Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

Document
D-98-01

Architektur für ein System zur Dokumentanalyse im Unternehmenskontext

—

Integration von Datenbeständen, Aufbau- und Ablauforganisation

Stephan Baumann, Jürgen Lichter, Michael Malburg, Heiko Maus, Harald Meyer auf'm Hofe, Claudia Wenzel

April 1998

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH

Postfach 20 80
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 (681) 302-5252
Fax: + 49 (681) 302-5341
E-Mail: info@dfki.de

WWW: http://www.dfki.de

Deutsches Forschungszentrum für Künstliche Intelligenz

DFKI GmbH

German Research Center for Artificial Intelligence

Founded in 1988, DFKI today is one of the largest nonprofit 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 commercialization.

Based in Kaiserslautern and Saarbrücken, the German Research Center for Artificial Intelligence ranks among the important "Centers of Excellence" worldwide.

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 115 full-time employees, including 95 research scientists with advanced degrees. There are also around 120 part-time research assistants.

Revenues for DFKI were about 24 million DM in 1997, half from government contract work and half from commercial clients. The annual increase in contracts from commercial clients was greater than 37% during the last three years.

At DFKI, all work is organized 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 six research departments are directed by internationally recognized research scientists:

- Information Management and Document Analysis (Director: Prof. A. Dengel)
- Intelligent Visualization and Simulation Systems (Director: Prof. H. Hagen)
- Deduction and Multiagent Systems (Director: Prof. J. Siekmann)
- Programming Systems (Director: Prof. G. Smolka)
- Language Technology (Director: Prof. H. Uszkoreit)
- Intelligent User Interfaces (Director: Prof. W. Wahlster)

In this series, DFKI publishes research reports, technical memos, documents (eg. 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

Architektur für ein System zur Dokumentanalyse im Unternehmenskontext — Integration von Datenbeständen, Aufbau- und Ablauforganisation

**Stephan Baumann, Jürgen Lichter, Michael Malburg,
Heiko Maus, Harald Meyer auf'm Hofe, Claudia Wenzel**

DFKI-D-98-01

This work has been supported by a grant from The Federal Ministry of Education, Science, Research, and Technology (FKZ ITW-9702).

ISSN 0946-0098

Architektur für ein System zur Dokumentanalyse im Unternehmenskontext — Integration von Datenbeständen, Aufbau- und Ablauforganisation

Stephan Baumann, Jürgen Lichter, Michael Malburg,
Heiko Maus, Harald Meyer auf'm Hofe, Claudia Wenzel

Februar 1998

Zusammenfassung

Workflowmanagementsysteme werden im Bürobereich verstärkt zur effizienten Geschäftsprozeßabwicklung eingesetzt. Das bereits Mitte der 70er Jahre propagierte papierlose Büro bleibt jedoch gegenwärtig immer noch Utopie. Dieser Widerspruch liegt darin begründet, daß die Handhabung von papierintensiven Vorgängen in hohem Maße abhängig ist von einer Identifkation und Aufbereitung der in den Dokumenten enthaltenen Informationen. Allerdings müssen solche Daten z.B. bei eingehender Post immer noch von Hand eingegeben werden.

In diesem Dokument wird die Architektur eines System vorgestellt, das diesen Medienbruch überwinden soll. Techniken aus dem Gebiet der Dokumentanalyse und des Dokumentverstehens werden in den Workflowkontext integriert und nutzen das dort verfügbare Wissen zur Steigerung der Erkennungsqualität.

Das Architekturdokument beruht auf einer ebenfalls dokumentierten Anforderungsanalyse (DFKI Dokument D-97-05). Es enthält eine statische und eine dynamische Beschreibung der benötigten Klassenkategorien und erklärt deren Funktionalität anhand eines umfassenden Beispiels.

Inhaltsverzeichnis

1 Allgemeines 1
1.1 Abkürzungen 1
1.2 Definitionen 1
1.3 Referenzierte Dokumente 4
2 Einleitung 5
3 Datenflußmodelle 7
3.1 Konfigurationsphase 7
3.2 Abarbeitung mehrerer DAU-Aktivitäten 10
3.3 Verifikations- und Akquisitionsphase 13
4 Ein Beispiel 15
4.1 Beispieldokument und -umgebung 15
4.1.1 Dokumenteigenschaften 15
4.1.2 Firmenwissen 15
4.1.3 Dokumentwissen 17
4.1.4 Strategisches Wissen 18
4.1.5 Erwartungshaltung und DAU-Aktivitätsauftrag 20
4.1.6 Workflow-Definition 21
4.1.6.1 Verwendete Datenstrukturen 24
4.1.6.2 Besonderheiten 24
4.2 Konfigurationsphase 25
4.3 Abarbeitung des Beispieldokumentes 28
4.4 Verifikations- und Akquisitionsphase 31
4.4.1 Verifikation 31
4.4.2 Akquisition 32
5 Klassenkategorien 33
5.1 Übersicht 33
5.1.1 Klassenkategorien des WFMS 34
5.1.2 Klassenkategorien des DAU-Systems 34
5.2 Aus Systemanforderungen abgeleitete Klassenkategorien 35
5.2.1 Workflow-Definition 35
5.2.1.1 Methode: CreateNewWorkflowDefinition 36
5.2.2 Workflow Engine 36
5.2.2.1 Methode: StartProcess 36
5.2.2.2 Methode: StartEngine 37
5.2.2.3 Methode: StopEngine 37
5.2.2.4 Methode: EnactActivity 37
5.2.3 Client Application 37
5.2.3.1 Methode: StartProcess 38
5.2.3.2 Methode: WorkProcess 38
5.2.4 Konfigurationskomponente 38
5.2.4.1 Methode: ViewConcepts 39
5.2.4.2 Methode: AddConcept 39
5.2.4.3 Methode: AddConstraint 39
5.2.4.4 Methode: AddConfigurationManually 40
5.2.4.5 Methode: AddConfigurationAutomatically 40
5.2.4.6 Methode: AddSubsumptionRelation 40
5.2.4.7 Methode: AddPartWholeRelation 41
5.2.4.8 Methode: ModifyAttribute 41

5.2.4.9 Methode: AddEquivalency ... 41
5.2.4.10 Methode: TransformDocConcept ... 42
5.2.4.11 Methode: NormalizeKB ... 42
5.2.5 DAU-Monitor ... 42
5.2.5.1 Methode: ViewSystemConfiguration ... 42
5.2.5.2 Methode: ChangeSystemConfiguration ... 43
5.2.6 Verifikations-GUI ... 43
5.2.6.1 Methode: VerifyResults ... 43
5.2.7 Akquisitions-GUI ... 44
5.2.7.1 Methode: CheckAll ... 44
5.2.7.2 Methode: AcquireLexicalKnowledge ... 45
5.2.7.3 Methode: AcquirePatterns ... 45
5.2.7.4 Methode: AcquireNewClass ... 45
5.2.7.5 Methode: AcquireNewGraphicalDocumentFeatures ... 46
5.2.7.6 Methode: AcquireNewData ... 46
5.2.8 DAU-Spezialisten ... 46
5.2.8.1 Bildverarbeitung I – Bildaufbereitung und Merkmalsextraktion ... 47
5.2.8.2 Logoerkennung ... 47
5.2.8.3 Segmentierung ... 47
5.2.8.4 Bildverarbeitung II – Aufbereitung für die OCR ... 48
5.2.8.5 Formularerkennung ... 48
5.2.8.6 OCR ... 48
5.2.8.7 OCR-Voting ... 48
5.2.8.8 Lexikalischer Abgleich ... 49
5.2.8.9 Worttagging/Morphologie ... 49
5.2.8.10 Strukturparser E-Mail ... 49
5.2.8.11 Strukturparser HTML ... 49
5.2.8.12 Labeling für bestimmte Objekte ... 50
5.2.8.13 Informationsextraktion mittels Parsing ... 50
5.2.8.14 Informationsextraktion mittels Pattern-Matching ... 50
5.2.8.15 Regelbasierte Dokumentklassifikation ... 51
5.2.8.16 Tabellenanalyse ... 51
5.3 Zusätzliche relevante Klassenkategorien ... 51
5.3.1 Reaktiver Planer ... 51
5.3.1.1 Methode: InitializePlanning ... 52
5.3.1.2 Methode: GeneratePlan ... 52
5.3.1.3 Methode: PlanAndRun ... 52
5.3.1.4 Methode: CallSpecialist ... 53
5.3.1.5 Methode: TerminatePlanning ... 53
5.3.2 Workflow-Modul ... 53
5.3.2.1 Methode: QueryErwartungshaltung ... 55
5.3.2.2 Methode: DeleteErwartungshaltung ... 55
5.3.2.3 Methode: ContactVermittler ... 55
5.3.2.4 Methode: RequestDAUTask ... 55
5.3.2.5 Methode: EndDAUSignal ... 56
5.3.3 DAU/WFMS-Vermittler ... 56
5.3.3.1 Methode: SetErwartungshaltung ... 58
5.3.3.2 Methode: DeleteErwartungshaltung ... 58
5.3.3.3 Methode: RequestDAUTask ... 58
5.3.3.4 Methode: SendDAUEndSignal ... 58
5.3.3.5 Methode: DeliverExpectedDocument ... 59
5.3.3.6 Methode: DeliverDAUTaskResult ... 59
5.3.3.7 Methode: StartVerifyingProcess ... 59
5.3.4 Postkorbbearbeitung ... 59
5.3.4.1 Methode: GetDocument ... 59

5.3.4.2 Methode: StartPlaner 60
5.3.5 Dokumentwissensrepräsentation 60
5.3.5.1 Kommunikation mit anderen Klassenkategorien 61
5.3.5.2 Kommunikationsschnittstelle 62
5.3.6 Zwischenablage 65
5.3.6.1 Methode: InitClipboard 65
5.3.6.2 Methode: SaveClipboard 65
5.3.6.3 Methode: LoadClipboard 65
5.3.6.4 Methode: SetBackupDevice 65
5.3.6.5 Methode: SetSwapDevice 66
5.3.6.6 Methode: LoadTIFFImage 66
5.3.6.7 Methode: SaveTIFFImage 66
5.3.6.8 Methode: LoadOCRResults 66
5.3.6.9 Methode: SaveOCRResults 67
5.3.6.10 Weitere Methoden 67

1 Allgemeines

1.1 Abkürzungen

API: Application Programming Interface

DAU: Document Analysis and Understanding

HTML: Hypertext Markup Language

OCR: Optical Character Recognition

ROI: Region of Interest

TIFF: Tagged Image File Format

WfMC: Workflow Management Coalition

WFMS: Workflow Management System

1.2 Definitionen

Aktivität: Hierbei kann es sich in Abhängigkeit vom Kontext sowohl um eine Workflow-Aktivität als auch eine DAU-Aktivität handeln.

Business process: siehe *Geschäftsprozeß*.

Corporate knowledge: siehe *Firmenwissen*.

DAU-Aktion: Eine elementare, nicht teilbare Aufgabe, die durch Kompetenz/Rollenvergaben in der Aufbauorganisation (oder im Strategiewissen der DAU-Konfigurationseinheit) potentiellen DAU-Spezialisten zugeordnet ist, da sich für diese Aufgabe der Einsatz von DAU-Verfahren anbietet.

DAU-Aktivität: Eine DAU-Tätigkeit, die aus einer oder mehreren DAU-Aktionen bestehen kann und die in der Workflow-Definition verwendet werden kann.

Document knowledge: siehe *Dokumentwissen*.

Dokumentverifikation: Eine Workflow-Aktivität zur Verifikation der Ergebnisse der Dokumentanalyse (bzw. des DAU-Systems) durch einen Sachbearbeiter. Dabei muß über die übliche Rollenzuordnung die notwendige Kompetenz (z. B. Posteingangserfassung, Formularfeldverifikation) für diese Aufgabe sichergestellt werden.

Dokumentwissen (engl.: *document knowledge*) bezeichnet sowohl strukturelles als auch relationales Wissen über die Dokumente der Domäne (z. B. Layout-Objekte, Logik-Objekte, Inhalte).

Firmenwissen (engl.: *corporate knowledge*) bezeichnet Wissen über die Geschäftsprozesse und Datenstämme einer Firma (z. B. Aufbauorganisation, Rollen, Kunden, Produkte).

Geschäftsprozeß (engl.: *business process*): Der Geschäftsprozeß ist als höchste organisatorische Ebene für die Dauer einer Geschäftsabwicklung mit einem Kunden oder Lieferanten zu verstehen. Grundsätzlich läßt sich zwischen wertschöpfenden und nicht wertschöpfenden Prozessen unterscheiden. Erstere sind entscheidend für den Geschäftserfolg, da sie mit Umsätzen und Erlösen unmittelbar zusammenhängen. Nicht wertschöpfende Geschäftsprozesse sind indirekt für den Geschäftserfolg wirksam, da sie lediglich der internen Verwaltung dienen. Geschäftsprozesse enthalten sowohl manuelle als auch automatisierbare, z. B. rechnergestützte, Prozeßanteile.

Wird im folgendem Dokument von Geschäftsprozessen gesprochen, ist der reale Vorgang gemeint. Im Gegensatz dazu ist der Workflow die Abbildung des Geschäftsprozesses im WFMS.

Off-Line-Learning: Hierunter wird eine Form des Lernens verstanden, die in der Konfigurationsphase stattfindet, auf größeren Dokumentmengen arbeitet und für das Füllen der Wissensbasen mancher Spezialisten unbedingt notwendig ist.

Online-Teaching: Eine Workflow-Aktivität, die im Bedarfsfall (Versagen der automatischen Dokumentanalyse) ausschließlich von einem geschulten Sachbearbeiterkreis zum Zwecke der Akquisition bzw. Verfeinerung von Dokumentanalysewissen durchgeführt wird.

ROI/Region of Interest: Ein Begriff aus der Dokumentanalyse, der auf einem Dokumentbild die (rechteckigen) Bereiche kennzeichnet, die für die weitere Analyse von Interesse sind, also z. B. den Absender. Eine ROI kann z. B. durch ihre Koordinaten spezifiziert werden.

Specialist knowledge: siehe *Spezialistenwissen.*

Spezialist: Im vorliegenden Dokument werden unter dem Begriff Spezialist immer Dokumentanalysemodule (z. B. Segmentierung, Parser, ...) verstanden. Dabei ist es vom Kontext abhängig, ob es sich um einen generischen Spezialisten mit mehreren Wissensbasen für verschiedene Aufgaben handelt oder ob es sich um einen instantiierten Spezialisten mit genau einer Wissenbasis für die Lösung einer spezifischen Aufgabe handelt.

Spezialistenwissen (engl.: *specialist knowledge*) bezeichnet das prozedurale Wissen im Algorithmus des Spezialisten als auch das deklarative Wissen des Spezialisten (z. B. Grammatikregeln, Parameter).

Standarderwartung: Als Standarderwartung bezeichnen wir Dokumente, die jederzeit und unaufgefordert eintreffen können, in unserer Anwendungsdomäne z. B. eine Werbung oder eine Mahnung.

Strategic knowledge: siehe *Strategisches Wissen.*

Strategisches Wissen (engl.: *strategic knowledge*) beschreibt den Analyseablauf (z. B. auf welchen Daten und in welchem Kontext geschieht der nächste Analyseschritt).

System: Im folgenden Dokument versteht man unter dem Begriff „System" das zu entwickelnde Gesamtsystem, das aus zwei Subsystemen besteht. Hierbei handelt es sich zum einen um ein kommerzielles WFMS, das erweitert und spezifisch adaptiert werden muß, daneben wird ein zu entwickelndes DAU-System beschrieben.

WFMS/Workflow Management System: Ein kommerzielles Workflow-System (optional inklusive einer Geschäftsprozeß-/Workflow-Modellierungskomponente), das hinsichtlich der in diesem Dokument spezifizierten Anforderungen erweitert wird.

Workflow-Aktivität: Ein logischer Schritt eines Workflows. Eine Aktivität ist die kleinste Einheit, die durch eine Workflow Engine während der Prozeßausführung eingeplant werden kann. Innerhalb einer Aktivität können einem Benutzer aber auch mehrere Aufgaben aufgetragen werden.

Workflow[prozeß][Definition]: Der Teil des Geschäftsprozesses, der aus einem Netzwerk automatisierbarer Tätigkeitsfolgen (Workflow-Aktivitäten) besteht. Dieser Begriff ist synonym zum Begriff Workflow-Spezifikation, der im Anforderungsdokument verwendet wurde.

Workflow[prozeß]instanz: Die zur Laufzeit von einem WFMS verwaltete Instanz eines Workflows, die sich aus dem zugehörigen generischen Workflow durch die konkrete Variablenbelegung ergibt.

1.3 Referenzierte Dokumente

[1] Systemanforderungen Virtual Office: DFKI Document D 97-05

[2] Proposal Virtual Office: DFKI internes Dokument /project/vo/BMBF/Proposal/VO.Buch

[3] Alan D. Davis: Software Requirements, PTR Prentice-Hall Inc., Englewood Cliffs, 1993

[4] G. Booch: Object-oriented analysis and design with applications. Second edition, The Benjamin/Cummings Publishing Company, Redwood City, 1994

[5] Workflow Management Coalition: Terminology & Glossary, WfMC-TC-1011, Version 2.0, 1996, http://www.aiim.org/wfmc/

2 Einleitung

Dieses Dokument beschreibt die Architektur für ein System, dessen Anforderungen in [1] beschrieben wurden. Es setzt die Lektüre von [1] und [2] voraus.

Das beschriebene System besteht aus zwei Subsystemen. Hierbei handelt es sich zum einen um ein kommerzielles WFMS, das erweitert und spezifisch adaptiert werden muß, daneben wird ein zu entwickelndes DAU-System beschrieben.

Das Dokument ist nach fortschreitendem Detaillierungsgrad aufgebaut. Kapitel 3 beinhaltet eine vergleichsweise oberflächliche Beschreibung der Phasen, in denen sich das System befinden kann. Dieses Kapitel stellt dadurch eine dynamisch-logische Gesamtsicht dar. In Kapitel 4 werden die drei Phasen mithilfe eines durchgängigen Beispiels in einer feineren Granularität beschrieben. Hier wird bereits häufig auf die dafür relevanten Klassenkategorien Bezug genommen. Schließlich beschreibt Kapitel 5 alle Klassenkategorien. Dies geschieht zuerst in Form einer Übersicht, danach folgen genauere Erklärungen mit einer Aufzählung der benötigten Methoden. Die Klassenkategorien sind aufgeteilt in solche, die aus den Systemanforderungen direkt abgeleitet werden konnten und in diejenigen, die sich zusätzlich bei der Verfeinerung des Entwurfs herauskristallisiert haben. Die Beschreibung der Klassenkategorien stellt die statisch-logische Sicht auf die Systemarchitektur dar.

3 Datenflußmodelle

Die folgende Darstellung (Abbildung 1) der Systemarchitektur basiert auf der Verwendung von Datenflußmodellen [3]. Diese Modelle wurden gewählt, um dem phasenorientierten Entwurf Rechnung zu tragen. Hierbei wird auf eine explizite Darstellung des Kontrollflusses verzichtet.

