Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

Document

D-98-02

Techniques for Organizational Memory Information Systems

Andreas Abecker, Ansgar Bernardi, Knut Hinkelmann,
Otto Kühn, Michael Sintek

February 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

Techniques for Organizational Memory Information Systems

Andreas Abecker, Ansgar Bernardi, Knut Hinkelmann,
Otto Kühn, Michael Sintek

DFKI-D-98-02

This work has been supported by a grant from The Federal Ministry of Education, Science, Research, and Technology (FKZ ITW-9705).

ISSN 0946-0098

Techniques for Organizational Memory Information Systems

Andreas Abecker, Ansgar Bernardi, Knut Hinkelmann,
Otto Kühn, Michael Sintek

German Research Center for Artificial Intelligence — DFKI GmbH
P.O.Box 2080, D–67608 Kaiserslautern, Germany
Phone: +49-631-205-3467, Fax: +49-631-205-3210
{`aabecker,bernardi,hinkelma,kuehn,sintek`}`@dfki.uni-kl.de`

Abstract

The KnowMore project aims at providing active support to humans working on knowledge-intensive tasks. To this end the knowledge available in the modeled business processes or their incarnations in specific workflows shall be used to improve information handling. We present a representation formalism for knowledge-intensive tasks and the specification of its object-oriented realization. An operational semantics is sketched by specifying the basic functionality of the Knowledge Agent which works on the knowledge intensive task representation.

The Knowledge Agent uses a meta-level description of all information sources available in the Organizational Memory. We discuss the main dimensions that such a description scheme must be designed along, namely information content, structure, and context. On top of relational database management systems, we basically realize deductive object-oriented modeling with a comfortable annotation facility. The concrete knowledge descriptions are obtained by configuring the generic formalism with ontologies which describe the required modeling dimensions.

To support the access to documents, data, and formal knowledge in an Organizational Memory an integrated domain ontology and thesaurus is proposed which can be constructed semi-automatically by combining document-analysis and knowledge engineering methods. Thereby the costs for up-front knowledge engineering and the need to consult domain experts can be considerably reduced. We present an automatic thesaurus generation tool and show how it can be applied to build and enhance an integrated ontology/thesaurus. A first evaluation shows that the proposed method does indeed facilitate knowledge acquisition and maintenance of an organizational memory.

Contents

Introduction

The systematic management of knowledge has been recognized as a necessity to enhance a company's survival and success in the global market place. In order to be effective, organizational knowledge management has to improve the capitalization on existing knowledge assets and facilitate the creation of new knowledge.

An **Organizational Memory (OM)** captures, stores, disseminates, and eases context-dependent utilization of valuable corporate knowledge and is thus a central prerequisite for the information-technology part of knowledge management.

Contributions from Artificial Intelligence mostly focus on formal knowledge representations for capturing individual expertise, e.g., in expert systems. However, empirical research shows only few systems based on formal knowledge-bases operational in daily industrial practice [Davenport *et al.*, 1996], mainly due to too difficult knowledge acquisition and maintenance. Our own industrial experiences (see [Kühn and Abecker, 1997, Tschaitschian *et al.*, 1997]) revealed that practical solutions should address the following issues:

Identification of core activities: Spending costs for sophisticated computer support is often only accepted for some small part of the whole bunch of business processes which lies at the heart of the business value creation, is especially hard and knowledge intensive, and heavily influences all other parts of the business.

Further analysis shows that these *core activities* typically exhibit the characteristics of so-called *wicked problems* which were extensively examined by Rittel and his coworkers in the early 70'ties [Rittel, 1972, Rittel and Webber, 1973] and reconsidered in depth by Conklin [Conklin and Weil, 1997] and others [Shum, 1997] in the discussion of *Organizational Memories.*

Typical examples for such problems could be: design a new product, formulate a mission statement for a group, determine the strategic direction for a company's development, divide a software design problem into subproblems which determine the module structure of the resulting program. The crux with wicked problems lies in the fact that the usual activity when tackling them is not problem-solving but constructing social commitments in a group of stakeholders, instead. As convincingly pointed out by Conklin [Conklin and Weil, 1997]—because of this different structure of the problem—there is no hope that tackling wicked problems can satisfactorily be supported by highly structured methods designed for essentially "tame" problems.

As a consequence, the KnowMore approach leaves the problem-tackling initiative with the user and tries to support it by providing the appropriate knowledge and information available in the company.

Sparse formalization: Also for the central knowledge-work processes, cost-optimized solutions cannot rely on a heavy, deep formalization of domain and knowledge sources. This is not only due to the rigid cost-benefit analyses which must be done in industrial applications and which often make formal knowledge acquisition and maintenance infeasible [Davenport *et al.*, 1996]; it is also due to the fact that the most valuable corporate knowledge is often tacit knowledge [Sveiby, 1996, Nonaka and Takeuchi, 1995] which is *per definitionem* hard or impossible to formalize because of its many implicits etc. Of course, the more formal one wants to capture such knowledge, the more difficult the task becomes.

Consequently, an OM should build wherever possible upon existing formal notions (like formalized business process models, or database schemata), and prevalent knowledge-sources (like document archives, or technical documentation). It should make extensive use of natural-language based, graphical, or multimedia documents which provide natural means for transporting tacit knowledge, even if it is implicitly contained in these documents but can be constructed by the user if the documents are presented in the appropriate context accompanied with the appropriate meta information.

