen Tasks grafisch dar.

[3]Der Übersicht wegen wurden hier die Konzeptannotationen (`task-concept`) bei den Instanzen weggelassen. `_s` stehen für die implizit vorliegenden Nachfolger-Beziehungen.

Zur besseren Übersicht und zum besseren Verständnis werden Namen von Konzepten, Modellen und Instanzen mit einem Präfix ausgezeichnet, der direkt sichtbar macht, worum es sich bei einem angesprochenen Objekt handelt (`C:Konzept`, `M:Modell`, `I:Instanz`).

Prozesslogik

Wissensprozesse lassen sich, aufgrund ihrer hohen Dynamik und schweren Planbarkeit, schlecht mittels traditioneller Prozesslogik beschreiben und ausführen. Es liegt daher nahe, sowohl die Spezifikation als auch die Art der Ausführung solcher Prozesse auf deren Beschaffenheit und Bedürfnisse hin anzupassen.

Die Komplexität wissensintensiver Aufgaben schlägt sich in dem Vorkommen und Interagieren von oft sehr vielen Unteraufgaben nieder, weshalb eine zentral gehaltene, flache (beispielsweise Flussdiagramm-ähnliche) Strukturierung der Unteraufgaben von vorneherein auszuschließen ist. Stattdessen erscheint es zweckmäßiger, die Beziehung zwischen Aufgaben eben in diesen Aufgaben selbst abzulegen, was bedeutet, dass man *nicht* die Reihenfolge von Aufgaben mittels einer direkten zeitlichen Ordnungsrelation (in planer Sichtweise) modelliert. Vielmehr werden in den Aufgaben selbst Vorgänger und Nachfolger in Form von Ablauf-*Constraints* spezifiziert. Die konkrete Abarbeitungsreihenfolge wird aus diesen Constraints dann zur Laufzeit *inferiert*. Zur Spezifikation und Steuerung der Ablaufconstraints können in den Aufgaben Attribute definiert werden, deren Werte in die Auswertung der Constraints eingehen. Die Constraints können somit aus Termen über Aufgabenzustände bestehen, wobei die Aufgabenzustände mittels der gespeicherten Attributwerte abgefragt werden können.

Es wird eine Reihe verschiedener Ablaufconstraints angeboten – mindestens jedoch die traditionellen Konstrukte wie *and-/or-/xor-Split* und *and-/or-/xor-Join*. Darüber hinaus werden auch spezielle Konstrukte angeboten, die dynamische Anfragen an den aktuelle Kontext erlauben (beispielsweise die Verfügbarkeit bestimmter Ressourcen). Außerdem werden noch zeitliche Bedingungen wie "muss spätestens am 31.12.2001 abgeschlossen sein" angeboten, also zeitliche Constraints zur Spezifikation von Deadlines, Dauer und frühestmöglichem Beginn einer Aufgabe.

4.3.2 Ausführung

Die Geschäftsprozesse liegen also in Form von hierarchisch strukturierten Aufgaben vor, deren Ablaufbeziehungen untereinander durch Ablaufconstraints spezifiziert sind. Diese Constraints werden zur Laufzeit ausgewertet, um dem Benutzer aktuell durchführbare Aufgabenangebote zu unterbreiten. Gegebenenfalls kann sogar eine zukünftige Abarbeitungsreihenfolge von nachfolgenden Aufgaben errechnet und vorgeschlagen werden. Wohlgemerkt handelt es sich hierbei jedoch bloß um Vorschläge, nicht um Vorschriften.

Aufgrund seiner menschlichen Kompetenz oder gerade neu erworbenen Informationen kann der Benutzer diesen Vorschlag ignorieren und andere Entscheidungen treffen. Idealerweise würde der Benutzer eine begründbare Entscheidungsänderung damit untermauern, dass er die entsprechende Aufgabe mit den zur Änderung führenden Faktoren nachmodelliert, so dass bei erneuter Berechnung der Ablaufconstraints nun der "richtige" Vorschlag vom System kommt.

In [Schwarz, 2000] haben wir die Ablaufbeziehungen verschiedener Aufgaben bereits durch Constraints in Form von Termen über Aufgabenattributen spezifiziert. In einer prototypischen Beispielapplikation wurden dort alle diese Constraints mittels eines Constraint-Solvers

ausgewertet und damit sowohl eine *aktuelle* Menge von durchführbaren Aufgaben als auch eine *mögliche* Ablaufreihenfolge zukünftiger Nachfolge-Aufgaben inferiert. Letztere wurden dem Benutzer präsentiert, indem ein entsprechendes Gantt-Diagramm berechnet wurde, das dem Benutzer einen übersichtlichen Vorschlag zur günstigen, zukünftigen Ablaufreihenfolge in grafischer Form unterbreitet.

4.3.3 Informationsunterstützung

Um die anvisierte Informationsunterstützung zu ermöglichen, muss an den verschiedensten Stellen des Workflow-Systems angesetzt werden. So müssen bei allen beteiligten Workflow-Komponenten geeignete Schnittstellen zur Verfügung gestellt werden, die Zugriff auf relevante Informationen zulassen [Abecker *et al.*, 2000b]. Damit erlaubt z.B. der Zugriff auf Historiendaten abgeschlossener Workflow-Instanzen die Bereitstellung von Informationen für die aktuelle Instanz, um etwa Lösungsschritte für die anstehende Aufgabe vorzuschlagen.

Andererseits gestatten bereits die vorgestellten neuen Konzepte des WfMS die Erschließung zusätzlicher Informationsquellen. So kann die Beziehung der Tasks zu einem beschreibenden Task-Konzept von externen Applikationen benutzt werden. Sie können damit in Erfahrung bringen, um welche Art von Aufgabe es sich handelt.

Der herkömmliche Datenfluss von WfMS, wie etwa in [Leymann and Roller, 2000, Jablonsky and Bussler, 1996] beschrieben, genügt den Anforderungen in *FRODO* nicht. Daher wird der Workflow-Datenfluss zu einem Informationsfluss aufgewertet werden. Hierzu erhalten die im Workflow verwendeten Variablen, zusätzlich zu ihrem Variablentyp, einen Verweis auf ein Konzept der im Framework verwendeten Ontologien. Damit erhalten die Variablen eine "Aussagekraft" gegenüber externen Applikationen, wie sie klassische WfMS nur implizit besitzen, und zwar durch die von dem Workflow-Designer bestimmte Verwendung.

Dieser Informationsfluss hilft auch bei der geforderten unvollständigen Modellierung. Somit kann einer Variable nur ein Konzept zugewiesen werden, das beschreibt deren Wert darstellen soll (etwa "Dokument"), ohne direkt den Inhalt oder die Quelle anzugeben. Daraus kann dann zur Laufzeit der Informationsbedarf einer Task inferiert werden.

Weiterhin bildet der Wissensarbeiter selbst eine wertvolle Quelle für Informationen, die eine adäquate Unterstützung ermöglicht: so etwa seine Interessen, die ein Filterkriterium für relevante Nachrichten liefern, oder seine Kompetenzen, die eine neue Art von Arbeitszuteilung ermöglichen (dies wurde bei klassischen WfMS bisher nur in WorkParty umgesetzt [Ruppietta and Wernke, 1994]). Dazu wird ein ausdrucksstarkes Benutzermodell benötigt, das in das vom Workflow-System benutzte Organisationsmodell eingebunden ist. Weitergehende Betrachtungen hierzu wurden in [van Elst and Abecker, 2001] präsentiert.

4.3.4 Arbeit eines Wissensarbeiters (Szenario)

Wir wollen im folgenden die Vision über die Arbeitsweise von Wissensarbeitern anhand des Beispiels aus Abschnitt 4.1.1 vorstellen.

Dazu stellt Abbildung 4.6 die im folgenden erarbeiteten Objekte und Relationen bereits dar. Die Abbildung dient somit parallel als Orientierungshilfe zur folgenden Szenariobeschreibung.

Nehmen wir als Voraussetzung an, es gäbe noch keinerlei (abrufbare) Informationen über das Know-How, wie ein Projekt durchgeführt wird. Typischerweise wird sich in diesem Fall ein

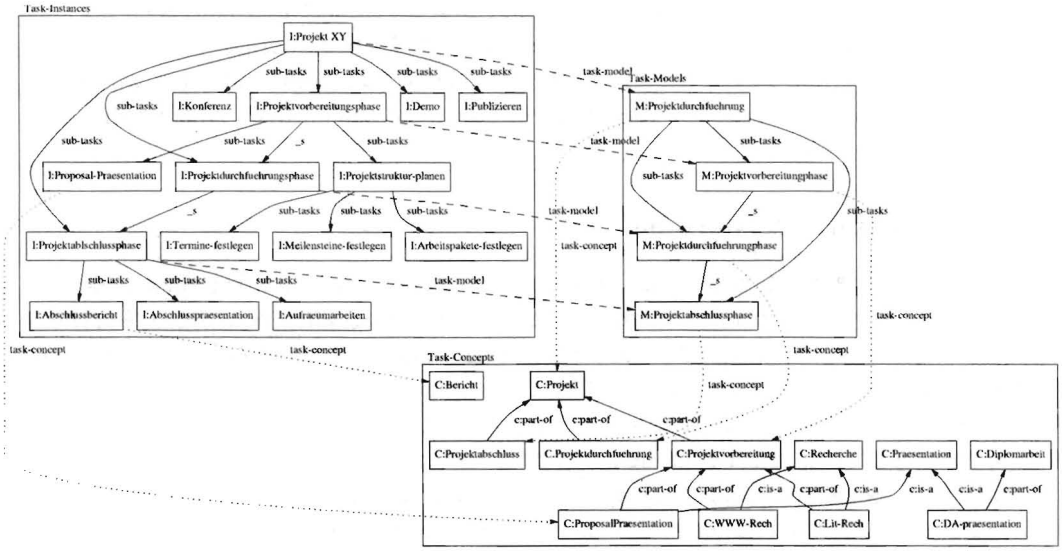

Abbildung 4.6: Objekte und Relationen im hier beschriebenen Szenario.

ausgezeichneter Mitarbeiter mit Leitungsposition, beispielsweise der Projektleiter, zu Beginn ein grobes Bild des Problems machen und erhält in etwa folgende grobe Struktur:

Projektdurchführung – Grobstruktur:

- Projektvorbereitungphase
- Projektdurchführungphase
- Projektabschlußphase

Er wird ein neues Modell (Task-Modell) `M:Projektdurchführung` erzeugen und drei Teilaufgaben als Sub-Tasks dieser Task darin anlegen, d.h. die drei Teilaufgaben wird er zunächst als "Black-Boxes"[4] realisieren, aber bereits mit dem Wissen anreichern, dass sie zumindest in der folgenden Reihenfolge angeordnet werden sollten: `M:Projektvorbereitungphase` → `M:Projektdurchführungphase` → `M:Projektabschlußphase`.

Am Projektanfang steht das Projektproposal als Input und am Projektabschluß steht der Abschlußbericht als Output. Daher werden diese auch als solche in der `M:Projektdurchführung` modelliert (alternativ dazu könnte man diese Information natürlich ebenso gut auf `M:Projektvorbereitungphase` und `M:Projektabschlußphase` verteilen).

Der Projektleiter muss nun nicht das gesamte Modell bis ins kleinste Detail modellieren. Stattdessen lässt er es nun dabei und instanziiert es als `I:ProjektXY`. Genauer bedeutet dies, dass eine instanziierte Kopie von `M:Projektdurchführung` angelegt wird, welche dann `I:ProjektXY` (eine Task-Instanz) ist. Die Sub-Tasks von `M:Projektdurchführung` werden dabei ebenfalls kopiert, woraufhin die Tasks (Task-Instanzen) `I:Projektvorbereitungsphase`, `I:Projektabschlussphase` und

[4] Black-Boxes sind (noch) unspezifizierte Teilaufgaben, die lediglich mit einer informellen Beschreibung versehen sind. Das Lösen einer solchen Aufgabe verlangt menschliche Erfahrung / Intelligenz und wird nicht weitergehend maschinell unterstützt.

I:Projektdurchführungsphase als Sub-Tasks von I:ProjektXY entstehen.
Er startet daraufhin beispielsweise mit der Bearbeitung der Task I:Projektvorbereitungsphase. Zu dieser Bearbeitung stehen ihm folgende Möglichkeiten offen:
(a) Task vollends bearbeiten und abschließen, sozusagen als eine zusammenhängende Aufgabe, die vom System nur als Black-Box angesehen wird.
(b) Aus früherer Erfahrung kennt er bereits wichtige Teile der Projektvorbereitungsphase und modelliert diese direkt hinzu, und zwar durch Verfeinerung dieser Task mit den Sub-Tasks I:Proposal-Präsentation und I:Projektstruktur-planen. Dabei unterteilt er auch gleich die Task I:Projektstruktur-planen in weitere Sub-Tasks. Es entstehen I:Meilensteine-festlegen, I:Arbeitspakete-festlegen und I:Termine-festlegen. In der Teilaufgabe I:Arbeitspakete-festlegen hinterlegt er eine Information, dass bei dieser Aufgabe die Zuständigkeiten an die Mitarbeiter verteilt werden sollen (informelle Beschreibung der Task, die zur Zeit lediglich als Black-Box modelliert ist).

Zur gleichen Zeit hilft ein Kollege bei der aktuellen Modellierung des Projektes XY, indem er versucht, die Task I:Proposal-Präsentation weiter zu verfeinern. Zur Unterstützung bei dieser Tätigkeit kann er das Modell-Repository als Informationsquelle nutzen:

Er kann das Modell-Repository nach bereits existierenden Modellen absuchen, die eine Präsentation realisieren, beinhalten bzw. damit verwandt sind. Dazu sucht er in der Konzept-Ontologie nach einem entsprechenden Aufgaben-Konzept. Hat er ein passendes Konzept gefunden, wird dieses ihm Modelle liefern, die das Konzept realisieren (ein Task-Konzept kennt seine realisierenden Modelle, genauso wie auch die Modelle das Task-Konzept kennen).

Weiter im Beispiel bedeutet das konkret: Ausgehend von dem Aufgabenkonzept C:Diplomarbeit, sucht er das die Aufgabe "Präsentation der Diplomarbeit" repräsentierende Konzept C:DA-praesentation (über die part-of-Beziehung) und von dort das Mutterkonzept (is-a) C:Präsentation.
Daraufhin wird er ein neues Konzept C:ProposalPräsentation als Unterkonzept (is-a) von C:Präsentation erstellen. Er nimmt ein Modell, welches das Konzept C:DA-praesentation realisiert, instanziiert es, passt es auf den Proposal-Fall an und verbindet die neu entstandene Instanz I:Proposal-Präsentation mit dem Aufgaben-Konzept C:ProposalPräsentation.

Man beachte, dass er also ein bereits vorhandenes Modell als Informationsquelle nutzen und als Baustein in eine andere Modellierung einbauen kann. Man beachte außerdem, dass die Suche nach einem passenden "ähnlichen" Modell durch Browsen in der Task-Konzept-Ontologie geschieht. Dementsprechend wird also nicht auf einer syntaktischen Ebene oder einer willkürlich definierten Struktur, sondern auf einer semantischen Ebene gesucht.

Kann kein passendes Modell gefunden werden, aber es ist bekannt, dass eine Instanz eine Teilaufgabe mit ähnlicher Zielsetzung realisiert, so kann diese Instanz zumindest als Vorlage und Informationsquelle dienen. Ein direktes Kopieren einer Instanz ist ein nicht-trivialer Akt und muss mit Vorsicht und Expertenwissen erledigt werden, erfordert es doch meist eine korrekte Generalisierung mit nachfolgender Instanziierung. Genaugenommen erfordert dieser Akt die Generierung eines (temporären) Modells einer Instanz innerhalb einer *Reengineeringphase.* Anstatt eine Instanz maschinell zu kopieren, muss also vielmehr aus dieser Instanz (von einem versierten Anwender) zunächst ein Modell extrahiert werden, welches dann *in einem zweiten Schritt* wiederum instanziiert werden kann.

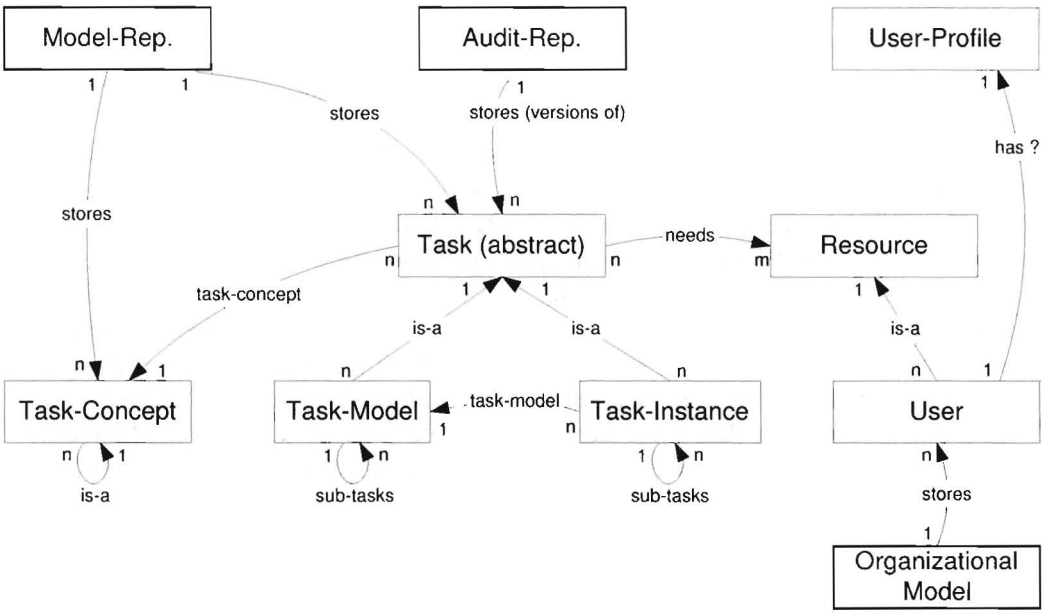

Abbildung 4.7: Daten-Objekte im FRODO Workflow-System.

Diese Vorgehensweise ist aufgrund der bereits beschriebenen nicht-trivialen Problematik zwingend, bedeutet aber keinerlei Modellierungsbeschränkung, da trotzdem jeder Anwender sein Modell an die Vorlage anpassen kann.

Mit dieser Bemerkung soll die Diskussion des Szenarios beendet werden. Ein erster Einblick in die angestrebte Wissensarbeit konnte hiermit sicherlich geboten werden.

4.4 Komponenten im Workflow-System

Nachdem die Objekte und Relationen in vorhergehenden Abschnitten oberflächlich behandelt wurden, soll dieser Abschnitt dazu dienen, konkrete Komponenten innerhalb des FRODO-Frameworks zu identifizieren und ihre Funktion sowie ihr Zusammenspiel zu beschreiben. Im späteren Verlauf des Projektes werden diese Komponenten dann innerhalb einer Agentenplattform tatsächlich realisiert und implementiert.

Im folgenden werden zunächst die Daten-Objekte und danach die Agenten beschrieben.

Zur Verdeutlichung der folgenden Beschreibung der Daten-Objekte, die innerhalb der Agentenplattform von Bedeutung sind, sei auf Abbildung 4.7 hingewiesen, welche die Daten-Objekte und ihre Beziehungen zueinander darstellt.

Model-Repository: Das Model-Repository speichert und verwaltet Task-Modelle, -Instanzen und -Konzepte. Somit enthält dieses Repository das komplette, aktuelle Workflow-Wissen: Die Instanzen stellen die aktuell laufenden Workflows dar, die Modelle die Bauteile, aus denen diese Workflows zusammengesetzt wurden, und die Konzepte statten Aufgaben mit einer semantischen Beschreibung (Annotation) aus.

Task-Concept: Eine Task-Konzept-Ontologie verwaltet eine Menge von Aufgabenkonzepten (Task-Concepts), angereichert mit Beziehungen zwischen diesen Konzepten. Die Konzepte werden dadurch *formell* in Beziehung zueinander gesetzt, wobei zumindest die `is-a`-Beziehung vorhanden ist und somit mindestens eine Taxonomie von Aufgabenkonzepten in der Ontologie erstellt werden kann. Eingesetzt werden diese Konzepte in Task-Modellen und -Instanzen zur semantischen Annotation der entsprechenden Tasks. Der Infoagent kann die Konzepte und deren Beziehungen untereinander zu Inferenzzwecken nutzen, um semantikspezifische Informationsbereitstellung zu realisieren. Über die taxonomische Beziehung von Aufgabenkonzepten hinaus sollten weitere (gegebenenfalls attributierte) Beziehungstypen in der Konzeptontologie integriert werden, um die Aufgabenkonzepte bestmöglich an die reale Welt der Aufgaben/Tätigkeiten anzupassen. Ein idealisiertes Ziel ist ein Abbild der erforderlichen realen Aufgaben/Tätigkeiten in Form einer Task-Concept Ontology.

Task-Model: Die Task-Modelle stellen Aufgabenbeschreibungen für Standardvorgehensweisen dar (entspricht einem Workflow-Modell). Zum Lösen einer Aufgabe oder eines Auftrags werden diese Modelle in instanziierter Form in Task-Instanzen verwendet, wo sie ausgeführt werden können. Dazu zwei Anmerkungen: (1) Instanziierte Modelle können auch bloß als Teil einer Instanz Verwendung finden. (2) Modelle sind generalisierte Vorgehensweisen und liegen daher oft in parametrisierter Form vor, d.h. manche parametrisierte Modellierung muss erst durch Wertzuweisung von Parametern während der Instanziierung in eine greifbare Modellierung umgewandelt werden.

Oft besteht eine Aufgabe in einem Geschäftsprozess aus mehreren Unteraufgaben. Daher kann ein Modell beliebig viele Unter-Modelle (=Task-Modelle) enthalten, was der Aufteilung einer Aufgabe in Unteraufgaben entspricht. Realisiert wird diese Aufteilung durch die `sub-tasks`-Relation.

Jedes Modell (also auch jedes Unter-Modell) kann durch Verweisen auf ein Task-Konzept semantisch annotiert werden.

Task-Instance: Die Task-Instanzen stellen die wirklich ausführbaren Workflows dar (entspricht einer Workflow-Instanz). Auch sie haben, analog zu den Modellen, Verweise auf Task-Konzepte und Ressourcen.

Außerdem besitzen die Task-Instanzen noch Verweise auf die Modelle, aus denen sie (durch Instanziierung) entstanden sind. Zu beachten ist dabei die Tatsache, dass nicht alle Instanzen aus Modellen hervorgehen. Während der Ausführung können Unter-Instanzen gebildet werden, die manuell, also ohne Verwendung / Einbau von Modellen, erstellt werden. Diese Sub-Tasks besitzen dann auch keine `task-model` Beziehung zu einem Modell.

Audit-Repository: In diesem Repository werden alle Versionen (Revisions) von Task-Modellen und -Instanzen zusammen mit den entsprechenden Modellierungsänderungen gespeichert. Damit sind Recherchen auf älteren Versionen möglich, beispielsweise Sichten einer älteren Version oder Sichten der entsprechenden Änderungen, die zu einer neuen Version geführt haben oder Abfrage nach der Person, welche die Änderungen durchgeführt hat.

Außerdem werden sämtliche "Aktionen" gespeichert (*Audit Data* [WfMC, 1999]): Jedes Mal, wenn eine Task-Instanz (teilweise) abgearbeitet wird, wird diese "Aktion" im Audit-Repository mitprotokolliert.

Resource: Dieses Datenobjekt kapselt eine Ressource, beispielsweise einen Raum oder einen Overheadprojektor, aber auch eine Informationsquelle (Datenbank, o.ä.). Generell werden alle verfügbaren Ressourcen durch Ressource-Agenten vertreten. Der Zugriff auf Ressourcen wird ausschließlich über diese Agenten erfolgen. Stehen direkte Informationen zur Verfügung, gibt es ein Ressource-Objekt, auf das ein entsprechender Ressource-Agent zugreift, ansonsten gibt es *nur* den Ressource-Agenten.