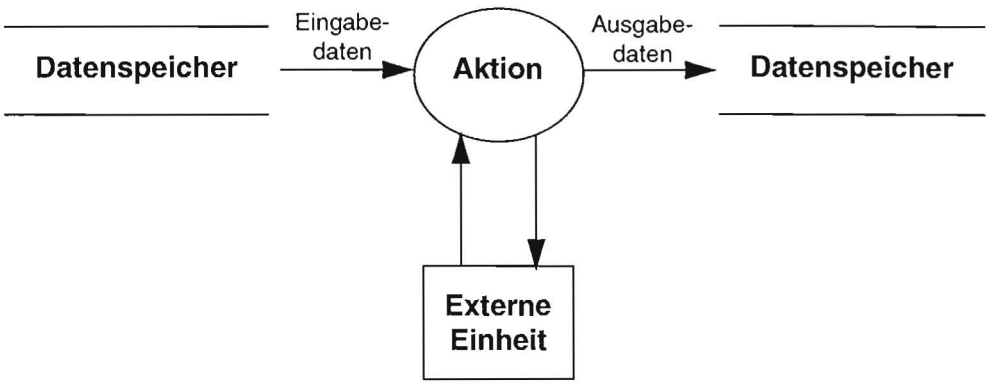

Abbildung 1: Symbolik von Datenflußmodellen

Bei den verwendeten Modellkonstrukten handelt es sich um Datenspeicher, auf die lesend oder schreibend zugegriffen werden kann. Entsprechende Pfeilkonstrukte können mit den konkreten Datenbezeichnern attributiert werden.

3.1 Konfigurationsphase

Die zur Konfigurationsphase benötigten Daten sind das initiale *Firmenwissen,* das initiale *Dokumentwissen* und das *strategische Wissen* (vgl. in Abbildung 2 oben). Alle diese „initialen" Datenquellen können auch leer sein. Da aber bereits existente Datenquellen in das System integriert werden sollen, kann man davon ausgehen, daß in der Regel auch solche Quellen vorhanden sind. Diese liegen dann für die Belange der Systemarchitektur systemextern vor.

Das *Firmenwissen* beinhaltet Geschäftsprozesse (und evtl. auch schon die Operationalisierung in Form einer Workflow-Definition) und Datenstämme einer Firma. Das initiale *Dokumentwissen* enthält strukturelles und relationales Wissen über die Dokumente der Domäne in eher allgemeiner Form.

Strategisches Wissen ist bereits im Rahmen der (deklarativen) DAU-Implementierung gegeben. Dort wird beschrieben, welche Spezialisten existieren, unter welchen Umständen Spezialisten arbeiten können, und welche Ergebnisse jeweils zu erwarten sind.

Durch die Konfiguration werden diese Datenquellen jeweils vergrößert (ebenfalls Abbildung 2).

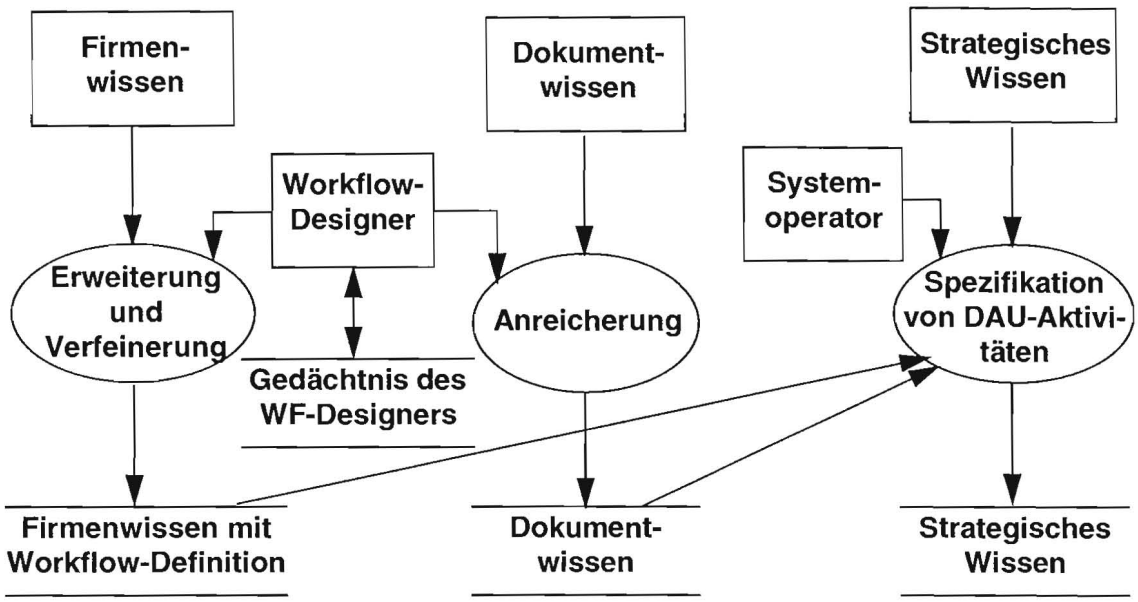

Abbildung 2: Datenflußdiagramm für die Konfigurationsphase

Das *Firmenwissen mit Workflow-Definition* (vgl. Abschnitt 5.2.2.2 „Schnittstelle Workflow-Anreicherung" im Dokument „Systemanforderungen" [1]) beinhaltet nach der Konfiguration über den reinen Workflow hinaus alle DAU-spezifischen Informationen über die Domäne. Die DAU wird explizit in Aktivitäten eingebunden bzw. von dort aus angesteuert.

Das *Dokumentwissen* wird dahingehend erweitert, daß möglicherweise konkretere Informationen über Briefe bestimmter Lieferanten eingefügt werden.

Das *strategische Wissen* wird erweitert um eine Aufstellung von sinnvollen Zuordnungen zwischen Workflow-Aktivitäten und DAU-Konfigurationen, die für die gegebene Gesamtsituation (bestehend aus Anwendungsdomäne, Workflow und DAU-Spezialisten)

in Frage kommen. Sie stellen für die Einsatzphase des Systems also das Inventar an möglichen DAU-Aufrufen im Workflow-Kontext dar.

Die Konfigurationsphase des Systems beinhaltet drei Aktivitäten, die von zwei unterschiedlichen Personen (Rollen) durchgeführt werden: Die Verfeinerung und Erweiterung des Workflows und die Anreicherung des Dokumentwissens werden vom Workflow-Designer übernommen, die Spezifikation von DAU-Aktivitäten vom Systemoperator (vgl. Abschnitt 6.1.2 „Konfigurationsphase" im Dokument „Systemanforderungen" [1]).

Bei der Verfeinerung und Erweiterung des Firmenwissens wird der bereits existente Geschäftsprozeß bzw. die Workflow-Definition aus dem Firmenwissen vom Workflow-Designer um relevante Wissensportionen erweitert. Außerdem kann der Workflow-Designer auch die Datenstämme bzgl. Lieferanten und Produkten erweitern. Das resultierende Firmenwissen mit Workflow-Definition wird später zur Laufzeit bei der Analyse durch die Planungskomponente verwendet und kann ggf. in der Akquisitions- und Verifikationsphase erweitert werden.

Bei der Anreicherung des Dokumentwissens fügt der Workflow-Designer neue, relevante Wissensportionen ein (s.o.).

Die Spezifikation von DAU-Aktivitäten wird durch den (DAU-) Systemoperator durchgeführt, nachdem der Workflow-Designer die beiden anderen Aktivitäten beendet hat. Neben dem erweiterten Firmenwissen mit der Workflow-Definition und dem bereits angereicherten Dokumentwissen kann hierbei das strategische Wissen über die Spezialisten verwendet werden. Der Systemoperator spezifiziert für alle möglichen Workflow-Aktivitäten mindestens eine Konfiguration von (beliebig vielen) DAU-Aktionen, welche eine spätere Realisierung erlauben. Das heißt, daß bereits existente, generische Definitionen von DAU-Aktivitäten den speziellen Anforderungen der aktuellen Anwendung angepaßt werden. Dies geschieht durch Spezialisierung. Die Notwendigkeit anwendungsspezifischer Erweiterungen der generischen Teile des strategischen Wissens in der Konfigurationsphase stellen die Motivation für eine objektzentrierte Darstellung dieses Wissens dar. Die neu entstandene Spezifikation wird dann wieder im strategischen Wissen abgelegt, das später zur Laufzeit von der Planungskomponente verwendet wird.

3.2 Abarbeitung mehrerer DAU-Aktivitäten

Die Initiierung der Analyse ist im allgemeinen abhängig von der in Abschnitt 3.1 „Konfigurationsphase" beschriebenen Konfiguration. Dies bedeutet, daß die Analyse aus dem WFMS heraus gestartet wird. Die Postkorbbearbeitung überprüft periodisch den Eingangspostkorb nach neu eingetroffenen Dokumenten. Das eingetroffene Dokument gelangt als TIFF-Bild in die Zwischenablage und eine Instanz des sog. reaktiven Planers wird diesem Dokument eineindeutig zugeordnet. Das heißt: Pro eingegangenem Dokument gibt es genau einen solchen Steuerungsprozeß, der die Analyse des Dokuments plant und kontrolliert.

Der reaktive Planer wird auf zwei Arten aktiv: erstens durch das Eintreffen eines neuen Dokumentes und der damit verbundenen Vorgangszuordnung und zweitens durch Aufträge, die durch eine Workflow-Instanz generiert wurden, die weitere DAU-Aktivitäten auf dem Dokument ausgeführt haben will.

Eine DAU-Aktivität wird unter Zugriff auf das *Dokumentwissen,* das *strategische Wissen*, das *Firmenwissen mit Workflow-Definition* und die *Erwartungshaltung* abgearbeitet. Die *Erwartungshaltung* wird durch ablaufende Geschäftsprozesse mit Daten zu den erwarteten Dokumenten gefüllt. So wird z. B. in einem Geschäftsprozeß „Bestellung" eine Erwartungshaltung „Rechnung erkennen" oder eine Erwartungshaltung „Auftragsbestätigung erkennen" generiert. Diese Erwartungshaltungen beziehen sich immer auf eine DAU-Aktivität, die in der Konfigurationsphase spezifiziert wurde. In der Konfigurationsphase müssen also die DAU-Aktivitäten „Rechnung erkennen" und „Auftragsbestätigung erkennen" spezifiziert worden sein. Desweiteren befinden sich in der Erwartungshaltung Angaben aus dem *corporate knowledge*, welche Relevanz für die DAU-Aktivität besitzen. Im Beispiel kann dazu etwa der Name der Firma gehören, an welche die Bestellung geschickt wurde. Außerdem gehören die Standarderwartungen zur *Erwartungshaltung*, die z. B. bei nicht eindeutigen Ergebnissen der Dokumentanalyse verwendet werden. Die Erwartungshaltung wird durch DAU/WFMS-Vermittler mit den benötigten Informationen gefüllt. Diese Vermittler werden durch Workflow-Aktivitäten gestartet, die bestimmte Dokumente erwarten.

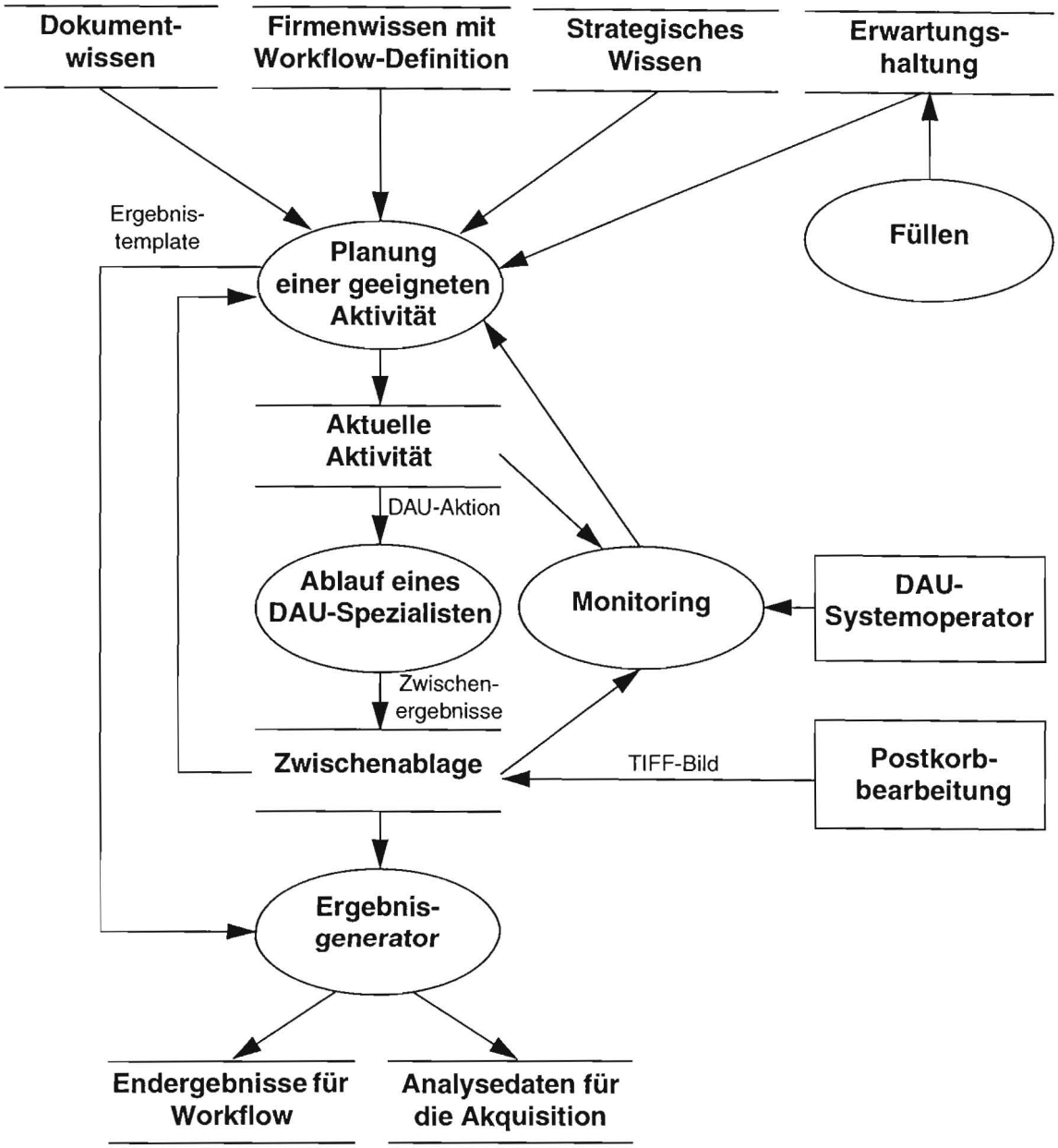

Abbildung 3: Datenflußdiagramm für die Analysephase

Planung einer geeigneten Aktivität heißt nun, daß der reaktive Planer eine Folge von (bedingten) DAU-Aktionen plant, die geeignet ist, alle in der *Erwartungshaltung* benannten DAU-Aktivitäten zu erfüllen. Dieser Plan wird zunächst als *aktuelle Aktivität* gespeichert. Die Komponente zum *Ablauf eines DAU-Spezialisten* wählt nun nacheinander die als nächstes auszuführende DAU-Aktion aus und startet den hier spezifizierten DAU-Spezialisten mit den angegebenen Parametern. Die DAU-Spezialisten erhalten

dabei immer einen Zugriff auf das Dokumentwissen. Die von ihnen erzeugten Zwischenergebnisse legen sie in der *Zwischenablage* ab. Die Endergebnisse der Aktivität werden zum einen im Speicher *Endergebnisse für Workflow* und zum anderen im Speicher *Analysedaten für die Akquisition* abgelegt.

Nach Ablauf des DAU-Spezialisten findet erneut die Operation *Planung einer geeigneten Aktivität* statt, falls das System ein unerwartetes Verhalten zeigt. Ansonsten wird die nächste DAU-Aktion durchgeführt, wenn die DAU-Aktivität noch nicht abgeschlossen wurde. Ist der Plan jedoch (erfolgreich) abgearbeitet, so wird dem *Ergebnisgenerator* durch das *Ergebnistemplate* mitgeteilt, welche Teile der *Zwischenablage* das erwünschte Ergebnis darstellen.

Es ist Teil der Spezifikation einer Aktivität, welche Zwischenergebnisse der Dokumentanalyse für die eventuell nachfolgende Akquisition von Dokumentwissen durch Einschreiben in den Speicher *Analysedaten für die Akquisition* persistent gemacht werden müssen (vgl. Abschnitt 6.1.3 „Analyse neu eingescannter Dokumente zur Laufzeit des WFMS" im Dokument „Systemanforderungen" [1]). Auch diese Anforderungen werden im *Ergebnistemplate* berücksichtigt. Anhand dieser Angaben schreibt der Ergebnisgenerator die erwünschten Daten in die Speicher *Endergebnisse für Workflow* und *Analysedaten für die Akquisition* ein.

Danach wendet sich der reaktive Planer an das Workflow-Modul und meldet das Vorliegen der Daten. Das Workflow-Modul im reaktiven Planer wendet sich über einen DAU/WFMS-Vermittler an das WFMS und initiiert ein Ereignis und/oder übergibt der wartenden Workflow-Aktivität bzw. dem wartenden Prozeß die verlangten Daten.

Der reaktive Planer wird durch ein spezielles Signal des Workflows (via Vermittler) terminiert, welches besagt, daß die Analyse „im Ganzen" beendet ist und keine Zwischenergebnisse mehr aufbewahrt werden müssen. Zu diesem Zeitpunkt sorgt der reaktive Planer dafür, daß die beiden persistenten Datenspeicher „Endergebnisse für den Workflow" und „Analysedaten für die Akquisition" – wie bereits erwähnt – erzeugt werden.

Über die Operation *Monitoring* hat der *DAU-Systemoperator* Zugriff auf Darstellungen der *aktuellen Aktivität* und der erbrachten Zwischenergebnisse der DAU-Spezialisten aus

der *Zwischenablage*. Anhand dieser Angaben kann er zu jeder Zeit durch Eingriff in die *Planung geeigneter Aktivität* die *aktuelle Aktivität* beeinflussen bzw. spezifizieren.

3.3 Verifikations- und Akquisitionsphase

Alle in Abbildung 4 dargestellten und in dieser Phase relevanten Datenspeicher wurden bereits in den vorhergehenden Kapiteln beschrieben.

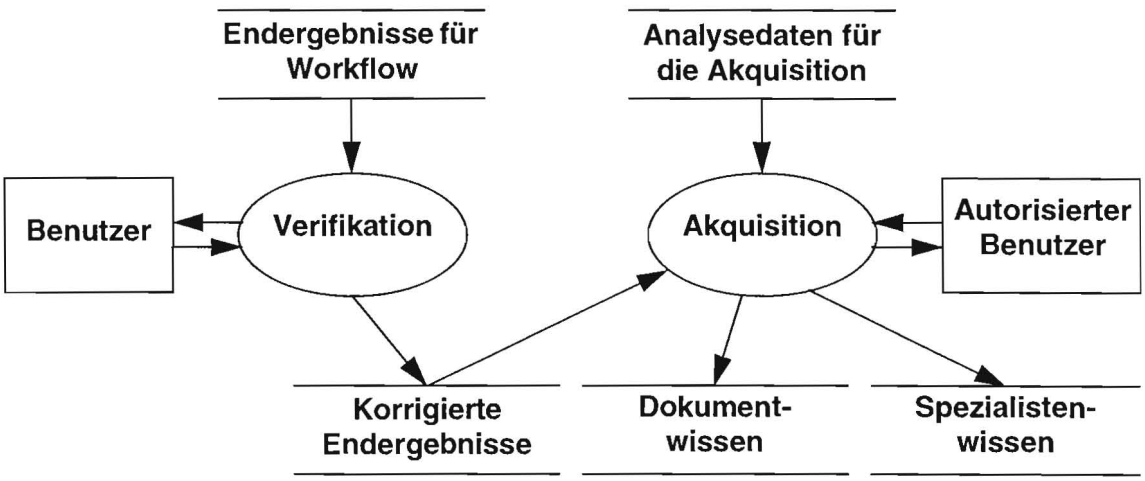

Abbildung 4: Datenflußdiagramm für Verifikations- und Akquisitionsphase

Abbildung 4 beschreibt 2 Aktionen: die *Verifikation* (vgl. Abschnitt 6.1.4 „Verifikation analysierter Dokumente durch den Benutzer" im Dokument „Systemanforderungen" [1]) und die *Akquisition* (vgl. Abschnitt 6.1.5 „Aktualisieren von Workflow- und Dokumentdaten durch den autorisierten Benutzer" im Dokument „Systemanforderungen" [1]). Der Zeitpunkt der Ausführung dieser Aktionen wird durch die dazugehörige Workflow-Definition bestimmt, so daß die beiden Aktionen auch zeitlich voneinander entkoppelt stattfinden können, wobei jedoch die Verifikation stets vor der Akquisition erfolgen muß. Die Verifikation, die vom einfachen Benutzer durchgeführt wird, findet meist nach der Analysephase statt. Sie hat den Zweck, Dokumentanalyseergebnisse, die letztendlich der Workflow Engine zugeführt werden sollen, zu überprüfen und Fehler zu korrigieren. Diese Aktion wird vom normalen Sachbearbeiter (= *Benutzer*) durchgeführt.