Integration into the workflow environment: There is no place for specialized, stand-alone assistant systems, assuming or even prescribing a specific kind of how the user does her work, but the support to be offered to the user must seamlessly tap into the usual flow of work established in the company.
This thesis is confirmed by the fact that wicked problem parts can be deeply integrated in more "tame" business processes (see [Bernardi, 1998] and [Decker *et al.*, 1997, Daniel *et al.*, 1997] for an example from design support): knowledge-intensive activities can be embedded into *business processes*; these processes are triggered and fed by *data* and *documents*, they are restricted, guided, and supported by formal corporate knowledge which is layed down, e.g., in *business rules* or *standards*, and they manipulate and produce again *documents*.
This suggests a deep integration of workflow management, document management, and knowledge management. It also promotes the idea that existing business process models as kind of formal models already used in a company (a) may provide a useful framework for launching *active* support and (b) could be exploited for establishing a representation of the work situation context that defines actual information needs and guides and constraints the retrieval process for satisfying these information needs.
To summarize, we propose the following basic support approach:

1. The ultimate goal of supporting situated work is *contextual assistance* [Nutt, 1996]; as argued above, it cannot be achieved to automate wicked-problem solving to a large extent. Consequently, we aim at **context-sensitive, active information provision** within the running working process.

2. Our experiences [Kühn and Abecker, 1997, Tschaitschian *et al.*, 1997] indicate that knowledge-intensive activities are nevertheless deeply embedded in more tame work processes and interlinked in manifold ways with other process stages. Hence, we propose to **use conventional workflow technology as a means for defining the context of wicked-problem solving**. This context can be exploited for constraining the search for useful and necessary information.

In particular, we will focus on three research topics that are of utmost practical importance for the realization of an OM:

- **Workflow-oriented knowledge management** in order to give *active* support with relevant knowledge in the context of a given task.
- **A common inference technique** for integrated processing and retrieval of formal, semi-structured, and non-formal knowledge.
- **Integrated ontology and thesaurus** for structuring and accessing formal and non-formal knowledge in the Organizational Memory.

Towards the realization of this ambitious vision we have made considerable progress that will be described in this report.

Chapter 1

Modeling Knowledge-Intensive Tasks

1.1 Introduction

An Organizational Memory supports knowledge works by *actively* providing relevant information in the context of a particular activity. The KnowMore research efforts focus on the support of so-called *knowledge-intensive tasks* – activities of sufficient importance and complexity that require a lot of knowledge and experience from the worker. The *workflow oriented knowledge management* has been identified as a specific instance of this research topic: The knowledge represented in business process models and their workflow realization provides the context for informational support in these knowledge-intensive tasks.

The integration of workflow-related information into the knowledge handling mechanisms provides answers to three guiding questions which play a crucial role in effective support: *What is the overall goal of a particular activity, and which support is needed?* (this is represented in the process model), *What contextual information is already known at this particular instance?* (this is answered by the workflow activity instance), and *When is the support appropriate?* (when the activity is started during workflow execution).

The pivot element towards the integration of workflow technology and organizational memory approaches is a suitable representation of the knowledge-intensive activities. The representation must fit into the overall business process representation and contain all necessary elements to facilitate the intended knowledge management operations. In this paper the representation formalism for knowledge-intensive tasks is presented. First, some key definitions of workflow technology are given. Then the description frame for knowledge-intensive tasks is introduced and its semantics is clarified by outlining the basic functionality of the knowledge agent. An example completes the presentation.

1.2 Business Process Modeling and Workflow Technology

The explicit modeling of business procedures in so-called *Business Process Models* (BPM) has proved valuable in a variety of situations. The explicit representation of business processes brings in the open many details about the usual way to achieve the business goals. The models may serve as a basis to reconsider the way things are done: Business Process Reengineering aims at streamlining business processes in order to reach optimal performance. To this end the models are analyzed, changed, simu-

lated and finally implemented in the enterprise. In day-to-day work the models serve as a guidance/control mechanism to survey the correctness and efficiency of business processes and support the short-time scheduling of the arising tasks in order to achieve a good workload balance among the resources. The implementation of business process models on some computer system in order to support the day-to-day work resulted in the so-called *Workflow* technology.
Both business process modeling and workflow technology are subject to intensive research and development. We base our research on the standards proposed and published by the *Workflow Management Coalition* (WfMC). The WfMC is a non profit organisation with the objectives of advancing the opportunities for the exploitation of workflow technology through the development of common terminology and standards [The Workflow Management Coalition, 1996].

1.2.1 Workflow terminology and reference model

In this section some basic concepts which will play a role in our context are given. The definitions are taken from the Workflow Management Coalition Terminology & Glossary [The Workflow Management Coalition, 1996].

Business Process A set of one or more linked procedures or activities which collectively realise a business objective or policy goal, normally within the context of an organisational structure defining functional roles and relationships.

- A business process has defined conditions triggering its initiation and defined outputs at its completion.
- A business process may consist of automated activities, capable of workflow management, and/or manual activities, which lie outside of the scope of workflow management

Thus the business process is what is done in an enterprise, regardless of possible support by workflow or other techniques. Its characteristics can be captured in a suitable representation:

Process Definition The representation of a business process in a form which supports automated manipulation, such as modelling, or enactment by a workflow management system. The process definition consists of a network of activities and their relationships, criteria to indicate the start and termination of the process, and information about the individual activities, such as participants, associated IT applications and data, etc.

- The process definition results from work during the process definition mode and may include both manual and workflow (automated) activities.
- A synonym is *Model Definition*, among others.
- Normally, workflow participants are identified in a process definition by reference to a role (cf. workflow participant definition, below)

Activity A description of a piece of work that forms one logical step within a process. An activity may be a manual activity, which does not support computer automation, or a workflow (automated) activity. A workflow activity requires human and/or machine resources to support process execution; where human resource is required an activity is allocated to a workflow participant.