User: Benutzer werden durch dieses Datenobjekt vertreten. Ein Benutzer ist eine spezielle Ressource und besitzt damit mindestens die Eigenschaften und Funktionen einer Ressource. Ein Benutzer kennt seine Rollen, die er ausführen kann und seine Position innerhalb des Unternehmens. Diese Information ist im Organizational Model hinterlegt.

Da Benutzer auch Ressourcen sind, haben sie demzufolge auch jeweils einen Agenten, der die Verfügbarkeit und Nutzbarkeit seines Benutzers mit anderen Komponenten des Agentenframeworks aushandelt.

User-Profile: Hier werden Daten zur Spezifizierung des Benutzerprofils gespeichert. Ein solches Benutzerprofil bietet folgende benutzerspezifischen Informationen über den Benutzer: Präferenzen, Gewohnheiten, Interessen, Know-How-Level (z.B. GUI-Bedienung), Knowledge-Level/-Typ (wie arbeitet der Benutzer mit Wissen / welche Art von Wissen erwartet der Benutzer?).

Organizational Model: Das Organizational Model umfasst alle Personen einer Organisation, Projekte und organisationsspezifische Beziehungen (Organigramm, Projektzuteilung, Über-/Untergebene von Personen etc.).

4.4.1 Agenten im FRODO Workflow-System

Nachdem die Daten-Objekte behandelt wurden, beschreibt der nun folgende Teil die Agenten innerhalb der Agentenplattform (siehe Abbildung 4.8 – aus Übersichtsgründen wurden nur die wichtigsten Kommunikationsformen zwischen den Agenten dargestellt).

In [Weiss, 1999] werden "intelligente, interagierende Agenten" wie folgt definiert:

> "Agents" are autonomous, computational entities that can be viewed as perceiving their environment through sensors and acting upon their environment through effectors. To say that agents are computational entities simply means that they physically exist in the form of programs that run on computing devices. To say that they are autonomous means that to some extent they have control over their behavior and can act without the intervention of humans and other systems. Agents pursue goals or carry out tasks in order to meet their design objectives, and in general these goals and tasks can be supplementary as well as conflicting.

Im folgenden wird zunächst beschrieben, was die dort angesprochenen Eigenschaften für die Agenten in unserem Workflow-System bedeuten. Danach folgt eine genauere Beschreibung der einzelnen Agenten.

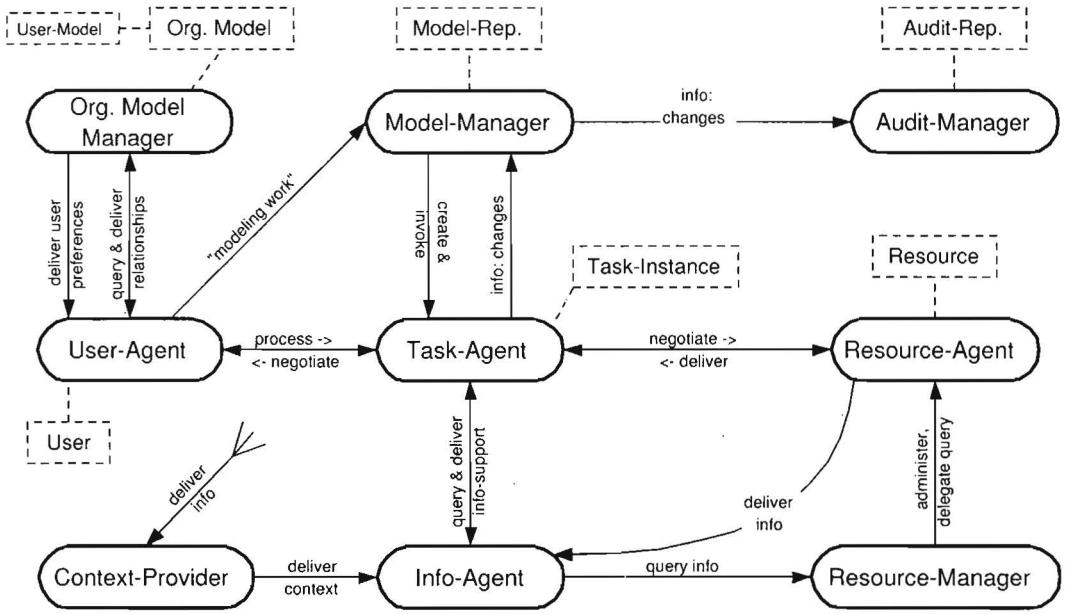

Abbildung 4.8: Agenten im FRODO Workflow-System (Agentenplattform).

Agenten erledigen individuelle Aufgaben selbstständig und autonom. Globale und gemeinsame Aufgaben lösen Agenten lediglich durch Verhandlung und Kommunikation.

Agenten besitzen eine Reihe von Fähigkeiten. Eine Reihe dieser Fähigkeiten bietet der Agent verpflichtend an (was sich allerdings während der Laufzeit ändern kann) und stellt sie der Öffentlichkeit (anderen Agenten) zur Verfügung. Diese werden im folgenden als "Dienste" bezeichnet.

Fähigkeiten, die Agenten selbst intern benutzen, um ihre Aufgaben zu erfüllen, werden niemals verpflichtend angeboten und werden auch vor der Öffentlichkeit geheim gehalten. Diese, im folgenden "Aktionen" genannt, werden lediglich aufgelistet, um die Möglichkeiten des Agenten hier zu demonstrieren und daraus Anforderungen für die Agentenplattform abzuleiten (S. 74 ff). Aktionen, die in natürlicher Weise aufgrund von analogen Diensten erforderlich sind, werden nicht extra aufgeführt.

In der Regel werden die Agenten im FRODO-Framework kooperativ zusammenarbeiten. Daher sind zur Ausführung der meisten Funktionen hier keine komplizierten, mehrschichtigen Verhandlungsprotokolle vorgesehen. Dennoch kann ein Agent jederzeit die Annahme eines Auftrags (Nutzen einer Funktion) verweigern, falls die Situation dies erfordert.

Zum Lösen der individuellen Aufgabe(n) eines Agenten besitzt er eigene *Ziele*, eigenes *Wissen* und gegebenenfalls eine Möglichkeit, seine *Umwelt* (teilweise) wahrzunehmen. Es ist offensichtlich, dass eine Umweltwahrnehmung insbesondere Wissen impliziert, daher wird triviales Wissen aufgrund von Umweltwahrnehmungen im folgenden nicht explizit angegeben.

Fehlen diese Eigenschaften (Ziele, Wissen, Umweltwahrnehmung), so handelt es sich weniger um echte Agenten, sondern mehr um (Hilfs-)Komponenten, die aber trotzdem hier vorgestellt werden, da sie im Zusammenspiel mit den Agenten nichtsdestoweniger wichtig sind (z.B. um komfortablen und geregelten Zugang zu Repositories zu ermöglichen).

Ausgehend von diesen Überlegungen werden nun die einzelnen Agenten aufgelistet.

Organizational Model Manager

Wissen: Organizational Model.

Dienste:

- R/W-Zugriff auf Daten des Organizational Models.
- Ontologietypische Anfragen an das Organizational Model

Model-Manager

Wissen: Model-Repository.

Ziele: Zu jeder Task-Instanz, die nicht durch Sub-Tasks *voll* subsumiert wird, muss ein eigener, zugehöriger Task-Agent existieren und laufen.

Dienste:

- R/W-Zugriff auf spezielle Modelle, Instanzen & Konzepte und ggf. darin vorliegende Daten.
- Suchanfragen nach Modellen, Instanzen & Konzepten anhand übergebener Eigenschaften (z.B. Ontologie-Eigenschaften).
- Suchanfragen nach laufenden Instanzen eines Modells.
- Anfragen bzgl. des Workflow-Meta-Modells.
- Klonen/Kopieren & Instanziieren eines Modells

Aktionen:

- Anfragen an Domänen-Ontologie(n)
- Erstellen und Starten eines Task-Agenten für eine Task-Instanz
- Übermitteln der Information über eine neue Modellierung bzw. eine Modellierungsänderung an den Audit-Manager

Audit-Manager

Wissen: Audit-Repository.

Dienste:

- Anfrage bzgl. Daten des Audit-Repositories (Modellierungen, -änderungen, alte Modellierungsversionen, "Aktionen" = Bearbeitungsschritte innerhalb einer Task).
- Speichern eines neuen Modells / einer neuen Instanz.
- Protokollieren einer Modellierungsänderung / einer "Aktion".

Task-Agent

Es besteht eine 1:1-Beziehung zwischen einem Task-Agenten und einer Task-Instanz.
Beachte: Eine Task-Instanz kann sowohl eine "Workflow-Instanz" (im herkömmlichen Sinne) als auch eine Task innerhalb einer solchen Workflow-Instanz sein. Das bedeutet insbesondere, dass es sehr viele Task-Agenten geben wird (pro Task ein Agent, inklusive Ober- und Sub-Tasks).
Allerdings sind diese Agenten, analog zu den zugehörigen Tasks, möglicherweise ebenfalls hierarchisch strukturierbar und gegebenenfalls auch hierarchisch (de-)aktivierbar etc.

Wissen:

- Task-Instanz plus zugehöriges Task-Modell und -Konzept.
- Domänen-Ontologie(n).

Ziele: Durchführung, (erfolgreiches) Beenden der Task-Instanz. Dazu gehört beispielsweise auch, dass passende Benutzer entsprechende Einträge in ihren "Worklists" vorfinden werden.

Dienste:

- Anfrage: Ist diese Task zur Zeit durch-/ausführbar?
- Aktion: Führe Task (teilweise) aus!
- Anfrage: zugehörige Task-Instanz / -Modell.

Aktionen:

- Anfragen an Domänen-Ontologie(n)
- Verhandlungen mit User-Agenten und Ressource-Agenten über deren Verfügbarkeit/Nutzbarkeit
- Verhandlungen mit Ressource-Agenten (OM-wrappers) über die Lieferung von Information bzw. Verhandlungen zur Auswahl *eines* Informationsproviders unter mehreren
- Modifikation der Modellierung der zugrunde liegenden Task-Instanz

Umwelt: Die Umwelt dieses Agenten ist gegeben durch die Stellung und den Aufbau der Task-Instanz sowie durch den "Kontext", der vom Context-Provider geliefert wird. Der Zugriff auf Domänen-Ontologie(n) wird außerdem noch zusätzliches Semantik-Wissen zur Umwelt vermitteln.

Context-Provider

Der Context-Provider liefert dem Info-Agenten Kontext auf Anfrage. Dazu nutzt der Provider viele Informationsquellen[5] und deren Beitrag zum Kontext. Der Kontext wird dynamisch

[5] Als Informationsquelle kommt für eine Task-Instanz der Verweis auf ein Aufgabenkonzept, das ursprüngliche Task-Modell, sowie zu Task-Instanzen dieses Modells in Frage. Abgesehen von diesen Beispielen sind noch weitere denkbar, die allerdings noch nicht Gegenstand dieses Meilensteinpapiers sein sollen.

zusammengestellt. Damit eine Anfrage nicht mit allen verfügbaren Informationen überfrachtet wird, besitzt der Context-Provider auch Heuristiken, welcher Kontext für wen zu welchen Zeitpunkt relevant ist (dienen als eine Art Filter).
Der Context-Provider ist nur als zusätzliche Informationsquelle gedacht, der etwa zu einer Anfrage des Info-Agenten an einen Retrieval-Agenten zusätzliche Informationen über den Task-Kontext gibt, Anfragen und Antworten aus früheren Task-Instanzen beifügt, das Interessenprofil des Benutzers bereitstellt, etc. Die Verwendung des Kontextes liegt dann in der Verantwortung des Retrieval-Agenten.

Umwelt/Wissen:

- "alle" Informationsquellen (Ressourcen) im Framework
- Informationsangebot je Ressource
- Heuristiken für die Kontextbereitstellung
- Kontext-Modell

Ziele:

- möglichst sinnvolle und angepasste Lieferung von Kontext

Dienste:

- Anfrage nach Kontext zu einer bestimmten Aktion (etwa Dokument-Retrieval)

Aktionen:

- Sammeln von Informationsquellen (Anfrage an Ressource-Manager nach Ressourcen)
- Anfrage der einzelnen Ressource nach Informationsart und anschließender Ablage
- Bereitstellung von Kontext durch Anfrage von Ressourcen und Verwendung der Heuristiken

Info-Agent

Der Info-Agent beschafft kontext-sensitive Information zu einem Problem. Dabei geht es um Probleme bei der Bearbeitung / Durchführung / Ausführung von Tasks (Instanzen). Der Info-Agent wird dazu vorwiegend von Task-Agenten befragt.

Umwelt/Wissen:

- Kontext (vom Context-Provider).
- Vom Resource-Manager verwaltet: Ressource-Agenten → diese sind nicht bloß Ressourcen wie Räume etc., sondern auch Informationsquellen (OMs).
- Kontext (vom Context-Provider) – Kontext ist hier sowohl Umwelt als auch Wissen. Zum Kontext gehört z.B. das Wissen über die durchzuführende Task, User-Profil (Interessen, ...) etc.
- Domänen-Ontologie(n) → Semantik-Information über Objekte und deren Beziehungen.

- User-Profil

Ziele: Möglichst gute, kontext-sensitive, aktuelle Information zu beschaffen und dies möglichst zeitnah.

Dienste:

- Anfrage: Informations-Support zu einem Informationsbedarf

Aktionen:

- Anfrage an Context-Provider: Kontext
- Anfrage an Resource-Manager: Informationen zu einem Informationsbedarf bzw. zur Verfügbarkeit

Resource-Manager

Umwelt: die ganze Welt der Ressourcen (Resource-Agents)...

Dienste:

- An- & Abmelden von Ressource-Agenten.
- Such-Anfrage nach einer Ressource mit bestimmten Eigenschaften / direkte Suche nach speziellem Ressource-Agenten
- Informations-Anfrage bzgl. bestimmter (Info-)Eigenschaften → Bestimmen von in Frage kommenden Ressource-Agenten und evtl. Verhandlung um die Lieferung der Information (evtl. auch Verhandlung bzgl. der Auswahl eines Ressource-Agenten unter vielen)

Resource-Agent

Umwelt/Wissen:

- zugrunde liegende Ressource oder Information (OM)
- evtl.: andere kooperierende Ressourcen (Resource-Agents).
- Information darüber, wie man Kontext (vom Context-Provider o.ä.) verarbeitet, um speziellere Anfragen beantworten zu können
- Information darüber, was eine Ressource überhaupt an Kontext-Informationen bietet

Dienste:

- Informations-Anfrage bzgl. bestimmter (Info-)Eigenschaften – evtl. Verhandlung über Lieferung der Information – ggf. unter Einbeziehung eines Kontexts.
- Verhandlung über Nutzen der Ressource.
- Nutzen & Freigeben der Ressource.

Aktionen:

- Anfrage an Context-Provider (o.ä.): Kontext
- Verhandlung mit Task-Agent über Verfügbarkeit / Nutzbarkeit / Lieferung der Ressource / Information

User-Agent

Der User-Agent vertritt den Benutzer mit seinen Wünschen, Rechten, Pflichten und seiner Nutzbarkeit (als Ressource). Er stellt direkt oder indirekt die grafische Schnittstelle zum Benutzer dar. Die Darstellung und Interaktion wird mithilfe des Benutzerprofils des Users angepasst.

Umwelt/Wissen:

- User-Daten und User-Profil
- Tasks (jeweils mit Kontext), die der Benutzer zur Zeit bearbeitet / ausführt.
- Organizational Model (vom Org. Model Manager verwaltet)

Ziele: Kooperativ, solange der Benutzer noch nicht voll ausgelastet.

Dienste:

- alles, was eine Ressource auch anbietet...
- R/W-Zugriff auf Rollen, Position etc. des Benutzers ($\rightarrow$ Org. Model)
- Anfrage: durchführbare / durchführende Tasks (Task-Agents) $\rightarrow$ Verwalten der "Worklist" des Benutzers

Aktionen:

- alles, was eine Ressource auch anbietet...
- Ausführen einer Task (Verhandlung mit entsprechendem Task-Agent)
- Modellierungsänderung (z.B. Verfeinerung)

Workflow-Engine

Ein ausgezeichneter Agent, der einen indirekten, komfortablen Zugriff auf die Workflow-Objekte und -Agenten bietet, so dass die gesamte Funktionalität des Workflow-Enactments von außerhalb genutzt werden kann. Die Funktionen dieser Engine stellen eine von außen zugängliche Schnittstelle für Applikationen dar, die das Enactment als abgeschlossene Komponente nutzen möchten.

4.5 Existierende Workflow-Systeme

Dieser Abschnitt hat die Aufgabe verwandte Workflow-Systeme zu beleuchten und im Sinne der FRODO-Anforderungen zu bewerten. Er gibt einen Einblick in den aktuellen Stand der Technik bezüglich flexibler Workflow-Systeme.

4.5.1 AIS Workware Demonstrator (AWD)

Bei dem AIS Workware Demonstrator [Carlsen and Jorgensen, 1998, Carlsen, 1998] handelt es sich um ein Forschungsprojekt verschiedener norwegischer Forschungs- und Anwendungspartner, dessen Konzept nahezu identisch zu dem in dieser Arbeit vorgestellten Ansatz motiviert wird.

Ziele

- Das System soll für normale Endbenutzer handhabbar sein.
- Integration von Planung und Ausführung: Prozessmodelle sind veränderbar und die Ausführung von unfertigen Modellen ist möglich.
- Es werden Prozess-*Instanzen* modelliert, nicht Prozess-*Definitionen.* Vielmehr werden die Instanzen später archiviert und können in Schablonen umgewandelt werden.
- Hierarchische Dekomposition: Jedes "Work item" (wissensintensive Aktivität) kann in mehrere "Sub Work items" dekomponiert werden.
- Deskriptive Modellierung statt präskriptiver Modellierung.
- Modellierung mittels APM (Action Port Model) [Carlsen, 1998].
- Archivieren von alten Varianten von Workflows mit Speicherung von Bemerkungen bezüglich der Änderungen. Dies ermöglicht ein Dokumentieren gemachter Erfahrungen von alten Workflowphasen bis hin zu aktuellen Workflowinstanzen und -schablonen.

Bewertung

Die Zielsetzungen bei der Entwicklung des AIS Workware Demonstrator ist offensichtlich *sehr* ähnlich zu dem in dieser Arbeit vorgestellten Konzept. Dabei ist allerdings die Prozesslogik einfacher als in unserem Konzept. So sind keine komplexen Terme zur Ablaufsteuerung möglich, da lediglich die Operationen AND, OR und XOR bereitgestellt werden. Ferner ist beim AWD keine generische Wissensmanagement-Anbindung vorgesehen. Die Anbindung zielt stattdessen eher auf Informations- und Ressourcenmanagement ab. Perspektivisch interessant ist das Speichern von Erfahrungen während der Prozessausführung, wie auch das Speichern der alten Versionen (Entwicklungsphasen) eines Workflows.

4.5.2 Continuous Planning and Execution Framework (CPEF)

CPEF [Myers, 1998, Berry and Myers, 1998] realisiert eine intelligente Planung, Durchführung, Kontrolle und *Fehlerbehebung* von kontinuierlichen, komplexen Planungsaufgaben in unvorhersehbaren, dynamischen Umgebungen. Bei dem System handelt es sich um ein Forschungsprojekt von SRI International. Das Anwendungsszenario für dieses System sind militärische Luftschlachten.

Ziele

- Integration von Workflow-Management- und KI-Planungs-Komponenten. Dabei definieren die KI-Planungs-Komponenten eine intelligente, zielorientierte, ereignisgesteuerte, reaktive Kontrolle von Prozess-Aktivitäten.
- Die Komponenten agieren verteilt auf einer Multiagentenplattform. Dies ermöglicht eine fließende, einheitliche Integration der reaktiven Komponenten in das umgebende Workflow-Konzept.
- Die Planung und Fehlerhebung wird durch Verwenden eines "Hierarchical Task Networks" (HTN) realisiert. Dies ermöglicht den flexiblen Aufbau von komplexen, detaillierten Plänen.
- Das System unterscheidet eine *direkte* und *indirekte Ausführung*. Als direkte Ausführung werden Aktivitäten bezeichnet, die vom System selbst unternommen werden könen, während die indirekten Ausführungen alle diejenigen Aktivitäten bezeichnen, deren Ausführung vom System nur überwacht werden können. Bei letzteren handelt es sich um wissensintensive Aktivitäten, die typischerweise stark spontaner, dynamischer und (softwaretechnisch) unkontrollierbarer Natur sind.
- Pläne können aus archivierten Komponenten einer sogennanten "Prozedurbibliothek" zusammengestellt werden.

Bewertung

Bei CPEF lassen sich konzeptionelle Parallelen zu dem in dieser Arbeit vorgestellten Konzept entdecken, allerdings folgt man mit CPEF eher dem Drang nach einer Multiagenten-Achitektur und der damit verbunden Komponentenorientierung. Die fließende, einheitliche Integration von reaktiven Komponenten wird dadurch erleichtert. Ein Fokus von CPEF ist die Analyse und Behandlung von Planungsfehlern und die resultierende Umplanung. Dazu werden sowohl die Planung als auch die Behandlung von Planungsfehlern mittels HTN-Technologie realisiert.

4.5.3 InConcert Workflow

Bei InConcert [Marshak, 1997] handelt es sich um ein Produkt einer Tochterfirma von Xerox, das eine Flexible-Workflow-Architektur realisiert. Das System ist als aktuelle Version "InConcert 2000" ein lauffähiges, kommerzielles Produkt, das sich bereits auf dem Markt etabliert hat.

Ziele

- Objektorientiertes Datenmodell, beliebige Verfeinerungsstufen durch hierarchische Dekomposition.
- Steuerungs- und Speicherungsmöglichkeiten innerhalb einer Aktivität durch "Attribute" und Referenzen auf externe Daten.

- Benutzer sind einer Menge von "User-Pools" zugeordnet, die den "Rollen" im klassischen Workflow-Konzept entsprechen. Diese Rollen können für die Abarbeitung einer Aktivität gefordert werden.
- Vorbedingungen in den Aktivitäten steuern den Prozessablauf.
- Anbindung an Standardprogramme wie MS Project, MS Exchange etc.
- Bereitstellung der Dienste des Workflow-Systems in Form einer Programmierschnittstelle (API).
- Client-Server-Architektur mit zentraler Datenverwaltung im Server unter Einbeziehung einer externen Datenbank.
- Eine Prozess-Instanz kann von der zugehörigen Prozess-Definition abweichen, jedoch nur in einem vorgesehenen Rahmen einer einzelnen Aktivität. Der Kontrollfluss kann nicht geändert werden.
- Jede modellierte Prozess-Definition steht direkt als wiederbenutzbare Schablone zur Verfügung, und ebenso kann jede Prozess-Instanz als Basis neuer Prozesse benutzt werden.

Bewertung

Trotz der sehr ähnlichen Ziele dieses Konzepts im Vergleich zu unserem werden die Benutzer des InConcert-Systems, ganz im klassischen Sinne, in zwei grobe Lager aufgespalten: diejenigen, welche die Aktivitäten einfach nur ausführen und diejenigen, welche die Ausführung der Aktivitäten ändern und kontrollieren können. Dies widerspricht dem großen Schwerpunkt unseres Bestrebens, dass *jeder* Benutzer des Systems seine Arbeit so frei wie möglich ausführen kann, so dass ihn das Benutzen des Systems hinsichtlich seiner Entscheidungsfreiheit weder einschränkt noch hindert. Des weiteren ist eine Wissensmanagement-Unterstützung weder vorgesehen noch geplant.