Der Datenspeicher *korrigierte Endergebnisse* ist ein Teil der Eingabe für die Akquisition von neuem Wissen, die durch einen dazu *autorisierten Benutzer* durchgeführt wird. Der andere Teil der Eingabe besteht aus Zwischenergebnissen der Dokumentanalyse, die für die Akquisition relevant sein können (*Analysedaten für die Akquisition*). Die Akquisition selbst beinhaltet die Aufbereitung der Analyse- und Endergebnisse durch den *autorisierten Benutzer* und den Lernprozeß selbst. Als Ausgabe entsteht neues oder verfeinertes Wissen, das entweder im deklarativen Teil des *Spezialistenwissens* oder im *Dokumentwissen* abgelegt wird. Diese Zuordnung geschieht in Abhängigkeit davon, ob das Wissen für einen oder mehrere DAU-Spezialisten relevant sein kann.

Nachdem das neue Wissen akquiriert wurde, löscht der autorisierte Benutzer den Datenspeicher mit den Analysedaten für die Akquisition manuell.

4 Ein Beispiel

Dieses Kapitel konkretisiert die im vorherigen Kapitel beschriebenen drei Phasen anhand eines Beispieldokumentes mit dazugehörigem Workflow und offenen Vorgängen. Dazu werden zunächst die Eigenschaften des Beispieldokumentes und des restlichen Kontextes näher spezifiziert. Danach wird die Abarbeitung der drei Phasen für dieses Beispiel skizziert.

4.1 Beispieldokument und -umgebung

Dieses Unterkapitel beschreibt diverse Eigenschaften des Beispiels, die aber nicht notwendigerweise vor Durchlauf der drei Phasen bekannt sind. Sofern die Eigenschaften inferiert werden, wird darauf an den entsprechenden Stellen in den Beschreibungen der Phasen hingewiesen.

4.1.1 Dokumenteigenschaften

- TIFF-Dokument, das in Farbe vorliegt und von einem Nadeldrucker gedruckt wurde
- Formular mit Logo, das mit roter Farbe unterlegt ist
- Nachrichtentyp: Rechnung über Kauf eines OfficeAsk-Systems zum Preis von DM 111.199,50
- Absender: Firma SmartLab, Postfach 2080, 67608 Kaiserslautern
- Empfänger: Unibeschaffung Kaiserslautern
- Datum: 05.12.97, Univorgangsnummer 6025/11111/97, Rechnungsnummer 0815

4.1.2 Firmenwissen

- Aktuelle Prozeßinstanzen: Aktuell sind zwei Vorgänge offen: Der Vorgang, zu dem das Beispieldokument tatsächlich gehört, wartet entweder auf eine Rechnung oder einen Lieferschein von SmartLab. Der andere Vorgang wartet auf eine Auftragsbestätigung der Firma ClumsyComp, die kein Logo enthält. Als Standarderwartung bezeichnen wir Dokumente, die jederzeit und unauf-

gefordert eintreffen können, also z. B. in unserer Anwendungsdomäne eine Werbung oder eine Mahnung.

Ein Teil dieser Beschreibung der offenen Vorgänge stellt eine Erwartungshaltung dar, die natürlich erst zur Laufzeit generiert werden kann. Es wird also zur Analysezeit ein Inferenzprozeß durchgeführt, der aus den offenen Vorgängen Beschreibungen von den zu erwartenden Dokumenten erzeugt.

Die Vorgänge wurden als Workflow *Bestellung* modelliert, der in Kapitel 4.1.6 näher erläutert wird. Diese Workflow-Definition dient als Muster zur Instantiierung der beiden Vorgänge (als Geschäftsprozesse) zur Beschaffung von OfficeAsk bei Firma SmartLab und einer Garderobe bei Firma ClumsyComp. Beide Prozeßinstanzen haben die Aktivität *Bestellung versenden* beendet und warten nun auf den Eingang eines Dokumentes.

Aufgrund des Firmenwissens über SmartLab wird in dieser Prozeßinstanz keine Auftragsbestätigung erwartet, da SmartLab üblicherweise keine Auftragsbestätigungen verschickt. Für die zweite Prozeßinstanz (Firma ClumsyComp) ist das Dokument optional, d. h. es könnte eine Bestätigung eintreffen, muß aber nicht, da die Firma in dieser Beziehung nicht sehr zuverlässig ist.

- Sonstiges Firmenwissen: In der Adreßdatenbank findet sich SmartLab als Lieferant innovativer Softwaretechnologien mit Adreßangabe. Außerdem ist die Produktdatenbank lieferbarer Artikel dieses Anbieters vorhanden. Die bekannten Formularbeschreibungen aller Lieferanten und eventuell eingesetzte Logos sind ebenfalls per Datenbank zugreifbar, wie im folgenden vereinfacht dargestellt ist:

  - Adreß-DB: | SmartLab | Postfach 2080 | 67608 Kaiserslautern | innovative Software | verschickt keine Auftragsbestätigung |
  - Adreß-DB: | ClumsyComp | Feldweg 42 | 12345 Hintertupfingen | Gastronomiebedarf | Auftragsbestätigung möglich | lange Antwortzeit |
  - Produkt-DB: | SmartLab | OfficeASK FormClas ConPlan |
  - Produkt-DB: | ClumsyComp | Girlanda-Universalgarderobe Zapfanlagen Bars |

- Formular-DB: | ClumsyComp | Auftragsbestätigung | Auftragsnummer kursiv | kein Logo |
- Formular-DB: | SmartLab | Lieferschein | Auftragsnummer schattiert | Logo rot hinterlegt |

4.1.3 Dokumentwissen

Im Dokumentwissen werden die verschiedenen Nachrichtentyp-Ausprägungen modelliert, wie sie vom reaktiven Planer und von den DAU-Spezialisten benötigt werden. Im folgenden Beispiel beschränken wir uns auf die Definition eines (allgemeinen) Geschäftsbriefes und eines (allgemeinen) Lieferscheins.

Jedes im Dokumentwissen definierte Objekt wird als *Frame* bezeichnet und durch Attribute näher spezifiziert. Aus Gründen der Übersichtlichkeit sind diese Attribute aufgeteilt in allgemeine Attribute (nicht gruppiert), bildbezogene Attribute (unter dem Schlüsselwort `image` gruppiert), layoutbezogene Attribute (Schlüsselwort `layout`) und strukturbezogene Attribute (Schlüsselwort `logic`).

In unserem Beispiel ist der Frame Geschäftsbrief das oberste Objekt der Hierarchie (`:super-type nil`) und hat die Teile Empfänger, Absender und Schreibdatum (`:parts ...`). Die hinter den restlichen Attributen angegebenen Werte stellen Mengen möglicher Werte dar.

```
(frame Geschäftsbrief
       :super-type nil
       :parts ((Empfänger :frequency "n=1")
               (Absender :frequency "n=1")
               (Schreibdatum :frequency "n=0 v n=1")
               )
       (image :paper-size {:A4P}
              :printer-type {:needle :laser :ink}
              :background-color {:white}
              :foreground-color {:black}
              )
       (layout :font-size {:medium})
       )
```

Für den Lieferschein, der eine Spezialisierung des Geschäftsbriefes ist, schränken wir die Druckerqualität auf Nadel- und Tintenstrahldrucker ein (`:printer-type {:needle :ink}`). Desweiteren geben wir eine mögliche Formulierung für die Lieferscheinnummer an.

```
(frame Lieferschein
       :super-type Geschäftsbrief
       (image :printer-type {:needle :ink})
       (logic :pattern
              (:name Lieferscheinnummer
              :syntax "Lieferschein Nr. ?nummer :morph-cat Zahl")
              )
       )
```

Die hier gewählte Syntax stellt eine grobe Vereinfachung der im Projekt verwendeten Syntax dar.

4.1.4 Strategisches Wissen

Es wird eine hierarchische Attribut-/Wert-Beschreibung der verfügbaren Komponenten verwendet, d. h. es ist von einem DAU-Spezialisten bekannt, ob er z. B. zur Logoerkennung geeignet ist. In einem solchen Aktivitäts-/Konfigurationspaar ist neben der einfachen Zuordnung aufgrund der Struktur der zugrundeliegenden Wissensbasis auch strukturelles Wissen enthalten. Im folgenden werden z. B. wichtige Eigenschaften der DAU-Aktivitäten angedeutet:

```
DAU_activity -> (
    • time_effort : float,
    • memory_effort : float,
    • precision : [0..1],
    • input : set(DAU_data),
    • output : set(DAU_data)),
```

Eine Teilklasse der DAU-Aktivitäten besteht aus einer einzelnen Aktion:

```
DAU_action -> (
    • DAU_activity,
    • spec_id : symbol),
```

Als Beispiel für eine Aktion soll die OCR dienen:

```
OCR_action1 -> (
    • DAU_action,
    • spec_id = OCR1,
    • input.size = 1,
    • exists aTIFF in input with aTIFF : TIFF_data,
    • output.size = 1,
    • exists text in output with text : Char_alternatives,
    • memory effort = ...),
```

Konzepte zur Charakterisierung von (Zwischen-) Ergebnissen wie die hier verwendeten Konzepte `TIFF_data` und `Char_alternatives` sind ebenfalls Bestandteil des strategischen Wissens. Pläne werden aus DAU-Aktionen oder DAU-Aktivitäten zusammengestellt:

```
DAU_plan -> (
    • DAU_activity,
    • subplan = list(DAU_activity),
    • intermediate_result = list(set(DAU_data)),
    • ...)
```

wobei `subplan` die auszuführenden Aktivitäten enthält.

Weitere *constraints* in dieser Konzeptdefinition fordern u. a., daß alle Ausgaben einer an Stelle *i* eingeplanten Aktion zum *i*-ten Zwischenergebnis hinzugefügt werden, und daß alle Eingaben eingeplanter Aktionen bekannte Zwischenergebnisse sein müssen.

Für bestimmte Aufgaben sind i. d. R. speziellere Beschreibungen von Plänen möglich. So wird eine DAU-Aktivität, die von einem TIFF-Image bis zur Extraktion von Informationen aus dem Dokument geht, zunächst eine Bildvorverarbeitungsphase haben, danach u. U. eine Logoerkennung (sofern das Ergebnis der Logoerkennung für Klassifikation oder Extraktion interessant ist). Darauf wird i. d. R. eine OCR-Aktion gestartet, die wiederum auch aus einem Aufruf verschiedener OCRs bestehen kann, die von einem *voting* beschlossen werden. Abschließend erfolgt eine Aktion zur Klassifikation und Extraktion von Information.

Häufig werden DAU-Aktivitäten in Abhängigkeit vom erwarteten Nachrichtentyp definiert sein. Beispiel: Oben wurde eine Option auf den Einsatz der Logoerkennung eingeführt. Ob diese Aktion nun sinnvoll ist oder nicht, hängt natürlich von der erwarteten Häufigkeit eines Logos ab, die wiederum stark vom Typ des analysierten Dokuments abhängt.

Um derartige Kontexte notieren zu können, werden in der Konfigurationsphase Konzepte von DAU-Aufgaben definiert:

```
DAU_task -> (public:
    • DAU_plan,
    • name : string
    • ...)
```

Spezialisierungen dieses Konzeptes spezifizieren die Eigenschaften einer DAU-Aktivität genauer und repräsentieren damit eine Aufgabenstellung für die Dokumentanalyse, die durch einen `DAU_plan` unter Verschaltung der bekannten DAU-Aktionen erfüllt werden soll. Das Attribut Name dient als Bezeichner dieser Aufgabenstellung in den Workflow-Definitionen (vgl. Kapitel 5.2.4).

4.1.5 Erwartungshaltung und DAU-Aktivitätsauftrag

Folgend wird die Datenstruktur beschrieben, die für das Setzen der Erwartungshaltung und den Auftrag einer DAU-Aktivität vom Vermittler verwendet wird.

Die Datenstruktur ist wie folgt aufgebaut:

```
DokID: String
Nachrichtentyp: String
Believe[]
ProzeßID: String
AktivitätsID: String
EreignisID: String
DAUTask: TaskID
```

Falls sich auf ein bereits existierendes Dokument bezogen wird (d. h. bei einem DAU-Aktivitätsauftrag), muß die zugehörige Dokument-ID unter `DokID` angegeben werden.

`Nachrichtentyp` bezeichnet den zu erwartenden Nachrichtentyp (etwa Rechnung, Lieferschein, etc.).

`Believe` ist ein assoziatives Feld (Feld[Schlüssel] = Wert), in dem alle bereits im Workflow bekannten Eigenschaften des zu erwartenden Dokumentes aufgeführt sind. Dies wären z. B.

```
Believe[AuftragsNr] = 6025/11111/97
Believe[Firma] = SmartLab
```

Die Semantik der in `Believe` verwendeten Bezeichner folgt aus der Verwendung im strategischen Wissen. Die folgenden drei Attribute werden dazu verwendet, die Erwartungshaltung bzw. die Aufträge der korrekten Workflow-Aktivität wieder zuzuordnen bzw. das korrekte Ereignis im Workflow generieren zu können. Die Angabe der EreignisID kann entfallen, wenn man sich beim Eintreffen des entsprechenden Dokumentes an eine bestimmte Workflow-Aktivität im Prozeß wenden soll. Die AktivitätsID kann entfallen, wenn ein Ereignis generiert werden soll.

`DAUTask` ist eine Liste von Bezeichnern, die eindeutig eine DAU-Aktivität identifizieren. Ist das DAUTask-Attribut in einem Erwartungshaltung-Datensatz nicht leer, so sollen die angegebenen DAU-Aktivitäten direkt ausgeführt und die Ergebnisse mitgeliefert werden.

4.1.6 Workflow-Definition

Den beiden zu betrachtenden Vorgängen liegt eine einzige Workflow-Definition *Bestellung* (siehe Abbildung 5) zugrunde. Diese wird für jeden Bestellvorgang instantiiert.

Zu Beginn des Workflows werden die Aktivitäten *Bedarfsanmeldung bearbeiten* und *Bestellung versenden* ausgeführt. Danach teilt sich der Workflow zur Behandlung der einzelnen Dokumente (Rechnung, Lieferschein, Auftragsbestätigung) auf (parallele Ablaufzweige, AND-Split). Die Aktivität *Auswahl Ablaufzweige* bestimmt dabei, welche Zweige durchlaufen und welche evtl. ausgelassen werden. Näheres hierzu folgt in Kapitel 4.1.6.2.

Die erste Aktivität (*SetErwartungshaltung*) in dem jeweiligen Ablaufzweig setzt die Erwartungshaltung. Sie startet den Vermittler, der den übergebenen Datensatz in die

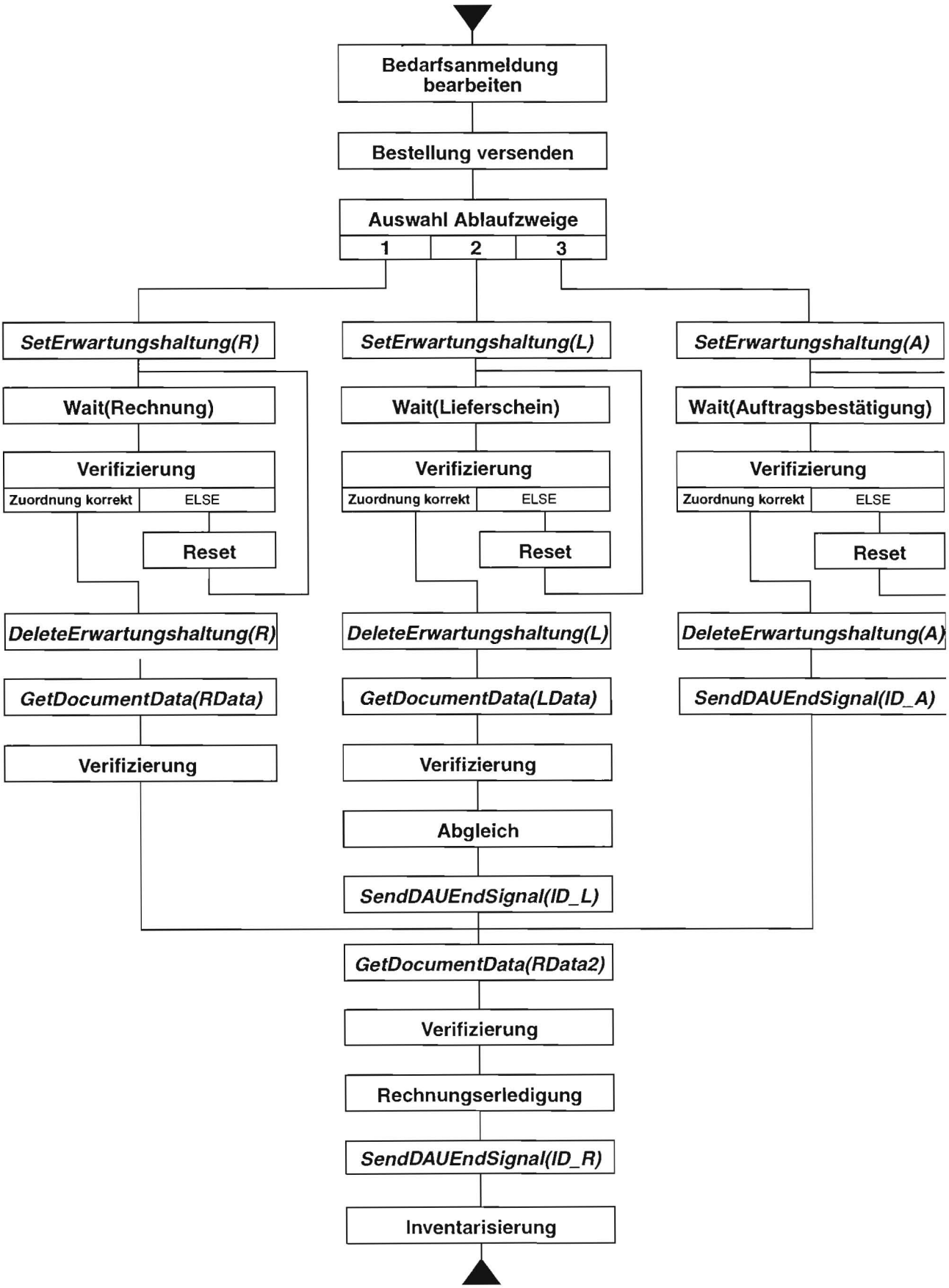

Abbildung 5: Workflow-Definition *Bestellung*

Erwartungshaltung einträgt, danach wird in der Aktivität *Wait* auf den Eingang des Dokumentes gewartet.