- An activity is typically the smallest unit of work which is scheduled by a workflow engine during process enactment, although one activity may result in several work items being assigend to a workflow participant.

Automated Activity An activity which is capable of computer automation using a workflow management system to manage the activity during execution of the business process of which it forms a part.

During process execution, an automated activty is managed by the workflow management system. This may result in

- an invoked application being activated directly by the workflow management system
- one or more work items being assigned to a workflow participant, with supporting tools or applications being invoked and managed by the workflow management system
- one or more work items being assigned for a workflow participant to process independently of the workflow management system, with the completion of the work items being notified to the workflow management system by the workflow participant *manually executed work items*

Manual Activity An activity within a business process which is not capable of automation and hence lies outside the scope of a workflow management system. Such activities may be included within a process definition, for example to support modelling of the process, but do not form part of a resulting workflow

The combination of the concepts defined above allows to capture the interesting aspects of a business process. Note that the process model completely describes the control flow of the process: The sequence of the activities and the various triggers are part of the process definition. The content of the activity, however, is described implicitly (designation of some *program* or *invoked application*, some work item (this will be detailed later) or simply the indication of a manual activity to be performed by human users) and with very little detail.
The various concepts mentioned until now all descibe entities with a class-like behaviour, that is, a process definition describes the class of all processes of this type, and so on. At runtime, these entities need to be instantiated into concrete instances (of process and activity, respectively). This is, where the various entities which do the work get more detailed:

Process/Activity Instance The representation of a single enactment of a process/an activity within a process instance. Including its associated data.

Workflow Participant A resource which performs the work represented by a workflow activity instance. This work is normally manifested as one or more work items assigned to the workflow participant via the worklist.

- The term *Workflow Participant* is normally applied to a human resource but it could conceptually include machine based resources such as an intelligent agent.
- Where an activity requires no human resource and is handled automatically by a computer application, the normal terminology for the machine based resource is *Invoked Application*

- A workflow participant is (normally) identified within the process definition by reference to a role, which can then be filled by one or more of the resources available to the workflow system to operate in that role during process enactment.

Work Item The representation of the work to be processed (by a workflow participant) in the context of an activity within a process instance.

- An activity typically generates one or more work items which together constitute the task to be undertaken by the user (a workflow participant) within this activity.
- The work items are normally presented to the user via a work list and a worklist handler.
- The control and progession of work items rests with the worklist handler and the user. The workflow engine is merely notified of the completion of particular work items.

As we will see, these definitions provide the necessary hooks for our intended enhancements.
Until now, the control flow and the various participants in the business process have been mentioned. But what about the data, which will be processed during any process instance? The Workflow Management Coalition distinguishes three important categories:

Application Data Data that is application specific and not accessible by the workflow management system.

- This is data that is needed by the various participants and applications, but which the workflow management system generally will never see. However, some of it might become workflow relevant data, if it is used by the workflow management system to determine a state change.
- As a process definition describes control flow, not data flow, most of the application data will not arise in its representation.

Workflow Relevant Data Data that is used by a workflow management system to determine the state transitions of a workflow instance, for example within pre- and post-conditions or workflow participant assignment.

- Workflow relevant data may be manipulated by workflow applications as well as by the workflow engine.
- Workflow relevant data may be made available to a subsequent activity of another process instance. Thus some kind of data flow can be modeled.
- Workflow relevant data may be of two types:
  - Typed - the structure of the data is implied by its type. Typically a workflow management system will understand the structure of the data and may be able to process it.
  - Untyped - the workflow management system will not understand the data structure. It may, however, pass the data to workflow applications.

Workflow Control Data Data that is managed by the workflow management system and/or a workflow engine. Such data is internal to the workflow management system and is not normally accessible to applications.

- This data represents the dynamic state of the workflow system and its process instances, but is not relevant to the business process itself or to the various applications.

This distinction is reflected in workflow implementation details as well as in our modeling concepts: The handling of application data is left to the workflow participant or the invoked applications. The workflow control data provides information about the state of a running process, where appropriate. The most interesting category is the workflow relevant data, as our modeling and enhancements will contribute to this category.

1.2.2 Generic workflow product structure - a reference model

The overall interaction of the various entities mentioned in the previous sections results in a reference model which depicts the generic workflow product structure (figure 1.1). At process definition time, the process definition is generated which refers to an Organisation/Role Model (an instance of an Enterprise Ontology) do describe the workflow participants. It also refers to the invoked applications and the application data and workflow relevant data, where necessary.

During runtime, the workflow management engine interprets and instantiates the process definition, calls the invoked applications and interacts with the other workflow participants (users) via the worklist handler. It performs the necessary control and scheduling using the available workflow relevant data.

The workflow participants, guided by the worklist handler, invoke their applications and influence the various data according to their need and beyond the reach of the workflow management engine. The latter is notified via the worklist handler if a particular work item has been completed.

The internal workflow control data maintained by the workflow management engine is an interesting object for administration and monitoring activities.

1.3 Representing a workflow in KnowMore – a basic formalism

To describe the business processes on an adequate level of detail, a basic BPM representation formalism has been realised. This formalism takes into account several objectives:

- In order to be compatible to a wide range of existing workflow management systems and applications, the principles of the repesentation formalism closely follow the proposals of the Workflow Management Coalition mentioned above.
- In KnowMore, the business process definitions are created using the ADONIS business process modeling tool. The representation formalism has to be able to capture the information provided by this tool.
- The necessary extensions of workflow models required by the KnowMore approach have to fit smoothly into the formalism.