4.5.4 OpenWater

OpenWater [Whittingham *et al.*, 1998, Whittingham, 1999] ist ein von IBM entwickelter Forschungsprototyp. Motiviert wird die Entwicklung dieser Applikation durch Identifizieren der Schwächen traditioneller Worflows für Ad-hoc-Situationen und Ausnahmen. Eine Analyse vieler wissensintensiver Prozesse ist aufwendig (teuer) und oft nicht machbar. Trotzdem wird dem "Prozesswissen" ein wichtiger Stellenwert bescheinigt. Ähnlich wie bei Wargitsch [Wargitsch, 1997] werden daher Geschäftsprozesse als wesentlicher Bestandteil des organisatorischen Lernens auf verschiedenen Abstraktionsebenen erachtet. Das System besteht in Form einer Java-Applikation. Eine Anbindung an eine Datenbank ermöglicht eine zentrale Datenhaltung. Eine Zusammenarbeit von IBM und Lotus verspricht eine eventuelle Anbindung von OpenWater an Lotus Notes.

Ziele

- Workflows müssen leicht modifizierbar sein — auch für Endbenutzer.

- Entwickle Workflow-Definitions-Methode für Prozessevolution und Ausnahmenbehandlung ohne Notwendigkeit, formale Modelle zu bearbeiten.
- Deskriptiv statt präskriptiv: "Workflow Support" statt "Workflow-Management". Bei OpenWater entscheidet der Benutzer selbst über den Prozessablauf. Das Routing von Arbeitsanweisungen basiert auf einem "Electronic Circulation Folder" und verläuft konkret per E-Mail-Kommunikation.
- Dieses Routing soll durch Lernalogrithmen automatisiert und verbessert werden. Beispielsweise wird ein bestimmtes Routing vom System vorgeschlagen, falls der Nachfolger von früher bekannt ist.
- "Entdecke" Prozesse aus der Beobachtung von Ausführungen.
- Integriere das organisatorische Lernen aus Prozessausführungen.

Bewertung

Die Konzeption dieses Systems ist sehr stark ad-hoc-orientiert. Die Prozesslogik geht nur unwesentlich über ein vom Benutzer festgelegtes Routing von Prozessen hinaus. Eine ausdrucksmächtigere Steuerungsmöglichkeit des Ablaufs, wie beispielweise die von uns angedeutete Prozesslogik, wäre zu begrüßen. Die Struktur von Prozessen und Daten ist flach realisiert und erlaubt somit weder eine hierarchische Gliederung noch eine Top-Down-Planung.

4.5.5 Plan Management Agent (PMA)

Der Plan Management Agent [Pollack *et al.*, 1999, Tsamardinos *et al.*, 2000] zielt auf die Unterstützung *eines* Benutzers bei der Verwaltung einer potentiell großen und komplexen Menge von Plänen ab. Das System wurde von Pollack und Tsamardinos (University of Pittsburgh), sowie Horty (University of Maryland) vorgestellt. Das System wird in Lisp (für Windows) implementiert, ist bereits lauffähig, befindet sich aber derzeit noch im Entwicklungsstatus.

Ziele

- Einsetzen von KI-Technologie zur Modellierung von "Plänen".
- Reasoning über ausdrucksstarke Pläne.
- Ausdrucksmächtige zeitliche Constraints, die Ausdrucksmöglichkeiten für zeitliche Ungewissheit beinhalten.
- Berechnung von Kosten für (1) die Erstellung eines neuen Planes und (2) das Ausführen eines Planes bezüglich des aktuellen Kontextes.
- Der Benutzer wird über nahende Deadlines informiert. Dabei wird beachtet, inwieweit der Zeitplan des Benutzers in Zukunft gefüllt ist. Im Falle eines zur Zeit unterbesetzten und in Zukunft dicht besetzten Zeitplanes schlägt das System bereits früh Aktionen vor, um die stark ungleichmäßige Arbeitsdichte etwas zu glätten.
- Dem Benutzer werden vom PMA Vorschläge unterbreitet. Die Entscheidung wird letztendlich jedoch stets dem Benutzer überlassen.

- Konflikte zwischen verschiedenen Plänen werden vom System erkannt. Das System schlägt dem Benutzer Lösungsmöglichkeiten zur Behandlung des Konfliktes vor, schafft aber selbstständig keine Abhilfe.
- Der Benutzer kann jederzeit Änderungen an den Plänen vornehmen.
- Bereits eingegebene Pläne oder Teile davon können in einer "Activity Library" zur Verfügung gestellt werden, um daraus neue Pläne zusammenzubauen.

Bewertung

Das System ist als Planungsunterstützung für einen einzelnen Benutzer konzipiert. Eine Kooperation mit anderen Mitarbeitern steht somit nicht im Vordergrund und wird auch nicht weiter beachtet. Gleiches gilt auch für eine Wissensmanagement-Anbindung.

Das System siedelt sich zwischen einfachen Kalender- / Erinnerungs-Tools und Workflow-Systemen an. Es sollen benutzerorientierte Pläne komfortabel und flexibel erstellt werden können. Dabei wird ein großer Schwerpunkt auf das Modellieren des zeitlichen Rahmens einer Aktivität (innerhalb eines Planes) gelegt.

Für eine Evaluierung bezüglich einer Integration im FRODO-Projekt bedeutet das konkret: Die *grundsätzliche* Konzeption von PMA wird bereits von der hier vorgestellten subsumiert. Interessant sind aber die Betrachtung der kleinen Details, wie beispielsweise die Kostenabschätzung für die Durchführung eines Planes. Ferner ist im Ausblick des Dokuments [Pollack *et al.*, 1999] die Rede von einer Erfolgsabschätzung für ein Projekt unter Beachtung des aktuellen Zustands und der anstehenden Deadlines.

4.5.6 *ADEPT*$_{flex}$

Bei *ADEPT*$_{flex}$ [Reichert and Dadam, 1998] handelt es sich um ein prototypisches WfMS, das an der Universität Ulm entwickelt wird. *ADEPT*$_{flex}$ behandelt dynamische Modellierungsänderungen, insbesondere Ausnahmen. Dazu wird in *ADEPT*$_{flex}$ ein formales Workflow-Modell zusammen mit einer Menge von ausgezeichneten, formal korrekten Modifikationsoperatoren (Task einfügen, löschen etc.) vorgestellt. Das Projekt beschäftigt sich zum größten Teil mit der Konsistenzproblematik bei Modellierungsänderungen. Die Verantwortung, die Konsistenz des Workflows bei Modellierungsmodifikationen zu wahren, soll nicht dem Benutzer sondern dem Workflow-System obliegen. Daher werden nur solche Modifikationsoperatorn zugelassen, die im aktuellen Zustand eine formal korrekte Operation gewährleisten. Diese kann ein Benutzer einsetzen, um die Modellierung eines Workflows an neue Gegebenheiten anzupassen.

Ziele

- Die Modellierung eines Workflows wird in einem *formalen* Workflow-Modell ("Workflow-Schema") festgehalten.
- Kleine Menge von Modellierungsoperatoren, deren Anwendung stets die Konsistenz im Workflow wahrt (mathematisch nachgewiesen).

- Die Modellierungsänderungen wirken sich nur auf laufende Workflow-Instanzen aus, nicht auf die zugrunde liegenden Workflow-Modelle, schließen aber strukturelle Änderungen mit ein.
- Ein Workflow enthält eine Menge von "Simple-Tasks", die allerdings nicht (hierarchisch) verfeinert werden können (z.B. durch Verfeinern einer Task in Sub-Tasks).
- Der Kontrollfluss wird explizit mit gerichteten Nachfolger-Kanten modelliert und ist demnach, bis auf Entscheidungen in OR-Joins, statisch.
- Erlaubte Verzweigungen sind AND-Split/AND-Join, OR-Split/OR-Join und AND-Split/OR-Join. Letztere erlaubt gleichzeitiges Beginnen paralleler Handlungsstränge (AND-Split) und später Entscheidung für einen Ast (OR-Join).
- Ausnahmen (Exceptions) werden zur Laufzeit aufgenommen und explizit mit sogenannten *failure-edges* modelliert. Diese Kanten führen zu einem gesondert behandelten Ablaufverhalten (eine Reihe von Aktionen werden zurückgesetzt).
- Datenfluss wird mit read-/write-Links zu *Workflow-globalen* Variablen modelliert.
- Schreibvorgänge, durch write-Links zu einer globalen Variablen, überschreiben diese Variable nicht, sondern legen eine neue Version von ihr an. Dadurch kann erstens ein Rollback auf frühere Versionen von Daten getätigt werden, und zweitens kann gleichzeitig auf verschiedenen Versionen der gleichen Datenquelle gearbeitet werden. Das ist beispielsweise in parallelen Handlungssträngen (AND-Split) vonnöten.

Bewertung

Das $ADEPT_{flex}$ WfMS hat interessante Ansätze zur Wahrung der Konsistenz trotz dynamischer, von Benutzern getätigter, Modellierungsänderungen. Diese ehrgeizige Zielrichtung wird allerdings begleitet von einer relativ einfachen Modellierungsstruktur. So ist zum einen eine Task nicht (hierarchisch) verfeinerbar und andererseits der Kontrollfluss zu statisch. Eine Integration als WfMS-Komponente innerhalb des FRODO-Projektes ist daher nicht zweckmäßig, da in FRODO das Hauptaugenmerk auf der Unterstützung schwach strukturierter Prozesse liegt, welche eine hierarchische Strukturierbarkeit fordert. Ebenso erscheint der statische Kontrollfluss nur wenig adäquat zur Behandlung stark dynamischer Prozessabläufe.

4.5.7 MOBILE

Bei MOBILE [Jablonski, 1994, Horn and Jablonski, 1998] handelt es sich um ein Workflow-Management-System, das auf auf eine Integration kooperativer Elemente in ein WfMS abzielt, um *typische Geschäftsprozesse* in Unternehmen adäquat zu unterstützen.

Ziele

- Integration verschiedener Prozesskategorien
- Reaktion auf kontinuierliche Änderungsanfragen

- Entwicklung neuer Kontrollkonstrukte zur kompakten, adäquaten Beschreibung des Ablaufs und zur Vereinfachung der Modellierung
- Unterstützung strikter Ablauf-Regeln, bei gleichzeitigem Zulassen schwach modellierter Aktivitäten (beide sind zur Erfassung typischer Geschäftsprozesse vonnöten)
- Ein Workflow-Modell unterteilt sich in *functional, behavioral, organizational* und *informational model*, was modulare und aspektorientierte Modellierungen bzw. Erweiterungen erlaubt.
- Sub-Workflows können als "Prozesselemente" in einen Workflow eingebaut werden.
- Workflow-Modelle ("Workflow-Typen") können adaptiert werden, was zu neuen Versionen / Varianten des alten Workflow-Modells führt. Bei der Adaption wird angegeben, inwieweit sich die Änderungen auf (laufende) Workflow-Instanzen auswirken soll.
- Änderungsmanagement in der Prozessverwaltung integriert

Bewertung

Trotz der scheinbar analogen Zielsetzung bezüglich der Unterstützung wissensintensiver Geschäftsprozesse werden Prozesse mit MOBILE zu formal und zu strikt modelliert. Eine Adaption von bestehenden Workflows wird lediglich in Form von Varianten dieser Workflows ermöglicht. (Laufende) Workflow-Instanzen können nicht *direkt* modifiziert werden, sondern nur indirekt über einer Änderung des zugehörigen Workflow-Modells. MOBILE erscheint daher zur Integration in FRODO nicht adäquat.

4.5.8 TeamWare Flow

TeamWare Flow [Group, 1997] ist ein kommerzielles WfMS bestimmt für Gruppenarbeit in einer lernenden Organisation. Es bietet Unterstützung in Planung, Ausführung, Protokollierung und Anpassung von Geschäftsprozessen. TeamWare bietet die Möglichkeit, Geschäftsprozesse mit ihrem Prozessmodellierungswerkzeug ProcessWise Workbench zu modellieren und nach TeamWare Flow zu exportieren. Weiterhin kann das System in Groupwareprodukte wie Lotus Notes oder Microsoft Exchange integriert werden.

Ziele

- Die Modellierung ist Aufgaben-basiert. Sie benutzt Pläne, welche aus einem Netzwerk von Phasen bestehen. Jede Phase besteht aus aus einer oder mehreren Aufgaben oder Fragen als ein spezifischer Schritt im Prozess. Solche Phasen können als Sequenz oder parallel ausgeführt werden und besitzen eine ausgezeichnete Rolle zur Bearbeitung.
- Pläne können Sub-Pläne besitzen, welche wiederum aus einer Menge von Aktivitäten bestehen und einen dedizierten Besitzer haben. Nur er kann dort Änderungen vornehmen. Ein Sub-Plan ist zur Ausführung einer bestimmten Aktivität gedacht. Die letzte Aktivität ist ein Exit-Knoten, der das Ergebnis an die aufrufende Aktivität liefert.
- Aktivitäten sind gedacht als Anfragen von einer Person (dem Planeigner) an eine oder mehrere Personen. Sie besitzen eine Freitext-Beschreibung

- Eine Aktivität besitzt eine oder mehrere Auswahlmöglichkeiten bzw. Aktionen (choices),. welche das Ergebnis einer Aktivität festhalten.
- Zu jeder Aktion kann eine Nachricht vom Benutzer formuliert werden. Diese Nachricht wird im Protokoll festgehalten und kann von anderen Benutzern gelesen werden.
- TeamWare Flow besitzt weiterhin einen Arbeitskorb für die Benutzer, Ereignis-Handling, eine Script-Sprache und automatische Aktivitäten, die auch ereignis- und bedingungsgesteuert sind.
- TeamWare betont die Einfachheit der Prozessmodellierung und dass der Benutzer jederzeit über seine ihm zugewiesene Aktivität entscheiden kann (Delegation, Modifikation, Verfeinerung durch Sub-Plan/Template)
- Es wird erlaubt, nicht vollständig spezifizierte Prozesse zu instanziieren (etwa fehlende Aktivitäts-Beschreibung) und diese zur Laufzeit nachzumodellieren. Damit wird eine Prozessmodellierung zur Laufzeit ermöglicht, in der die Bearbeiter den Prozess erst dann modellieren, wenn der Bedarf auftritt. Die Modellierungsänderungen betreffen nur die aktuelle Instanz. Die Modellierungsschritte werden protokolliert.

TeamWare Dolphin TeamWare Dolphin ist ein kommerzielles ad-hoc Workflow-System. Im Gegensatz zu TeamWare Flow ist Dolphin als "leichtgewichtige" Dokument-Routing und ad-hoc Workflow-Lösung gedacht. Das System soll hier nur der Vollständigkeit halber genannt werden. Dolphins Zielgruppen sind sogenannte Geschäftsnutzer, wie Gruppenleiter oder Manager ohne besondere Entwicklungs- oder Programmierkenntnisse, die einen einfachen und schnellen Zugang zu einer Workflow-Lösung brauchen [WorkflowWorld, 1998].

Ziele

- Einfacher Zugang zur Prozessmodellierung: Benutzer beschreibt Prozess als To-Do-Liste, wobei das System dazu eine graphische Repräsentation aufbaut
- Definition von Workflow-Teilnehmern (auch Gruppen), Prioritäten, Routing Optionen, Parallelität und zugehöriger Dokumente.
- Benutzerinterface zeigt den Prozessablauf mit der aktuellen Aufgabe hervorgehoben. Es erlaubt Zugriff auf zugehörige Dokumente und Steuerung des Prozessablaufs über die vordefinierten Optionen.
- Bei komplexeren Prozessen besteht die Möglichkeit, eine eingeschränkte Version des Workflow-Modellierungstools von TeamWare Flow zu nutzen.
- Dolphin bietet auch Templates für Benutzer an, die öffentlich oder nur lokal verfügbar sein können.
- keine Pre- und Post-Conditions
- Modifikationen zur Laufzeit sind möglich: Hinzufügen oder Ändern von Aktivitäten und Routing-Regeln
- Einbindung in Microsoft Outlook als Client-Oberfläche ist möglich

Bewertung

TeamWare Flow ist ein innovatives Workflow-System, welches Teamarbeit mit Prozessplanung und -ausführung unterstützt. Das System setzt einige der in Frodo anvisierten Ideen um und zeigt damit die Relevanz des Themas schwacher Workflow für den Workflow-Markt. Da es sich um ein kommerzielles Tool handelt, sind die Erweiterungsmöglichkeiten, die in FRODO nötig sind, sehr stark beschränkt.

4.5.9 Zusammenfassung

Die hier vorgestellten Workflow-Systeme realisieren Unterstützungen zu unterschiedlichen Problemstellungen, woraus unterschiedliche Anforderungen und Lösungen erwachsen. Ihnen allen gemeinsam ist jedoch, dass für ihre Problemstellungen das klassische Workflow-Konzept nicht ausreicht. So passt InConcert – mit Wurzeln in den klassischen Workflow-Management-Systemen – sein Workflow-Konzept an. Alle anderen präsentieren grundlegende Alternativen zu dem klassischen Workflow-Konzept. So z.B. sehen alle eine *dynamische* und *flexiblere* Modellierung von Geschäftsprozessen vor, anstelle der klassischen, statischen Modellierung.

Alle Systeme adressieren nur Teilaspekte der Wissensarbeit. Eine umfassende Unterstützung von Wissensarbeit, wie sie Frodo vorsieht, wird in keinem System unternommen. Abbildung 4.9 zeigt nochmals im Überblick die vorgestellten Systeme im Vergleich mit den Zielsetzungen in Frodo.

	Schwerpunkt der Assistenz		Verzahnung von Modellierung und Ausführung	Lazy/Late Modeling	hierarchische Dekomposition	Prozesslogik
	Modellierung	Ausführung				
AWD		X	X	X	X	Action Port Model
CPEF	X		X	X	X	Hierachical Task Networks
InConcert		X	nur bei Aktivitäten		X	klassisches Workflow Prozessmodell
OpenWater		X	ad-hoc Routing			Routing-Primitive
PMA	X		X	X		Constraints
Adept*flex*	X	X	X	X	X	Routing-Primitive
Mobile	X	X	X	X	X	Constraints
TeamWare Flow	X	X	X	X		Pläne/Aktivitäten
Frodo	X	X	X	X	X	Constraints

Abbildung 4.9: Vergleich der Systeme mit den Zielsetzungen in Frodo.

Kapitel 5

Informal–Formal Transitions

5.1 Introduction

One of the core elements of the FRODO framework for distributed organizational memories are intelligent agents that provide users with information that is relevant in their actual work context. In order to determine the relevance of available information objects, the intelligent agents not only have to have access to them, e.g. in form of identifiers, they must also have a representation of the objects' meaning. Of course *every representation* of an information object in a computer system is a kind of *formalization.* However, different formalizations allow for different services that operate on these formalizations, and the power of these services may vary extremely. While, for example, a "bag-of-words"–formalization of a text document can be well suited to determine the similarity of two documents, such a formalization is a poor starting point to answer detailed questions about the story a document contains and its implications. For this task, a logical representation of the story and formal inference rules seem to be more appropriate. On the other hand, it is normally very expensive to obtain such a logical representation. Therefore, a comprehensive OM framework should provide mechanisms for different levels of formalization depending on costs and benefits. In FRODO two types of formalizations are of special importance for text documents: Models from classical information retrieval (e.g. vector representations) and annotations with meta-data (e.g. from a formal ontology). The latter can capture document content as well as its context (e.g. creation time, author, tec.).

How to obtain more formal from less formal representations of texts has extensively been investigated in various fields of research, including natural language processing, cognitive science, information extraction and information retrieval. In [Engels and Bremdal, 2000] an excellent state-of-the-art overview with respect to the different approaches, goals, and achievements is presented.

Figure 5.1 summarizes the classical approach to informal–formal transitions from text. The result of processing on the *pre-text level* (e.g. speech or character recognition) is the text as a sequence of characters. On the *word level* tokens (single words, delimiters etc.) are identified. Furthermore, compounds and word types (part–of–speech) are determined. Stemming, i.e. identifying the basic word form, takes also place at the word level. On the *sentence level* the grammatical structure, i.e. sentence fragments and the particular roles of word, is analyzed. A semantic or knowledge representation is established, e.g. by populating a set of frames. The discourse structure is identified at the *document level* by segmenting the text and relating

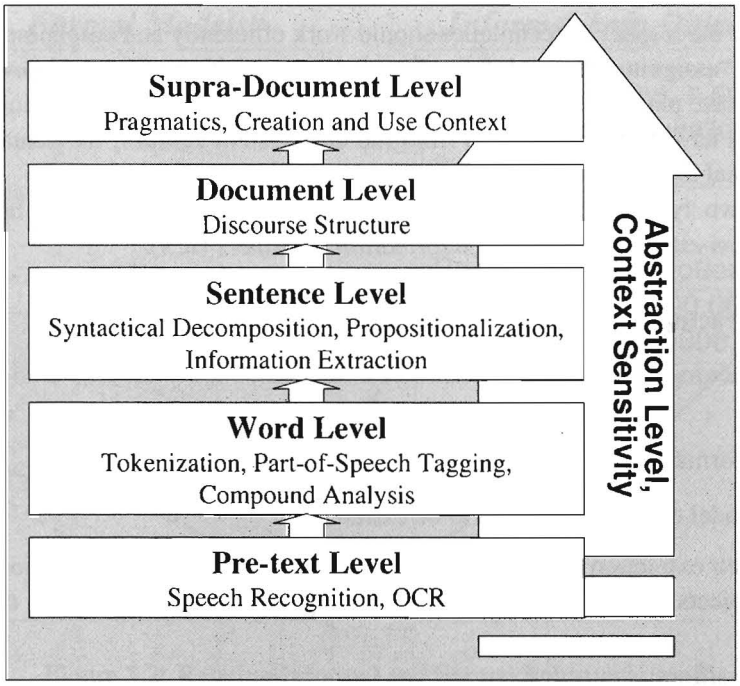

Figure 5.1: Classical Informal–Formal Transitions from Text.

the segments to discourse concepts (e.g. introducing new topics, conclusion, summary). The creation context, the author's intention when writing a document, the disposition of a potential reader and other forms of world knowledge that can be used to "understand" a text form the *supra-document level.*

Often these levels are seen as processing steps where level n is the input for level $n + 1$. However, research in human text understanding indicates that these levels are more interwoven. For instance, expectations and goals on a higher level can have a strong impact on the results of a lower level. The VirtualOffice project [Wenzel and Maus, 2001] is an example where this insight guided the design of a system that deploys informal–formal transitions for a commercial application.

5.2 Agents for Informal–Formal Transitions

In FRODO two types of informal–formal transitions are especially important:

- Assigning information objects to formal knowledge models
- Acquisition of formal knowledge models on the basis of text documents

5.2.1 Establishing Relations between Informal and Formal Knowledge

Relating formal models to informal information objects can happen either *ad hoc* to satisfy a specific information need or *continuously* whenever an information object is added to the system or changed. Due to the fact that in an OM information system this functionality will be

needed very often the respective techniques should work efficiently and autonomously.
In contrast to this "assigning informal–formal" task, the acquisition of formal knowledge models will typically take place asynchronously. Though we want to obtain suggestions when and how these models have to be maintained from the OM system at work, the actual process of changing the formal model will have to be directed by a human editor.

In FRODO two types of specialists for informal–formal transitions will be employed, namely *information extractors* (IEs) and *information classifiers* (ICs).

Information Extractors

Information Extractors work as follows:
They take

- a bag of informal information objects (e.g. text documents) and
- a formal model of the information to be extracted.

On the basis of their extraction expertise they find relations between the formal model and parts of the informal objects. This is a search task in the "bag of informal information objects".

Information Classifiers

Information Classifiers work as follows:
They take

- an informal information object (e.g. a text document or a passage) and
- a formal model of possible target concepts, in the simplest form e.g. a list of categories.

On the basis of their classifying expertise they find relations between the informal information object and parts of the formal model. This is a search task in the formal model.