Trifft z. B. die Rechnung ein, wird im Ablaufzweig 1 fortgefahren. Die nächste Aktivität *Verifizierung* läßt den Bearbeiter die Ergebnisse des DAU-Systems überprüfen und gegebenenfalls korrigieren. Eine solche Aktivität folgt allen Aktivitäten, die sich an das DAU-System wenden. Sollte sich bei der Verifikation herausstellen, daß das Dokument falsch zugeordnet wurde, muß das Dokument durch den Bearbeiter dem korrekten Vorgang zugeordnet werden. Der Bearbeiter erhält Einsicht in die Erwartungshaltung und kann direkt einen Vermittler starten, der die Übergabe an die korrekte Prozeßinstanz übernimmt. Im Workflow muß bei der Modellierung diese Situation berücksichtigt werden, damit nach einer falschen Zuordnung wieder in den ursprünglichen Zustand zurückgesetzt werden kann. Dieses Zurücksetzen übernimmt hier die Aktivität *Reset,* nach deren Ausführung wieder in *Wait* zurückgegangen wird.

Ist die Zuordnung korrekt, kann die Erwartung aus der Erwartungshaltung gelöscht werden (*DeleteErwartungshaltung*). Dies übernimmt der Workflow, da das DAU-System zur Zeit der Zuordnung nicht sicher sein kann, daß diese korrekt ist und die Erwartung befriedigt wurde. Für die spätere Bearbeitung wird die Rechnungsnummer benötigt. Diese erhält der Workflow von dem DAU-System durch den Vermittler, der mit den entsprechenden Angaben in der Aktivität *GetDocumentData* gestartet wird. Gleiches gilt für die Zweige 2 und 3. Bei der Bestätigung der Lieferung durch den Lieferschein wird ein Abgleich der gelieferten Artikel mit den bestellten Artikel vorgenommen und bei Unstimmigkeiten entsprechend gehandelt. Eine Liste der gelieferten Artikel fordert die Aktivität *GetDocumentData* in Zweig 2 an.

Sind alle Dokumente eingegangen, werden die parallelen Zweige wieder zusammengeführt (AND-Join). Danach wird die Artikelliste aus der Rechnung angefordert, um in den Aktivitäten *Rechnungserledigung* und *Inventarisierung* alle benötigten Daten zur Verfügung zu haben. Dies ist ein Beispiel dafür, warum erst gegen Ende des Workflows das Ende-Signal für die Rechnung an das DAU-System gesendet wird, da das Dokument bzw. ihre Daten noch benötigt werden.

Die Aktivitäten *SendDAUEndSignal* teilen dem DAU-System mit, daß das bezeichnete Dokument nicht mehr benötigt wird. Die reaktive Planerinstanz für das jeweilige Dokument wird seine Zwischenablage löschen und sich dann beenden.

4.1.6.1 Verwendete Datenstrukturen

Erwartungshaltungen

```
R: Believe[Nachrichtentyp]= Rechnung
```

Ergebnis: `DokID = ID_R`

```
L: Believe[Nachrichtentyp]= Lieferschein
```

Ergebnis: `DokID = ID_L`

```
A: Believe[Nachrichtentyp]= Auftragsbestätigung
```

Ergebnis: `DokID = ID_A`

DAU-Aktivitätsauftrag

```
RData:   DokID = ID_R
         DAUTask = GetRechnungsNr
```

Ergebnis: `RechnungsNr`

```
LData:   DokID = ID_L
         DAUTask = GetArtikel
```

Ergebnis: `Artikel`

```
RData2:  DokID = ID_R
         DAUTask = GetArtikel
```

Ergebnis: `Artikel`

Die Datenstruktur für Erwartungshaltung und DAU-Aktivitätsauftrag ist identisch. Für die Auflistung der einzelnen Ausprägungen siehe Anhang 1.

4.1.6.2 Besonderheiten

In unserem Beispiel weichen die beiden betrachteten Firmen von dem modellierten Bestellvorgang ab. ClumsyComp sendet nur von Zeit zu Zeit eine Auftragsbestätigung

und SmartLab sendet nie eine. Beide Vorgehensweisen werden in der Geschäftswelt toleriert. Um nun solches auch hier abbilden zu können, wurde der normale Vorgang modelliert. Abweichungen einer Firma von diesem Vorgang werden im *corporate knowledge* vermerkt. In der jeweiligen Workflow-Instanz wird dann entschieden, ob bestimmte Aktivitäten auch ausgelassen werden können. Im Beispiel würde in der Workflow-Instanz für die Bestellung bei der Firma SmartLab Zweig 3 nicht durchlaufen werden. Diese Entscheidung übernimmt die Aktivität *Auswahl Ablaufzweige*. Sie ist als automatische Aktivität gedacht, die keine Benutzereinwirkung benötigt. In der Aktivität wird auf das *corporate knowledge* zugegriffen, die benötigte Information wird extrahiert und ausgewertet und je nach Ergebnis wird die Workflow Engine veranlaßt, bestimmte Zweige zu durchlaufen bzw. auszulassen.

Wie dies im einzelnen erreicht wird und ob es überhaupt möglich ist, hängt vom jeweiligen WFMS ab. Der reaktive Planer macht darüber keine Annahmen.

Es werden für die Auftragsbestätigung folgende Konventionen befolgt: Es wird ein Eintrag in die Erwartungshaltung vorgenommen, wenn eine Auftragsbestätigung immer (d. h. obligatorisch) oder nur ab und zu (d. h. optional) gesendet wird. Wird keine Auftragsbestätigung erwartet, wird auch kein Eintrag getätigt.

4.2 Konfigurationsphase

In der Konfigurationsphase wird der bestehende Workflow mit Dokumentwissen angereichert und die Aktivitäts-/Konfigurationspaare werden erstellt.

Bezüglich des Dokumentwissens besteht zu diesem Zeitpunkt bereits eine Definition eines allgemeinen Lieferscheins. Da dem Workflow-Designer zum Zeitpunkt der Konfiguration genaueres Wissen über den Lieferschein vorliegt, reichert er das Dokumentwissen mit dieser Beschreibung an:

```
(defframe Lieferschein_SmartLab
    :super-type Lieferschein
    :data-base-reference Formular-DB
    (image :printer-type {:needle}
           :paper-size {:A4P}
```

```
        :background-color {:red}
        :logo-structure (:background-color {:red}
                         :rlc-code (:data-base-reference DB-Logo) )
        )
(logic :word-list (OfficeASK Formclas ConPlan))
)
```

Die Beschreibung des Logos der Firma SmartLab wird in der Datenbank abgelegt:

- Logo-DB: | SmartLab | RLC-Structure |

Lieferantenabhängige Spezialisierungen können über den Slot *:data-base-reference* auf dem Toplevel eines Dokumentframes aus dem Firmenwissen inferiert werden. Ebenso ist die Angabe kompletter Spezialisierungshierarchien mit *:data-base-reference* aus den Parts eines Dokumentframes möglich.

Aus den Änderungen im Dokumentwissen werden (durch entspr. Konverter) die Datenmodelle für die Darstellung von `Believe`-Daten (Erwartungshaltung) und von Ergebnissen der DAU-Aktivitäten geändert. Analog wird ein Teil des Dokumentwissens in das strategische Wissen projiziert, um z. B. Objektstrukturen für die Verwaltung der Erwartungshaltung und der Extraktionsergebnisse zu definieren.

Außerdem werden in der Konfigurationsphase die generischen Definitionen von DAU-Aktivitäten den speziellen Anforderungen der aktuellen Anwendung angepaßt. Dies geschieht durch Spezialisierung. Die Notwendigkeit anwendungsspezifischer Erweiterungen der generischen Teile des strategischen Wissens in der Konfigurationsphase stellen die Motivation für eine objektzentrierte Darstellung dieses Wissens dar.

Zum Beispiel: Zur Bearbeitung der Workflow-Aktivität „Rechnung bearbeiten" sollen folgende DAU-Aktivitäten implementiert werden: Eine Aktivität „Vorgangszuordnung" muß spezifiziert werden, um ein Dokument aus dem Postkorb im TIFF-Format einer offenen Prozeßinstanz im WFMS zuzuordnen (vgl. Kapitel 5.3.4). Im gegebenen Beispiel soll diese DAU-Aktivität zusätzlich zumindest den Nachrichtentyp ermitteln.

Die DAU-Aktivität „GetRechnungsNr" soll die Rechnungsnummer extrahieren, „GetArtikel" extrahiert aus dem Dokument den Bezeichner der bezogenen Ware. Im Kontext

dieser DAU-Aktivität ist der ablaufende Prozeß klar (vgl. Kapitel 4.1.6), so daß eine Vorgangszuordnung nicht mehr notwendig ist.

Für jede dieser Aufgaben wird während der Konfigurationsphase eine Spezialisierung des Konzepts `DAU_task` erstellt, deren Attribut `name` den o.g. Bezeichnern für die DAU-Aktivität entspricht. Wird im WFMS einer dieser Bezeichner verwendet, so ist über die in der Konfigurationsphase erstellte Aufgabenbeschreibung definiert, welche DAU-Funktionalität erforderlich ist. Mit Bezug auf das Dokumentwissen bzw. die zur Charakterisierung von (Zwischen-) Ergebnissen bereitgestellten Konzepte können Ein- und Ausgaben der erwünschten Funktionalität festgelegt werden. Durch *constraints* über das Attribut `precision` bzw. Speicheraufwand und verfügbare Rechenzeit kann in der Konfigurationsphase ein flexibles Anforderungsprofil für die im Prozeßmodell notierten DAU-Aktivitäten erstellt werden. Desweiteren ist die Formulierung von harten bzw. weichen Bedingungen an den durchzuführenden Plan für die spezifizierte DAU-Aktivität erwünscht und möglich. Dazu werden dem Systemoperator im Dialog automatisch berechnete Pläne vorgelegt, die mit den bislang genannten Spezifikationen verträglich sind bzw. Konfigurationskonflikte werden angezeigt. Kritikpunkte an diesen automatisch generierten Plänen können dann zu erweiterten Anforderungen führen. Die letztlich akzeptierten Instanzen von `DAU_plan` werden gespeichert, um ggf. dem reaktiven Planer während des Betriebs als Hilfe bei der Plangenerierung zu dienen.

Zum Beispiel: Im Beispiel macht der Systemoperator es sich leicht und verwendet generische Konzepte für Pläne, die jeweils die Sequenz auszuführender DAU-Aktionen durch allgemeinere Konzepte spezifizieren, die dann von der reaktiven Planung jeweils passend mit Aktionen gefüllt werden.

- Voraussetzungen: TIFF-Dokument.
  Ergebnis: Dokumentklasse und Vorgangszuordnung.
  Die Aktivität „Vorgangszuordnung“ hat folgende Konfiguration:
  Vorverarbeitung – Logoerkennung – Segmentierung – Formularerkennung – OCR – Nachrichtentypbestimmung – Adreßerkennung – Vorgangsdatenermittlung

- Voraussetzungen: TIFF-Dokument, Nachrichtentyp ist „Rechnung", Vorgang. Ergebnis: Ermittlung der Rechnungsnummer.
  Die Aktivität „GetRechnungsNr" hat folgende Konfiguration:
  Vorverarbeitung – Segmentierung – OCR – Extraktion der Rechnungsnummer

- Voraussetzungen: TIFF-Dokument, Nachrichtentyp ist „Rechnung", Vorgang. Ergebnis: Ermittlung der Artikelliste.
  Die Aktivität „GetArtikel" hat folgende Konfiguration:
  Vorverarbeitung – Liniendetektion – Segmentierung – OCR – Tabellenanalyse

4.3 Abarbeitung des Beispieldokumentes

Nach der Erzeugung des Prozesses zur Analysesteuerung, d. h. durch Instantiierung eines reaktiven Planers, ist im Beispiel die folgende Situation ausschlaggebend.

Folgende Tabelle veranschaulicht die aktuelle Erwartungshaltung:

(ID)	(Nachrichtentyp)	(Firma)	(AuftragsNr)	(ProzeßID)	(AktivitätsID)	(Datum)
1	Rechnung	SmartLab	6025/11111/97	Bestellung1	A1_2	05.12.1997
2	Lieferschein	SmartLab	6025/11111/97	Bestellung1	A2_2	05.12.1997
3	Rechnung	ClumsyComp	6025/11322/97	Bestellung2	A1_2	07.12.1997
4	Auftragsbest.	ClumsyComp	6025/11322/97	Bestellung2	A2_2	07.12.1997
5	Lieferschein	ClumsyComp	6025/11322/97	Bestellung2	A3_2	07.12.1997

Zusätzlich sind noch die Standarderwartungen (Nachrichtentyp) Angebot und (Nachrichtentyp) Mahnung enthalten.

Aufgrund des aktuellen Datums (08.12.1997) erscheint das Eintreffen eines Briefes der Firma SmartLab wahrscheinlicher als das Eintreffen eines Briefes von ClumsyComp (insbesondere da letztere bisweilen etwas träge reagieren). Dadurch ist eine starke Erwartungshaltung zugunsten eines SmartLab-Dokuments gegeben. Im Dokumentwissen wurde in der Konfigurationsphase notiert, daß SmartLab-Dokumente mit einem Logo versehen sind.

Planung: Der reaktive Planer wird nun mit folgender Spezifikation gestartet: Eingabe ist ein TIFF-Bild des Dokuments und als Ausgabe sollen sowohl Rechnungen, Lieferscheine und Auftragsbestätigungen der Firmen SmartLab und ClumsyComp erkannt werden können. Für jede dieser Teilaufgaben liegt aus der Konfigurationsphase sowohl eine Spezifikation als auch ein Beispielplan vor. Der Planungsprozeß sucht nun nach einem Plan, welcher der Konjunktion der Spezifikationen der Teilaufgaben genügt. Eine sinnvolle Verwendung von Kontextwissen führt für die Vorgangszuordnung z. B. zu folgendem Verfahren, das womöglich als Skelettplan in der Konfigurationsphase spezifiziert wurde: Nach der Logoerkennung ist bereits unter dem gegebenen Kontext (`Believe`-Daten) eine Zuordnung zur Firma SmartLab möglich, weil sich die erwarteten Dokumente in der Verwendung von Logos unterscheiden. Es verbleibt als weitere Aufgabe die Erkennung des Nachrichtentyps. Diese kann wiederum auf vielerlei Art und Weise geschehen. Während die üblichen Formen der Nachrichtentyperkennung direkt das verfügbare Dokumentwissen ohne weitere Verwendung des Kontexts betrachten, kann im strategischen Wissen z. B. spezifiziert sein, daß es zur Erkennung bzw. Widerlegung der Hypothese *Rechnung der Firma SmartLab* genügt, nach dem Wort Rechnung zu suchen. Als Voraussetzung für die Vorgangszuordnung ist auf jeden Fall eine OCR-Operation zu betrachten, die am besten anhand der zu erwartenden Druckqualität ausgewählt wird. Unter dem vorliegenden Kontext reichen Firma und Nachrichtentyp zur Vorgangszuordnung. Im allgemeinen Fall oder falls eine höhere Genauigkeit der Vorgangszuordnung erforderlich ist, können durch eine generische Nachrichtentyperkennung weitere Textmuster gesucht oder Dokumentteile extrahiert werden. Im vorliegenden Falle der mutmaßlichen Analyse von Rechnungen soll im strategischen Wissen vermerkt sein, daß das generische Verfahren beschritten wird, weil es sicherer ist und i. d. R. auf die Vorgangszuordnung in diesen Fällen weitere Aufgaben zur Informationsextraktion folgen.

Planablauf: Die Logoerkennung liefert uns nun die Information, daß das eingetroffene Dokument tatsächlich von der Firma SmartLab stammt und in exzellenter 3-Farben-Tiefdruckqualität mit Laseraufdruck vorliegt. Die OCR wird durchgeführt. Zur Verifikation wird der String *„SmartLab"* gesucht und gefunden. Als weitere Aktion wird nun die generische Nachrichtentyperkennung gestartet. Diese sollte, wenn alles gut gegangen ist,

den Typ Rechnung zurückliefern. Im Zuge der generischen Nachrichtentyperkennung werden Auftragsnummer und Rechnungsnummer erkannt (letztere allerdings falsch als 4711). Aufgrund der extrahierten Eigenschaften findet eine Vorgangszuordnung statt.

Das Ergebnistemplate bestimmt nun, welche Analysedaten (Logotyp, Nachrichtentyp, Vorgangsnummer, Absender) in den Datenspeicher „Endergebnisse für den Workflow" geschrieben werden. Diese werden nach dem Ende der Aktivität geschrieben.

Beispiel für ein Ergebnistemplate (nur der Teil für die Endergebnisse):

((Angebot (Datum ...) (AngebotsNr ...) (Artikel ...) (Angebotssumme ...))
(Auftragsbestätigung (Datum ...) (AngebotsNr ...))
(Lieferschein (Datum ...) (AuftragsNr ...) (LieferscheinNr ...))
(Rechnung (Datum ...) (AuftragsNr ...) (RechnungsNr ...) (Artikel ...))
(Mahnung (Datum ...) (AuftragsNr ...)))

Beispiel für die Ergebnisablage (Rechnung der Fa. SmartLab):

(Rechnung (Datum 08.12.1997) (AuftragsNr 6025/11111/97) (RechnungsNr 4711) (Artikel OfficeAsk-System) ProzeßID (Bestellung1) AktivitätsID (A1_2))

Die Ergebnisse werden pro Dokument abgelegt.

Desweiteren wird bestimmt, welche Analysedaten für die Akquisition (Datenspeicher „Analysedaten für die Akquisition") bereitgestellt werden sollen. Diese Daten sollen es dem autorisierten Benutzer ermöglichen, bei Fehlern im Ergebnistemplate nachvollziehen zu können, warum der Fehler auftrat. Dies können z. B. Segmentierungsfehler, OCR-Fehler oder Fehler aufgrund mangelndem Dokumentwissens sein. Im Beispiel werden im Datenspeicher das Original-Dokumentbild, Logo-, OCR- und Segmentierungsergebnisse und die Ergebnisse der beteiligten Spezialisten zur inhaltlichen Analyse abgelegt.

Die Zwischenergebnisse werden solange in der Zwischenablage gehalten, bis entweder der komplette Vorgang bearbeitet worden ist (Default-Einstellung) oder die Aktivität „Zwischenablage löschen" im Workflow explizit aufgerufen wird.

Zweite DAU-Aktivität des Beispiels: Wie im Kapitel 4.1.6 beschrieben, wird nun eine weitere DAU-Aktivität zum Auslesen der Rechnungsnummer gestartet. Nun ist dieses Attribut der Rechnung bereits extrahiert worden (wenn auch fehlerhaft), so daß für diese Aktivität keine DAU-Aktion mehr gestartet werden muß. Der Datenaustausch erfolgt analog zum Verfahren im ersten Teil dieses Abschnittes. Allerdings wird hier nicht der globale `Believe` verwendet, sondern ein speziell für diesen Aufruf generierter `Believe`, der nur aus dem bekannten Nachrichtentyp und den bereits extrahierten Informationen besteht. Der reaktive Planer kann das aufgrund der Homogenität von `Believe`-Daten und DAU-Ergebnisse erkennen. Aufgrund der dokumentzentrierten Speicherung werden automatisch zuvor berechnete Ergebnisse von DAU-Aktivitäten als `Believe` der nachfolgenden Aktivitäten behandelt. Die Planung erkennt, daß bereits eine Vermutung über die Rechnungsnummer besteht und beendet die DAU-Aktivität ohne Starten einer DAU-Aktion.

4.4 Verifikations- und Akquisitionsphase

Verifikations- und Akquisitionsphase werden gemeinsam betrachtet, da sie beide nach der Dokumentanalyse stattfinden. Allerdings sind die beiden Phasen darüberhinaus zeitlich völlig unabhängig voneinander zu sehen.

4.4.1 Verifikation

Da im Workflow nach den entsprechenden DAU-Aktivitäten jeweils die Verifizierung der DAU-Ergebnisse als Aktivität definiert wurde (vgl. Kapitel 4.1.6), findet nun die Überprüfung des Ergebnistemplates statt. Der „einfache" Benutzer kann das Originalbild des Briefes mit den gefundenen Ergebnissen vergleichen und diese, falls nötig, korrigieren. Außerdem kann der Benutzer OCR-Fehler korrigieren.