These requirements result in the following principles: The business process is represented as a sequence of *activities*. Each activity implies the execution of a *task* and is linked to its *preceeding activities* and its *subsequent activities*. Each activity is identified by a symbolic name. Splits of the control flow from one activity to several subsequent

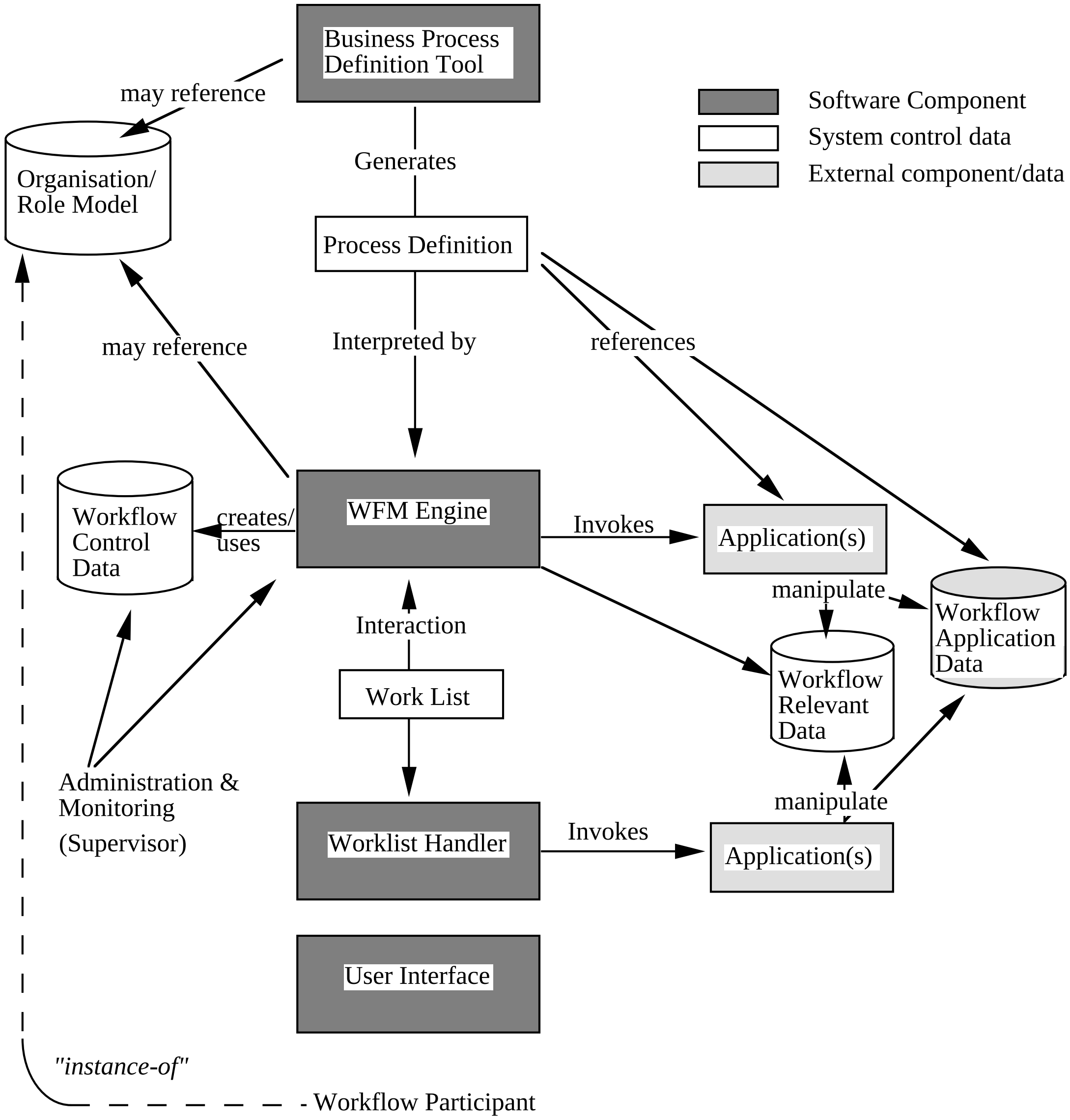


Figure 1.1: Generic workflow product structure adopted from the WfMC

activities may result in the execution of several activities in parallel. Several execution tracks can be joined again in a single activity.

Each activity in the business process handles some data objects from the enterprise environment. Each activity references its relevant data objects by means of *variables*: *Input variables* indicate the objects which are needed as prerequisite for the particular activity, *output variables* describe the result of the activity[1].

1.3.1 Context and Goals of an activity

A central topic in the KnowMore approach is the use of information about the context and the goal of a particular activity in order to provide suitable support. The workflow representation needs to offer this information. The *context* of an activity is represented

---

[1] Whether the data described by these variables is passed from one activity to the other (data-flow approach) or is globally accessible from various activities (blackboard approach) is an implementation detail of the workflow management system. We will try both approaches, where appropriate.

1. in the identification of the particular process definition, which the activity is a part of (*global context*)
2. and by the set of all input variables of the activity *local context*).

At process execution time, the activity instance thus knows its actual context from the identification of the process instance it belongs to and from the instantiation of its input variables. Analogously, the *goals* of an activity are seen in

1. the identification of the particular process definition (*global goal*), as the objective of the business process is well-defined
2. and in the set of all output variables of the activity (*local goal*), as these denote the contribution of the particular activity to the process.[2]

At process execution time the activity instance contributes to the global goal by instantiating its output variables, performing particular tasks to do so.