Figure 5.2 shows the characteristics of the two types of specialists. The left hand side depicts two simple formal knowledge models, a taxonomy of text types and a conceptualization of an advertisement. On the right hand side there are two documents.
An information classifier would start with the documents, trying to find elements in the formal models to which they can be related. As the formal model doesn't contain a concept like "fairy-tale", no link to document D_1 can be established. D_2 on the other hand can be classified as an advertisement text.
An information extractor would work quite complementary. It gets a part of the formal model, e.g. the conceptualization of an advertisement. For this fixed part, the IE tries to find portions of the informal objects, i.e. parts of the texts, which can be related to the elements of an advertisment: "Real Estates Lmt." is a seller, the "luxury villa" the object to be sold, and the price is "1,000,000$" where "1,000,000" is the amount and "$" is the currency.
A few remarks to this simple example:

- Often, categorization is a much simpler task than extraction. The search space "formal models" is normally much smaller and better structured than the search space "text documents". Text documents contain a huge number of atomic elements (tokens), it is not clear how these atoms can be grouped etc. One of the main reasons for the techniques applied on the sentence and document level (cf. figure 5.1) is to better structure this search space.

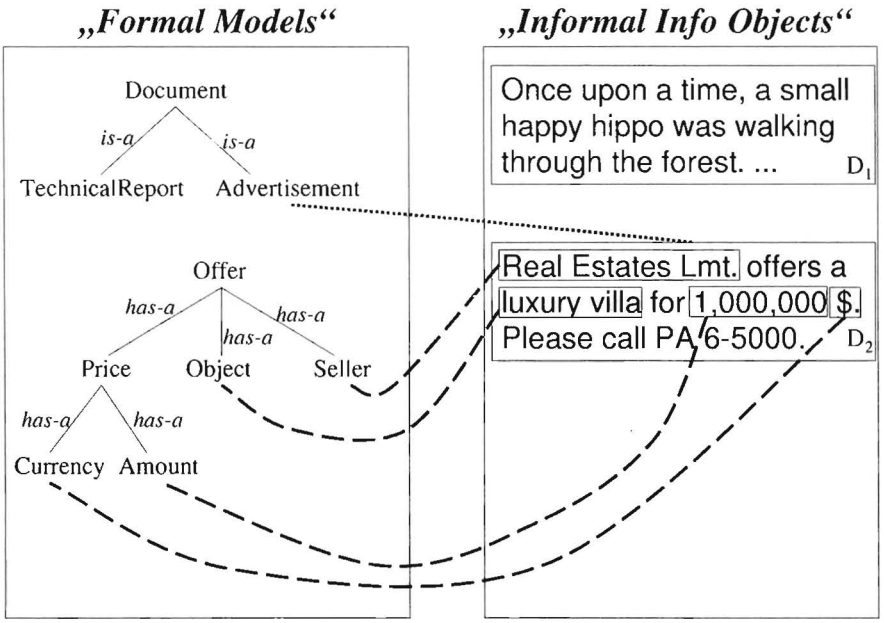

Figure 5.2: Relating Informal and Formal Representations

- One task can be utilized to perform the other: For example, to check if "Real Estates Lmt." should fill the slot "seller" or "object" or "price" a categorization task can be performed. On the other hand, an operationalization of an "Advertisement Document"–classifier could be to check wether a price and an object can be found. Hence it is desireable to allow for a dynamic configuration of IEs and ICs to satisfy an actual information need.

- A richer formal model can greatly help to reduce the search space for information extraction. For example constraints like "sellers can only be companies from the yellow pages" or "an amount can only be an integer" help a lot in extracting the information desired (compared to "a seller is a sequence of string", "an amount is a number").

- Both extraction and classification can only utilize the formal models given, but can not extend them. For example, document D_1 remains completely uninterpreted due to the lack of a fairy-tale concept. Parts of D_2 remain uninterpreted in the extraction task due to the lack of a concept "phone" though it might be useful. However, an IC could note if there are lots of documents that can't be classified with the actual formal model and therefore trigger a maintenance cycle.

These rather abstract characterizations of information extraction and classification fit well into the FRODO framework. Here the formal models can be delivered by the domain ontology agents (cf. section 2.2). So IEs and ICs take at least the role of *passive users* of the relevant ontologies. If a feedback mechanism (see above) is integrated, IEs and ICs would be *active users*.

Typical customers of IEs and ICs in FRODO are — besides the interplay mentioned above — the information agents that realize the distributed inferences (cf. section 2.3). They can

combine classification and extraction results together with formal reasoning steps to satisfy information needs that are specified by the task at hand and its user.
IEs as well as ICs maintain knowledge about their own capabilities so that they can deny requests they can not serve.

5.2.2 Acquisition of Formal Knowledge Models

In FRODO formal conceptualizations managed by ontology agents play a central role. The acquisition of ontologies from informal information objects (e.g. from texts) is a research topic with growing interest in several communities (NLP, ML, KR, KM). Typically, a combination of NLP, information extraction, and statistical techniques is utilized to find concepts and relations (or to support the user in finding them). However, the question why the extracted information should be *ontological in nature* is hardly tackled by now. With the concept of *ontological societies* in an OM we hope to lay the basis for a proper treatment of this question.

Kapitel 6

Agentenplattformen

Die Beschreibung des OM-Frameworks in den vorangegangenen Kapiteln hat gezeigt, dass die Verteiltheit von Daten und dessen Nutzern sowie die Bildung sozialer Strukturen zur Lösung von Problemen in vielen Stellen des FRODO-Szenarios, den Einsatz des Agentenparadigmas sinnvoll macht.

Ein System, in dem Agenten existieren und interagieren sollen, muss ein Grundmaß an Infrastruktur bereitstellen. Der Lebenszyklus eines Agenten bedarf einer entsprechenden Verwaltung. Agenten müssen erzeugt, angemeldet, deaktiviert und wieder abgemeldet werden sobald sie nicht mehr benötigt werden. Es muss Verzeichnisse geben, in denen Agenten ihre Ansprechpartner mit den gesuchten Kompetenzen nachschlagen können, um eigene Aufgaben erfüllen zu können. Agenten benötigen eine entsprechende Kommunikationsinfrastruktur, um mit anderen Agenten in Kontakt zu treten und Informationen auszutauschen. Aus dem Blickwinkel der Agentenentwicklung und der wissenschaftlichen Analyse der Agenteninteraktion ist die Visualisierung und Dokumentation der Interaktion (z.B. in mitgeführten Logfiles) notwendig.

Idealerweise wird solch eine Infrastruktur durch eine bereits existierende Agentenplattform bereitgestellt. So gesparte Ressourcen für die Neuentwicklung einer eigenen mit hoher Wahrscheinlichkeit proprietären Ablaufumgebung können in die wissenschaftlichen Kernprobleme investiert werden; bereits für die Plattform entwickelte Hilfsprogramme, wie z.B. Visualisierungstools, können direkt verwendet oder auf die eigenen Bedürfnisse angepaßt werden.

Voraussetzung für die Verwendung einer Agentenplattform ist allerdings ihre Eignung für das oben präsentierte FRODO-Szenario. Zu diesem Zweck werden im Anschluss zunächst Anforderungen an eine zugrundeliegende Agentenplattform vorgestellt, die aus der oben vorgestellten Präsentation des angestrebten OM-Frameworks abgeleitet wurden. Explizit wird in diesem Zusammenhang auch auf Gesichtspunkte verwiesen, die in der Anforderungsanalyse keine Berücksichtigung finden. Es wird dargelegt, welche Eigenschaften FIPA-konforme Agentenplattformen aufweisen, die sie einsetzbar im FRODO-Kontext machen und warum die Betrachtung auf diese standardisierten Plattformen beschränkt wurde. Der Vorstellung von vier FIPA-konformen Agentenplattformen und der abschließenden Auswahl einer der vier Kandidaten, geht zunächst noch eine Vervollständigung der Vergleichskriterien voraus. Denn nicht nur wissenschaftliche Aspekte spielen bei der Suche einer geeigneten Agentenplattform eine Rolle, sondern auch software-technische und weitere nicht-wissenschaftliche Anforderungen – hier ist z.B. das Angebot an Visualisierungshilfsmitteln zu nennen, das für eine spezielle Plattform zur Verfügung steht.

6.1 Anforderungen aus dem OM-Framework

Die in den vorangegangenen Kapiteln erfolgten Ausführungen über das OM-Framework lassen eine Reihe von Bedingungen ableiten, die an eine geeignete Ablaufumgebung zu stellen sind. Neben Punkten mit allgemeinerem Charakter, z.B. im Hinblick auf die Agentenverwaltung, lassen die Beschreibungen auch schon Bereiche erkennen, die zunächst einer genaueren wissenschaftlicheren Betrachtung bedürfen und für die eine unterstützende Agentenplattform an diesen Stellen dementsprechende Freiräume zur Verfügung stellen muss (beispielhaft seien hier die Interaktionsprotokolle erwähnt, die im Bereich des Ontologiemanagements benötigt werden. Eine Plattform, die nur die Verwendung von fest vordefinierten Protokollen erlaubt, wäre im Rahmen von FRODO eher ungeeignet). Besonders erwähnt sei hier noch einmal, dass alle hier vorgestellten Forderungen noch keine Aussagen über deren konkrete Realisierung enthalten. Die letztendliche Umsetzung wird von der gewählten Agentenplattform und den von ihr zur Verfügung gestellten Hilfsmitteln und Programmierschnittstellen bestimmt.

6.1.1 Agentenverwaltung

Agentenerzeugung: Starten von Agenten muss auf Betreiben von anderen Agenten des Frameworks möglich sein. Man betrachte z.B. den Model-Manager (vgl. Abschnitt 4.4.1 auf Seite 55) welcher neue Task-Agenten erzeugt und sie mit der Durchführung des zugeordneten Tasks beauftragt.

Dienstverzeichnisse (yellow pages): Das Framework muss eine Reihe von Verzeichnisdiensten bieten, über die Agenten andere Teilnehmer mit benötigter Kompetenz ermitteln. Innerhalb der sozialen Strukturen des OM-Managements (Abschnitt 2.2.2) beispielsweise müssen Editoren ihre Verhandlungspartner kennen, mit denen sie sich über ein Update zu einigen haben. Im Falle von Änderungen müssen aktive Ontologie-Benutzer informiert werden. Passive Benutzer müssen ihre Ansprechpartner für ihre Anfragen finden. Je nach eingenommener Rolle steht einem Teilnehmer somit eine eigene Sicht auf seine Kommunikationspartner zur Verfügung, die zur Erfüllung seiner eigener Ziele herangezogen werden können. Diese rollenabhängige Sicht auf Dienstanbieter muss demnach auch in einem zur Verfügung gestellten Yellow-Pages-Dienst Berücksichtigung finden.

Die starke Verteiltheit des FRODO-Szenarios (vgl. Abschnitte 2.2 und 4.4.1) sowie eine potentiell hohe Anzahl von darin existierenden, hierarchisch organisierten Agenten erfordert auch von den Yellow-Page-Diensten entsprechend strukturiert werden zu können.

6.1.2 Agentenkommunikation

Nachrichteninhalt: Verschiedenste Informationen werden im gesamten Framework transportiert – Faktenwissen, Kontextangaben, User-Profile, etc. Da erst eine genauere Untersuchung im Verlaufe des Projekts klären kann, ob diese einheitlich durch eine einzige Nachrichten-Inhaltssprache (message content language) beschrieben werden kann, sollte die Nachrichteninfrastruktur einer unterstützenden Agentenplattform eine flexible Einbindung zusätzlicher Inhaltssprachen erlauben.

Elementare Sprechakte und Interaktionsprotokolle: Verteilte Inferenzen, das Ontologiemanagement und die Durchführung von Workflow-Tasks erfordern eine reichhaltige

Kommunikation aller Teilnehmer. Die dabei zum Tragen kommenden Sprechakte und Interaktionsprotokolle sollten in die Ablaufumgebung, welche durch eine Agentenplattform bereitgestellt wird, integrierbar sein. Idealerweise bietet eine Plattform bereits eine sinnvolle Auswahl solcher Sprechakte und Interaktionsprotokolle an.

Integration weiterer Datenquellen: Inzwischen weit verbreitete HTML-Interfaces für Datenbanken und weitere Informationsquellen wie z.B. Suchmaschienen sollen nutzbar sein. Das heißt, Agenten sollten, sei es durch die Kommunikationsmechanismen der Agentenplattform oder durch agentenverwaltete Kommunikation, mit der Außenwelt in Kontakt treten können und durch eine Agentenplattform in dieser Hinsicht keinen Restriktionen unterworfen sein.

6.1.3 Agentenverhalten

Dynamische Rollenwechsel: Die verschiedenen Rollen, die ein Agent bei der Nutzung und Pflege von Ontologien einnehmen kann, implizieren auch wechselnde Berechtigungen bei der Agenteninteraktion. Passive Nutzer dürfen Verbesserungen vorschlagen aber keine Änderungen einpflegen usw. In diesem Zusammenhang kann ein Agent bezüglich verschiedenen Ontologien auch gleichzeitig unterschiedliche Rollen innehaben. Wünschenswert wäre also eine Modellierungsmöglichkeit der Interagentenkommunikation, die objektorientiert die einzelnen Rollen beschreiben kann.

6.1.4 Benutzerinteraktion

Graphische User-Interfaces: User-Agenten (siehe Abschnitt 4.4.1), die u.a. die Aufgabe haben, den Benutzern in einem OM-unterstützten Arbeitsumfeld nützliche Informationen zu präsentieren, lassen eine entsprechende Unterstützung durch die Agentenplattform sinnvoll erscheinen.

6.1.5 Nicht relevante Gesichtspunkte

Die folgende Auflistung enthält eine Reihe von Anforderungen, die **nicht** für das FRODO-Szenario relevant sind und die deshalb bei der weiteren Anforderungsanalyse keine weitere Erwähnung mehr finden:

Robustheit: Bei allen Betrachtungen insbesondere den Kommunikationsprotokollen wird die Verfügbarkeit aller Framework-Komponenten voraus gesetzt. Unvorhergesehene Ausfälle z.B. durch Systemabstürze und eventuell dadurch verursachte Inkonsistenzen werden im FRODO-Szenario nicht berücksichtigt.

Sicherheit: Es gibt keine Teilnehmer innerhalb des Frameworks mit dem Ziel, mutwillig Schaden anzurichten. Es gibt z.B. keine Angriffe, die versuchen bestimmte Systemdienste außer Kraft zu setzen (denial of service attacks) oder Daten mutwillig zu fälschen oder zu löschen.

Mobilität: Alle im Framework existierenden Agenten bleiben für ihre gesamte Lebenszeit an ihren „Geburtsort" gebunden. Es gibt keine Anforderungen, die den Einsatz von mobilen Agenten, d.h. von Agenten, die im Laufe ihrer Lebenszeit ihren Aufenthaltsort (Rechner) wechseln können, erfordern.

Performanz: Im gesamten Framework werden keine formalen Garantien bezüglich Auftragsbearbeitungszeiten gegeben/benötigt.

Quality of Service (QoS): Nachrichten zwischen Agenten der Plattform werden mit dem QoS des zugrundeliegenden Nachrichtentransportprotokolls übermittelt. Es gibt keine Anforderung im FRODO-Szenario den QoS in irgendeiner Art und Weise zu verschärfen bzw. abzuschwächen.

6.2 FIPA-Standard

Die „Foundation for Intelligent Physical Agents" (FIPA) hat eine Spezifikation für Softwareagenten erarbeitet, um die Zusammenarbeit von Agenten verschiedener Plattformen zu ermöglichen . Dazu enthält sie Festlegungen für eine abstrakte Plattformarchitektur [FIPA, 2000a], die Agentenkommunikation und den dazugehörigen Nachrichtentransport umfasst (Nachrichten und Interaktionsprotokolle, Transportprotokolle). Außerdem enthält sie Spezifikationen zum Agentenmanagement.

Obwohl diese Spezifikationen darauf abzielen, Interoperabilität zwischen Agenten verschiedener Plattformen zu ermöglichen, ist der Hauptgrund eine FIPA-konforme Agentenplattform als Basis für das FRODO-Framework zu verwenden der, eine Infrastruktur zur Verfügung zu haben, deren Bestandteile als wohlstrukturiert und wohlspezifiziert angesehen werden können. Im Gegensatz zu proprietären Lösungen, die unter Umständen eine Überspezialisierung und eine damit verbundene fehlende Anpassungsfähigkeit aufweisen.

Im Folgenden werden die wesentlichen für das FRODO-Framework relevanten Bausteine der abstrakten FIPA-Agentenplattform-Architektur vorgestellt und aufgezeigt, dass diese Bausteine eine sinnvolle Grundlage für das FRODO-Framework bilden.

6.2.1 Abstrakte Plattformarchitektur

Wie schon oben erwähnt, zielt die FIPA-Spezifikation auf die Interoperabilität von Agenten ab. Die Architektur umfaßt die Bereiche Agentenmanagement und Plattformdienste sowie die Agentenkommunikation:

Plattformdienste

Agent Management System (AMS): Ein White-Page-Dienst; alle Agenten der Plattform sind hier unter einer eindeutigen Agenten-ID (AID) registriert. Hier sind die Adressen der Agenten gespeichert unter denen sie erreichbar sind. Das AMS verwaltet die Life-Cicles der angemeldeten Agenten.

Directory Facilitator (DF): Ein Yellow-Pages-Dienst, bei dem alle Agenten ihre Kompetenzen eintragen können. Der Dienst bietet entsprechend Suchmöglichkeiten, um geeignete Agenten (mit passender Kompetenz) zu finden. Eine hierarchische Verknüpfung, die mehrere DFs zu sog. „DF federations" zusammenschließt, ist vorgesehen.

Message Transport Service (MTS): Transportdienst für die Versendung von Nachrichten zwischen Agenten. Der MTS sorgt für den physikalischen Transport und die dazu notwendigen Weiterleitungen einer Nachricht, ggf. über weitere Zwischenstationen. Dies

umfasst unter Umständen auch einen Transport über Plattformgrenzen hinweg. Der Zugriff auf den Transportservice wird über einen „Agent Communication Channel" (ACC) bereitgestellt. Der MTS ermöglicht eine Verteilung der Agentenplattform über mehrere Rechner hinweg.

Management-Ontologien: Der FIPA-Standard definiert eine Reihe von Ontologien, die bei der Nutzung der Plattformdienste Verwendung finden. Dazu zählt u.a. das An- und Abmelden eines Agenten beim AMS, das Suchen nach Agenten mit geeigneter Kompetenz beim DF oder das Versenden von Nachrichten über das MTS.

Agentenkommunikation

Agentensprache (FIPA ACL): Der FIPA-Standard beschreibt die abstrakte Struktur einer Agentennachricht und die notwendigen Informationen, die sie mindestens enthalten muss, unabhängig von der vorliegenden Repräsentation [FIPA, 2000b]. FIPA ACL-Nachrichten bestehen zunächst aus zwei logischen Teilen: den Versendeinformationen (message envelope) und der eigentlichen Nachricht (Nachrichteninhalt, sowie Angaben über dessen Repräsentation und darin verwendete Ontologien)

Nachrichteninhalt Der Inhalt einer einer FIPA ACL-Nachricht ist kann in unterschiedlichen Sprachen verfasst sein. Angaben über die im konkreten Fall verwendete Sprache ist in der Nachricht selbst nocheinmal explizit angegeben. Die Kommunikationspartner erkennen alleine an dieser Angabe die verwendete Inhaltssprache. Prinzipiell ist die Sprache frei wählbar. Für die Kommunikation beispielsweise mit den Service(agenten) der Plattform, müssen jedoch in der FIPA-Spezifikation enthaltene Sprachen, z.B. SL, gewählt werden. Siehe auch [FIPA, 2000e].

Elementare Sprechakte: In der „FIPA Communicative Act Library" (CAL) werden ein Reihe von Sprechakten (communicative acts) definiert. Jede FIPA ACL-Nachricht muss einem dieser Sprechakte zugeordnet sein. Eine vollständige Liste findet sich in [FIPA, 2000d]. Das mit den Sprechakten verknüpfte Modell macht die formale Beschreibung der Sprechakt-Semantik möglich und kann als Ausgangspunkt für weitere formale Betrachtungen dienen.

Interaktionsprotokolle: Aufbauend auf die elementaren Sprechakte, definiert der FIPA-Standard komplexere Interaktionsprotokolle (IPs) [FIPA, 2000f], die die folgenden Eigenschaften aufweisen: zwei oder mehr Kommunikationspartner sind daran beteiligt, jeder Kommunikationspartner nimmt im Rahmen der Interaktion eine bestimmte Rolle ein, z.B. Initiator. Eventuell kann der Agent auch in verschiedenen Kontexten gleichzeitig mehrere dieser Rollen innehaben (vgl. OM-Agenten-Rollen in Abschnitt 2.2). IPs selbst können ineinander verschachtelt (komplex verknüpft) auftreten.

Nachrichtentransport

[FIPA, 2000c] enthält Festlegungen zum eigentlichen Nachrichentransport. Hier sind zu erwähnen: die Repräsentation der gesamten FIPA-ACL-Nachricht und Transportprotokolle, die bei der Übertragung einer Nachricht Verwendung finden.

Transportprotokolle Der eigentliche physikalische Transport wird über ein standard Transportprotokoll abgewickelt. Derzeit definiert sind beispielsweise IIOP, HTTP oder WAP.

Nachrichtenrepräsentationen Nachrichten selbst können auf unterschiedliche Weisen repräsentiert sein. Derzeit spezifiziert sind eine String-Repräsentation, eine Repräsentation in XML und eine platzsparende sog. bit-efficient-Repräsentation.

6.3 Software-technische Anforderungen

Die vorgestellte abstrakte FIPA-Plattformarchitektur beschreibt die logischen Bestandteile der Plattform. Sie macht allerdings keinerlei Aussagen über eine konkrete Realisierung derselben. Konkrete Plattformumsetzungen weisen daher immer noch zum Teil signifikante Unterschiede im Plattformdesign auf. Diese beeinflussen insbesondere die angebotenen APIs einer Plattform, die u.U. den Fokus auf bestimmte Eigenschaften wie Mobilität oder Performanz setzt.

Aus diesem Blickwinkel heraus ergeben sich zusätzliche Anforderungen, die eine zu verwendende Agentenplattform erfüllen sollte:

1. Unterstützung des Anwendungsprogrammierers bei der Kodierung von Agenten, Erweiterungen der Plattformfunktionalität und Integration neuer Dienste durch geeignete APIs.

2. Aktivitäten müssen überwacht und für eine spätere Analyse protokolliert werden können.

3. Konfigurationen sollten automatisiert, d.h. ohne manuelle Eingriffe, eingerichtet werden können.

4. Problemlose Einbettung in die potentiell heterogene Betriebssystem-Umgebung soll möglich sein.

5. Nutzung neuster Softwaretechnologie in den Bereichen XML und RDF sollte möglichst direkt einsetzbar sein.

zu Punkt 1: Die bereitgestellte API der Agentenplattform muss bestimmten Qualitätskriterien genügen. Die Systemarchitektur und die darin enthaltenen Klassen werden informell danach bewertet, wie schlüssig sich ihre Strukturierung darstellt. Wie wird Funktionalität angeboten? Welcher Wert wird auf Austauschbarkeit und Wiederverwendbarkeit von zentralen Modulen gelegt?

zu Punkt 2: Die Agentenplattform sollte idealerweise graphische Tools zur Verfügung stellen, mit deren Hilfe eine Protokollierung der Plattformkommunikation möglich wird.

zu Punkt 3: Analysen bestimmter Szenarien und Systemtests erfordern das Speichern und (Re)Aktivieren verschiedener Teilnehmerkonstellationen. Eine Agentplattform sollte daher auch Unterstützung im Bereich automatisierter, d.h. nicht manueller, Plattformkonfiguration bieten.

zu Punkt 4: Da durch die Verteiltheit im betrachteten FRODO-Szenario auch potentiell mehrere verschiedene Betriebsysteme Verwendung finden, sollte eine Agentenplattform auf allen gängigen Betriebssystemen (Windows, Solaris, Linux, Mac) lauffähig sein.