In unserem Beispiel wurde die Rechnungsnummer fälschlicherweise als „4711" identifiziert. Die korrekte Rechnungsnummer lautet „0815". Der Sachbearbeiter ändert dieses Ergebnis also ab. In einer Log-Datei wird zusätzlich abgespeichert, daß eine Korrektur an den Ergebnissen zu diesem Dokument stattgefunden hat und welcher Art die Korrektur ist.

4.4.2 Akquisition

Der genaue Zeitpunkt der Akquisition kann auf verschiedene Arten spezifiziert werden:

- Der autorisierte Benutzer möchte das Dokument- oder das Spezialistenwissen erweitern, da die Domäne erweitert wird (z. B. „zukünftig werden Briefe eines neuen Typs oder eines neuen Lieferanten eintreffen“ oder „zukünftig müssen andere Informationen gefunden werden“) oder
- in einem eigenen Workflow wurde zu bestimmten Zeiten die Akquisition als Aktivität definiert oder
- die Log-Datei (vgl. Kapitel 4.4.1) hat eine bestimmte Größe erreicht.

Ein autorisierter Benutzer bearbeitet im 2. und 3. Fall zu diesem Zeitpunkt alle Einträge der Log-Datei, d. h., er bearbeitet alle die DAU-Fehler, die sich auf das Endergebnis ausgewirkt haben. Für unser Beispiel befaßt er sich also mit der fehlerhaften Rechnungsnummer und findet folgende Situation: Das Dokument enthält ein Pattern: „Telefon-Nummer: 4711“ und die Rechnungsnummer taucht in folgendem Kontext auf: „Rechnung der Firma SmartLab Nr. 0815“. Durch OCR-Fehler wurde die Telefonnummer als Rechnungsnummer toleriert und aufgrund eines fehlenden Patterns wurde die korrekte Rechnungsnummer nicht gefunden.

Eventuell kann die OCR durch Justierung von Parametern verbessert werden, dieser Fall ist allerdings nicht sehr wahrscheinlich, da die OCR bereits versucht, sich automatisch zu adaptieren. Das fehlende Pattern aber kann der autorisierte Benutzer ohne Probleme in das Dokumentwissen aufnehmen. Dazu nimmt er folgende Erweiterung im Dokumentwissen vor:

```
(defframe Rechnung_SmartLab
    :super-type Rechnung
    (logic :pattern
          (:name Rechnungsnummer
           :syntax "Rechnung der Firma SmartLab Nr. ?nummer :morph-cat
     Zahl")
    ))
```

In Zukunft kann dadurch der aufgetretene Fehler vermieden werden.

5 Klassenkategorien

5.1 Übersicht

In Anlehnung an Booch [4] werden unter Klassenkategorien sinnvolle Gruppierungen von Klassen verstanden. Sie haben den Zweck, die statisch-logische Systemsicht auf einer hohen Abstraktionsebene zu beschreiben.

Abbildung 6 beinhaltet alle für das System relevanten Klassenkategorien. In der oberen Hälfte befinden sich alle Klassenkategorien, die zum WFMS gehören und in der unteren

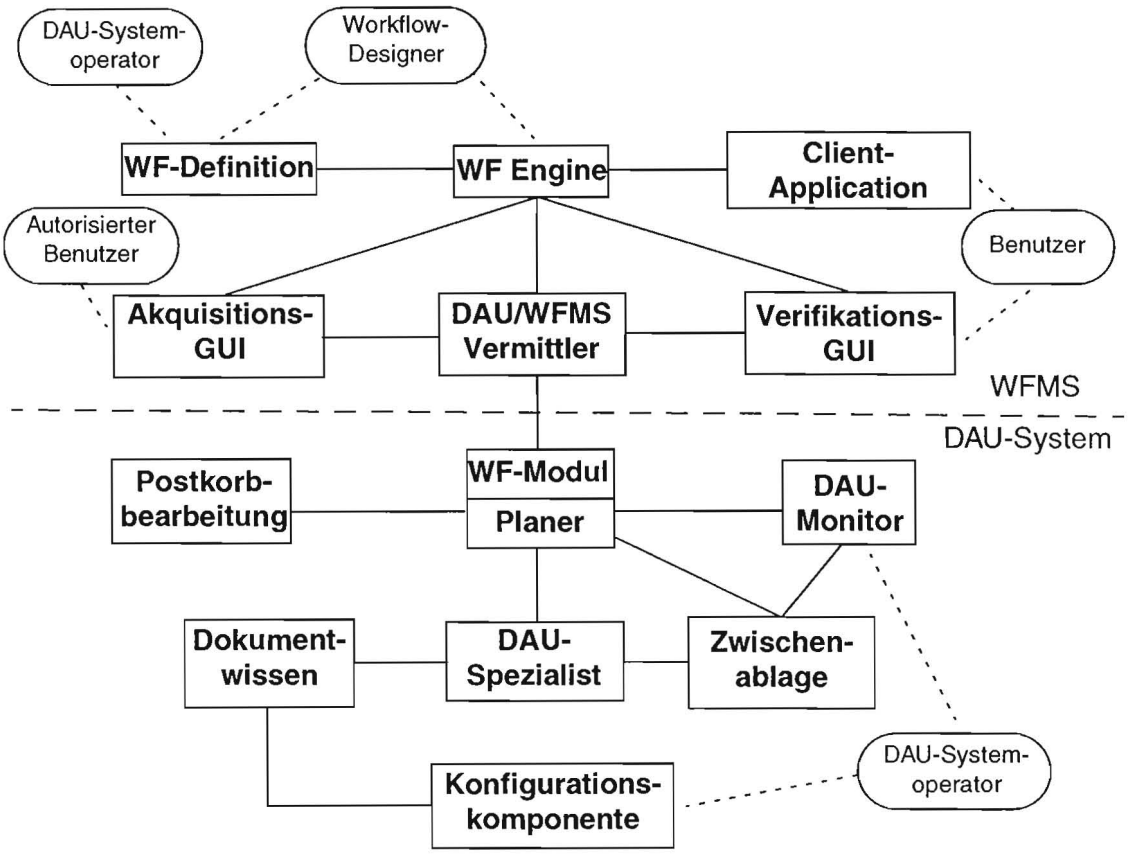

Abbildung 6: Übersicht über alle Klassenkategorien

Hälfte (Trennung durch gestrichelte Linie) finden sich alle Klassenkategorien, die zum DAU-System gehören. Beteiligte Rollen sind in Ovalen, die Klassenkategorien selbst sind in Rechtecken dargestellt. Verbindungslinien zwischen Klassenkategorien haben die

Bedeutung von assoziativen Verbindungen. Im folgenden wird Abbildung 6 näher erläutert.

5.1.1 Klassenkategorien des WFMS

Die Workflow Engine liefert die Laufzeitumgebung für einen Workflow-Prozeß. Sie ist dabei zuständig für die Interpretation der Workflow-Definition und die Ausführung und Verwaltung der Prozeßinstanzen. In der Workflow-Definition wird der reale Geschäftsprozeß abgebildet, so daß dieser durch die Workflow Engine ausführbar ist. Zusätzlich muß in der Workflow-Definition die Anbindung an das DAU-System modelliert werden.

Die Client Application ist die Schnittstelle des Benutzers zur Workflow Engine. Hier können z. B. die administrativen Aufgaben getätigt werden. Die Verifikations-GUI dient dem normalen Sachbearbeiter zur Überprüfung der DAU-Ergebnisse und über die Akquisitions-GUI gibt der autorisierte Benutzer Lerndaten für die DAU-Spezialisten ein.

Die DAU/WFMS-Vermittler dienen als Schnittstelle zwischen DAU (bzw. reaktivem Planer) und WFMS. Das WFMS wendet sich immer mit seinen Aufträgen an diese DAU/WFMS-Vermittler, d. h. der reaktive Planer ist für das WFMS nicht sichtbar.

5.1.2 Klassenkategorien des DAU-Systems

Das Workflow-Modul (WF-Modul) im reaktiven Planer ist auf der Seite der DAU zuständig für die Kommunikation zwischen Planer und DAU/WFMS-Vermittler. Die Postkorbbearbeitung überprüft periodisch den Eingangspostkorb nach neu eingetroffenen Dokumenten. Je eingegangenem Dokument wird eine reaktive Planer-Instanz mit der Aufgabe der „Vorgangszuordnung“ generiert und dieser das jeweilige TIFF-Bild mit der globalen Erwartungshaltung aus dem WFMS übergeben. Der reaktive Planer erzeugt dazu eine lauffähige Sequenz von DAU-Spezialisten, welche die gewünschte DAU-Aktivität ausführen können. Werden diese DAU-Spezialisten dann ausgeführt, so bedienen sie sich des Dokumentwissens, das strukturelles und relationales Wissen über die Dokumente der Domäne enthält.

Alle anfallenden Ergebnisse der DAU-Spezialisten werden in der Zwischenablage gespeichert. Sie stellt also alle Funktionen bereit, die benötigt werden, um die Ergebnisse der Spezialisten abzulegen bzw. zu laden.

Wesentliche Aufgabe des DAU-Monitors ist die Visualisierung des aktuellen DAU-Systemverhaltens. Dazu gehören aktive DAU-Spezialisten inklusive der Ausprägungen ihrer Parameter und ihrer Ein- bzw. Ausgabebeziehungen.

Die Konfigurationskomponente wird in der Konfigurationsphase benötigt. Sie verwaltet das strategische Wissen, aufgrund dessen der reaktive Planer zur Laufzeit Pläne erzeugen kann.

5.2 Aus Systemanforderungen abgeleitete Klassenkategorien

Aus den Systemanforderungen wurden die Klassenkategorien abgeleitet, die nach außen hin sichtbar sind, also von einer Rolle benötigt werden. Folgende Änderungen haben sich bei dieser Ableitung ergeben:

- Umbenennung Workflow-Spezifikation zu Workflow-Definition
- Trennung der Klassenkategorien Verifikation und Akquisition
- Umbenennung Workflow-Applikation zu Client Application

5.2.1 Workflow-Definition

In der Workflow-Definition wird der reale Geschäftsprozeß abgebildet, so daß dieser durch das WFMS ausführbar ist. Zusätzlich muß in der Definition die Anbindung an das DAU-System modelliert werden. Dazu gehört u. a. der Aufruf des Vermittlers, die Spezifizierung der benötigten Datenstrukturen und der Übergabe an den Vermittler, die Weiterverarbeitung der durch den Vermittler gelieferten Daten und die Einbindung einer Aktivität zur Verifizierung der erhaltenen Daten. Wie dies in der konkreten Workflow-Definition aussieht, hängt stark von der herstellerspezifischen Modellierungskomponente und der Mächtigkeit der zur Verfügung gestellten Workflow-Definitionssprache ab.

Die WfMC arbeitet im Rahmen ihrer Standardisierungsbemühungen im Workflow-Referenzmodell an einer Definition für die Aufbau- und Ablauforganisation (Interface 1). Zu dieser Definition ist bis jetzt jedoch keine nähere Information erhältlich.

5.2.1.1 Methode: CreateNewWorkflowDefinition

Argumente: Name der Workflow-Definition

Ausgabe: Oberfläche, in welcher der Workflow modelliert werden kann.

Beschreibung: Der hier definierte Workflow dient als Muster zur Erzeugung von Prozeßinstanzen.

5.2.2 Workflow Engine

Die Workflow Engine liefert die Laufzeitumgebung für eine Prozeßinstanz. Sie ist dabei zuständig für die Interpretation der Workflow-Definition und die Ausführung und Verwaltung der Prozeßinstanzen. Weiterhin steuert die Engine den Ablauf zwischen den einzelnen Aktivitäten, gestützt auf die in der Workflow-Definition modellierten Übergänge.

Die Mächtigkeit der von extern verfügbaren Methoden ist abhängig vom WFMS und der angebotenen API, i. d. R. sind Prozeßverwaltung (Instantiierung, Terminierung von Workflow-Instanzen) und eine Kommunikation über eine Schnittstelle mit gestarteten externen Programmen möglich.

Als Methoden sollen im folgenden nur die wichtigsten aufgeführt werden. Sonstige Verwaltungsmethoden, wie etwa SuspendProcess oder ResumeProcess werden aus Gründen der Übersichtlichkeit außen vor gelassen.

5.2.2.1 Methode: StartProcess

Argumente: Daten zum speziellen Geschäftsprozeß (Kundenname, Dokumente), Name der Workflow-Definition

Ausgabe: neue Prozeßinstanz mit Bezug zum spezifischen Geschäftsprozeß

Beschreibung: Das Starten eines neuen Prozesses kann durch das DAU-System geschehen, wenn z. B. ein Dokument eintrifft, das keinem offenen Prozeß zugeordnet werden

kann. Es ist aber auch durch den Benutzer möglich, deswegen erscheint diese Methode auch in der Client Application.

5.2.2.2 Methode: StartEngine

Argumente: –

Ausgabe: –

Beschreibung: Nach Aufruf ist das WFMS bereit, so daß Bearbeiter sich über die Client Application anmelden und ihre Arbeitsliste bearbeiten können. Die Workflow Engine API ermöglicht nun Aufrufe externer Programme.

5.2.2.3 Methode: StopEngine

Argumente: –

Ausgabe: –

Beschreibung: Das WFMS wird kontrolliert heruntergefahren. Nach diesem Aufruf kann sich kein Benutzer mehr anmelden und es werden keine API Aufrufe mehr bedient.

5.2.2.4 Methode: EnactActivity

Argumente: –

Ausgabe: –

Beschreibung: Die Workflow Engine führt eine Aktivität aus. Sie stellt u. a. die Datenstrukturen (systeminterne und benutzerdefinierte) zur Verfügung, damit in der Aktivität gestartete Programme darauf zugreifen können.

5.2.3 Client Application

Die *Client Application* (auch *Front-End Application*) ist die Schnittstelle des Benutzers zur Workflow Engine. Hier können etwa administrative Aufgaben getätigt, die Arbeitslisten bearbeitet und Workflows modelliert, instantiiert und verwaltet werden. Welche Funktionalitäten die eingesetzte Client Application besitzt, hängt vom Hersteller ab. Die

WfMC definiert in ihrem Workflow-Referenzmodell eine Schnittstelle zwischen Client Applications und der Workflow Engine, das sogenannte Interface 2.

WorkDesk ist die Client Application des im Projekt eingesetzten WFMS *CSE WorkFlow.* Mit WorkDesk können Workflows instantiiert und verwaltet werden. Desweiteren kann der jeweilige Benutzer seine Arbeitsliste bearbeiten. Die Definition eines Workflows findet jedoch im sogenannten *ProcessDesigner* statt.

Auch hier werden die vielfältigen Verwaltungsmethoden (Verwaltung des Arbeitskorbes, eines Workflows, ...), die eine Client Application bieten soll, aus Gründen der Übersichtlichkeit nicht vollständig aufgelistet.

5.2.3.1 Methode: StartProcess

Argumente: Daten zum speziellen Geschäftsprozeß (Kundenname, evtl. Anlagen wie Dokumente), Name der Workflow-Definition

Ausgabe: neue Prozeßinstanz mit Bezug zum spezifischen Geschäftsprozeß

Beschreibung: Der Benutzer kann eine neue Prozeßinstanz erzeugen, um einen Geschäftsprozeß zu bearbeiten.

5.2.3.2 Methode: WorkProcess

Argumente: eine offene Prozeßinstanz

Ausgabe: Ausführung der in der anstehenden Aktivität definierten Aktion

Beschreibung: Der Benutzer bearbeitet hier die anstehende Aktivität der Prozeßinstanz. Nach Erledigung wird der Prozeß gemäß der Workflow-Definition zur nächsten Aktivität übergehen.

5.2.4 Konfigurationskomponente

Abstrakte Anforderungen an die Konfiguration der DAU-Spezialisten werden mittels einer Konzepthierarchie spezifiziert, wie sie beispielhaft in Kapitel 4.1.4 angegeben ist. Daher besteht die Benutzerschnittstelle der Konfigurationskomponente im wesentlichen aus einem Konzepteditor, der zum einen die Beziehungen der verwendeten Konzepte

anzeigen und zum anderen Elemente der verwendete Konzeptsprache einfügen kann. Bestandteile dieser Konzeptsprache sind explizite Subsumptionsbeziehungen, Attribute, Integritätsbedingungen (Constraints) zwischen Attributen und eine Partonomie und explizite Äquivalenz von Konzeptausdrücken, die mittels eines normalisierten Termersetzungssystems zur Normalformbildung in der Wissensbasis verwendet werden. Ergebnis ist eine abstrakte Beschreibung spezieller DAU-Aktivitäten durch Mengen kompatibler Systemkonfigurationen. Diese enthalten die verwendeten DAU-Spezialisten inklusive ihrer „Verschaltung" und Parametrisierung.

5.2.4.1 Methode: ViewConcepts

Argumente: Datenbank der definierten Konzeptbeschreibungen

Ausgabe: Graphische bzw. Browser-Darstellung der taxonomischen Hierarchie der definierten DAU-Aktivitäten, ihrer Attribute und ihrer Komponenten

Beschreibung: Über diese Methode navigiert der Systemoperator in der Konfigurationsphase, um sich einen Überblick über die bislang definierten DAU-Aktivitäten zu verschaffen und ggf. anwendungsspezifische Spezialisierungen durchführen zu können.

5.2.4.2 Methode: AddConcept

Argumente: Eindeutiger Name eines neuen Konzepts einer DAU-Aktivität

Ausgabe: Veränderte Hierarchie der definierten Konzeptbeschreibungen

Beschreibung: In der Konfigurationsphase fügt der Systemoperator mittels dieser Methode ein neues Konzept (neue DAU-Aktivität) in die Datenbank ein.

5.2.4.3 Methode: AddConstraint

Argumente: Konzept einer DAU-Aktivität, ein Vektor mit Attributnamen oder Verweis auf die Kardinalität einer Komponente, Beschreibung der Extension des Constraints

Ausgabe: Darstellung der Beschreibung des modifizierten Constraints

Beschreibung: Mittels dieser Methode führt der Systemoperator eine Integritätsbedingung zwischen Attributen des angegebenen Konzepts ein. Es können auch ererbte Attri-

bute bezeichnet werden. Durch Verweis auf die Kardinalität einer Komponente kann auch das Vorkommen einer Komponente in Abhängigkeit der Ausprägung anderer Attribute eingeschränkt werden.

5.2.4.4 Methode: AddConfigurationManually

Argumente: Konzept einer DAU-Aktivität, eine mit dieser Aktivität kompatible Systemkonfiguration

Ausgabe: Systemkonfiguration zur angegebenen Aktivität, wenn eine berechnet wurde. Ansonsten werden zu dem angegebenen Konzept ähnliche Aktivitäten ausgegeben, die realisiert werden können.

Beschreibung: Die angegebene Systemkonfiguration wird auf Kompatibilität mit der angegebenen Aktivität getestet und, wenn sie kompatibel ist, in die Datenbank ausführbarer Systemkonfigurationen aufgenommen.

5.2.4.5 Methode: AddConfigurationAutomatically

Argumente: Konzept einer DAU-Aktivität

Ausgabe: Systemkonfiguration zur angegebenen Aktivität, wenn eine berechnet wurde. Ansonsten werden zu dem angegebenen Konzept ähnliche Aktivitäten ausgegeben, die realisiert werden können.

Beschreibung: Die automatische Konfigurationskomponente ermittelt aus der Beschreibung der bekannten DAU-Spezialisten und der angegebenen DAU-Aktivität eine konkrete Aggregation der DAU-Spezialisten (Systemkonfiguration). Der berechnete Ausführungsplan wird in die Datenbank ausführbarer Systemkonfigurationen aufgenommen.