1.3.2 Technical details of our BPM representation

A formal description of the extended business process models comprises the elements mentioned below. The notation follows the realization in the OCRA formalism 2.

variable(*name,type,description* **)** denotes data objects of the business process. *name* is a symbol to identify the variable, *type* describes the possible fillers of this variable (range information). *description* is an optional comment. Typically, *type* is a symbol refering to some ontology which models the environment of the business process.

During process execution variables are instantiated by suitable values. The default value is *null*.

activity (*name, description, task, successors:*{*activity*}*,*
fireswhen:constraintObject, input:{*variable*}*, output:*{*variable*})
models the various stages in the control flow of the business process. *name* identifies the activity, *description* is a comment, *task* denotes the work item to be executed. A set of *successors* denotes the subsequent activities. The *fireswhen* object refers to workflow control information (e.g. *all predecessors completed* or *at least one predecessor completed*) or workflow relevant data (e.g. *value greater than some limit*) in order to indicate when the activity will be started.

The task initiated by the activity can be represented in varying levels of detail:

task(*name***)** denotes the work item initiated by the activity. For a manual activity this is usually sufficient as representation.

simpletask:task(*execute, input:*{*variable*}*,output:*{*variable*}**)** indicates some program (invoked application) to be executed together with its input and output parameters. Both input and output variables must form subsets of the respective variable sets of the activity.

---

[2] A dataflow-oriented realization approach might blurr the description of local goals, as possibly many input variables are simply passed on unprocessed to subsequent activities, thus occuring as meaningless output variables. In this case, only the output variables influenced by the task performed by the activity will be considered.

knowledgeIntensiveTask:simpletask enhances the description of a simpletask by a *support specification* in order to enable the automatic provision of relevant information. The most simple realization is the name of some retrieval program where some of the *input* variables are used as search keys for the retrieval. Further possibilities are at the heart of this paper and will be detailed in the next sections.

1.4 Knowledge-Intensive Tasks and their characteristics

KnowMore focuses on the support of activities in a business process which are not reduceable to some well-known workflow application but need knowledge in various and complex ways to accomplish their goals. According to [Davenport *et al.*, 1996], activities dealing with the acquisition, creation, packaging and application of knowledge are at the heart of any knowledge work which in turn can be increasingly identified inside the core competencies of modern enterprises. In traditional workflow management these tasks lead to manual activities which are in principle beyond the scope of the workflow management system.

A *knowledge-intensive task* (KIT) is thus handled by a human workflow participant. This participant can be supported in his work by suitable information. In addition the human participant might use some invoked application, hence the identification of a KIT as an extension of a simple task.

Based on industrial surveys, Davenport et al. [1996] identified several promising approaches to support the knowledge-intensive tasks, among them the more detailed description of steps and contributions inside the task (*changing the unit of knowledge*) and the sophisticated use of supporting information systems of adequate complexity (*employing technological enablers*)[3] The KnowMore project subscribes to these approaches. From this point of view, a KIT is characterized by the existence of (one or many) *information needs* which need to be satisfied in order to achieve the goals of the task. The detailed representation of the information needs realizes the first approach. To fulfill an information need, actions of varying complexity can be imagined, ranging from data base queries using well-defined selections to extended keyword association on the world-wide web. The integration of suitable knowledge sources corresponds to the second approach.

1.4.1 Output of KIT processing

The processing of a knowledge-intensive task by the workflow system shall produce a two-fold result: The simple task leads to the execution of some application and to some user interaction. The evaluation of the information needs produces an amount of information which supports the task.

This information has to be presented to the user in an adequate way, depending on its form (formal, semi–formal, or non–formalized). As far as the presentation only depends on this, the respective post–processing is independent of the task/process at hand and may be left to the knowledge agent.

Should the need arise to configure the presentation from the particular variables of specific tasks, this must be represented in the KIT. (An obvious example is, whenever 'post-processing' comprises the processing of formal data. In this case the respective algorithms must be specified in the KIT)

---

[3] It is interesting to note that classical expert systems are one possible realisation of this approach. According to [Davenport *et al.*, 1996], however, the maintenance of these proved too hard for practical success in many of the surveyed projects.

1.4.2 Control structure in a KIT

A KIT forms a unit of sense, but further details might be given by specifying several (partial) information needs. Each information need will result in some information which supports a particular aspect of the complete KIT. The contributions of the various information needs might overlap. The result of one information need might be a subset or a superset of another's result — everything is possible.

As far as these dependencies among the information needs are known at process definition time, they are to be represented in the KIT description. This is done by *preconditions* in the various information needs, and by a set of *processing rules.* The knowledge represented here influences the way the information needs will be interpreted and fulfilled during process execution. However, this information shall not be interpreted as defining a complete and well-defined control structure for executing information needs (if this is a correct interpretation, then the KIT can be replaced by a well-defined subprocess comprising of several activities of a simpler structure, thus obliviating the need for the complex KIT representation).

The proposed representation of knowledge-intensive tasks takes into account these characteristics.