6.4 Analyse der FIPA-Plattformen

Aus den vorangegangenen Kapiteln haben sich nun folgende Anforderungspunkte an eine FRODO-geeignete Agentenplattform ergeben:

[VERW1]	Starten und Beenden von Agenten durch andere Agenten ist möglich.
[VERW2]	Es stehen Dienstverzeichnisse mit folgenden Eigenschaften zur Verfügung: - verteilt - hierarchisch strukturierbar - benutzerrollen-unterscheidend
[KOMM1]	Neue Nachrichteninhaltssprachen sind leicht integrierbar.
[KOMM2]	Einführung neuer elementarer Sprechakte ist möglich.
[KOMM3]	FIPA-Interaktionsprotokolle sind verfügbar. Die Integration neuer IPs ist möglich.
[VERH1]	Unterstützt wird eine objektorientierte Rollenmodellierung.
[GUI]	Unterstützung der Agent-GUI-Anbindung.
[TECH1]	Die API Qualität unterstützt eine Agentenentwicklung.
[TECH2]	Visualisierungs- u. Protokollierungstools sind vorhanden.
[TECH3]	Automatisierte Plattformkonfigurierung wird angeboten.
[TECH4]	Betriebssystemunabhängigkeit ist gegeben.
[TECH5]	Aktuelle XML und RDF Technologie ist integrierbar.

Zu den Anforderungen noch ein paar Bemerkungen: die Punkte [TECH4] und [TECH5] sind erfüllt, wenn eine Plattform vollständig in JAVA implementiert ist (alle nachfolgend betrachteten Plattformen erfüllen dieses Kriterium). Beide Punkte tauchen deshalb in den Bewertungstabellen nicht mehr auf.

Anforderung [VERW1] wird bei allen Plattformen durch die Bereitstellung einer API mit entsprechender Funktionalität garantiert und kann somit als erfüllt angesehen werden.

Stellt sich bei der Untersuchung der Anforderungen [KOMM1-3] bzw. [VERW2] schon eine fehlende Eignung heraus, entfällt eine genauere Betrachtung weiterer Punktes softwaretechnischen Anforderungen, beispielsweise der API-Qualität [TECH1].

6.4.1 FIPA-OS 1.3.3

FIPA-OS ist ein experimentelles Agentenframework der kanadischen Nortel Network Corporation[1] und wurde in den britischen Harlow Laboratories entwickelt. Es unterstützt die Entwicklung von Multi-Agentensystemen durch die Bereitstellung FIPA-konformer Dienste (AMS, DF, etc.).

Anforderung	Bewertung
[VERW2]	Keine federated DFs möglich.
[KOMM1]	Keine Factory/Interface-Unterstützung für die Verwendung eigener Nachrichteninhaltssprachen. Nutzung neuer Inhaltssprachen nur über Umwege.
[KOMM2+3]	Komplette FIPA-Spezifikation für die Sprechakte (communicative acts) unterstützt. Neue Interaktionsprotokolle dynamisch ladbar.
[VERH1]	Agentenverhalten wird in `Task`-Objekten modelliert.
[VERH2]	JESS-Anbindung wird unterstützt.
[GUI]	Synchronisation von Graphik-Events und Agenten-Tasks liegen in der Hand des Agenten-Entwicklers.
[TECH1]	Schwächen in der internen Plattformstrukturierung (z.B. Nachrichtenklasse `ACL` im Ontologie-Package). Eingeschränkte Unterstützung von Erweiterbarkeit.
[TECH2]	GUI zur Plattform-Konfigurierung.
[TECH3]	Plattform- und Agentenprofile werden in XML-Dateien verwaltet.

6.4.2 JADE 2.1 - Java Agent Development Environment

JADE ist eine Agentenentwicklungs-Framework der italienischen CSELT[2]. Die Plattform stellt eine flexible, FIPA-kompatible Basis zur Entwicklung von Multi-Agentensystemen dar. Insbesondere graphische Tools für Debugging und das Management der Agentenplattform sind enthalten.

[1] http://www.nortelnetworks.com

[2] Centro Studi e Laboratory Telecomunicazioni; http://www.cselt.it

Anforderung	Bewertung
[VERW2]	Eine hierarchische und verteilte Strukturierung des Directory Facilitators (DF) ist möglich. Es existiert ein GUI zur Visualisierung und Modifikation der DF-Einträge.
[KOMM1]	Beliebige Inhaltssprachen und die zugehörigen Verarbeitungsroutinen können unter einer eigenen ID bei der Plattform angemeldet werden.
[KOMM2]	Keine direkte Unterstützung durch eine API. Es stehen die Begriffe der FIPA-Ontologie für Sprechakte zur Verfügung.
[KOMM3]	Interfaces zur Implementierung neuer Interaktionsprotokolle existieren. Implementierungen liegen bereits für die FIPA-Protokolle `Request`, und `ContractNet` vor.
[VERH1]	Agentenverhalten wird in `Behaviour`-Objekten gekapselt. Nebenläufige Ausführung verschiedener `Behaviours` wird transparent vom System koordiniert.
[VERH2]	Keine besonderen Hilfsmittel. Eine Anbindung muss bei der Agentenrealisierung geschehen. Beispiel für eine JESS[3]-Anbindung existiert.
[GUI]	Spezielle Klassen erleichtern die Anbindung von GUI-Events an das in `Behaviour`-Klassen modellierte Agentenverhalten.
[TECH1]	Klarer Aufbau, für alle wichtigen Bereiche existieren entsprechende Basisklassen und Interfaces als Grundlage für die Agentenrealisierung.
[TECH2]	**Remote Management Agent (RMA):** Management-Tool für das AMS. **DF GUI**: Visualisiert die Yellow-Pages (DF). **Dummy Agent**: Erlaubt das Versenden und Empfangen von Nachrichten, die auch geladen bzw. gespeichert werden können. **Sniffer Agent**: Visualisiert die Kommunikation zwischen Agenten(gruppen) der Plattform
[TECH3]	Agentenkonfiguration weder gespeichert noch geladen werden. Eigene Startup-Skripte müssen zu diesem Zweck eingesetzt werden.

6.4.3 Grasshopper 2.1

Grasshopper ist ein kommerzielles System, gedacht für die Anforderungen der Telekommunikation (z.B. für UMTS-Anwendungen). Die Plattform wird entwickelt von der berliner IKV++ GmbH[4]. Sie legt besondere Schwerpunkte in den Bereichen Mobilität, Zuverlässigkeit und Performanz. Ein separater FIPA-Aufsatz stellt die FIPA-Kompatibilität her.

[4] http://www.ikv.de

Anforderung	Bewertung
[VERW2]	Die Funktionalität eines DFs wird auf die Directory Dienste des Basissystem abgebildet. Eine hierarchische Strukturierung der Yellow Pages ist möglich.
[KOMM1]	Verwendung unterschiedlicher Nachrichteninhaltssprachen ist nur durch Modifikation von Klassen der Plattform möglich.
[KOMM2+3]	Die Plattform bietet keine Unterstützung bei der Implementierung und Verwendung von Interaktionsprotokollen. Unterstützung der Sprechakte beschränkt sich auf einen bereitgestellten Nachrichtenparser. Die Verwaltung von Interaktionsprotokollen muss komplett vom Agent übernommen werden.

6.4.4 ZEUS - An Agent Building Tool-Kit

Die ZEUS-Plattform der britischen Telekom[5] zielt darauf, von den Implementierungsdetails eines verteilten Multi-Agentenszenarios zu abstrahieren und den Fokus auf die konzeptionellen Probleme der jeweiligen Domäne zu richten. Unterstützt wird das Design von (frameorientierten) Ontologien, regelbasierte Taskdefinitionen der eingesetzten Agenten, sowie eine Organisation der Agenten (Zusammenarbeit, Rollendefinition). Die ablauffähigen Agenten des System werden aus diesen Angaben automatisch generiert.

Anforderung	Bewertung
[KOMM1-3]	Durch die Fokussierung auf das Konfigurieren eines Agenten über eine entsprechende Oberfläche und das letztendliche Erzeugen durch einen Kodegenerator, ist das Gesamtsystem ohne Änderung von Systemklassen nicht geeignet anzupassen.

6.4.5 Fazit

Die vorgestellten Systeme ZEUS und Grasshopper sind nicht für die in diesem Projekt benötigte Flexibilität konzipiert worden. Entsprechend wenig Freiräume lassen die bereitgestellten Programmierschnittstellen bei der Agentenentwicklung.

Lediglich FIPA-OS und JADE können in dieser Hinsicht den Anforderungen genügen. Der Vergleich der beiden Plattformen zeigt eine insgesamt besserere Eignung des JADE-Systems. Mit seinem insgesamt klareren internen Aufbau und der besseren Erweiterungsfähigkeit läßt es den größten Spielraum bei den im FRODO-Szenario zu entwickelnden Agenten. Die verfügbaren graphischen Tools, z.B. für die Protokollierung und Speicherung von Nachrichten, untersteichen die Führungsrolle weiter.

[5] http://www.labs.bt.com

Kapitel 7

Methodologie

FRODO fokussiert auf das Wissensmanagement im betrieblichen Umfeld. Dabei sind die dort vorhandenen Geschäftsprozesse von besonderem Interesse: einerseits als Ort des WM-Bedarfs und andererseits als Informationsquelle für das WM. Daher wird die zu entwickelnde Methodik den Bereich Geschäftsprozess-orientiertes Wissensmanagement abdecken. Die prozessorientierte Betrachtung von Wissensmanagement ist ein relativ neues Gebiet. Durchgängige Methoden werden zwar verstärkt gefordert und angerissen, bislang ist aber keine solche präsentiert worden, obwohl ein Bedarf existiert [Maier and Remus, 1999].

Im folgenden soll anhand einer Einteilung des Gebietes in verschiedene Phasen der Systemgestaltung existierende Ansätze diskutiert und interessante Teilaspekte für ein Vorgehensmodell hervorgehoben werden. Diese Phasen sind:

1. **Systemdesign:** Sowohl WM- als auch BPM-Projekte erfordern eine umfassende Planungs-, Analyse- und Einführungsphase. Es stellt sich die Frage, inwiefern hierfür abgestimmte, integrierte Methoden entwickelt werden können.

2. **Systemnutzung:** Zur ''Laufzeit'', also im operativen Betrieb eines BPM-Systems (konkret bedeutet dies dann, eines Workflow-Tools) und eines WM-Systems, können diese interoperieren. Diese führt zu beiderseitigen Nutzeffekten und neuen Systemdienstleistungen.

3. **Systemevolution:** Im Sinne der kontinuierlichen Prozessverbesserung sollte während der Systemlaufzeit laufend systematisch Verbesserungspotential identifiziert und genutzt werden. Diese ''Metaebene'' zum eigentlichen Operativsystem kann i.w. als WM-Ebene betrachtet werden.

Diese drei Phasen, bzw. Ebenen der Systembetrachtung werden im folgenden jeweils in einem eigenen Kapitel diskutiert.

7.1 Systemdesign: Wissensorientierte Organisationsanalyse und Strategisches Wissensmanagement

Wissensmanagement als Management-Aktivität mit weitreichenden Implikationen für die Arbeitsabläufe einer Organisation, muss sich wie jede andere Management-Aktivität an den globalen Zielsetzungen der Unternehmung ausrichten, in einer initialen Analysephase Schwach-

stellen analysieren und Ziele setzen, und in einer Konzeptionsphase den Fokus der zuerst anzugehenden WM-Maßnahmen setzen. Praktisch alle strukturierten Ansätze für die Durchführung von WM-Initiativen sind im Kern als Top-Down-Ansätze konzipiert und sehen solche Schritte vor.

Bei **Karl Wiig** beispielsweise wird als eines von acht wichtigen Themengebieten bei der Durchführung eines WM-Programms die Fokussierung auf klar definierte WM-Projekte genannt:

"Pursue **targeted KM focus** *determined from the knowledge landscape mapping insights and other opportunities and based on KM priorities that align with enterprise objectives: Undertake "bite-sized" and sharply targeted KM initiatives with clear benefit expectations that cumulatively build to implement the KM vision."* [Wiig, 1998].

Dabei wird jedoch weder im Detail spezifiziert, wie man denn die Wissenslandschaft effektiv kartographiert, noch wie man zu Entscheidungen über sinnvolle Schwerpunktsetzungen kommt.

In der **Know-Net Methode** [Apostolou *et al.*, 2000, Know-Net Consortium, 2000] bietet sich ein ähnliches Bild, wenn auch manche Details besser beschrieben sind.

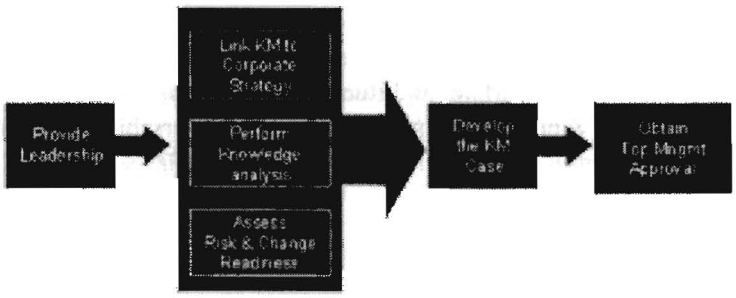

Abbildung 7.1: Stufe I der Know-Net Wissensmanagement-Methode.

So wird beispielsweise in Schritt 2 der Stufe I *"Strategic Planning for Knowledge Management"* immerhin vage eine Vorgehensweise angegeben, welche den ersten Schritten im Geschäftsprozess-Reengineering naturgemäß nicht unähnlich ist: Hier wird zumindest ange-

Step 2: Link KM strategy with corporate strategy	Organize workshop / conduct interviews Analysis o identify vision, strategy, and objectives o identify critical success factors o link strategy to critical success factors, improvement needs, key people and pocesses Select the key business area and process of focus

Abbildung 7.2: Schritt 2 von Know-Net, Stufe I.

deutet, dass eine Analyse top-down von der organisatorischen Vision über die kritischen Erfolgsfaktoren zu den dafür relevanten Prozessen und Knowledge Assets erfolgen kann. Auch bei der Umsetzung konkreter WM-Aktivitäten wird das in Geschäftsprozessen anfallende / benutzte Wissen immerhin als Interventionsbereich identifiziert, wenn sich auch hier, ähnlich wie

bei Wiig die Vorschläge zur Vorgehensweise in Grenzen halten.

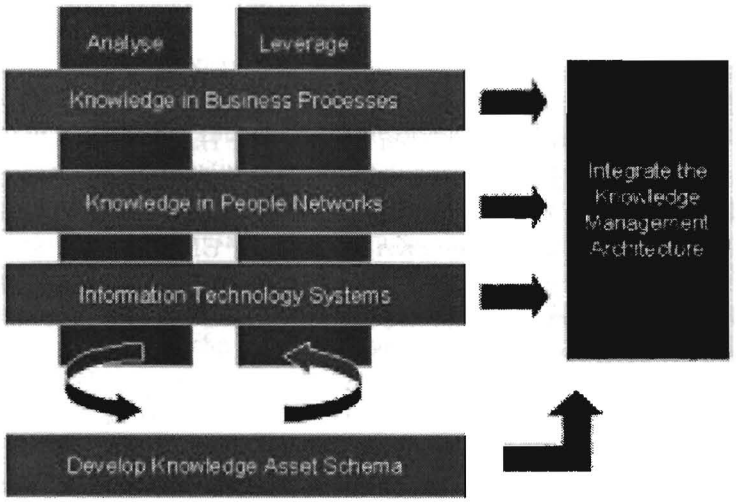

Abbildung 7.3: Stufe II der Know-Net Wissensmanagement-Methode.

Macintosh et al. [Macintosh and Kingston, 1999] geben in dieser Hinsicht schon etwas deutlichere Ratschläge. Neben einer Reihe anderer Analyse- und Modellierungstechniken stellen sie *Knowledge Asset Roadmaps* [Macintosh *et al.*, 1998] vor, eine Weiterführung der Idee der Technologie-Roadmaps für die strategische Planung. Bei diesem Ansatz setzt man für den betrachteten Planungshorizont verschiedene für die angestrebten Geschäftsziele relevanten Facetten – wie Schlüsselprojekte, involvierte Wissensprozesse, oder eben auch wichtige

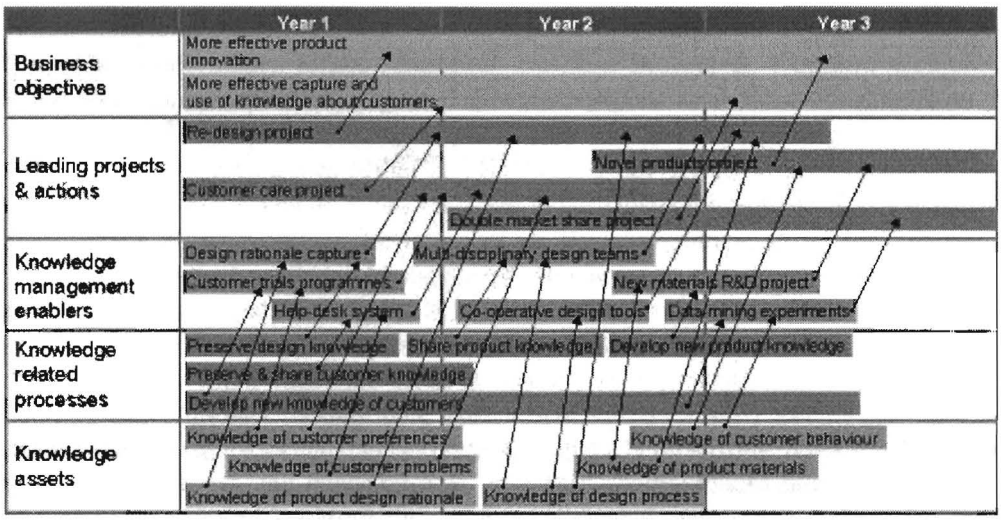

Abbildung 7.4: Beispiel für eine Knowledge Asset Roadmap.

Die **CommonKADS-Methode** [Akkermans *et al.*, 1999, Schreiber *et al.*, 1999], ursprüng-

lich entwickelt als methodisch fundierte Vorgehensweise für den Bau von Expertensystemen, bietet einen umfassenden Rahmen zur Entwicklung von WM-Systemen. Hier wird, wie bei keiner anderen uns bekannten Methode, ein durchgängiges Vorgehensmodell und ein umfassender Analyseansatz, mit Geschäftsprozessen und Aufgabenbeschreibungen im Mittelpunkt, angeboten. Weitere, für die praktische Umsetzung wesentliche, Module befassen sich beispielsweise mit Machbarkeits- und Kosten-Nutzen-Analysen für mögliche Handlungsoptionen.

Verglichen mit den zuvor genannten Methoden von Wiig und Know-Net bewegt sich das CommonKADS-Modell auf einer feinkörnigeren Ebene und gibt daher etwas ‘handfestere‘ Hilfestellung. So stützt sich CommonKADS etwa auf UML (Unified Modelling Language [Rumbaugh *et al.*, 1999]), einer visuellen Modellierungssprache die eine standardisierte Sammlung von Modellierungskonzepten darstellt. Diese reichen von Use-Cases bis hin zu den Klassendiagrammen der objektorientierten Programmierung.

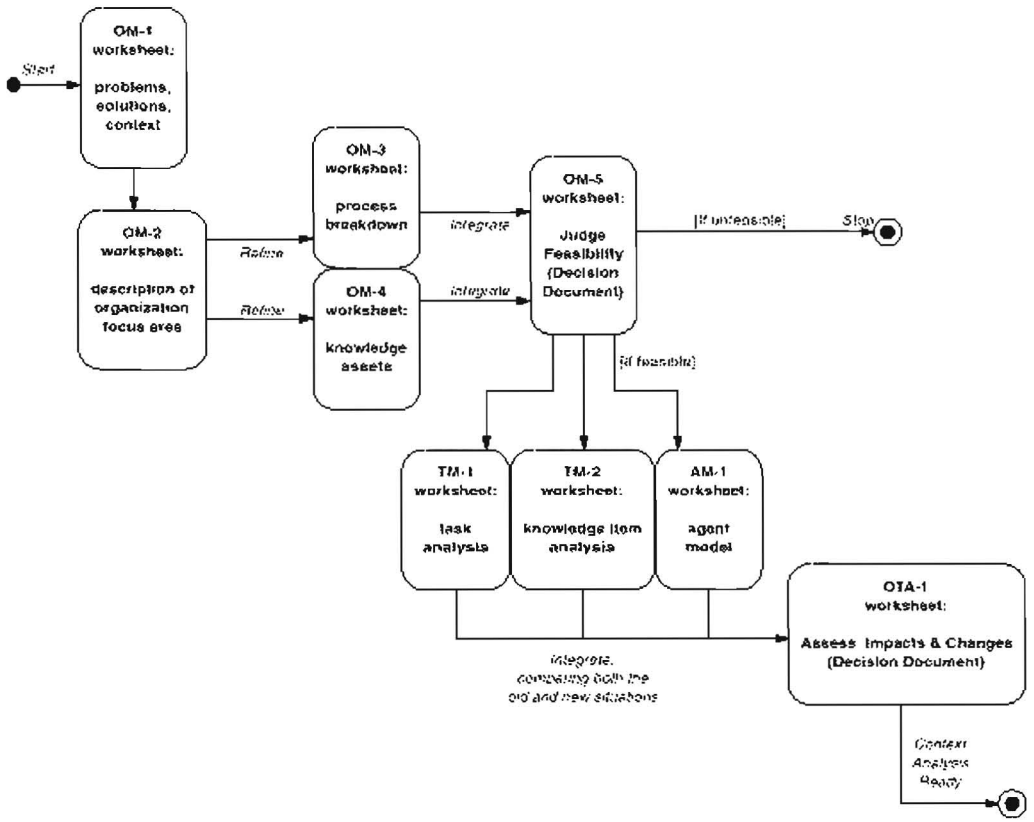

Abbildung 7.5: Fahrplan zur Durchführung von WM-Projekten nach CommonKADS.

Für die frühen Phasen der Unternehmensanalyse mag dieser Detaillierungsgrad allerdings eventuell zu hoch sein, bedingt er doch umfangreiche Interview- (bzw. allgemeiner: Wissenserfassungs-) und Modellierungstätigkeiten. Hier erscheint uns der Roadmap-Ansatz von Macintosh *et al.* angemessener. Jedenfalls bestätigen die Praxiserfahrungen der Partner im DECOR-Projekt [DECOR Consortium, 2000], dass eine rigide Prozessorientierung zur Strukturierung der wissensorientierten Organisationsanalyse eine wertvolle Hilfe ist, um zielführend arbeiten zu können [Müller and Herterich, 2001].

Weiterhin kann man feststellen, dass sowohl Macintosh *et al.* als auch CommonKADS bei ihren Darstellungen der erforderlichen Analysetätigkeiten i.a. auf der Ebene der Identifikation von Wissensquellen (Tickermeldungen von Reuters, die Unternehmensbibliothek, der Kollege XY, ...) oder groben Wissensbereichen (Wissen über Wettbewerber, Wissen über neue Produkt-Markt-Kombinationen, ...) enden. In Know-Net dagegen wurde der Ansatz verfolgt, zum Zwecke vereinfachter inhaltsorientierter Navigation in Unternehmensarchiven und verbesserter automatischer Informationsbeschaffung aus diesen Archiven bzw. anderen Quellen, auch eine weitergehende inhaltsbezogene Charakterisierung von Wissensinhalten und -bedarfen auf der Basis von Domänenmodellen bzw. -ontologien anzustreben [Benjamins *et al.*, 1998, Sintek *et al.*, 2000] (dies spiegelt sich in Abbildung 7.3 im Arbeitsschritt ''develop knowledge asset schema'' wider, der sowohl die Analyse benutzter Wissensquellen und grober Inhaltsbereiche als auch feinerer Inhaltsbeschreibungen und Wissensstrukturen umfasst). Auch für diese verfeinerten Domänenanalysen, die durch die o.g. Methoden nicht sehr weitgehend unterstützt werden, erscheint uns eine Vorgehensweise vernünftig zu sein, welche durch die Aufgabenanalyse der Geschäftsprozessmodellierung getrieben wird.