5.2.4.6 Methode: AddSubsumptionRelation

Argumente: Name des Oberkonzepts einer DAU-Aktivität, Name des Unterkonzepts. Beide Konzepte müssen in der Datenbank der Konzeptbeschreibungen von DAU-Aktivitäten vorkommen und es darf noch keine Subsumptionsbeziehung bestehen.

Ausgabe: Veränderte Hierarchie der definierten Konzeptbeschreibungen

Beschreibung: In die Hierarchie der definierten Konzeptbeschreibungen wird eine neue Subsumptionsbeziehung zwischen zwei Konzepten eingeführt. Mittels dieser Methode ist es dem Systemoperator in der Phase Systemkonfiguration möglich, anwendungsspezifische Spezialisierungen generischer Konzepte von DAU-Aktivitäten einzufügen.

5.2.4.7 Methode: AddPartWholeRelation

Argumente: Name der Zerlegungsrelation, Konzept (DAU-Aktivität) der Aggregate, Konzept der Komponenten (DAU-Aktivität), Unter- und Obergrenze der Kardinalität

Ausgabe: Definierte Teil-/Ganzes-Beziehungen zum angegebenen Konzept der Aggregate.

Beschreibung: Über diese Methode wird eine Partonomie über die angegebenen Konzepte definiert. Wurden unter einem Relationsnamen mehrere Zerlegungen eingegeben, so handelt es sich um alternative Zerlegungen.

5.2.4.8 Methode: ModifyAttribute

Argumente: Name des zu modifizierenden Attributs, Name des modifizierten Konzepts einer DAU-Aktivität, Datentypbezeichnung für das Attribut

Ausgabe: Darstellung des modifizierten Konzepts

Beschreibung: Durch diese Methode kann der Systemoperator in der Konfigurationsphase den Attributvektor eines Objektes modifizieren. Kennt das Konzept den angegebenen Attributnamen nicht, so wird es neu angelegt. Ansonsten wird der Datentyp des bestehenden Attributs entsprechend verändert.

5.2.4.9 Methode: AddEquivalency

Argumente: Zwei Konzeptbeschreibungen und eventuell eine Reduktionsrichtung

Ausgabe: durch Aufruf von *NormalizeKB* normalisierte Wissensbasis.

Beschreibung: Von nun an gelten die beiden genannten Konzeptbeschreibungen als äquivalent. Diese Gleichsetzungen werden in der Methode *NormalizeKB* verwendet.

5.2.4.10 Methode: TransformDocConcept

Argumente: Ein Konzeptname zu einem Nachrichtentyp

Ausgabe: Eine mit dem strategischen Wissen kompatible Beschreibung des Nachrichtentyps.

5.2.4.11 Methode: NormalizeKB

Argumente: Wissensbasis

Ausgabe: normalisierte Wissensbasis

Beschreibung: Normalisierung der Wissensbasis in eine logisch äquivalente, aber kompaktere und effizienter zu verarbeitende Form.

5.2.5 DAU-Monitor

Wesentliche Aufgabe des DAU-Monitors ist die Visualisierung des aktuellen Systemverhaltens. Dazu gehören aktive DAU-Spezialisten inklusive der Ausprägungen ihrer Parameter und ihrer Ein- bzw. Ausgabebeziehungen.

5.2.5.1 Methode: ViewSystemConfiguration

Argumente: –

Ausgabe: Browser-Darstellung der aktuellen Systemkonfiguration mit aktiven DAU-Spezialisten, ihren Parametern und den aktuellen Ein- bzw. Ausgabebeziehungen. Die Anzeige umfaßt analog zum `ps`-Werkzeug des UNIX-Systems Laufzeitdaten wie aufgewendete Speicher- und Rechenzeitressourcen sowie interne Statusmeldungen der DAU-Spezialisten in der aktuellen Konfiguration.

Beschreibung: Diese Methode implementiert im wesentlichen die Benutzerschnittstelle Monitoring (vgl. Abbildung 3).

5.2.5.2 Methode: ChangeSystemConfiguration

Eingabe: Systemkonfiguration, aktive DAU-Aktivität

Ausgabe: –

Beschreibung: Analog zu Kapitel 5.2.4.4 wird hier eine Systemkonfiguration festgelegt. Hier jedoch wird diese Konfiguration für die angegebene aktive DAU-Aktivität vorgeschrieben und eventuell berechnete Zwischenergebnisse werden gelöscht, sofern sie nicht in der angegebenen neuen Systemkonfiguration verwendet werden. Zusätzlich können in dieser Schnittstelle mittels der Methode *AddConstraint* (vgl. Kapitel 5.2.4.3) neue Integritätsbedingungen in die Konzeptbeschreibungen eingeführt werden. Dadurch kann die erneute Berechnung fehlerhafter Systemkonfigurationen verhindert werden.

5.2.6 Verifikations-GUI

Die Verifikations-GUI wurde im Anforderungsdokument noch gemeinsam mit der Akquisitions-GUI betrachtet. Mittlerweile hat sich eine Trennung als sinnvoll erwiesen, da sich die Funktionalitäten dieser beiden Klassenkategorien und die beteiligten Rollen stark voneinander unterscheiden.

Die Verifikations-GUI dient dem normalen Sachbearbeiter zur Überprüfung der DAU-Ergebnisse.

5.2.6.1 Methode: VerifyResults

Argumente: Brutto-TIFF Bild (ohne Bildbereinigung), Texterkennungsergebnisse, Ergebnistemplate, Vorgangszuordnung, Erwartungshaltung

Ausgabe: Korrigierte Texterkennungsergebnisse, Korrigiertes Ergebnistemplate, Korrigierte Vorgangszuordnung

Beschreibung: Wird durch einen Benutzer aufgerufen. Angezeigt werden nach dem Ablauf der DAU Brutto-Bild, Texterkennungsergebnisse, die extrahierte Information in Form des Ergebnistemplates und die Vorgangszuordnung. Der Benutzer kann mittels einer intuitiven, einfachen Oberfläche diese Daten korrigieren. Korrigierte Daten werden in der Workflow-Datenbank abgelegt. Zudem werden sie mit den fehlerbehafteten Daten

in einer Log-Datei zwischengespeichert, damit der autorisierte Benutzer später darauf zugreifen kann. Außerdem können sie auch direkt an Lernkomponenten lernfähiger DAU-Spezialisten weitergeleitet werden, die daraufhin ihre Wissensbasen anpassen.

5.2.7 Akquisitions-GUI

Der genaue Zeitpunkt der Akquisition kann auf verschiedene Arten spezifiziert werden:

- Der autorisierte Benutzer möchte das Dokument- oder das Spezialistenwissen erweitern, da die Domäne erweitert wird (z. B. „zukünftig werden Briefe eines neuen Typs oder eines neuen Lieferanten eintreffen“ oder „zukünftig müssen andere Informationen gefunden werden“) oder
- in einem eigenen Workflow wurde zu bestimmten Zeiten die Akquisition als Aktivität definiert oder
- die Log-Datei (vgl. Kapitel 4.4.1) hat eine bestimmte Größe erreicht.

Welche der drei Möglichkeiten gewählt werden bzw. tatsächlich erlaubt sind, modelliert der Workflow-Designer im Workflow. Im ersten Fall benötigt der autorisierte Benutzer einen direkten Zugang zum Dokumentwissen, um es abzuändern oder zu erweitern (On-Line Teaching), oder er muß die benötigten Eingabedaten für die Lernkomponenten bestimmter Spezialisten bereitstellen und dann die Lernkomponenten aktivieren (Off-Line-Learning).

Im zweiten und dritten Fall bearbeitet der autorisierte Benutzer alle Einträge der Log-Datei, d. h., er bearbeitet alle die DAU-Fehler, die sich auf die jeweiligen Endergebnisse ausgewirkt haben. Dazu kann er sich über eine graphische Schnittstelle Dokumentwissen und alle relevanten Zwischenergebnisse ausgeben lassen. Dadurch kann er Fehlerquellen zurückverfolgen und zukünftig durch die Akquisition neuen Wissens ausschalten. Auch dies kann sowohl über On-Line Teaching als auch über Off-Line Learning geschehen. Der erste Fall wird allerdings die Regel sein.

5.2.7.1 Methode: CheckAll

Argumente: DokID

Ausgabe: Ursprüngliches Ergebnistemplate, Korrigiertes Ergebnistemplate, Relevante DAU-Zwischenergebnisse, Dokumentwissen

Beschreibung: Der autorisierte Benutzer kann sich gleichzeitig über eine graphische Schnittstelle die ursprünglichen DAU-Ergebnisse, die korrigierten DAU-Ergebnisse, das Dokumentwissen und alle Zwischenergebnisse ausgeben lassen. Dadurch kann er Fehlerquellen zurückverfolgen.

5.2.7.2 Methode: AcquireLexicalKnowledge

Argumente: Wort, Lexikalische Eigenschaft

Ausgabe: –

Beschreibung: Der autorisierte Benutzer kann hier neue Wortinformationen eingeben. Dies sind Synonym-Beziehungen und Hyper-/Hyponym-Beziehungen sowie morphologische Wortinformationen. Die Daten werden in einer Datenbank abgelegt.

5.2.7.3 Methode: AcquirePatterns

Argumente: Eine oder mehrere ROIs, Bezeichner für ROIs, zu extrahierende Informationseinheiten innerhalb der ROIs, feste Bestandteile einer Wortsequenz

Ausgabe: –

Beschreibung: Der autorisierte Benutzer kann hier Regionen kennzeichnen, in denen bestimmte Wortsequenzen (Pattern) gefunden und bestimmte Informationseinheiten extrahiert werden sollen. Die Region wird mit bisher generierten generischen Pattern verglichen. Abhängig von der Ähnlichkeit zu bestehenden Pattern wird entweder ein bestehendes Pattern erweitert oder ein neues Pattern generiert.

5.2.7.4 Methode: AcquireNewClass

Argumente: Eine beliebige Menge von Dokumenten (TIFF-Bild oder ASCII-Text), Name für Dokumentklasse, evtl. Klassenfeatures.

Ausgabe: –

Beschreibung: Der autorisierte Benutzer kann für eine beliebige Menge von Dokumenten eine neue Klasse festlegen (z. B. Nachrichtentyp, Produktkategorie, Formulartyp/-nummer). Aus den Eingabedaten wird eine Erweiterung der bestehenden Regeln für diese neue Klasse generiert.

5.2.7.5 Methode: AcquireNewGraphicalDocumentFeatures

Argumente: Dokument als TIFF-Bild, Formular- und Tabelleneigenschaften des Dokuments in Form einer Liste mit Namen für Regions-Of-Interest (ROIs), deren geometrische Position und graphischen Eigenschaften

Ausgabe: –

Beschreibung: Der autorisierte Benutzer kann die Eigenschaften, die für ein Referenzdokument für Tabellen- und Formularerkennung benötigt werden, eingeben. Dazu gehören u. a. Angaben über Linien, geometrische Positionen von ROIs, mögliche Farben des Vorder-/Hintergrunds und die Verteilung von Farben im Dokument. Diese Angaben werden gespeichert.

5.2.7.6 Methode: AcquireNewData

Argumente: Datenbankbezeichner, Eintrag

Ausgabe: –

Beschreibung: Der autorisierte Benutzer kann die bestehenden, firmenspezifischen Datenbanken hier erweitern (z. B. Adressen, Leute, Firmen, Produkte). Die Daten werden in der jeweiligen Datenbank abgelegt.

5.2.8 DAU-Spezialisten

Die DAU-Spezialisten lösen Teilaufgaben der Dokumentanalyse. Sie werden im folgenden getrennt nach ihren Aufgaben aufgelistet. Dabei wird unter einem Spezialisten auch immer seine (evtl. vorhandene) Lernkomponente subsumiert.

Es gibt drei Arten von Argumenten:

- aktuelles Dokument (mehrseitige TIFF-Bilder, bisherige Analyseergebnisse, ...)
- aktuelles sonst. Umfeld (Erwartungshaltung, Dokumentwissen),
- sonst. Parameter (eigene Wissensbasen, Steuerungsinfo für den reaktiven Planer, ...)

Die ersten beiden Argumente werden jedem Spezialisten übergeben. Aus diesem Grund werden sie bei den Argumenten der Spezialisten nicht mehr aufgelistet.

5.2.8.1 Bildverarbeitung I – Bildaufbereitung und Merkmalsextraktion

Argumente: Binär-, Grauwert- oder Farb-TIFFs

Ausgabe: aufbereitetes TIFF, Drehwinkel, Linienbeschreibung, Texturen, Hintergrundbeschreibung

Beschreibung: Schmutzpartikel werden entfernt, Schiefstellung des Dokumentes wird behoben, Linien werden entfernt, Texturen und Farbhintergründe werden entfernt.

5.2.8.2 Logoerkennung

Argumente: Binär-, Grauwert- oder Farb-TIFFs

Ausgabe: Wenn Logo gefunden und erkannt wird, wird der Absender geliefert. Ansonsten kein Output.

Beschreibung: Logo wird an den Stellen gesucht, die durch das Dokumentwissen spezifiziert sind. Fehlt dieses Wissen, muß zuvor der Schritt Logofinden durchgeführt werden.

5.2.8.3 Segmentierung

Argumente: Binär-, Grauwert- oder Farb-TIFFs

Ausgabe: Objekthierarchie aus Block, Zeile, Wort, Zusammenhangsgebiete; Merkmale: Text/Nichttext, Fax-/Scan-Document, Fontattribute, Zeilenwinkel, etc.

Beschreibung: Erzeugung einer Objekthierarchie von dem eingehenden TIFF, wobei parallel eine Klassifikation in Text/Graphik und in Fax-/Scan-Dokument stattfindet. Die letztere Klassifikation kann dabei sowohl auf der Erkennung des spezifischen Datenformats als auch auf einer Faxkopf-Erkennung beruhen.

5.2.8.4 Bildverarbeitung II – Aufbereitung für die OCR

Argumente: TIFF, Objekthierarchie des Layouts

Ausgabe: TIFF (je nach Erkenner ein Binärbild oder ein Grauwertbild), evtl. Zonefile

Beschreibung: Aus dem TIFF und der Objekthierarchie wird ein TIFF, evtl. mit Zonefile, für die OCR generiert. Das Format basiert – je nach Anforderung der OCR – entweder auf Binär- oder Grauwertdaten.

5.2.8.5 Formularerkennung

Argumente: Zeilen (nur als Layoutobjekte)

Ausgabe: Formulartyp und a-priori-Blockstruktur: Bekannte Fontattribute im Sinne einer ROI (Hintergrund, Vordergrund, Farbe, OCR, Fontattribute, syntaktische und semantische Label).

Beschreibung: Basierend auf den Zeilenkoordinaten des Dokuments wird durch einen Vergleich mit den Zeilenstrukturen der bekannten Formulare (Dokumentwissen) der Formulartyp erkannt.

5.2.8.6 OCR

Argumente: Objekthierarchie von Block, Zeile, Wort.

Ausgabe: Objekthierarchie-Zeichen: Liste von Alternativen

Beschreibung: Kommerzielle OCR-Komponenten führen die Erkennung der Zeichen durch.

5.2.8.7 OCR-Voting

Argumente: Wort mit Alternativen pro Zeichen

Ausgabe: „beste" Hypothese pro Zeichen

Beschreibung: Das Voting ermittelt aus Zeichenalternativen verschiedener Erkenner die wahrscheinlichste Zeichenhypothese.

5.2.8.8 Lexikalischer Abgleich

Argumente: Zeichenhypothesengraph

Ausgabe: gültiges Wort aus Lexikon oder gemäß FSA-Struktur

Beschreibung: Zeichenhypothesen werden zu gültigen Wörtern zusammengefaßt.

5.2.8.9 Worttagging/Morphologie

Argumente: textuell erkanntes Wort als Buchstabenfolge (evtl. mit Alternativen)

Ausgabe: Wort mit Attributen (Wortgrundform, weitere morphologische Informationen, (morpho-) syntaktische Kategorie)

Beschreibung: Die morphologischen Attribute des Wortes werden berechnet.

5.2.8.10 Strukturparser E-Mail

Argumente: E-Mail

Ausgabe: Blockstruktur (Zeile/Wort) mit semantischen Labeln

Beschreibung: Eine E-Mail wird in ein Format transformiert, welches zusammen mit den Ergebnissen der Bildanalyse einer einheitlichen Nachbearbeitung zugeführt werden kann.

5.2.8.11 Strukturparser HTML

Argumente: HTML

Ausgabe: Blockstruktur (Zeile/Wort) mit semantischen Labeln und Layoutinformation

Beschreibung: Eine HTML-Seite wird in ein analysierbares Format transformiert.

5.2.8.12 Labeling für bestimmte Objekte

Argumente: Gesamtdokument mit Worthypothesen, spezifische Information zur gesuchten ROI, z. B. eine Wortliste (logisches Wörterbuch), es sind in Abhängigkeit von der Güte der Komponente auch andere Argumente denkbar

Ausgabe: eine oder mehrere etikettierte ROIs

Beschreibung: ROIs werden erkannt.

5.2.8.13 Informationsextraktion mittels Parsing

Argumente: Gesamtdokument mit Ergebnissen des passenden lexikalischen Abgleichs und ROIs, evtl. Verweis auf die zu verwendende Grammatik

Ausgabe: je nach Bedarf syntaktische und/oder semantische Analyse eines interessierenden Teils

Beschreibung: Führt eine durch die Grammatik genau spezifizierte Analyse eines Textteils durch, die als Ergebnis einen syntaktischen Parsebaum und/oder die semantische Analyse des entsprechenden Bereichs liefert.

5.2.8.14 Informationsextraktion mittels Pattern-Matching

Argumente: Gesamtdokument mit Ergebnissen des passenden lexikalischen Abgleichs und ROIs sowie Koordinatenangaben und Fontattribute von Wörtern, evtl. Patterndatei mit Angabe der zu findenden Pattern

Ausgabe: Gefundene Pattern mit Stellenangabe (=Syntax) und extrahierten Informationsteilen innerhalb einer Case-Frames (=Semantik)

Beschreibung: Vorher spezifizierte, relevante Informationen werden aus dem Dokument extrahiert. Die Patterndatei wird explizit angegeben, wenn es nicht erwünscht ist, daß sie aus dem Dokumentwissen automatisch generiert wird.

5.2.8.15 Regelbasierte Dokumentklassifikation

Argumente: Gesamtdokument mit Ergebnissen des passenden lexikalischen Abgleichs und ROIs sowie Koordinatenangaben und Fontattribute von Wörtern, Regeldatei

Ausgabe: Gewichtete Liste von Klassennamen

Beschreibung: Dem Dokument wird eine inhaltliche Beschreibung (Klasse) zugeordnet. Die Regeldatei wird explizit angegeben, wenn es nicht erwünscht ist, daß sie aus dem Dokumentwissen automatisch generiert wird.

5.2.8.16 Tabellenanalyse

Argumente: Gesamtdokument mit Ergebnissen des passenden lexikalischen Abgleichs und ROIs sowie Koordinatenangaben von Wörtern, Linienangaben

Ausgabe: Segmentierte Tabelle (Zellen mit Koordinatenangaben), Informationen, die in den Spalten enthalten sind (z. B. Preis)

Beschreibung: Im Dokument enthaltene Tabellen werden in ihre Bestandteile segmentiert.

5.3 Zusätzliche relevante Klassenkategorien

Zusätzlich zu den aus den Systemanforderungen abgeleiteten Klassenkategorien wurden weitere identifiziert, die nach außen hin nicht sichtbar sind und in keiner Kommunikation mit den Rollen stehen.

5.3.1 Reaktiver Planer

Der reaktive Planer besteht im wesentlichen aus den Inferenzdiensten, die im Kapitel 5.2.4 zur Unterstützung der Konfigurationsphase genannt wurden. Dazu gehört z. B. die Herstellung lokaler Konsistenz zwischen den Eigenschaften erweiterter Konzeptbeschreibungen, Normalformbildung usw. Diese Methoden werden von der Konfigurationskomponente übernommen. Lediglich die Vorgangszuordnung, die Übertragung von Daten, das Ausschreiben von Daten und die Kontrolle der Suche sind spezielle Aufgabenfelder dieser Klassenkategorie.