1.5 Representation of Knowledge-Intensive Tasks

As mentioned above, the KIT representation extends the *simple task* representation by the *support specification*. The various attributes of the simple task are inherited by the KIT. Additionally the KIT representation may refer to the process context. The support specification inside the KIT representation employs a description frame as follows:

Context information, inherited from simple task	
name,	a symbol identifying the KIT
execute,	denotes the application which is supported by the KIT
input:{variable}	denotes the local context of the KIT
output:{variable}	denotes the local goal of the KIT
Process context, provided by the runtime environment	
#callingActivity, #processInstance	These symbols may be used in the following to refer to the relevant workflow control data. These symbols indicate the actual instances of the activity which initiated the KIT and of the process which contains the activity. This information is provided at runtime by the workflow management system.
Support specification, *containing a set of information needs which detail the KIT and connect between interface variables and information retrieval queries*	
local-variables:{variable}	declaration of additional variables used in the KIT description
infoneeds:{	
(name,	a symbol
description,	a comment
precondition:{ *constraint-object*}	a set of constraints on any of the variables accessible from inside the KIT. The information need is only evaluated if these preconditions are fulfilled.
agent-spec,	a string, containing the specification of the information needed which can be given at process definition time.
parameters:{variable},	a subset of the relevant input variables, local variables, or the above-mentioned symbols denoting references to the calling activity and the process instance. The values of these interface elements are only known at process execution time.
from:*info-source-description*,	a set of symbols denoting info sources. This might be omitted, if the knowledge agent which processes the KIT is able to compute the relevant information sources.
contributes-to:{variable}	local variables or output variables which are filled using the result of this info need.
) }	
processing:{ if *constraintobject* do *action* }	A set of forward rules working on the result of the information needs.

This representation frame merits further comments.

The *process environment* is made accessible by the defined symbols. During process definition, this interconnection is sufficiently represented by the link process – activity – KIT. To refer to the enclosing process, however, the KIT needs these explicit identifiers.

Details of the available workflow control data which can be accessed via this interface rely on the particular workflow enactment system.

The *support specification* realizes the concepts elaborated above: The KIT is detailed by information needs. Their contributions, preconditions and processing rules describe their interconnection, but it is not suitable to create a strict sequence of info needs.

The **precondition** allows to restrict the evaluation of information needs dependend e.g. on the state of their parameters (only execute if some variables are allready non–null, or: if some parameter is already known, skip this need) or on the state of the process (skip if time is critical).

The **agent-spec** description of the relevant information is interpreted as a remote procedure call to a specific knowledge agent. This agent is responsible to retrieve the relevant information. The *about* string thus contains the name of a program. It might also contain additional parameters which can already be specified during process definition. At runtime, the knowledge agent is invoked and provided with the *parameters*.

The **from** description of the info sources is a temporary construct. In principle, determining the info sources which are relevant for a particular information need is a central objective of the knowledge agent. By computing *info-source = f (parameters,expected-output,callingActivity,processInstance)*, the knowledge agent finds the knowledge source according to the goal and context information.

As a first step, however, we identify suitable info sources at process definition time, e.g. the well-known data bases of the enterprise. Thus *from* contains a list of relevant info sources.

Further development will stride towards the automatic computation of relevant info sources. The *from* parameter will be obsolete as soon as this is realized.

The **contributes-to** field indicates the goal of the particular info need: It shall help in finding values for the variables mentioned here. On the basis of this information, the interconnection between the different information needs can be deduced and evaluated by the knowledge agent, see below.

The **processing** rules govern a certain amount of post-processing of the retrieved information. As detailed in 1.4.1, the result of the evaluation of the information needs is usually presented to the user. In certain cases, however, it is possible to specify further operations (e.g. a formal knowledge item is used for direct computation by some algorithm). The result of some information need—seen in meta-information from the knowledge agent—can also be used to trigger further operations. The *constraintobject* may contain expressions about any variable accessible inside the KIT or about meta-information which is provided by the knowledge agent. Examples of meta-information are e.g. *empty result* or *count of produced information objects*. Action comprises the calculation of values, the setting of variables, or the activation of information needs.

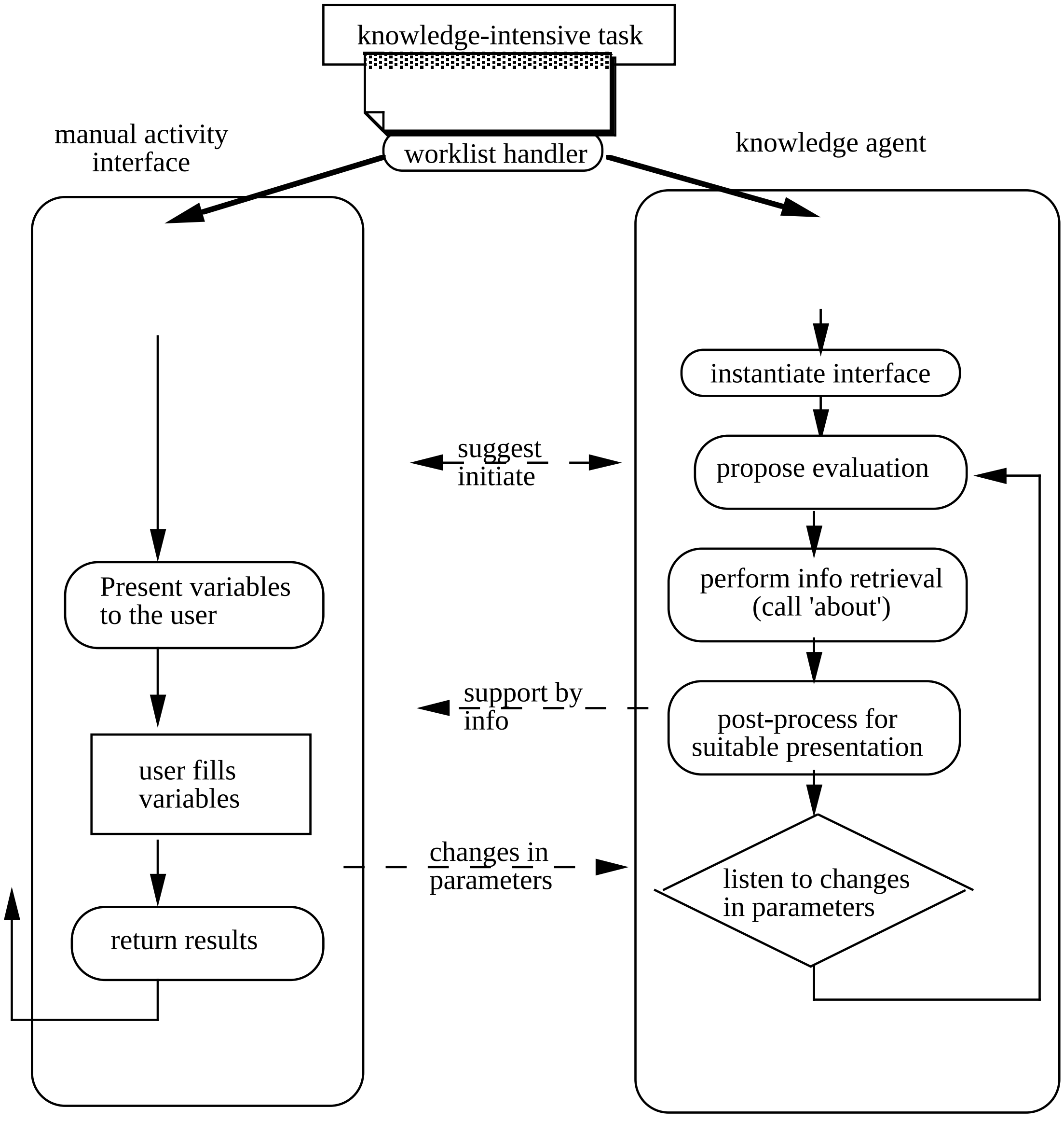
During process execution, KIT representation frames are passed to a knowledge agent after the interface section has been instantiated. The basic functionality of the knowledge agent – and thus the operational semantics of the KIT representation – is sketched in the next section.