Motiviert aus dem engeren Kontext der Wissensakquisition für Expertensysteme, stellen **Speel et al.** [Speel *et al.*, 1999] für diese Detailanalysen u.a. eine Methode vor, die auf der Basis von Problem-Cause-Solution Diagrammen das graphische *Knowledge Mapping* unterstützt. Solche aufgabengetriebenen Analysemethoden könnten eine sinnvolle Ergänzung einer prozessorientierten ''äußeren Schleife'' in der Tradition von Know-Net und CommonKADS sein.

7.1.1 Workflow Einführung

Da die in *FRODO* zu entwerfende Methodologie Geschäftsprozesse fokussiert und deren Operationalisierung durch WfMS geschieht, besitzt die Modellierung von Workflows eine zentrale Bedeutung. Jedoch sind, wie Remus und Lehner [Remus and Lehner, 2000] feststellen, die klassischen Methoden der Geschäftsprozess-Modellierung nicht ausreichend für wissensintensive Prozesse und daher müssen diese um geeignete Aspekte erweitert werden. Da dies bereits Thema des vorigen Abschnittes war, werden wir im folgenden einige Vorgehensweisen zur Workflow Einführung betrachtet.

Die GIGA Information Group 'European Business Process & Workflow Conference' [GIGA, 1998] thematisierte die gegenseitige Abhängigkeit von Wissensmanagement und Geschäftsprozessen. Dabei wurde die zentrale Rolle von Workflow Management betont als Bindeglied und technologisches Mittel zur Umsetzung des Wissensmanagements in Unternehmen. Campbelli [Campbelli, 1998] sieht bei der Realisierung von Wissensmanagement-Projekten einen Trend Workflow Management als Basistechnologie zu nehmen, da etwa in den USA der Begriff "Knowledge Management" abgenutzt ist und eine greifbare Umsetzung fehlte. Jedoch nennt er als Grund des Scheiterns vieler Workflow-Projekte die Unkenntnis der Unternehmen über ihre Geschäftsprozess-Strukturen. Daher kommen Vorgehensmodellen zur Einführung von WfMS eine große Bedeutung zu. Betrachtet man nun Methoden zur Einführung von Workflow Management Systemen in Unternehmen, so fällt auf, dass diese sehr stark Prozess- und anwendungsgetrieben sind und kaum Wissensmanagement als ganzheitlichen Ansatz betrachten. So beschreibt etwa Christine Lenz von dem WfMS-Hersteller Staffware in [Lenz, 1998] eine Workflow-Einführungsstrategie, die sich in mehrere Phasen unterteilen lässt:

- Die *Vision* umfasst die Unternehmensvision und die Positionierung des WfMS darin,

woraus die Vorgaben abgeleitet werden sollen.

- Anschließend werden die *Ziele* identifiziert und priorisiert, woraus
- die *Prozessevaluation* die konkreten Prozesse, beteiligten Personen und Ressourcen ableiten kann.
- Das nachfolgende *Prozessdesign* teilt sich auf in Modellierung, Simulation, Bewertung und Optimierung, welches wiederum Input ist
- für die *Implementierung*, wobei das WfMS in die Unternehmensumgebung integriert wird, die Prozesse realisiert werden, und eine Schulung und Einführung durchgeführt wird.

Dabei hebt Lenz zwar die Bedeutung der Unternehmensanalyse und die Mitarbeiterakzeptanz als wichtige Vorbedingungen zum Erfolg der Einführung. Eine explizite Berücksichtigung von Wissensmanagement - abgesehen von der Einbindung vorhandener Technologien, wie eMail und Dokumenten Management – erfolgt jedoch nicht.

Striemer et al. präsentieren in [Striemer *et al.*, 1997] einen Überblick unterschiedlicher Vorgehensmodelle im Bereich Workflow Management. Sie stellen fest, dass es sich dabei meist um Derivate allgemeiner Vorgehensmodelle für die Informationssystem-Entwicklung handelt, die um spezielle Eigenschaften der einzelnen Ansätze erweitert werden.

Als ein Beispiel hierzu sind Jablonski und Stein [Jablonski and Stein, 1998, Stein, 1999] zu nennen, die aus dem Phasenmodell der Informationssystem-Entwicklung ein Vorgehensmodell für Workflow Management Anwendungen ableiten. Die Modellierung stützt sich dabei auf unterschiedliche Perspektiven, welche die verschiedenen Aspekte des MOBILE-Meta-Modells [Jablonski, 1994] abbilden, darin u.a. Funktions-, Informations-, Verhaltens-, Organisations- und Operationsaspekte (siehe Abbildung 7.6). Die Einzelaspekte werden analysiert und fließen dann an die dafür vorgesehenen Stellen des MOBILE-WfMS ein (etwa der Organisationsaspekt in das Aufbaumodell des Unternehmens). Damit ergibt sich eine Synergie zwischen Analysephase des Anwendungsszenarios und der Implementierungsphase des WfMS. Es ist möglich weitere Aspekte hinzuzufügen, welche dann jedoch auch von dem MOBILE-System abgedeckt werden müssen.

Weiter unterscheiden Striemer et al. unterschiedliche Ansätze, wie die Geschäftsprozess (GP)- und Workflow-Modellierung in die Vorgehensmodelle einfließen: Im *isolierten* Ansatz werden Geschäftsprozesse und Workflows getrennt voneinander betrachtet, also geschieht die Workflow-Modellierung ohne auf die Grundlage einer GP-Modellierung zurückzugreifen. Nahezu jedes WfMS bietet hierzu geeignete Mittel an, wie etwa ein graphisches Workflow-Modellierungstool. Im Gegensatz dazu bildet im *sequentiellen* Ansatz eine GP-Modellierung die Grundlage für die Workflow-Modellierung. Der *integrierte* Ansatz ist eine Weiterführung des sequentiellen, mit dem Zusatz, dass zu jeder Zeit ohne Verluste in frühere Schritte gegangen werden kann. Dies fordert aber einen ganzheitlichen Ansatz zur Modellierung, wie ihn z.B. [Deiters, 1997] fordert.

Damit eng verknüpft ist der Begriff des *Workflow-Lebenszyklus*, der mit der Erhebung und Modellierung des Geschäftsprozesses beginnt, die GP-Modelle in Workflow-Modelle überführt, welche dann ausgeführt werden. Anschließend wird die Audit Data für eine Verbesserung der zugrundeliegenden Prozessmodelle ausgewertet, die wiederum in die Workflow-Modellierung einfließen [Galler and Scheer, 1995].

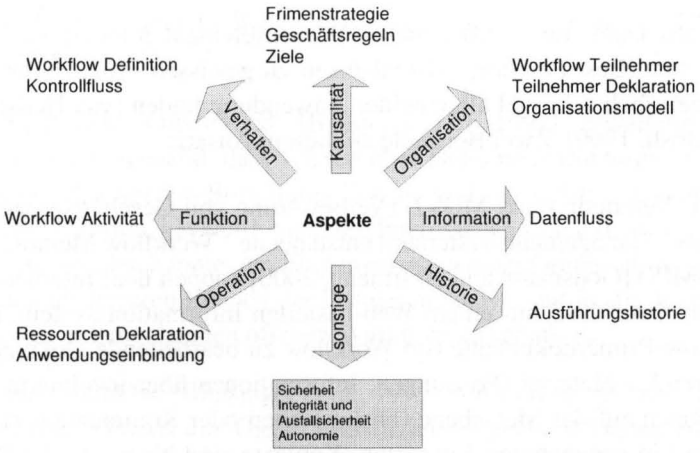

Abbildung 7.6: Aspekte und zugehörige Elemente des WfMS.

Einen neuen Weg beschreitet das POLIVEST-Projekt [Dietel *et al.*, 1997], das statt der bekannten top-down ein bottom-up-Vorgehen – also vom Benutzer getrieben – verwendet. POLIVEST schlägt eine zweigeteilte Vorgehensweise zur Einführung eines Workflow Systems vor. Zuerst werden dazu geeignete Arbeitsabläufe eines Benutzers als sequentielle Arbeitsschritte in SmartAssist – einem Makro-Editor und Ausführungswerkzeug – modelliert und dann als Makros ausgeführt und angepasst. Komplexe Prozesse können damit jedoch nicht modelliert werden. Somit werden Benutzer einerseits an die Automatisierung ihrer Arbeit herangeführt, andererseits erhält man damit – nach einigen Iterationsstufen von Verwendung und Anpassung – automatisierte Teile des Gesamtprozesses. In der folgenden Phase wird das endgültige Workflow-Modell durch die Verwendung der so isolierten Prozessteile gewonnen und in WorkParty – ein klassisches WfMS – überführt.

7.2 Systemnutzung: Kopplung von Wissensmanagement und Workflow

[Goesmann *et al.*, 1998] stellen zu Recht fest, dass sich besondere Ansatzpunkte zur Unterstützung der Durchführung von Geschäftsprozessen durch ein WM-System immer dort ergeben, wo komplexe, wissens- und informationsintensive Aktivitäten unterhalb des Granularitätsniveaus der Modellierung vorliegen. Gerade bei solchen Aktivitäten ist es besonders nützlich, wenn der Benutzer auf zusätzliches Wissen zur Bearbeitung der Aufgabe zugreifen kann. Für diesen Zugriff auf bereits vorhandenes Wissen während der laufenden Workflow-Ausführung ergeben sich nun interessante Nutzeffekte, die das WM-System bei der Bereitstellung abgelegten Wissens vom Workflow-Management-System ziehen kann:

Prozessorientierte Archivorganisation

Die einfachste Art der Synthese ergibt sich, wenn man als eine (ggf. von mehreren) Indexierungsdimension für ein Wissensarchiv die Geschäftsprozesse und/oder Aufgabenmodellierungen der Organisation benutzt, so dass man jedes Wissenselement über den Prozess bzw.

die Aufgabe auffinden kann, für den/die dieses Wissenselement relevant ist. Diese Sichtweise ist zumindest für ''Best Practice'' Datenbanken zu gewissen Prozessschritten naheliegend, kann aber sicher noch sehr viel allgemeiner Anwendung finden [van Heijst *et al.*, 1996, Kingston and Macintosh, 1999]. Zwei Beispiele zu diesem Ansatz:

- Das im BMBF-Verbundprojekt MOVE (Verbesserung von Geschäftsprozessen mit flexiblen Workflow-Management-Systemen) entstandene ''Workflow Memory Information System'' (WoMIS) [Goesmann and Hoffmann, 2000] koppelt über Internet-Technologie ein WfMS der Firma CSE mit einem Web-basierten Informationssystem. Dieses organisiert Workflow-Primärdokumente (im Workflow zu bearbeitende Dokumente) ebenso wie unterstützendes Material (Weisungen, Informationen über involvierte Kunden, ...) und Informationen auf der Metaebene (Diskussionen oder Kommentare zur Aufgabenausführung etc.) in sogenannten Kontexten. Kontexte sind hierarchische Baumstrukturen, die alle im Workflow involvierten Objekte (Aktivitäten, Werkzeuge, Dokumente, Rollen, Geschäftsprozesse und Geschäftsfälle bzw. Instanzen) reflektieren und als Container für alle mit diesem Objekt befassten Informationsobjekte dienen können. Diese direkte Entsprechung zwischen Workflow-Objekten und Informationsstrukturierung erleichtert einen nahtlosen Übergang zwischen Workflow-Primärprozess und Wissensprozessen.
- Eine sehr ähnliche Funktionalität stellt das CognoVision Tool der DHC GmbH [Müller and Herterich, 2001, DHC – Dr. Herterich & Consultants GmbH, Saarbrücken, 2000] zur Verfügung. Dieses bietet äußerst mächtige und flexible Möglichkeiten, Dokumentbasen durch annotiert verlinkte Begriffsnetzwerke zu organisieren. Anwendungsszenarien des Tools umfassen beispielsweise hochgradig dokumentlastige Änderungsprozesse für SAP R/3-Installationen in sicherheitskritischen Anwendungsbereichen. Für solche Vorgänge lassen sich die aufgabenspezifischen Dokumente (Systemhandbücher, Input- und Zwischenergebnis-Dokumente, Standard-Vorgehensmodelle und -Testfälle, Dokumentationsschablonen, ...) für den einfachen Benutzerzugriff nach verschiedenen Sichten ordnen, beispielsweise mithilfe einer Modellierung der Änderungsprozedur. Hierzu kann man z.B. mit dem ARIS-Toolset erzeugte Visualisierungen von Prozessmodellen importieren und diese als graphische Zugangsschnittstelle zum Dokumentarchiv benutzen. Ein ähnliches Ziel (Prozessmodelle zur Navigationshilfe) verfolgt der gedion Prozessnavigator [Fillies *et al.*, 2001].

Aktive Informationslieferung

Die nächste Stufe von Systemdiensten organisiert nicht nur Wissensinhalte nach ihrem Prozessbezug, sondern nutzt die Prozessausführung direkt, um von ihr aktiv Informationsbereitstellungsdienste zur Prozesslaufzeit anstoßen zu lassen. Damit können dem Nutzer proaktiv Information geliefert werden, die er selber u.U. gar nicht erwartet oder gesucht hätte. Es entfällt ferner der manuelle Suchaufwand. Prinzipiell könnten so automatisch aufgefundene Informationen sogar vor der Anzeige beim Benutzer noch weiterverarbeitet werden, z.B. zur benutzerorientierten Präsentationsaufbereitung (Personalisierung von Information), zur Vorverarbeitung (z.B. Abstracting zur Konzentration auf das Wesentliche) oder zur Berechnung

von Lösungsvorschlägen. Einfache Beispiele für die automatisch vom WfMS gesteuerte Informationsverarbeitung:

- Das DFKI-Projekt OfficeMaid [Baumann *et al.*, 1997] erweiterte die Workflow-Modellierung dahingehend, dass Wissen über Dokumentstrukturen und typische Dokumentketten (z.B. Angebot–Bestellung–Rechnung–Lieferschein) zusammen mit zu Mail-Accounts assoziierten, keyword-basierten Interessensprofilen (bzw. Zuständigkeitsbereichen) benutzt werden konnte, um Eingangspost in einer Organisation nach dem Einscannen und den niedrigeren Ebenen der Dokumentanalyse automatisch an den zuständigen Bearbeiter im entsprechenden offenen Workflow zu schicken.

- Das Produkt IntelliDoc [Bleisinger *et al.*, 1999] von COI setzte die OfficeMaid Ideen in die kommerzielle Praxis um. Dieses klassifiziert eingehende Dokumente und verteilt sie entweder direkt in die persönlichen Postkörbe der ermittelten Empfänger (über personell vorgegebene Profile bezüglich Dokumentenart und extrahierten Inhalten), oder indirekt durch den Start eines entsprechenden Workflows (anhand der Dokumentenart).

Dynamischer Prozesskontext

Geht man noch einen Schritt weiter, kann man auch noch den dynamischen Kontext der aktuellen Prozessinstanzen heranziehen, um spezifischere Informationsrecherchen durchführen zu können. Eine erweiterte Prozessmodellierung, die inhaltliche Aspekte der zu bearbeitenden Aufgaben umfaßt, kann zur Prozessausführungszeit instanzenspezifische Recherchen anstoßen.

Am DFKI wurde dieser Ansatz verschiedentlich erprobt, insbesondere in den Projekten *VirtualOffice* [Wenzel, 1998, Abecker *et al.*, 2000b] und *KnowMore* [Abecker *et al.*, 2000a], die wir weiter unten in einigem Detail erläutern werden. Auf dem *KnowMore* Konzept beruhende Ansätze wurden inzwischen beispielsweise auch am AIFB der Uni Karlsruhe entwickelt [Staab and Schnurr, 1999, Staab and Schnurr, 2000].

Betrachtet man die drei genannten Prinzipien des Prozessangebundenen Wissensmanagements, so verwirklichen sie i.w. die beiden Ideen:

Knowledge is information made actionable.

und

Knowledge is information in context.

7.3 Systemevolution: Kontinuierliche Prozessverbesserung als Wissensmanagement-Aufgabe

Im letzten Abschnitt sind wir immer davon ausgegangen, dass eine *gegebene, stabile* Workflow-Modellierung benutzt wird, um Informationsobjekte für die manuelle Navigation zu organisieren, aktiv im Workflow-Ablauf zu präsentieren und dabei auch neues Wissen zu erfassen bzw.

bestehendes zu evolvieren. Beachtet man nun, dass WM-Initiativen sich i.a. mit *wissensintensiven Prozessen* [Davenport *et al.*, 1996] befassen, die von Haus aus schwer a priori planbar sind, häufig individuell sehr unterschiedliche Arbeitsabläufe aufweisen und oft sehr dynamischen Umweltbedingungen (Randbedingungen, Optimierungsgrößen, Informationslage, etc.) unterliegen, so wird klar, dass konventionelle Geschäftsprozessmodellierungen und traditionelle Workflow Systeme hier nicht sehr nützlich sind (vgl. z.B. [Goesmann *et al.*, 1998, Remus and Lehner, 2000, Schwarz *et al.*, 2001, Wargitsch, 1997]). Offensichtlich ist zur Unterstützung solcher Prozesse ein hohes Maß an Flexibilität und dynamischer Adaptivität erforderlich. Für ein solches, hochgradig adaptives Workflow System wird nun auch die Geschäftsprozessmodellierung und -ausführung selber zur unterstützenswerten Aufgabe, die von einem WM-Support profitieren kann.

Das Gebiet ist noch zu jung, um hier abschließende Strukturierungen zu finden. Ein erster Überblick findet sich bei Goesmann *et al.* [Goesmann *et al.*, 1998], die als Unterstützungspotentiale u.a. beschreiben:

- Zugriff auf bereits erfasstes Unternehmenswissen, das die Entscheidung über den Prozessablauf betrifft, z.B. Normen, Regulierungen, Monitoring-Daten früherer Prozessinstanzen
- Erfassung von Begründungen für Entscheidungen über Prozessabläufe, die in anderen Geschäftsfällen genutzt werden können
- die Identifikation ähnlicher, bereits aufgetretener Geschäftsfälle – hierfür bieten sich Techniken des fallbasierten Schließens an
- wissensbasierte Vervollständigung von Modellen durch erweiterte Prozessmodelle, die z.B. Prozessvarianten und Auswahlbedingungen enthalten

Grundprämisse ist dabei jeweils, dass schon die konkrete Art der Ausführung eines wissensintensiven Prozesses selber eine wissensintensive Aufgabe ist, die durch Informationslieferung und Kommunikationsunterstützung erleichtert werden kann, so dass Geschäftsprozesse und Prozessinstanzen selber zum Inhalt des WM-Systems werden, über den auch diskutiert und der über die Zeit hinweg evolviert werden kann und muss.

Sehr weitgehende Ideen wurden zu diesem Thema bereits im WorkBrain-System [Wargitsch, 1997, Wargitsch *et al.*, 1998] umgesetzt, wo man geschäftsfallspezifische Workflow-Instanzen aus fallbasiert im Organisationsgedächtnis aufgefundenen Prozessschemata und Schemabausteinen zusammensetzen kann, im Prozessverlauf verfeinert oder adaptiert und dabei auch über assoziierte Diskussionsgruppen das Prozesswissen im Diskurs mit den Kollegen erweitern kann. Im MILOS-System [Maurer and Holz, 1999] befasst man sich mit langlaufenden Prozessen, z.B. in der Software-Entwicklung oder der Raum- und Umweltplanung, die von vielen, teilweise zur Prozesslaufzeit veränderlichen, äußeren Faktoren beeinflusst werden, beispielsweise Kundenwünschen oder rechtlichen Bestimmungen bzw. juristischen Entscheidungen. Die Unterstützungsanforderungen für solche Aufgaben bewegen sich schon mehr in den Bereich des Projektmananagements als der Prozesssteuerung, wobei der Nachverfolgbarkeit von Entscheidungen und Revidierbarkeit bei veränderten Rahmenbedingungen besondere Aufmerksamkeit gewidmet wird.

In *FRODO* wird aus einer aus ähnlichen Grundmotivationen heraus, ein Workflow-Begriff für schwach strukturierte, wissensintensive Prozesse definiert und auf der Basis kooperie-

render Software-Agenten flexible und robuste Implementierungen entwickelt, die nahtlos mit Informations- und persönlichen Filter- und Präsentationsagenten zusammenarbeiten.

7.4 Zusammenfassung

Auf der Ebene der wissensorientierten Organisationsanalyse als Ausgangsbasis für die anvisierte Methodologie haben wir argumentiert, dass eine enge Verschränkung von Geschäftsprozess- und wissensorientiertem Vorgehen, wie bei CommonKADS umgesetzt, vielversprechend ist, wenn man es in eine umfassendere WM-Methodik (wie von Know-Net vorgestellt) einbettet, um geeignete Methoden zur Identifizierung von Knowledge Assets erweitert (wie von Macintosh und Speel) und im Bereich der Inhaltsmodellierung von Informationsobjekten um geeignete Ontologieakquisitionsmethoden ergänzt (wiederum KnowNet).

Im Bereich der Einführung von Workflows in Unternehmen, wird sich *FRODO* auf die Unterstützung von Wissensarbeit konzentrieren und damit alle technik- und anwendungsbasierten Ansätze außen vor lassen. Daher wird ausgehend von der Idee des Workflow-Lebenszyklus, eine eigene Vorgehensweise zur Einführung entwickelt, die auf eine kontinuierliche Workflow-Evolution durch die Benutzer setzt. Damit ist diese Vorgehensweise eng mit der Systemevolution verbunden.

Bei der Systemnutzung stützt sich *FRODO* auf die Weiterführung und Entwicklung der pro-aktiven Informationslieferung aus KnowMore und der Einführung des Konzeptes des dynamischen Prozesskontextes.

Zusammengefasst zeigt Tabelle 7.1 eine nach den Systemphasen strukturierte Aufstellung der von uns als sinnvoll erachteten Elemente der verschiedenen Methoden.

Wie im FRODO Projektantrag erwähnt, wird das Know-Net-Framework als Ausgangsbasis dienen. D.h. für das FRODO Methodologie Grundgerüst wird von Know-Net übernommen:

- die Leitprinzipien
- grundlegende Definitionen (etwa Knowledge Assets)
- die Komponenten der Wissensmanagement Infrastruktur:
  - WM-Strategie
  - WM-Struktur
  - WM-Prozesse
  - WM-Systeme (gefüllt mit FRODO-Komponenten)
- Knowledge Networking Levels

Die oben aufgeführten Ansätze werden in dieses Framework eingepasst. Deren Kombination und Zusammenspiel zu einer durchgängigen Methode, soll anhand des *FRODO* Anwendungsszenarios getestet werden. Dabei wird sich auch herausstellen, welche Analyse-Methoden am besten geeignet sind, um eine reibungslose Überführung der Modellierung in das System zu gewährleisten.

Phase	Teilphase	Referenz	Ansatz
Design	1. Phase der Unternehmens-modellierung	Macintosh	Knowlege Asset Roadmaps
		KnowNet	Geschäftsprozess-orientierte Vorgehensweise
	Wissenserfassung	CommonKADS	Interviews/Fragebögen
		Speel	Knowledge Mapping
	verfeinerte Domänenanalyse	KnowNet	inhaltsbezogene Charakterisierung von Wissensinhalten und -bedarfen auf Basis von Domänenmodellen bzw. -ontologien
	Workflow	Jablonski	aspekten-orientierte Analyse für WfMS
Einführung	Workflow	u.a. Galler Scheer	Workflow-Lebenszyklus
Nutzung	Wissensarbeit	KnowMore	pro-aktive Informationsliefe-rung
		FRODO	dynamischer Prozesskontext
Evolution	Workflow	FRODO	Workflow-Evolution

Tabelle 7.1: Methodologie: Verwendung von Ansätzen in den Systemphasen

Kapitel 8

Konzept und Realität - Ein Anwendungsbeispiel

Die hier vorgestellten Konzepte gehen von zentralen Annahmen über die Existenz und den Charakter mehrerer verteilter Organizational Memories in einem gemeinsamen Unternehmenskontext aus. Es bleibt zu beweisen, daß diese Annahmen in der Realität zutreffen und daß die vorgestellten Konzepte Lösungen von praktischer Relevanz darstellen.