Die Postkorbbearbeitung startet eine reaktive Planerinstanz beim Eintreffen eines Dokumentes. Bezüglich des Ergebnisses der Vorgangszuordnung ergeben sich folgende Szenarien:

- Das Dokument kann keinem offenen Vorgang zugeordnet werden und wird damit an eine Standarderwartung gereicht. Dieser Standarderwartung entsprechend muß eine neue Workflow-Instanz initiiert werden. Normalerweise dürfte in diesem Geschäftsprozeß eine Korrektur der DAU-Ergebnisse mit Hilfe der Verifikations- und Akquisitionsschnittstelle durchgeführt werden.
- Das Dokument kann einem offenen Vorgang (d. h. einer bestehenden Workflow-Instanz) zugeordnet werden: Ergebnisgenerierung und Information werden durch das Workflow-Modul an die Workflow-Instanz weitergereicht.

5.3.1.1 Methode: InitializePlanning

Argumente: Erwartungshaltung (`Believe`-Daten)

Ausgabe: Umwandlung in internes Datenformat der Planung

Beschreibung: –

5.3.1.2 Methode: GeneratePlan

Argumente: Zeitlimit, Konzeptbeschreibung einer DAU-Aktivität

Ausgabe: Instanz dieser Aktivität

Beschreibung: –

5.3.1.3 Methode: PlanAndRun

Argumente: Erwartungshaltung (`Believe`-Daten), Standarderwartung (optional)

Ausgabe: Bildung des Ergebnistemplates und Rückschreiben der errechneten Ergebnisse

Beschreibung: Wird gestartet, um eine DAU-Aktivität durchzuführen. Verwendet die anderen Methoden dieser Kategorie als Unterprogramme.

5.3.1.4 Methode: CallSpecialist

Argumente: Plan einer DAU-Aktivität (aktuelle Systemkonfiguration), Zeiger auf aktuell auszuführende DAU-Aktion

Ausgabe: –

Beschreibung: Zentrale Ausführung des aktuellen Plans.

5.3.1.5 Methode: TerminatePlanning

Argumente: –

Ausgabe: Bildung des Ergebnistemplate und Rückschreiben der errechneten Ergebnisse.

Beschreibung: –

5.3.2 Workflow-Modul

Das Workflow-Modul (Wf-Modul) im reaktiven Planer ist zuständig für die Kommunikation zwischen Planer und Vermittler bzw. WFMS (siehe Abbildung 7). Dabei ist das

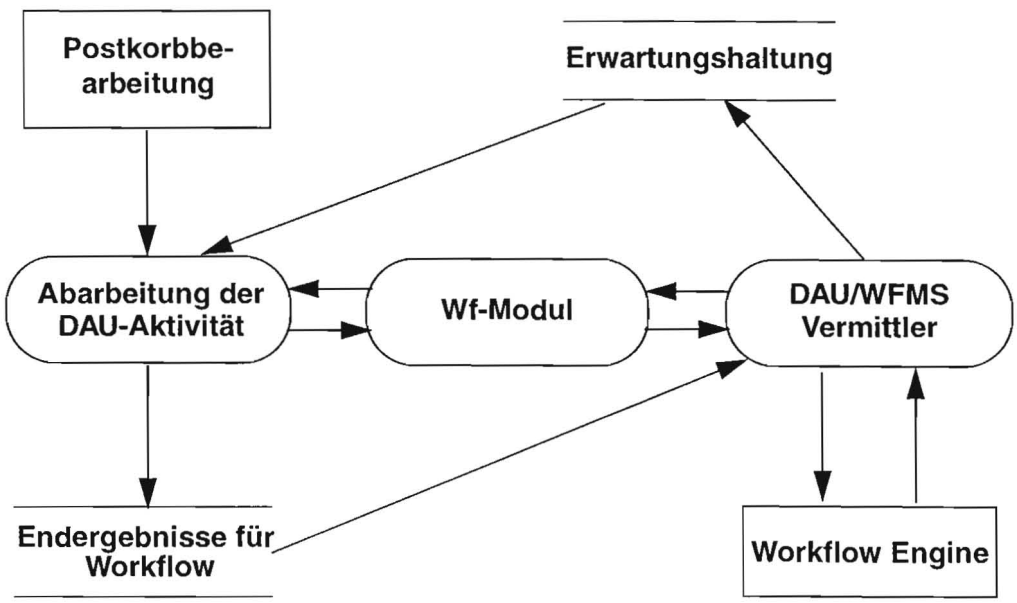

Abbildung 7: reaktiver Planer/WFMS

Modul unabhängig von dem verwendeten WFMS, da die Kommunikationsschnittstelle

zwischen dem Workflow-Modul und den Vermittlern allgemeingültig gehalten wird (siehe Abbildung 8). Im Unterschied dazu ist ein Teil des DAU/WFMS-Vermittlers WFMS-spezifisch und die Vermittler wenden sich direkt an das WFMS bzw. werden direkt von ihm aufgerufen. Die Aktivität *Abarbeitung der DAU-Aktivität* steht für die gesamte Phase 2, wie sie in Abbildung 3 dargestellt wurde.

Das Workflow-Modul übernimmt alle WFMS-bezogenen Aktionen und Anfragen, so daß der Planer kein Wissen über das WFMS haben muß. Das Workflow-Modul erhält vom Planer Aufträge und Meldungen (wie etwa *Endergebnis für DAU-Task liegt vor*), und ist dann verantwortlich für die Übergabe und die Umsetzung der Ergebnisse im WFMS. Gleiches gilt für die umgekehrte Richtung. Die Aufträge des WFMS an den reaktiven Planer (bzw. aus WFMS Sicht an das DAU-System) werden vom Workflow-Modul in WFMS-unabhängige Aufträge umgesetzt.

Die Aktivitäten eines Prozesses im WFMS melden ihre Erwartungen über den DAU/WFMS-Vermittler an (d. h. führen das Programm aus und übergeben ihm den Datensatz). Der Vermittler trägt diese Erwartung zusammen mit Informationen aus dem Workflow (ProzeßID, AktivitätsID, Lieferschein von Firma XY, ...) in die *Erwartungshaltung* ein.

Der reaktive Planer greift nach Bedarf auf die *Erwartungshaltung* zu und liest den aktuellen Zustand aus. Für das Löschen zugeordneter bzw. nicht mehr benötigter Datensätze in der Erwartungshaltung ist die jeweilige Workflow-Instanz zuständig, da nur diese entscheiden kann, wann eine Erwartungshaltung befriedigt ist (da die Verifikation erst in der Workflow-Instanz stattfindet). In unserem Beispiel im Workflow *Bestellung* wird eine Erwartungshaltung gelöscht in dem Fall, daß Rechnung und Lieferschein eingetroffen sind – aber noch keine Auftragsbestätigung. Man kann dann davon ausgehen, daß auch keine Auftragsbestätigung mehr eintreffen wird. Die bereits gesetzte Erwartungshaltung wird durch einen Vermittler gelöscht. Dasselbe gilt nach einer erfolgreich durchgeführten Verifikation.

Der Vermittler ist nötig, da WFMSe i. d. R. nicht direkt mit externen Programmen kommunizieren können. Je nach Architektur des WFMS wird der Vermittler beendet, der

Workflow suspendiert und ein Ereignis generiert oder aber der Vermittler wartet solange, bis das Ergebnis vorliegt und dieser vom Workflow-Modul benachrichtigt wurde.

Die Endergebnisse, die dem WFMS wieder zugänglich gemacht werden sollen bzw. die angefordert wurden, werden in die Ablage *Endergebnisse für Workflow* geschrieben. Das Workflow-Modul macht dann diese Daten dem jeweiligen Vermittler zugänglich bzw. referenziert diese.

5.3.2.1 Methode: QueryErwartungshaltung

Argumente: –

Ausgabe: aktuelle Erwartungshaltung

Beschreibung: Liefert die aktuelle Erwartungshaltung.

5.3.2.2 Methode: DeleteErwartungshaltung

Argumente: ID eines Eintrages in der Erwartungshaltung

Ausgabe: –

Beschreibung: löscht eine bereits zugeordnete (befriedigte) Erwartungshaltung

5.3.2.3 Methode: ContactVermittler

Argumente: AktivitätsID oder EreignisID, ProzeßID, Verweis auf Ergebnis

Ausgabe: –

Beschreibung: Kontaktiert einen DAU/WFMS-Vermittler, der das Ergebnis dem Workflow übergeben soll.

5.3.2.4 Methode: RequestDAUTask

Argumente: DokID, DAUTask, AktivitätsID oder EreignisID, ProzeßID

Ausgabe: Referenz auf Ergebnis des DAUTask

Beschreibung: Fordert vom reaktiven Planer eine DAUTask an. Diese Methode wird vom DAU/WFMS-Vermittler aufgerufen.

5.3.2.5 Methode: EndDAUSignal

Argumente: DokID oder Liste von DokIDs

Ausgabe: –

Beschreibung: Fordert die betroffenen reaktiven Planer auf, sich zu terminieren und ihre Daten in der Zwischenablage zu löschen. Diese Methode wird vom DAU/WFMS-Vermittler aufgerufen.

5.3.3 DAU/WFMS-Vermittler

Die Vermittler dienen als Schnittstelle zwischen reaktivem Planer und WFMS. Das WFMS wendet sich immer mit seinen Aufträgen an einen Vermittler, d. h. der reaktive Planer ist für das WFMS nicht sichtbar. Damit erreicht man, daß der reaktive Planer das WFMS nicht direkt ansprechen muß. Der reaktive Planer wendet sich an sein Workflow-Modul, das sich dann an einen bestimmten Vermittler wendet. Eventuell ist ein Server für die Kommunikation vom reaktiven Planer hin zum WFMS sinnvoll, z. B., wenn keine offene Aktivität existiert oder ein Vermittler wartet.

Damit der Vermittler unabhängig von einem konkreten WFMS genutzt werden kann, ist er modular aufgebaut und beinhaltet ein WFMS-spezifisches Modul, das die Verbindung zum jeweiligen WFMS übernimmt (siehe Abbildung 8).

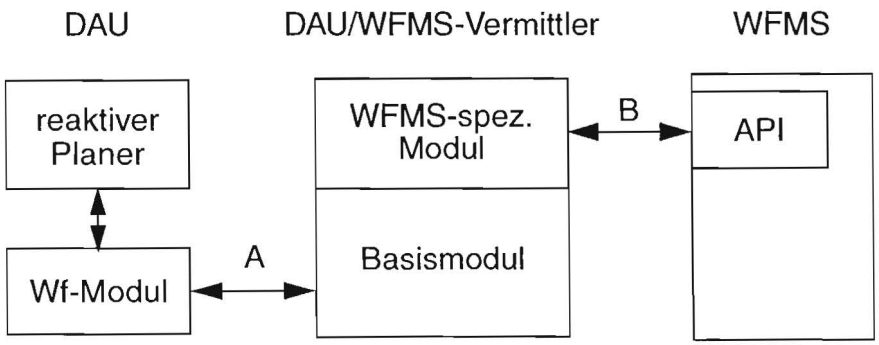

Abbildung 8: Architektur DAU/WFMS Vermittler

Kommunikationsschnittstellen:

A: Kommunikation in der Regel über Rechnergrenzen hinweg, etwa CORBA (speziell CORBA Workflow Management facility), DCE, Sockets

B: WFMS-spezifisch: eigene API, Datenaustauschschnittstelle, Schnittstelle zu aufgerufenen Programmen (WfMC: *invoked applications*); ideal: WfMC-Interface 3: Invoked Applications, Status: noch kein Draft vorhanden

Es ergeben sich folgende Szenerien für den Einsatz eines Vermittlers:

- **WFMS: Erwarten** eines bestimmten Dokumentes (z. B. Lieferschein und/ oder Rechnung). Der Vermittler trägt diese Erwartung zusammen mit zusätzlichen Informationen aus dem Workflow in die Erwartungshaltung ein.
- **WFMS: Befriedigung/Verfall** einer Erwartung: Workflow-Instanz startet den Vermittler, der den Datensatz aus der Erwartungshaltung löscht.
- **WFMS: DAU-Aktivitätsauftrag:** Anfordern einer bestimmten Information aus einem existierenden Dokument: Aufruf des Vermittlers und Angabe der DokID und der auszuführenden DAU-Aktivität (`DAUTask`). Der Vermittler wendet sich an das Workflow-Modul und übergibt den Auftrag.
- **WFMS: DAU-Ende für ein Dokument:** Mitteilung an das DAU-System, daß ein bestimmtes Dokument nicht mehr benötigt wird. Das DAU-System kann nun die Zwischenablage löschen und die reaktive Planerinstanz beenden.
- **DAU-System: Zuordnung eines Dokumentes** zu einer Workflow-Instanz wegen der Übereinstimmung des Dokumentes mit einer Erwartungshaltung: Workflow-Modul wendet sich an einen Vermittler und übergibt Daten. Der Vermittler ermittelt die korrekte Workflow-Instanz und übergibt Dokument und evtl. Daten.
- **DAU-System: Ergebnisse eines DAU-Aktivitätsauftrages liegen vor:** Das Workflow-Modul wendet sich an einen Vermittler und übergibt Referenz auf Ergebnisse. Der Vermittler wendet sich an die Workflow-Instanz, die den Auftrag aufgegeben hatte und übergibt die Daten.

- **DAU-System: Keine Zuordnung eines Dokumentes möglich:** Das Workflow-Modul wendet sich an einen Vermittler, der neuen Workflow zur Überprüfung startet und Dokument übergibt.

5.3.3.1 Methode: SetErwartungshaltung

Argumente: Erwartung (`Believe`-Daten)

Ausgabe: ID des Eintrages

Beschreibung: trägt eine Erwartung in den Datenspeicher *Erwartungshaltung* ein.

5.3.3.2 Methode: DeleteErwartungshaltung

Argumente: ID eines Eintrages in der Erwartungshaltung

Ausgabe: –

Beschreibung: löscht eine Erwartung im Datenspeicher *Erwartungshaltung*

5.3.3.3 Methode: RequestDAUTask

Argumente: DokID, DAUTask

Ausgabe: –

Beschreibung: Wird von einer Workflow-Aktivität angefordert. Der Vermittler leitet einen solchen Auftrag an das Workflow-Modul weiter, holt das Ergebnis ab und übergibt es der Workflow-Instanz.

5.3.3.4 Methode: SendDAUEndSignal

Argumente: DokID oder Liste von DokIDs

Ausgabe: –

Beschreibung: Wird von einer Workflow-Aktivität angefordert. Der Vermittler leitet das Signal an das Workflow-Modul weiter. Es bedeutet, daß auf die angegebenen Dokumente nicht mehr zugegriffen wird und somit alle Daten gelöscht werden können.

5.3.3.5 Methode: DeliverExpectedDocument

Argumente: Referenz auf Ergebnis in Datenspeicher *Ergebnis für Workflow*

Ausgabe: –

Beschreibung: Wird vom Workflow-Modul aufgerufen. Der Vermittler liest den Datensatz und übergibt das Dokument (und evtl. Ergebnis) an die im Datensatz bezeichnete Workflow-Instanz.

5.3.3.6 Methode: DeliverDAUTaskResult

Argumente: Referenz auf Ergebnis in Datenspeicher *Ergebnis für Workflow*

Ausgabe: –

Beschreibung: Wird von Workflow-Modul aufgerufen. Der Vermittler liest den Datensatz und übergibt das Ergebnis an die im Datensatz bezeichnete Workflow-Instanz.

5.3.3.7 Methode: StartVerifyingProcess

Argumente: Referenz auf Ergebnis in Datenspeicher *Ergebnis für Workflow*

Ausgabe: –

Beschreibung: Startet einen Workflow, in dem das Dokument überprüft werden soll, das keiner offenen Workflow-Instanz zugeordnet werden konnte.

5.3.4 Postkorbbearbeitung

Die Postkorbbearbeitung überprüft periodisch den Eingangspostkorb nach neu eingetroffenen Dokumenten. Je eingegangenem Dokument wird eine reaktive Planer-Instanz mit der Aufgabe der „Vorgangszuordnung“ generiert und dieser das jeweilige TIFF-Bild mit der Erwartungshaltung übergeben.

5.3.4.1 Methode: GetDocument

Argumente: –

Ausgabe: {TIFF-Bild, NULL}

Beschreibung: Nimmt – falls vorhanden – ein Dokument aus dem Postkorb.

5.3.4.2 Methode: StartPlaner

Argumente: TIFF-Bild

Ausgabe: –

Beschreibung: Startet eine neue reaktive Planer-Instanz mit einer (u. U. näher spezifizierten) Aufgabe der Vorgangszuordnung und übergibt das TIFF-Bild und die Erwartungshaltung.

5.3.5 Dokumentwissensrepräsentation

Das Dokumentwissen umfaßt laut Definition strukturelles und relationales Wissen über die Dokumente der Domäne. Dazu zählen primär folgende Wissensportionen:

- Nachrichtentypen: dies sind die verschiedenen Klassen von Dokumenten; z. B. Angebot, Bestellung, Lieferschein, Rechnung, Mahnung;
- Semantische Struktur von Dokumenten: hier wird die inhaltliche Struktur eines Dokuments beschrieben, voraussichtlich in Form sog. *conceptual dependency graphs*;
- Logische Struktur von Dokumenten: dies beinhaltet die syntaktische Struktur gewisser Äußerungen (also auf textueller Ebene);
- Layoutmerkmale von Dokumenten: hier wird neben der geometrischen Position von (logischen) Objekten v. a. Schriftinformation und die Ausrichtung von Absätzen abgelegt;
- Bildinformation: hier werden die verbleibenden bildorientierten Wissensportionen zusammengefaßt, wie z. B. Papierformate, Hintergrundfarben und -texturen sowie Druckerqualitäten.

Zur Kapselung dieser Wissenseinheiten werden sogenannte *frames* definiert, die jeweils eine kognitiv adäquate Modellierung eines interessierenden Objekts darstellen. Dazu noch eine ergänzende Anmerkung zum Beispiel aus Abschnitt 4.1.3 „Dokumentwissen":

Der Frame *Geschäftsbrief* ist ein Nachrichtentyp, der seinerseits Spezialisierungen wie Angebot, Bestellung etc. haben kann. Da ein Nachrichtentyp sich auf ein ganzes Dokument bezieht, werden die Wissensabteilungen Layout und Bild entsprechend gefüllt: typischerweise eine oder mehrere DIN A4 Seiten, beliebiger Druckertyp, meist schwarz auf weiß geschrieben. Zum anderen besteht das Nachrichtenobjekt *Geschäftsbrief* aus logischen Objekten wie Empfänger, Absender und Schreibdatum. Diese werden nun ihrerseits wiederum als Frames definiert, die analog Attribute wie typische Position bzgl. des umgebenden Dokuments, Anzahl der Zeilen usw. enthalten.

Für die Systemarchitektur sind nun vor allem drei Fragen interessant:

- Mit welchen Systemkomponenten muß eine Abstimmung (etwa als Schnittstellendefinition) erfolgen?
- Wie läuft die Kommunikation zwischen dem Modul zur Dokumentwissensrepräsentation und dem restlichen System ab?
- Wie sieht die interne Repräsentation des Dokumentwissens aus?

Hierbei kann einzig der letzte Punkt (bedingt) noch etwas vernachlässigt werden, da er das definitive Darstellungsformat für das Wissen betrifft. Wichtig ist in jedem Fall, welche Arten von Wissen repräsentiert werden und wie diese insgesamt organisiert werden (etwa bzgl. Vererbungshierarchien und Hierarchiearten). Die Dokumentation der Wissensrepräsentationssprache selbst wird in einem extra Dokument beschrieben.