1.6 Basic functionality of the knowledge agent

The execution of a knowledge intensive task in accordance with the principles discussed here presupposes some basic functionality of the knowledge agent and the worklist handler.

The central instance to work on the KIT is the human workflow participant. He is responsible for solving the problem at hand as denoted by the simpletask instance (remember that a KIT is-a simpletask). Thus the worklist handler simply presents the workflow participant with the KIT name and the input and output variables. The human user solves the task at hand (which is identified by the name) and fills the output variables.

In parallel, the KIT representation is passed to the knowledge agent. The knowledge agent evaluates the information needs and instantiates the parameters. It then presents the various information needs as *support offers* to the user, using the name and the comment of the information needs (cf. Figure 1.2).

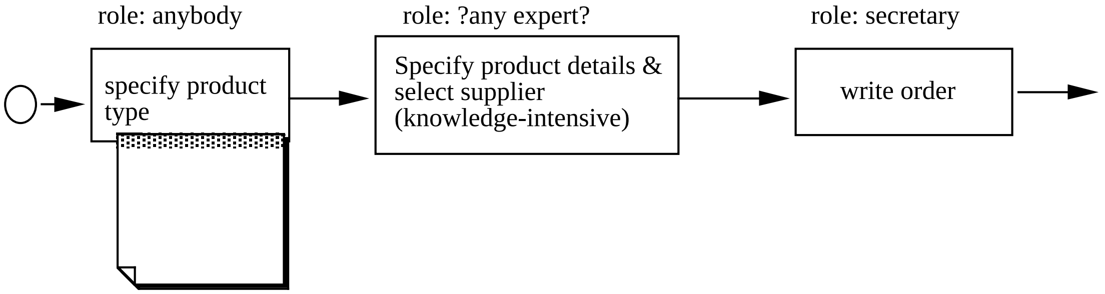


As soon as the user completes the task and the filling of output variables, a message is passed to the worklist handler (as already indicated in the generic model by the workflow management coalition). Automatically the knowledge agent receives a *close* signal for this particular KIT, closes the display windows under its responsibility and exits.

In summary, the information needs modeled in the KIT representation are satisfied by the knowledge agent, under control of the human workflow participant and in close interaction with the worklist handler. The integration of the knowledge agent into existing workflow environments thus only needs the extension of the business process definition by the KIT representation and the implementation of a suitable worklist handler, leaving the remaining workflow enactment services untouched.

1.7 An example of a modeled KIT

To illustrate the modeling of a knowledge-intensive task we refer to a section of a business process in the purchasing department of an enterprise (see Figure 1.3).



- Previous experiences about products and suppliers are collected as written notes.

Thus the KIT is represented as follows:

```
KIT:
( name:           Specify-product-kit,
  relevant-input: {product-type},
  expected-output:{product-name,product-id-no,price,supplier-id},
  infoneeds:{
   (name:       available-products,
    description:"Products of the wanted type, from database",
    precondition:{},
    agent-spec:  "databasey-agent select $p"
    parameters:  {product-type},
    from:        {product-database}
    contributes-to:{product-name}
   ),
   (name:        ask-specialist,
    description: "email to specialist for the wanted product"
    precondition:{product-name==null} // ask only if no idea yet
    agent-spec:  "person-competence-agent",
    parameters:  {product-type},
    from:        {enterprise-competence-base}
    contributes-to:{product-name,supplier-id}
   ),
   (name:        relevant-suppliers,
    precondition:{product-name!=null},
    agent-spec:  "database-agent select($p-type,$p-name)",
    parameters:  {product-type,product-name}
    from:        {list-of-suppliers}
    contributes-to:{product-id-no, price, supplier-id}
   ),
   (name:        prev-experiences,
    agent-spec:  "full-text-retrieval keywords $*",
    parameters:  {product-type, supplier-id}
    from:        {notes-archive}
    contributes-to:{product-name, supplier-id}
   )}
  processing:{
    if (price>100) propose prev-experiences
    if (supplier.specialconditions) price=0.98*price
  }
)
```