Im folgenden wird daher ein Anwendungsszenario vorgestellt, das die angenommenen charakteristischen Eigenschaften tatsächlich aufweist, mithin zur Erprobung der vorgestellten Konzepte geeignet ist. Im weiteren Verlauf des Projekts *FRODO* werden zentrale Projektergebnisse in diesem Szenario erprobt und eingesetzt werden.

8.1 Problemsituation: Bewahren von Know-How in der Kerntechnik

Diei Betreiber kerntechnischer Anlagen in der Bundesrepublik Deutschland sehen sich zunehmend mit einem außerordentlich kritischen Problem konfrontiert: Die erfahrenen Mitarbeiter und Wissensträger erreichen die Altersgrenze und verlassen die Unternehmen, gleichzeitig gibt es keinerlei Nachwuchskräfte. Aufgrund der politischen Entwicklungen und der gesellschaftlichen Einschätzung der Kernenergie sind entsprechende Ausbildungs- und Studiengänge nicht mehr gefragt bzw. eingestellt worden. Andererseits ist zum Betrieb, aber auch zur Abwicklung und zum Abbau kerntechnischer Anlagen eine Fülle von kritischem Know-How erforderlich, das nunmehr akut gefährdet ist.

Vor diesem Hintergrund wird nach Möglichkeiten gesucht, das kritische Wissen personenunabhängig und über lange Zeiten zu bewahren und verfügbar zu halten. Alle Ansätze des Wissensmanagements haben daher in diesem Anwendungsfeld eine besondere Bedeutung.

8.2 Vorhandene Informationsquellen

Günstig erweist sich, daß wesentliche Teile des relevanten Wissens bereits seit langem dokumentiert und strukturiert vorliegen. Da viele Aspekte im Zusammenhang mit kerntechnischen Anlagen in Form von Gesetzen und Verwaltungsvorschriften geregelt sind, existiert

auch eine Fülle von prozeßbezogenen Dokumentationen. Vorgänge und Verfahren sind vielfach vollständig modelliert und dokumentiert.

Jeder Versuch, in diesem Anwendungsfeld Techniken des Wissensmanagements zum Einsatz zu bringen, kann also auf eine Fülle von Informationsquellen und Ansätze für Modelle und Ontologien zurückgreifen.

8.3 Verteiltheit

Andererseits sind in wesentlichen Aufgaben im Umfeld kerntechnischer Anlagen eine Reihe von Teilnehmern involviert, die jeweils spezifische Aspekte beachten und zur Lösung der Aufgabe kooperieren müssen:

- Die Betreiber der Anlage
- die Genehmigungsbehörden der einzelnen Bundesländer
- die Gesellschaft für Reaktorsicherheit
- das Öko-Institut Darmstadt als 'neutrale' Dienstleistungsinstanz
- das Umweltbundesamt und ggf. weitere Ministerien auf Bundesebene

Jeder dieser Teilnehmer verfügt über spezifisches Fachwissen, das es zu bewahren gilt und das zur Verfügung stehen muß. Jeder Teilnehmer ist aber auch eifersüchtig darauf bedacht, daß die Hoheit über sein spezifisches Wissen bei ihm verbleibt. Ein zentralistischer Ansatz im Sinne einer gemeinsamen OM-Architektur ist daher von vorn herein zum Scheitern verurteilt.

Der von FRODO vorgeschlagene Weg, die bei den verschiedenen Teilnehmern vorhandenen Informationsquellen und Repositories als lokale Organizational Memories zu betrachten und den wechselseitigen Zugriff im Sinne kooperierender Agenten zu lösen, ist hingegen der Situation angemessen und wird akzeptiert.

8.4 Prozesse und Weak Workflows

Wie bereits erwähnt, sind im Umfeld kerntechnischer Anlagen vielfältige Vorgehensweisen detailliert dokumentiert und vorgeschrieben. Andererseits sind gerade die vielfältigen Genehmigungsprozesse durchaus im Einzelfall den jeweiligen Gegebenheiten entsprechend zu konfigurieren, so daß unter Einsatz von Wissen aus den vorhandenen Standard-Aktionsfolgen die spezifische Prozedur entsteht. Diese Vorgehensweise entspricht aber genau dem in FRODO formulierten Konzept der schwach modellierten Prozesse bzw. weak workflows.

8.5 Konkretes Beispiel: Transportgenehmigung

Der Transport kerntechnischen Materials, etwa aus der Wiederaufarbeitungsanlage in La Hague / Frankreich in die Zwischenlager oder ein Endlager in der Bundesrepublik wird minutiös geplant und unterliegt in allen Einzelheiten vielfältigen Genehmigungsvorschriften. Die korrekte Durchführung eines solchen Transports ist deshalb bereits in der Genehmigungsphase als wissensintensive Aktivität anzusehen.

Die Vorgänge, die zur korrekten Erstellung eines Genehmigungsantrags und zur Bewertung und Genehmigung eines solchen durchzuführen sind, sollen als ein Anwendungsbeispiel modelliert und operationalisiert werden. Die dabei in Frage kommenden Informations- und Wissensquellen bei den verschiedenen Teilnehmern werden wie oben skizziert als kooperative Organizational Memories im Hoheitsbereich der jeweiligen Teilnehmer erfaßt. Durch dieses Zusammenspiel ist das Gesamtsystem in der Lage, dem mit der Transportplanung und -beantragung befaßten Menschen zu jeder Zeit effektive Hilfestellung über die zu leistenden Schritte, zu berücksichtigenden Vorschriften, zu erhebenden Daten usw. zu geben und damit sowohl eine effektive Antragsstellung als auch eine korrekte Bearbeitung und Genehmigung zu unterstützen.

8.6 Organisatorisches: begleitendes Pilotprojekt

Im Auftrag des Bundesministeriums für Umwelt und unter Führung der Gesellschaft für Reaktorsicherheit wird ein Pilotprojekt durchgeführt, das Möglichkeiten des Wissensmanagements zur Erhaltung des Know-Hows im Bereich kerntechnischer Anlagen untersuchen und erproben soll. Die DFKI GmbH liefert konzeptionelle Grundlagen für dieses Projekt. Im Laufe der nächsten drei Jahre werden die in FRODO entwickelten Konzepte in dieses Projekt Eingang finden und in dem skizzierten Ansendungsbeispiel praktisch erprobt werden. Der Start des Pilotprojekts ist für April 2001 vorgesehen.

Literaturverzeichnis

[Abecker *et al.*, 1998a] A. Abecker, M. Sintek, and H. Wirtz. From hypermedia information retrieval to knowledge management in enterprises. In *IFMIP-98: First International Forum on Multimedia & Image Processing*, Anchorage, Alaska, USA, May 1998. TSI Press, Albuquerque, New Mexico, USA.

[Abecker *et al.*, 1998b] A. Abecker, A. Bernardi, K. Hinkelmann, O. Kühn, and M. Sintek. Techniques for organizational memory information systems. DFKI Document D-98-02, DFKI GmbH, February 1998.

[Abecker *et al.*, 1998c] A. Abecker, A. Bernardi, K. Hinkelmann, O. Kühn, and M. Sintek. Toward a Technology for Organizational Memories. *IEEE Intelligent Systems*, 13(3):40–48, June 1998.

[Abecker *et al.*, 2000a] A. Abecker, A. Bernardi, K. Hinkelmann, O. Kühn, and M. Sintek. Context-Aware, Proactive Delivery of Task-Specific Knowledge: The KnowMore Project. *Int. Journal on Information Systems Frontiers (ISF), Special Issue on Knowledge Management and Organizational Memory, Kluwer*, 2(3/4):139–162, 2000.

[Abecker *et al.*, 2000b] A. Abecker, A. Bernardi, H. Maus, M. Sintek, and C. Wenzel. Information Supply for Business Processes – Coupling Workflow with Document Analysis and Information Retrieval. *Knowledge-Based Systems, Special Issue on AI in Knowledge Management, Elsevier*, 13(5):271–284, November 2000.

[Abecker *et al.*, 2000c] A. Abecker, A. Bernardi, and M. Sintek. Proactive knowledge delivery for enterprise knowledge management. In G. Ruhe and F. Bomarius, editors, *Learning Software Organizations – Methodology and Applications*, number 1756 in Lecture Notes in Computer Science. Springer, 2000.

[Abecker *et al.*, 2001] A. Abecker, A. Bernardi, M. Sintek, and K. Hinkelmann. Enterprise information infrastructures for active, context-sensitive knowledge delivery. In S. Barnes, editor, *Knowledge Management Systems: Theory and Practice*. International Thomson Business Press, 2001. To appear.

[Akkermans *et al.*, 1999] H. Akkermans, P.-H. Speel, and A. Ratcliffe. Hot issues and cool solutions in knowledge management: An industrial case study. In *[Gaines* et al., *1999]*, 1999.

[Apostolou *et al.*, 2000] D. Apostolou, G. Mentzas, R. Young, and A. Abecker. Consolidating the Product Versus Process Controversy in Knowledge Management: The Know-Net

Approach. In J. Domingue, editor, *PAKeM 2000, The Third International Conference and Exhibition on The Practical Application of Knowledge Management, Manchester, UK*, April 2000.

[Baumann *et al.*, 1997] S. Baumann, M. Ben Hadj Ali, A. Dengel, T. Jäger, M. Malburg, A. Weigel, and C. Wenzel. Message extraction from printed documents - a complete solution. In *Fourth International Conference on Document Analysis and Recognition (ICDAR 97), Ulm, Germany, August 18-20*, 1997.

[Benjamins *et al.*, 1998] V. Benjamins, D. Fensel, and A. G. Pérez. Knowledge management through ontologies. In *[Reimer, 1998]*, 1998.

[Berners-Lee, 1999] T. Berners-Lee. *Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web by Its Inventor.* Harper San Francisco, September 1999.

[Berry and Myers, 1998] P. M. Berry and K. L. Myers. Adaptive Process Management: An AI Perspective. In *Towards Adaptive Workflow Systems*, 1998.

[Bleisinger *et al.*, 1999] R. Bleisinger, M. Müller, P. Hartmann, and T. Dörstling. Intelligente Eingangspostverarbeitung mit wissensbasierter Dokumentanalyse. *Wirtschaftsinformatik*, (8), 1999.

[Bonifacio *et al.*, 2000] M. Bonifacio, P. Bouquet, and A. Manzardo. A distributed intelligence paradigm for knowledge management. In *Proceedings of the AAAI Symposium on Bringing Knowledge to Business Processes.* AAAI, Menlo Park, March 20–22, 2000.

[Buckingham Shum, 1997] S. Buckingham Shum. Negotiating the construction and reconstruction of organisational memories. *Journal of Universal Computer Science*, 3(8):899–928, 1997. Auch als Report KMI-TR-56, Knowledge Media Institute, The Open University, Milton Keynes, UK, URL: `http://kmi.open.ac.uk/publications/techreports.html`.

[Budzik and Hammond, 1999] J. Budzik and K. Hammond. Watson: Anticipating and contextualizing information needs. In *Proc. of the Sixty-second Annual Meeting of the American Society for Information Science.* Information Today, Inc., Medford, NJ, 1999.

[Budzik and Hammond, 2000] J. Budzik and K. Hammond. User interactions with everyday applications as context for just-in-time information access. In *Proc. of Intelligent User Interfaces 2000.* ACM Press, 2000.

[Campbelli, 1998] W. Campbelli. Workflow – what are the business drivers? In *[GIGA, 1998]*, 1998.

[Carlsen and Jorgensen, 1998] S. Carlsen and H. Jorgensen. Emergent Workflow: The AIS Workware Demonstrator. In *ACM CSCW '98 Conference Workshop: Towards Adaptive Workflow Systems, Seattle*, 1998.

[Carlsen, 1998] S. Carlsen. Action Port Model: A Mixed Paradigm Conceptual Workflow Modeling Language. In *3rd IFCIS Conference on cooperative Information systems - CoopIs '98, New York*, 1998.

[Cohen, 1995] W. Cohen. Learning to classify english text with ilp methods. In L. D. Raedt, editor, *Advances in ILP*. IOS Press, 1995.

[COI GmbH,] COI GmbH. URL: `http://www.coi.de/coiupdate/info/info materialien_de.asp`.

[Conklin and Weil, 1997] E. J. Conklin and W. Weil. Wicked problems: Naming the pain in organizations. White Paper of Group Decision Support Systems, Inc., URL: `http://www.gdss.com/wicked.htm`, 1997.

[Craven and Mahling, 1995] N. Craven and D. Mahling. Goals and Processes: A Task Basis for Projects and Workflows. Technical report, Department of Information Science, University of Pittsburgh, May 1995.

[Davenport *et al.*, 1996] T. H. Davenport, S. L. Javenpaa, and M. C. Beers. Improving Knowledge Work Processes. *Sloan Management Review*, 37(4):53–65, Summer 1996.

[DCMI, 2001] DCMI. Dublin Core Metadata Initiative, 2001. URL: `http://purl.org/dc/`.

[Decker *et al.*, 1998] S. Decker, D. Brickley, J. Saarela, and J. Angele. A Query and Inference Service for RDF. In *QL'98 – The Query Languages Workshop*, 1998. URL: `http://www.w3.org/TandS/QL/QL98/pp/queryservice.html`.

[DECOR Consortium, 2000] DECOR Consortium. DECOR: Delivery of Context-Sensitive Organizational Knowledge, 2000. URL: `http://www.dfki.uni-kl.de/decor/`.

[Deiters, 1997] W. Deiters. Prozeßmodelle als grundlage für ein systematisches management von geschäftsprozessen. *Informatik Forschung und Entwicklung*, 12(2), 1997.

[DHC – Dr. Herterich & Consultants GmbH, Saarbrücken, 2000] DHC – Dr. Herterich & Consultants GmbH, Saarbrücken. , 2000. URL: `http://www.dhc-gmbh.com/`.

[Dieng and Vanwelkenhuysen, 1996] R. Dieng and J. Vanwelkenhuysen, editors. *Tenth Knowledge Acquisition for Knowledge-Based Systems Workshop KAW'96, Special Track on Corporate memory and Enterprise Modeling*. November 1996.

[Dietel *et al.*, 1997] C. Dietel, G. Schneider, and J. Schweitzer. POLIVEST - Integrierte Televerwaltung. In *GI Workshop Informatik '97, Rechnergestützte Kooperation in Verwaltungen und großen Unternehmen, Aachen*, 1997.

[Domingue, 1998] J. Domingue. Tadzebao and WebOnto: Discussing, Browsing, and Editing Ontologies on the Web. In *[Gaines and Musen, 1998]*, 1998.

[Engels and Bremdal, 2000] R. Engels and B. Bremdal. Information extraction: State-of-the-art report. Technical report, CognIT a.s; Asker; Norway, 2000.

[Fillies *et al.*, 2001] C. Fillies, F. Weichhardt, and G. Koch-Süwer. Prozessmodellierungswerkzeuge und das Semantic Web. In *[Müller* et al., *2001]*, 2001. Eingereicht.

[FIPA, 2000a] FIPA. Fipa abstract architecture specification, 2000. URL: `http://www.fipa.org/specs/fipa00001/`.

[FIPA, 2000b] FIPA. Fipa acl message structure specification, 2000. URL: http://www.fipa.org/specs/fipa00061/.

[FIPA, 2000c] FIPA. Fipa agent message transport service specification, 2000. URL: http://www.fipa.org/specs/fipa00067/.

[FIPA, 2000d] FIPA. Fipa communicative act library specification, 2000. URL: http://www.fipa.org/specs/fipa00037/.

[FIPA, 2000e] FIPA. Fipa content languages specification, 2000. URL: http://www.fipa.org/specs/fipa00007/.

[FIPA, 2000f] FIPA. Fipa interaction protocol library specification, 2000. URL: http://www.fipa.org/specs/fipa00025/.

[Fox *et al.*, 1993] M. S. Fox, J. F. Chionglo, and F. G. Fadel. A common-sense model of the enterprise. In *Proceedings of the 2nd Industrial Engineering Research Conference*, pages 425–429, 1993.

[Gaines and Musen, 1998] B. R. Gaines and M. A. Musen, editors. *11th Workshop on Knowledge Acquisition, Modeling and Management (KAW'98).* Knowledge Science Institute, University of Calgary, April 1998.

[Gaines *et al.*, 1999] B. Gaines, M. Musen, and R. Kremer, editors. *12th Workshop on Knowledge Acquisition, Modeling and Management (KAW'99).* Knowledge Science Institute, University of Calgary, October 1999.

[Galler and Scheer, 1995] J. Galler and A. W. Scheer. Workflow-Projekte: Vom Geschäftsprozeßmodell zur unternehmensspezifischen Anwendung. *Information Management*, 1:20–27, 1995.

[GIGA, 1998] *Proceedings of the GIGA European Business Process & Workflow Conference, Geneva, Switzerland.* GIGA Information Group, October 1998.

[Goesmann and Hoffmann, 2000] T. Goesmann and M. Hoffmann. Unterstützung wissensintensiver Geschäftsprozesse durch Workflow-Management-Systeme. In *DCSCW 2000.* Springer Verlag, Heidelberg, 2000.

[Goesmann *et al.*, 1998] T. Goesmann, E. Föcker, and R. Striemer. Wissensmanagement zur Unterstützung der Gestaltung und Durchführung von Geschäftsprozessen. Technical Report 48/98, Fraunhofer Institut Software- und Systemtechnik (ISST), September 1998.

[Group, 1997] T. Group. TeamWare Flow 2.0. White Paper, 1997. http://www.teamware.com.

[Gruber, 1995] T. R. Gruber. Toward principles for the design of ontologies used for knowledge sharing. *International Journal of Human-Computer Studies*, 43(5,6):907–928, 1995.

[Han *et al.*, 1998] Y. Han, A. Sheth, and C. Bussler. A taxonomy of adaptive workflow management. In *Proc. CSCW 98 Workshop Towards Adaptive Workflow Systems, Seattle, USA*, 1998.

[Horn and Jablonski, 1998] S. Horn and S. Jablonski. An Approach to Dynamic Instance Adaption in Workflow Management Applications, 1998.

[Insiders, 2000] Insiders. MindAccess. Insiders Information Management, 2000. URL: `http://www.im-insiders.de/html/smartfinder.html`.

[Jablonski and Stein, 1998] S. Jablonski and K. Stein. *Ein Vorgehensmodell für Workflow-Management-Anwendungen.* 1998.

[Jablonski, 1994] S. Jablonski. Mobile: A modular workflow model and architecture. In *Proc. International Working Conference on Dynamic Modelling and Information Systems, Noordwijkerhout, NL*, 1994.

[Jablonsky and Bussler, 1996] S. Jablonsky and C. Bussler. *Workflow Management. Modeling Concepts, Architecture and Implementation.* International Thomson Computer Press, 1996.

[Kidd, 1994] A. Kidd. The marks are on the knowledge worker. In U. M. Borghoff and R. Pareschi, editors, *Proc. ACM CHI'94: Human Factors in Computing Systems, Boston, Mass*, pages 186–191. ACM Press, 1994.

[Kifer *et al.*, 1995] M. Kifer, G. Lausen, and J. Wu. Logical foundations of object-oriented and frame-based languages. *Journal of the ACM*, 42:741–843, July 1995.

[Kingston and Macintosh, 1999] J. Kingston and A. Macintosh. Knowledge management through multi-perspective modelling: Representing and distributing organisational memory. In *[Gaines* et al., *1999]*, 1999.

[Know-Net Consortium, 2000] Know-Net Consortium. Know-Net: Knowledge Management with Intranet Technologies, Solution Homepage, 2000. URL: `http://www.know-net.org/`.

[Lenz, 1998] C. Lenz. Workflow einführen - aber wie ? *Client Server Computing, Dokumenten Management Special*, 1998.

[Leymann and Roller, 1994] F. Leymann and D. Roller. Business Process Management with FlowMark. In *Proceedings of Compcon 94, San Francisco*, page 230f, September 1994.

[Leymann and Roller, 2000] F. Leymann and D. Roller. *Production Workflow - Concepts and Techniques.* Prentice Hall PTR, Upper Saddle River, New Jersey, 2000.

[Lloyd and Topor, 1984] J. Lloyd and R. Topor. Making prolog more expressive. *Journal of Logic Programming*, 3:225–240, 1984.

[Ludäscher *et al.*, 1999] B. Ludäscher, G. Yang, and M. Kifer. FLORA: The secret of object-oriented logic programming. Technical report, SUNY at Stony Brook, 1999.

[Macintosh and Kingston, 1999] A. Macintosh and J. Kingston. Knowledge asset management–practical modelling & analysis techniques. Tutorial given at PAKeM-1999, Manchester, UK, 1999.

[Macintosh *et al.*, 1998] A. Macintosh, I. Filby, and A. Tate. Knowledge asset road maps. In *[Reimer, 1998]*, 1998.

[Maier and Remus, 1999] R. Maier and U. Remus. Towards a Framework for Knowledge Management Strategies: Process-Orientation as Strategic Starting Point. Technical report, Dep. for Information Management Systems, University of Regensburg, Germany, 1999.

[Marshak, 1997] R. T. Marshak. InConcert Workflow. Article in Patricia Seybold's Workgroup Computing Report, Available under URL: `http://www.inconcert.com/press/articles/iar_seybold.html`, 1997. URL: `http://www.inconcert.com/products/`.

[Maurer and Holz, 1999] F. Maurer and H. Holz. Process-Oriented Knowledge Management For Learning Software Organizations. In *[Gaines* et al., *1999]*, 1999.

[Müller and Herterich, 2001] S. Müller and R. Herterich. Prozessorientiertes Wissensmanagement mit CognoVision. In *[Müller* et al., *2001]*, 2001. Eingereicht.

[Müller *et al.*, 2001] H.-J. Müller, A. Abecker, K. Hinkelmann, and H. Maus, editors. *Workshop Geschäftsprozeßorientiertes Wissensmanagement auf der WM'2001, Baden-Baden.* March 2001.

[Myers, 1998] K. L. Myers. Towards a Framework for Continuous Planning and Execution. In *AAAI Fall Symposium on Distributed Continual Planning*, 1998. URL: `http://www.ai.sri.com/~cpef/`.

[Newell, 1982] A. Newell. The knowledge level. *Artificial Intelligence*, 18:87–127, 1982.

[Nutt, 1996] G. J. Nutt. The evolution toward flexible workflow systems. In *Distributed Systems Engineering*, number 3, pages 276–294, December 1996.

[Petrie *et al.*, 1999] C. Petrie, S. Goldmann, and A. Raquet. Agent-Based Project Management. Technical report, Center for Design Research, Stanford University, 1999.

[Pollack *et al.*, 1999] M. E. Pollack, I. Tsamardinos, and J. F. Horty. Adjustable Autonomy for a Plan Management Agent. In *AAAI Spring Symposium on Adjustable Autonomy, Stanford, CA*, March 1999.

[Reichert and Dadam, 1998] M. Reichert and P. Dadam. ADEPTflex – Supporting Dynamic Changes of Workflows Without Loosing Control. In *Journal of Intelligent Information Systems (JIIS), Special Issue on Workflow and Process Management*, volume 10, No. 2, 1998.

[Reimer, 1998] U. Reimer. *PAKM-98: Practical Aspects of Knowledge Management. Proc. of the Second Int. Conference.* October 1998. URL: `http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-13/`.

[Remus and Lehner, 2000] U. Remus and F. Lehner. The Role of Process-oriented Enterprise Modeling in Designing Process-oriented Knowledge Management Systems. In *[Staab and O'Leary, 2000]*, 2000.

[Rittel and Webber, 1973] H. Rittel and M. Webber. Dilemmas in a general theory of planning. *Policy Sciences*, 4, 1973.