5.3.5.1 Kommunikation mit anderen Klassenkategorien

Die Klassenkategorie zur Repräsentation des Dokumentwissens muß von mehreren anderen Klassenkategorien aus zugreifbar sein. Für viele Klassenkategorien genügt dabei ein rein lesender Zugriff, manche benötigen aber auch die Möglichkeit, das Dokumentwissen zu ändern. In Abbildung 9 sind die Klassenkategorien, die einen Zugriff auf das Dokumentwissen benötigen, graphisch dargestellt.

- die Konfiguration (siehe Abschnitt 5.2.4 auf Seite 38): hier kann es bei der Erstellung der Aktivität-/Konfigurationspaare nötig sein, gewisse Kontexte im Dokumentwissen zu berücksichtigen.

- das Akquisitions-GUI (siehe Abschnitt 5.2.7 auf Seite 44): der autorisierte Benutzer kann über das Akquisitions-GUI die verschiedenen Wissensarten des Systems modifizieren, darunter auch das Dokumentwissen. Da Sinn und Zweck der Akquisition die Erstellung bzw. Korrektur und Verfeinerung von Wissen ist, muß hier ein lesender und schreibender Zugriff erlaubt sein.
- die Spezialisten (siehe Abschnitt 5.2.8 auf Seite 46): Sinn und Zweck des Dokumentwissen ist dessen Verwendung durch die Dokumentanalyse-Spezialisten bei der Analyse. Daher benötigen diese einen lesenden Zugriff auf das Dokumentwissen. Zusätzlich benötigen manche Spezialisten auch einen Schreibzugriff, da deren Lernkomponenten bisweilen neues Wissen akquirieren.
- der reaktive Planer (siehe Abschnitt 5.3.1 auf Seite 51): dieser benötigt für die Zwecke der Planung einen lesenden Zugriff auf das Dokumentwissen.

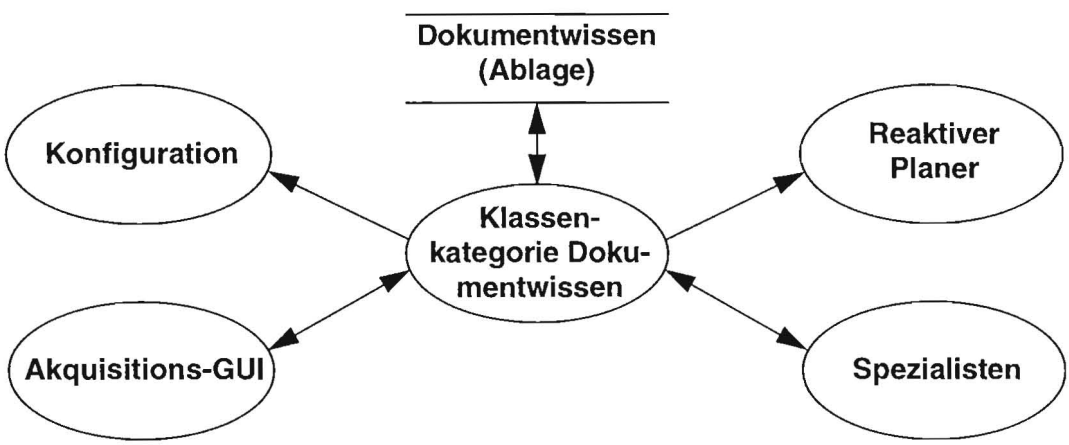

Abbildung 9: Zugriff auf das Dokumentwissen

5.3.5.2 Kommunikationsschnittstelle

Zur Abfrage bzw. Änderung des Dokumentwissens werden einige wenige Methoden zur Verfügung gestellt, die nur äußerst wenig Voraussetzungen über die Struktur des Dokumentwissens machen.

Methode GetTopFrame()

Argumente: –

Ausgabe: ein Frame

Beschreibung: liefert das oberste Element der Frame-Hierarchie im Dokumentwissen.

Methode GetAllFrames()

Argumente: –

Ausgabe: eine Liste von Frames

Beschreibung: liefert eine Liste aller Frames im Dokumentwissen

Methode GetFrameName()

Argumente: ein Frame

Ausgabe: ein String

Beschreibung: liefert den Namen des angegebenen Frames.

Methode GetFrameSuperType()

Argumente: ein Frame

Ausgabe: ein Frame

Beschreibung: liefert zu dem angegebenen Dokument-Frame den nächsten in der Spezialisierunghierarchie übergeordneten Dokument-Frame. Ist die Eingabe gleich `GetTopFrame()`, so wird NIL, Null oder ähnliches zurückgegeben.

Methode GetFrameDescription()

Argumente: ein Frame

Ausgabe: ein String

Beschreibung: liefert die textuelle Beschreibung des gegebenen Frames.

Methode GetFrameAllAttributes()

Argumente: ein Frame

Ausgabe: eine Liste von Attributen

Beschreibung: liefert zu einem gegebenen Frame die Liste aller in ihm definierten Attribute (die selbst eine Klasse sind).

Methode GetFrameAttributeValue()

Argumente: ein Frame, ein Attribut

Ausgabe: eine Liste von Attributwerten

Beschreibung: liefert den Wert des angegebenen Attributs innerhalb des angegebenen Frames; der Wert eines Attributs ist eine Sammlung aus dem Wertebereich dieses Attributs, z. B. gegeben als Liste von Möglichkeiten, sowie deren Wahrscheinlichkeiten und andere Fazetten. Wie solche Attributwerte aussehen, ist noch festzulegen.

Methode GetFrameAllParts

Argumente: ein Frame

Ausgabe: eine Liste von Parts

Beschreibung: liefert zu einem gegebenen Frame die Liste aller ihm untergeordneten Parts (also untergeordnet bzgl. der partonomischen Subordination).

Methode GetFramePartAnnotation()

Argumente: ein Frame, ein Part

Ausgabe: eine Liste von Annotations (Fazetten)

Beschreibung: liefert zu dem angegebenen Frame und dessen Part die Liste aller Annotations, die dort gesetzt sind.

5.3.6 Zwischenablage

Die Zwischenablage speichert alle anfallenden Endergebnisse der DAU-Spezialisten. Die Daten werden, wenn möglich, im Speicher gehalten. Die Zwischenablage stellt die Funktionen bereit, die benötigt werden, um die Ergebnisse der Spezialisten abzulegen bzw. zu laden. Hierzu muß diese Klasse von allen externen Datenstrukturen der einzelnen Spezialisten Kenntnis haben. Weiterhin ist ein Verdrängungsalgorithmus (LRU, Zeitstempelverfahren, etc.) sinnvoll, da man nicht davon ausgehen kann, daß alle Ergebnisse im Speicher gehalten werden können.

5.3.6.1 Methode: InitClipboard

Argumente: –

Ausgabe: Leere Zwischenablage

Beschreibung: Alle Zwischenergebnisse werden aus der Zwischenablage entfernt. Desweiteren wird der Speicher, der durch die Objekte belegt war, wieder freigegeben.

5.3.6.2 Methode: SaveClipboard

Argumente: –

Ausgabe: gesicherte Zwischenablage

Beschreibung: Die Zwischenablage wird auf das BackupDevice gesichert.

5.3.6.3 Methode: LoadClipboard

Argumente: –

Ausgabe: wiederhergestellte Zwischenablage

Beschreibung: Die gesicherte Version wird in die Zwischenablage eingespielt. Daten, die sich vorher in der Zwischenablage befunden haben, sind verloren.

5.3.6.4 Methode: SetBackupDevice

Argumente: neues BackupDevice

Ausgabe:

Beschreibung: Mit dieser Methode wird das Medium, File bzw. Datenbank, für die Sicherung der Zwischenablage festgelegt.

5.3.6.5 Methode: SetSwapDevice

Argumente: neues SwapDevice

Ausgabe:

Beschreibung: Das SwapDevice (File bzw. Datenbank) nimmt alle Zwischenresultate auf, die aus Platzgründen aus der Zwischenablage verdrängt worden sind.

5.3.6.6 Methode: LoadTIFFImage

Argumente: eindeutige Kennung

Ausgabe: Verweis auf ein TIFF-Bild, falls in der Zwischenablage vorhanden

Beschreibung: Diese Methode liefert das TIFF-Bild mit der eindeutigen Kennung zurück, sofern es vorhanden ist.

5.3.6.7 Methode: SaveTIFFImage

Argumente: Verweis auf ein TIFF-Bild

Ausgabe: eindeutige Kennung

Beschreibung: Diese Methode speichert das TIFF-Bild und liefert eine eindeutige Kennung zurück.

5.3.6.8 Methode: LoadOCRResults

Argumente: eindeutige Kennung

Ausgabe: Verweis auf die OCR Ergebnisse, sofern in der Zwischenablage vorhanden.

Beschreibung: Diese Methode liefert die entsprechenden OCR-Ergebnisse zurück.

5.3.6.9 Methode: SaveOCRResults

Argumente: Verweis auf die OCR-Ergebnisse

Ausgabe: eindeutige Kennung

Beschreibung: Diese Methode speichert die OCR-Ergebnisse und liefert eine eindeutige Kennung zurück.

5.3.6.10 Weitere Methoden

Die bisher in diesem Kapitel beschriebenen Methoden decken nur teilweise den Bedarf der DAU-Spezialisten ab. Der weitere Bedarf klärt sich erst, wenn entschieden ist, ob Instanzen des Dokumentwissens zur Ablage von Zwischenergebnissen dienen sollen oder nicht. Diese Entscheidung kann zum jetzigen Zeitpunkt noch nicht getroffen werden.

Anhang 1: Ausprägungen der Datenstrukturen im Beispiel

Erwartungshaltung

Rechnung

```
DokID = -
Nachrichtentyp = Rechnung
Believe[AuftragsNr] = 6025/11111/97
Believe[Firma] = SmartLab
ProzeßID = Bestellung1
AktivitätsID = A1_1
EreignisID = -
DAUTask = -

Ergebnis: DokID = 2333
```

Lieferschein

```
DokID = -
Nachrichtentyp = Lieferschein
Believe[AuftragsNr] = 6025/11111/97
Believe[Firma] = SmartLab
ProzeßID = Bestellung1
AktivitätsID = A2_2
EreignisID = -
DAUTask = -

Ergebnis: DokID = 2334
```

Auftragsbestätigung

```
DokID = -
Nachrichtentyp = Auftragsbestätigung
Believe[AuftragsNr] = 6025/11111/97
Believe[Firma] = SmartLab
ProzeßID = Bestellung1
AktivitätsID = A3_2
EreignisID = -
DAUTask = -
```

Ergebnis: DokID = 2335

DAU-Aktivitätsauftrag

GetRechnungsNr

DokID = 2333

Nachrichtentyp = Rechnung

Believe[]

ProzeßID = PID

AktivitätsID = A1_5

EreignisID = -

DAUTask = GetRechnungsNr

Ergebnis: RechnungsNr = 0815

GetArtikel

DokID = 2334

Nachrichtentyp = Lieferschein

Believe[]

ProzeßID = PID

AktivitätsID: A_4

EreignisID: -

DAUTask: GetArtikel

Ergebnis: Artikel = OfficeAsk-System

Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

-Bibliothek, Information
und Dokumentation (BID)-
PF 2080
67608 Kaiserslautern
FRG

Telefon (0631) 205-35(
Telefax (0631) 205-321

e-mail
dfkibib@dfki.uni-kl.
WWW
http://www.dfki.un
sb.de/dfkibib

Veröffentlichungen des DFKI

Die folgenden DFKI Veröffentlichungen sowie die aktuelle Liste von allen bisher erschienenen Publikationen können von der oben angegebenen Adresse oder (so sie als per ftp erhaeltlich angemerkt sind) per anonymous ftp von ftp.dfki.uni-kl.de (131.246.241.100) im Verzeichnis pub/Publications bezogen werden. Die Berichte werden, wenn nicht anders gekennzeichnet, kostenlos abgegeben.

DFKI Publications

The following DFKI publications or the list of all published papers so far are obtainable from the above address or (if they are marked as obtainable by ftp) by anonymous ftp from ftp.dfki.uni-kl.de (131.246.241.100) in the directory pub/Publications.
The reports are distributed free of charge except where otherwise noted.

DFKI Research Reports

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-95-16
Martin Müller, Tobias Van Roy
Title not set
0 pages

Note: The author(s) were unable to deliver this document for printing before the end of the year. It will be printed next year.

RR-95-15
Joachim Niehren, Tobias Van Roy
Title not set
0 pages

Note: The author(s) were unable to deliver this document for printing before the end of the year. It will be printed next year.

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

1994

RR-94-39
Hans-Ulrich Krieger
Typed Feature Formalisms as a Common Basis for Linguistic Specification.
21 pages

RR-94-38
Hans Uszkoreit, Rolf Backofen, Stephan Busemann, Abdel Kader Diagne,
Elizabeth A. Hinkelman, Walter Kasper, Bernd Kiefer, Hans-Ulrich Krieger,
Klaus Netter, Günter Neumann, Stephan Oepen, Stephen P. Spackman.
DISCO–An HPSG-based NLP System and its Application for Appointment Scheduling.
13 pages

RR-94-37
Hans-Ulrich Krieger, Ulrich Schäfer
TDL - A Type Description Language for HPSG, Part 1: Overview.
54 pages

RR-94-36
Manfred Meyer
Issues in Concurrent Knowledge Engineering. Knowledge Base and Knowledge Share Evolution.
17 pages

RR-94-35
Rolf Backofen
A Complete Axiomatization of a Theory with Feature and Arity Constraints
49 pages

RR-94-34
Stephan Busemann, Stephan Oepen, Elizabeth A. Hinkelman,
Günter Neumann, Hans Uszkoreit
COSMA – Multi-Participant NL Interaction for Appointment Scheduling
80 pages

RR-94-33
Franz Baader, Armin Laux
Terminological Logics with Modal Operators
29 pages

RR-94-31
Otto Kühn, Volker Becker, Georg Lohse, Philipp Neumann
Integrated Knowledge Utilization and Evolution for the Conservation of Corporate Know-How
17 pages

RR-94-23
Gert Smolka
The Definition of Kernel Oz
53 pages

RR-94-20
Christian Schulte, Gert Smolka, Jörg Würtz
Encapsulated Search and Constraint Programming in Oz
21 pages

RR-94-19
Rainer Hoch
Using IR Techniques for Text Classification in Document Analysis
16 pages

RR-94-18
Rolf Backofen, Ralf Treinen
How to Win a Game with Features
18 pages

RR-94-17
Georg Struth
Philosophical Logics—A Survey and a Bibliography
58 pages

RR-94-16
Gert Smolka
A Foundation for Higher-order Concurrent Constraint Programming
26 pages

RR-94-15
Winfried H. Graf, Stefan Neurohr
Using Graphical Style and Visibility Constraints for a Meaningful Layout in Visual Programming Interfaces
20 pages

RR-94-14
Harold Boley, Ulrich Buhrmann, Christof Kremer
Towards a Sharable Knowledge Base on Recyclable Plastics
14 pages

RR-94-13
Jana Koehler
Planning from Second Principles—A Logic-based Approach
49 pages

RR-94-12
Hubert Comon, Ralf Treinen
Ordering Constraints on Trees
34 pages

RR-94-11
Knut Hinkelmann
A Consequence Finding Approach for Feature Recognition in CAPP
18 pages

RR-94-10
Knut Hinkelmann, Helge Hintze
Computing Cost Estimates for Proof Strategies
22 pages

RR-94-08
Otto Kühn, Björn Höfling
Conserving Corporate Knowledge for Crankshaft Design
17 pages

RR-94-07
Harold Boley
Finite Domains and Exclusions as First-Class Citizens
25 pages

RR-94-06
Dietmar Dengler
An Adaptive Deductive Planning System
17 pages

RR-94-05
Franz Schmalhofer, J. Stuart Aitken, Lyle E. Bourne jr.
Beyond the Knowledge Level: Descriptions of Rational Behavior for Sharing and Reuse
81 pages

RR-94-03
Gert Smolka
A Calculus for Higher-Order Concurrent Constraint Programming with Deep Guards
34 pages

RR-94-02
Elisabeth André, Thomas Rist
Von Textgeneratoren zu Intellimedia-Präsentationssystemen
22 Seiten

RR-94-01
Elisabeth André, Thomas Rist
Multimedia Presentations: The Support of Passive and Active Viewing
15 pages

DFKI Technical Memos

1998

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

1994

TM-94-05
Klaus Fischer, Jörg P. Müller, Markus Pischel
Unifying Control in a Layered Agent Architecture
27 pages

TM-94-04
Cornelia Fischer
PAntUDE – An Anti-Unification Algorithm for Expressing Refined Generalizations
22 pages

TM-94-03
Victoria Hall
Uncertainty-Valued Horn Clauses
31 pages

TM-94-02
Rainer Bleisinger, Berthold Kröll
Representation of Non-Convex Time Intervals and Propagation of Non-Convex Relations
11 pages

TM-94-01
Rainer Bleisinger, Klaus-Peter Gores
Text Skimming as a Part in Paper Document Understanding
14 pages

DFKI Documents

1998

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

D-95-01
Susanne Biundo, Wolfgang Tank (Hrsg.)
PuK-95, Beiträge zum 9. Workshop „Planen und Konfigurieren", Februar 1995
169 Seiten

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

1994

D-94-15
Stephan Oepen
German Nominal Syntax in HPSG
— On Syntactic Categories and Syntagmatic Relations —
80 pages

D-94-14
Hans-Ulrich Krieger, Ulrich Schäfer
TDL - A Type Description Language for HPSG, Part 2: User Guide.
72 pages

D-94-12
Arthur Sehn, Serge Autexier (Hrsg.)
Proceedings des Studentenprogramms der 18. Deutschen Jahrestagung für Künstliche Intelligenz KI-94
69 Seiten

D-94-11
F. Baader, M. Buchheit, M. A. Jeusfeld, W. Nutt (Eds.)
Working Notes of the KI'94 Workshop: KRDB'94 - Reasoning about Structured Objects: Knowledge Representation Meets Databases
65 pages

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

D-94-10
F. Baader, M. Lenzerini, W. Nutt, P. F. Patel-Schneider (Eds.)
Working Notes of the 1994 International Workshop on Description Logics
118 pages

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

D-94-09
Technical Staff
DFKI Wissenschaftlich-Technischer Jahresbericht 1993
145 Seiten

D-94-08
Harald Feibel
IGLOO 1.0 - Eine grafikunterstützte Beweisentwicklungsumgebung
58 Seiten

D-94-07
Claudia Wenzel, Rainer Hoch
Eine Übersicht über Information Retrieval (IR) und NLP-Verfahren zur Klassifikation von Texten
25 Seiten

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

D-94-06
Ulrich Buhrmann
Erstellung einer deklarativen Wissensbasis über recyclingrelevante Materialien
117 Seiten

D-94-04
Franz Schmalhofer, Ludger van Elst
Entwicklung von Expertensystemen: Prototypen, Tiefenmodellierung und kooperative Wissensevolution
22 Seiten

D-94-03
Franz Schmalhofer
Maschinelles Lernen: Eine kognitionswissenschaftliche Betrachtung
54 Seiten

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

D-94-02
Markus Steffens
Wissenserhebung und Analyse zum Entwicklungsprozeß eines Druckbehälters aus Faserverbundstoff
90 pages

D-94-01
Josua Boon (Ed.)
DFKI-Publications: The First Four Years
1990 - 1993
75 pages

D-98-01
Document

Architektur für ein System zur Dokumentanalyse im Unternehmenskontext
—
Integration von Datenbeständen, Aufbau- und Ablauforganisation

Stephan Baumann, Jürgen Lichter, Michael Malburg,
Heiko Maus, Harald Meyer auf'm Hofe, Claudia Wenzel