[Rittel, 1972] H. Rittel. *Second Generation Design Methods.* 1972. Reprinted in: Developments in Design Methodology, N. Cross (Ed.) 1984, pp. 317-327, J. Wiley & Sons.

[Rumbaugh *et al.*, 1999] J. Rumbaugh, I. Jacobson, and G. Booch. *The unified modeling language reference manual : UML.* Addison-Wesley, Reading, Mass., 1999.

[Ruppietta and Wernke, 1994] W. Ruppietta and G. Wernke. *Umsetzung organisatorischer Regelungen in der Vorgangsbearbeitung mit WorkParty und ORM.* 1994.

[Schreiber *et al.*, 1999] G. Schreiber, H. Akkermans, A. Anjeiwerden, R. de Hoog, N. Shadbolt, W. van de Velde, and B. Wielinga. *Knowledge Engineering and Management: The CommonKADS Methodology.* MIT Press, 1999.

[Schwarz *et al.*, 2001] S. Schwarz, A. Abecker, H. Maus, and M. Sintek. Anforderungen an die Workflow-Unterstützung für wissensintensive Geschäftsprozesse. In *[Müller* et al., *2001]*, 2001. To appear.

[Schwarz, 2000] S. Schwarz. Schwach strukturierte Workflows für das Wissensmanagement in Unternehmen. Master's thesis, Fachbereich Informatik, Universität Kaiserslautern, August 2000.

[Sheth, 1997] A. Sheth. From Contemporary Workflow Process Automation to Adaptive and Dynamic Work Activity Coordination and Collaboration. In *IEEE: Eighth International Workshop on Database and Expert Systems Applications, Toulouse, France*, pages 24–27, September 1997.

[Sintek *et al.*, 2000] M. Sintek, B. Tschaitschian, A. Abecker, A. Bernardi, and H.-J. Müller. Using Ontologies for Advanced Information Access. In J. Domingue, editor, *PAKeM 2000, The Third International Conference and Exhibition on The Practical Application of Knowledge Management, Manchester, UK*, April 2000.

[Speel *et al.*, 1999] P.-H. Speel, N. Shadbolt, W. de Vries, P. van Dam, and K. O'Hara. Knowledge mapping for industrial purposes. In *[Gaines* et al., *1999]*, 1999.

[Staab and O'Leary, 2000] S. Staab and D. O'Leary. *AAAI Spring Symposium: Bringing Knowledge to Business Processes.* AAAI Press, March 2000. URL: `http://www.aifb.uni-karlsruhe.de/~sst/Research/Events/sss00/`.

[Staab and Schnurr, 1999] S. Staab and H.-P. Schnurr. Knowledge and business processes: Approaching an integration. In *International Workshop on Knowledge Management and Organizational Memory (OM'99) at IJCAI-99, Stockholm, Sweden*, 1999.

[Staab and Schnurr, 2000] S. Staab and H.-P. Schnurr. Smart Task Support through Proactive Access to Organizational Memory. *Knowledge-Based Systems, Elsevier*, September 2000. URL: `http://www.aifb.uni-karlsruhe.de/~sst/Research/-Publications/Publications.htm`.

[Stein, 1999] K. Stein. *Integration von Anwendungsprozeßmodellierung und Workflow-Management.* PhD thesis, Institut für Mathematische Maschinen und Datenverarbeitung, Universität Erlangen-Nürnberg, July 1999.

[Striemer *et al.*, 1997] R. Striemer, M. Weske, and R. Holten. Beschreibung und Analyse von Vorgehensmodellen zur Entwicklung von betrieblichen Workflow-Anwendungen. In *4. Workshop der GI Fachgruppe 5.1.1: Vorgehensmodelle. GMD Studie Nr. 311. ISBN 3-88457-311-X. Berlin*, 1997.

[SUNY, 2000] SUNY. The XSB Programming System. Dept. of Computer Science, SUNY at Stony Brook, 2000. URL: `http://www.cs.sunysb.edu/~sbprolog/xsb-page.html`.

[Tsamardinos *et al.*, 2000] I. Tsamardinos, M. E. Pollack, and J. F. Horty. Merging Plans with Quantitative Temporal Constraints, Temporally Extended Actions, and Conditional Branches. In *Proceedings of the 5th International Conference on AI Planning Systems, Breckenridge, CO*, April 2000.

[van Elst and Abecker, 2001] L. van Elst and A. Abecker. Integrating Task, Role, and User Modeling in Organizational Memories. In *The 14. Int. FLAIRS Conference, Special Track on Knowledge Management, Key West, Florida, USA*, May 2001.

[van Heijst *et al.*, 1996] G. van Heijst, R. van der Spek, and E. Kruizinga. Organizing corporate memories. In *[Dieng and Vanwelkenhuysen, 1996]*, 1996.

[W3C, 2001a] W3C. Resource Description Framework (RDF) Schema Specification 1.0, 2001. URL: `http://www.w3.org/TR/2000/CR-rdf-schema-20000327/`.

[W3C, 2001b] W3C. Semantic Web Activity: Resource Description Framework (RDF), 2001. URL: `http://www.w3.org/RDF/`.

[Wagner, 1998] G. Wagner. *Foundations of Knowledge Systems with Applications to Databases and Agents*. Kluwer Academic Publishers, 1998.

[Wargitsch *et al.*, 1998] C. Wargitsch, T. Wewers, and F. Theisinger. An organizational-memory-based approach for an evolutionary workflow management system – concepts and implementation. In *Proceedings of HICCS'31, Vol. 1*, pages 174–183, 1998.

[Wargitsch, 1997] C. Wargitsch. WorkBrain: Merging Organizational Memory and Workflow Management Systems. Technical report, Bavarian Research Center for Knowledge-Based Systems (FORWISS), 1997.

[Weiss, 1999] G. Weiss, editor. *Multiagent Systems – A Modern Approach to Distributed Artificial Intelligence*. The MIT Press, 1999.

[Wenzel and Maus, 2001] C. Wenzel and H. Maus. Leveraging corporate context within knowledge-based document analysis and understanding. *International Journal on Document Analysis and Recognition, Special Issue on Document Analysis for Office Systems*, 2001. to be published.

[Wenzel, 1998] C. Wenzel. Integrating information extraction into workflow management systems. In *Natural Language and Information Systems Workshop (NLIS/DEXA 98), Vienna, Austria*, 1998.

[WfMC, 1999] WfMC. Terminology and Glossary, issue 3.0. Technical report, Workflow Management Coalition, 1999.

[Whittingham *et al.*, 1998] K. Whittingham, H. Ludwig, and M. Stolze. An alternative approach to business process support. In *CSCW-98 Workshop: Towards Adaptive Workflow Systems*, 1998. URL: `http://ccs.mit.edu/klein/cscw98/`.

[Whittingham, 1999] K. Whittingham. OpenWater White Paper. Technical report, IBM Research Division, Zurich Research Laboratory, 1999.

[Wiig, 1998] K. Wiig. Perspectives on introducing enterprise knowledge management. In *[Reimer, 1998]*, 1998.

[Wooldridge *et al.*, 2000] M. Wooldridge, N. Jennings, and D. Kinny. The gaia methodology for agent-oriented analysis and design. *Journal of Autonomous Agents and Multi-Agent Systems*, 3(3):285–312, 2000.

[WorkflowWorld, 1998] Dolphin leaps into view. *Workflow World*, 6(2), 1998.

Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

DFKI-Bibliothek
Postfach 20 80
D-67608 Kaiserslautern
Germany

Telefon (0631) 205-3506
Telefax (0631) 205-3210

bib@dfki.uni-kl.de
www.dfki.de/dfkibib

Veröffentlichungen des DFKI

Die folgenden DFKI Veröffentlichungen sowie die aktuelle Liste aller bisher erschienener Publikationen können über die URL http://www.dfki.uni-kl.de/dfkidok/publications bezogen werden. Die Berichte werden, wenn nicht anders gekennzeichnet, kostenlos abgegeben.

DFKI Publications

The following DFKI publications and the list of all published papers so far are also available online under http://www.dfki.uni-kl.de/dfkidok/publications The reports are distributed free of charge except otherwise noted.

DFKI Research Reports

2001

RR-01-03
Michael Schillo, Klaus Fischer, Christian Kray
The Eager Bidder Problem
A Fundamental Problem of DAI and Selected Solutions
18 pages

RR-01-02
Detlef Prescher
Inside-Outside Estimation Meets Dynamic EM
—GOLD—
14 pages

2000

RR-00-02
Michael Schillo
Vertrauen und Betrug in Multi-Agenten Systemen
Erweiterung des Vertrauensmodells von Castelfranchi und Falcone um eine Kommunikationskomponente
130 Seiten

1999

RR-99-04
Gero Vierke, Christian Ruß
The Matrix Auction: A Mechanism for the Market-Based Coordination of Enterprise Networks
11 pages

RR-99-03
Christian Gerber, Jörg Siekmann, Gero Vierke
Holonic Multi-Agent Systems
42 pages

RR-99-02
Michael Schillo, Jürgen Lind, Petra Funk, Christian Gerber, Christoph Jung
SIF - The Social Interaction Framework
System Description and User's Guide to a Multi-Agent System Testbed
30 pages

RR-99-01
Jürgen Lind, Stefan Philipps
Ein System zur Definition und Ausführung von Protokollen für Multi-Agentensysteme
61 Seiten

1998

RR-98-04
Bernd Kiefer, Hans-Ulrich Krieger
A Bag of Useful Techniques for Efficient and Robust Parsing
9 pages

RR-98-03
Heiko Mantel
Developing a Matrix Characterization for $\mathcal{MELL}$
59 pages

RR-98-02
Klaus Fischer, Christian Ruß, Gero Vierke
Decision Theory and Coordination in Multiagent Systems
134 pages

RR-98-01
Christoph G. Jung, Klaus Fischer
Methodological Comparison of Agent Models
58 pages

1997

RR-97-08
Stefan Müller
Complement Extraction Lexical Rules and Argument Attraction
14 pages

RR-97-07
Stefan Müller
Yet Another Paper about Partial Verb Phrase Fronting in German
26 pages

RR-97-06
Stefan Müller
Scrambling in German – Extraction into the *Mittelfeld*
24 pages

RR-97-05
Harald Meyer auf'm Hofe
Finding Regions of Local Repair in Hierarchical Constraint Satisfaction
33 pages

RR-97-04
Serge Autexier, Dieter Hutter
Parameterized Abstractions used for Proof-Planning
13 pages

RR-97-03
Dieter Hutter
Using Rippling to Prove the Termination of Algorithms
15 pages

RR-97-02
Stephan Busemann, Thierry Declerck, Abdel Kader Diagne, Luca Dini,
Judith Klein, Sven Schmeier
Natural Language Dialogue Service for Appointment Scheduling Agents
15 pages

RR-97-01
Erica Melis, Claus Sengler
Analogy in Verification of State-Based Specifications: First Results
12 pages

1996

RR-96-06
Claus Sengler
Case Studies of Non-Freely Generated Data Types
200 pages

RR-96-05
Stephan Busemann
Best-First Surface Realization
11 pages

RR-96-04
Christoph G. Jung, Klaus Fischer, Alastair Burt
Multi-Agent Planning
Using an *Abductive*
EVENT CALCULUS
114 pages

RR-96-03
Günter Neumann
Interleaving
Natural Language Parsing and Generation
Through Uniform Processing
51 pages

RR-96-02
E.André, J. Müller , T.Rist:
PPP-Persona: Ein objektorientierter Multimedia-Präsentationsagent
14 Seiten

RR-96-01
Claus Sengler
Induction on Non-Freely Generated Data Types
188 pages

1995

RR-95-20
Hans-Ulrich Krieger
Typed Feature Structures, Definite Equivalences, Greatest Model Semantics, and Nonmonotonicity
27 pages

RR-95-19
Abdel Kader Diagne, Walter Kasper, Hans-Ulrich Krieger
Distributed Parsing With HPSG Grammar
20 pages

RR-95-18
Hans-Ulrich Krieger, Ulrich Schäfer
Efficient Parameterizable Type Expansion for Typed Feature Formalisms
19 pages

RR-95-17
Hans-Ulrich Krieger
Classification and Representation of Types in TDL
17 pages

RR-95-14
Joachim Niehren
Functional Computation as Concurrent Computation
50 pages

RR-95-13
Werner Stephan, Susanne Biundo
Deduction-based Refinement Planning
14 pages

RR-95-12
Walter Hower, Winfried H. Graf
Research in Constraint-Based Layout, Visualization, CAD, and Related Topics: A Bibliographical Survey
33 pages

RR-95-11
Anne Kilger, Wolgang Finkler
Incremental Generation for Real-Time Applications
47 pages

RR-95-10
Gert Smolka
The Oz Programming Model
23 pages

RR-95-09
M. Buchheit, F. M. Donini, W. Nutt, A. Schaerf
A Refined Architecture for Terminological Systems: Terminology = Schema + Views
71 pages

RR-95-08
Michael Mehl, Ralf Scheidhauer, Christian Schulte
An Abstract Machine for Oz
23 pages

RR-95-07
Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, Werner Nutt
The Complexity of Concept Languages
57 pages

RR-95-06
Bernd Kiefer, Thomas Fettig
FEGRAMED
An interactive Graphics Editor for Feature Structures
37 pages

RR-95-05
Rolf Backofen, James Rogers, K. Vijay-Shanker
A First-Order Axiomatization of the Theory of Finite Trees
35 pages

RR-95-04
M. Buchheit, H.-J. Bürckert, B. Hollunder, A. Laux, W. Nutt, M. Wójcik
Task Acquisition with a Description Logic Reasoner
17 pages

RR-95-03
Stephan Baumann, Michael Malburg, Hans-Guenther Hein, Rainer Hoch, Thomas Kieninger, Norbert Kuhn
Document Analysis at DFKI
Part 2: Information Extraction
40 pages

RR-95-02
Majdi Ben Hadj Ali, Frank Fein, Frank Hoenes, Thorsten Jaeger, Achim Weigel
Document Analysis at DFKI
Part 1: Image Analysis and Text Recognition
69 pages

RR-95-01
Klaus Fischer, Jörg P. Müller, Markus Pischel
Cooperative Transportation Scheduling
an application Domain for DAI
31 pages

DFKI Technical Memos

2000

TM-00-01
Jürgen Lind
Specifying Agent Interaction Protocols with UML Activity Diagrams
12 pages

1999

TM-99-04
Christoph Endres
The MultiHttpServer - A Parallel Pull Engine
18 pages

TM-99-03
Jürgen Lind
A Process Model for the Design of Multi-Agent Systems
20 pages

TM-99-02
Hans-Jürgen Bürckert, Petra Funk, Gero Vierke
An Intercompany Dispatch Support System for Intermodal Transport Chains
12 pages

TM-99-01
Matthias Fischmann
The Smes Client/Server Protokol (SMESPR/1.0)
10 pages

1998

TM-98-09
Jürgen Lind
The EMS Model
15 pages

TM-98-08
Michael Schillo, Petra Funk
Spontane Gruppenbildung in künstlichen Gesellschaften
10 Seiten

TM-98-07
Markus Perling
The RAWAM: Relfun-Adapted WAM Emulation in C
49 pages

TM-98-06
Petra Funk, Gero Vierke, Hans-Jürgen Bürckert
A Multi-Agent Perspective on Intermodal Transport Chains
8 pages

TM-98-05
Jürgen Lind, Klaus Fischer
Transportation Scheduling and Simulation in a Railroad scenario: A Multi-Agent Approach
17 pages

TM-98-04
Hans-Jürgen Bürckert, Gero Vierke
Simulated Trading Mechanismen für Speditionsübergreifende Transportplanung
12 pages

TM-98-03
Petra Funk
Fast Loading and Unloading Devices: Planning and Scheduling Requirements
7 pages

TM-98-02
Christian Gerber, Christian Ruß, Gero Vierke
An Empirical Evaluation on the Suitability of Market-Based Mechanisms for Telematics Applications
20 pages

TM-98-01
Christian Gerber
Bottleneck Analysis as a Heuristic for Self-Adaption in Multi-Agent Societies
16 pages

1997

TM-97-03
Hans-Jürgen Bürckert, Klaus Fischer, Gero Vierke
TeleTruck: A Holonic Fleet Management System
10 pages

TM-97-02
Christian Gerber
Scalability of Multi-Agent Systems - Proposal for a Dissertation
49 pages

TM-97-01
Markus Perling
GeneTS: A Relational-Functional Genetic Algorithm for the Traveling Salesman Problem
26 pages

1996

TM-96-02
Harold Boley
Knowledge Bases in the World Wide Web:
A Challenge for Logic Programming
(Second, Revised Edition)
10 pages

TM-96-01
Gerd Kamp, Holger Wache
CTL — a description Logic with expressive concrete domains
19 pages

1995

TM-95-04
Klaus Schmid
Creative Problem Solving
and
Automated Discovery
— An Analysis of Psychological and AI Research —
152 pages

TM-95-03
Andreas Abecker, Harold Boley, Knut Hinkelmann, Holger Wache, Franz Schmalhofer
An Environment for Exploring and Validating Declarative Knowledge
11 pages

TM-95-02
Michael Sintek
FLIP: Functional-plus-Logic Programming
on an Integrated Platform
106 pages

TM-95-01
Martin Buchheit, Rüdiger Klein, Werner Nutt
Constructive Problem Solving: A Model Construction Approach towards Configuration
34 pages

DFKI Documents

2001

D-01-02
Heinz-Jürgen Müller, Andreas Abecker, Knut Hinkelmann, Heiko Maus
Geschäftsprozessorientiertes Wissensmanagement
Workshop im Rahmen der 1. Konferenz Professionelles Wissensmanagement - Erfahrungen und Visionen
Kongresshaus Baden-Baden, 14.-16. März 2001
329 Seiten

Note: This document is available for a nominal charge of 25 DM (or 15 US-$).

D-01-01
Andreas Abecker, Ansgar Bernardi, Ludger van Elst, Andreas Lauer, Heiko Maus, Sven Schwarz, Michael Sintek
FRODO: A Framework for Distributed Organizational Memories
Milestone M1: Requirements Analysis and System Architecture
112 Seiten

2000

D-00-01
Tilman Becker, Stephan Busemann
IMPACTS in Natural Language Generation
NLG Between Technology and Applications
Workshop at Schloss Dagstuhl, Germany
July 26-28, 2000
66 pages

Note: This document is available for a nominal charge of 25 DM (or 15 US-$).

1999

D-99-01
Tilman Becker, Stephan Busemann
May I Speak Freely? Between Templates and Free Choice in Natural Language Generation. Workshop at the 23rd German Annual Conference for Artificial Intelligence (KI '99), Bonn
14.-15. September 1999
69 pages

1998

D-98-03
Stephan Busemann, Karin Harbusch, Stefan Wermter(Hrsg.)
Hybride konnektionistische, statistische und regelbasierte Ansätze zur Verarbeitung natürlicher Sprache
Workshop auf der 21. Deutschen Jahrestagung für Künstliche Intelligenz, Freiburg, 9.–10. September 1997
75 Seiten

D-98-02
Andreas Abecker, Ansgar Bernardi, Knut Hinkelmann , Otto Kühn, Michael Sintek
Techniques for Organizational Memory Information Systems
66 pages

D-98-01
Stephan Baumann, Jürgen Lichter, Michael Malburg, Heiko Maus, Harald Meyer auf'm Hofe, Claudia Wenzel
Architektur für ein System zur Dokumentanalyse im Unternehmenskontext Integration von Datenbeständen, Aufbau- und Ablauforganisation
76 Seiten

1997

D-97-08
Christoph G. Jung, Klaus Fischer, Susanne Schacht
Distributed Cognitive Systems
Proceedings of the VKS'97 Workshop
50 pages

D-97-07
Harold Boley, Bernd Bachmann, Christian Blum, Christian Embacher, Andreas Lorenz, Jamel Zakraoui
PIMaS:
Ein objektorientiert-regelbasiertes System zur Produkt-Prozeß-Transformation
45 Seiten

D-97-06
Tilman Becker, Stephan Busemann, Wolfgang Finkler
DFKI Workshop on Natural Language Generation
67 pages

D-97-05
Stephan Baumann, Majdi Ben Hadj Ali, Jürgen Lichter, Michael Malburg, Harald Meyer auf'm Hofe, Claudia Wenzel
Anforderungen an ein System zur Dokumentanalyse im Unternehmenskontext
— Integration von Datenbeständen, Aufbau- und Ablauforganisation
42 Seiten

D-97-04
Claudia Wenzel, Markus Junker
Entwurf einer Patternbeschreibungssprache
für die Informationsextraktion
in der Dokumentanalyse
24 Seiten

D-97-03
Andreas Abecker, Stefan Decker, Knut Hinkelmann, Ulrich Reimer
Proceedings of the Workshop „Knowledge-Based Systems for Knowledge Management in Enterprises" 97
167 pages

D-97-02
Tilman Becker, Hans-Ulrich Krieger
Proceedings of the Fifth Meeting on Mathematics of Language (MOL5)
168 pages

Note: This document is available for a nominal charge of 25 DM (or 15 US-$).

D-97-01
Thomas Malik
NetGLTool Benutzeranleitung
40 Seiten

1996

D-96-07
Technical Staff
DFKI Jahresbericht 1995
55 Seiten

Note: This document is no longer available in printed form.

D-96-06
Klaus Fischer (Ed.)
Working Notes of the KI'96 Workshop on Agent-Oriented Programming and Distributed Systems
63 pages

D-96-05
Martin Schaaf
Ein Framework zur Erstellung verteilter Anwendungen
94 pages

D-96-04
Franz Baader, Hans-Jürgen Bürckert, Andreas Günter, Werner Nutt (Hrsg.)
Proceedings of the Workshop on Knowledge Representation and Configuration WRKP'96
83 pages

D-96-03
Winfried Tautges
Der DESIGN-ANALYZER - Decision Support im Designprozess
75 Seiten

D-96-01
Klaus Fischer, Darius Schier
Ein Multiagentenansatz zum Lösen von Fleet-Scheduling-Problemen
72 Seiten

1995

D-95-12
F. Baader, M. Buchheit, M. A. Jeusfeld, W. Nutt (Eds.)
Working Notes of the KI'95 Workshop:
KRDB-95 - Reasoning about Structured Objects: Knowledge Representation Meets Databases
61 pages

D-95-11
Stephan Busemann, Iris Merget
Eine Untersuchung kommerzieller Terminverwaltungssoftware im Hinblick auf die Kopplung mit natürlichsprachlichen Systemen
32 Seiten

D-95-10
Volker Ehresmann
Integration ressourcen-orientierter Techniken in das wissensbasierte Konfigurierungssystem TOOCON
108 Seiten

D-95-09
Antonio Krüger
PROXIMA: Ein System zur Generierung graphischer Abstraktionen
120 Seiten

D-95-08
Technical Staff
DFKI Jahresbericht 1994
63 Seiten

Note: This document is no longer available in printed form.

D-95-07
Ottmar Lutzy
Morphic - Plus
Ein morphologisches Analyseprogramm für die deutsche Flexionsmorphologie und Komposita-Analyse
74 Seiten

D-95-06
Markus Steffens, Ansgar Bernardi
Integriertes Produktmodell für Behälter aus Faserverbundwerkstoffen
48 Seiten

D-95-05
Georg Schneider
Eine Werkbank zur Erzeugung von 3D-Illustrationen
157 Seiten

D-95-04
Victoria Hall
Integration von Sorten als ausgezeichnete taxonomische Prädikate in eine relational-funktionale Sprache
56 Seiten

D-95-03
Christoph Endres, Lars Klein, Markus Meyer
Implementierung und Erweiterung der Sprache $\mathcal{ALCP}$
110 Seiten

D-95-02
Andreas Butz
BETTY
Ein System zur Planung und Generierung informativer Animationssequenzen
95 Seiten

FRODO: A Framework for Distributed Organizational Memories

Milestone M1: Requirements Analysis and System Architecture

A. Abecker, A. Bernardi, L. van Elst,
A. Lauer, H. Maus, S. Schwarz, M. Sintek

D-01-01

Document