Chapter 2

A Generic Knowledge-Description Formalism

An OM essentially acts as an active multimedia IR system which comprises existing information sources as its content. Formal knowledge integrated wherever reasonable partly automates problem-solving, but mainly helps finding more informal knowledge documents. Interoperability of representations is achieved by projection onto a common information space.
In this specification document, we will concentrate on information modeling issues, i.e., on the question how this common information space can be designed and represented. To this end, the following sections are organized as follows:

- In Section 2.1, we will outline our overall approach with a comprehensive document model base at its core. Knowledge-item descriptions (KIDs) describe a document (in the broadest sense, which means any information source) with respect to a number of relevant dimensions. KIDs are formulated using a configurable description formalism which is tailored with the help of a number of representational ontologies describing the respective modeling dimensions.
- In Section 2.2, we examine requirements for the realization of our knowledge description formalism which are either of general nature or coming from the special representation needs of the modeling dimensions under consideration.
- In Section 2.3, we define our basic knowledge description language.
- In Section 2.4, we show how our basic representation formalism is used for defining the ontologies needed in KnowMore and how specific document descriptions can look like on the basis of the so-configured information modeling language.
- In Section 2.5, we present our implementation approach.
- In Section 2.6, we discuss how our repesentation formalism relates to other formalisms used in the literature for similar purposes.

2.1 Overview of Our Modeling Approach

2.1.1 Information Modeling in Information Retrieval

The availability of almost every kind of information in electronic form together with the success of internet and intranets for comfortable document dissemination put completely new demands on Information Retrieval technology. Possibly the greatest potential for facing these challenges lies in the *logic-based approach to Information Retrieval (IR).*

Logic-Based Information Retrieval is based upon van Rijsbergen's seminal idea to understand retrieval as the task of finding all documents d for a given query q which are likely to *imply* q, i.e., $d \rightarrow q$ holds [van Rijsbergen, 1989]. Retrieval is seen as logical inference which can profit from different sources of background knowledge. The inference works on formal representations of both documents d and query q. Since a user's real information need is typically specified only rather vague in a query, and, on the other hand, the content of documents can only be modeled to a certain extent, it is clear that there is a lot of vagueness and uncertainty intrinsic to the inference process. This is reflected by probabilistic inferences which aim at computing the probability $P(d \rightarrow q)$ that d implies q.

Dimensions of Document Models Usually, document modeling in logic-based IR is concerned with three dimensions of document description [Meghini *et al.*, 1991]:

1. *the logical structure*, e.g., of a proceedings volume with sections as parts, articles as the sections' parts, and title, abstract, and text body as the articles' parts,
2. *the layout structure*, e.g., of a business letter with a rectangular bold-faced region in the upper left corner of the sheet, and
3. *the conceptual structure*, e.g., of a technical memo which describes the content of a document making, for instance, statements about a product's quality.

In addition to these document-intrinsic features, most IR systems use also some factual knowledge about the document, e.g., the author's name, the publisher etc. We will refer to these document-extrinsic features as document meta-content or document *contextual structure.*

The most interesting advantage of such a comprehensive modeling of documents (and any other information source) is the possibility to attach additional background knowledge to each of the modeled dimensions and let these knowledge bases interact. The most important example herefore is to have a sophisticated model of the domain the documents talk about and to index documents with pointers into this domain model. This *conceptual indexing* approach for sophisticated content representation allows, e.g., formulation of domain-specific search heuristics [Baudin *et al.*, 1992] or a more precise query formulation [van Bakel *et al.*, 1996]. It is not only a way for indexing non-text documents (e.g. video tapes or images) [Gordon and Domeshek, 1995], but also a natural means for integrating information from different sources with different vocabulary [Kindermann and Hoppe *et al.*, 1996].

Document Modeling Languages Having identified the dimensions useful for describing information sources, it has to be clarified what language is needed. From the examples (especially the detailed domain modeling) above it is already clear that at least the basic abstraction mechanisms are needed which constitute a structurally object-oriented formalism: classification of objects into classes, generalization of classes to superclasses, aggregation to express "part-of" relationships, and attribute-value assertions to specify certain class instances. We need also inferential capabilities for the formulation of search heuristics or to follow links in hypermedia documents. In order to provide the needed expressiveness plus a means for coping with the mentioned uncertainty of the inference, typical approaches are based on probabilistic extensions of, e.g., terminological logics, or deductive databases [Meghini and Straccia, 1996, Rölleke and Fuhr, 1996]. In the following section, we will show how these mechanisms can be employed for supporting corporate knowledge management.

2.1.2 Information Modeling for Corporate Memories

Figure 2.1 (taken from [Abecker *et al.*, 1998]) sketches our approach to grading up from *information* retrieval to *knowledge* management. We start with heterogeneous, multi- and hypermedia information sources which shall work together to provide comprehensive problem-solving support. To this end, all sources are described according to a homogeneous, comprehensive description schema (*knowledge item descriptions*, KIDs). Recent investigations on OM organization principles [van Heijst *et al.*, 1996b] reveal the following factors to be essential for determining the knowledge which is useful to support an activity: the *task* to be performed, the *role* the actor plays for this task, and the *domain* the task is done within. [Hofer-Alfeis and Klabunde, 1996] concretize these factors in enterprise terminology as *business process activity, organisational role,* and *product* to be processed. This view gives us a first specialization of the general IR scenario for the enterprise knowledge management problem:

1. *conceptual structure:* the topics a knowledge item is dealing with are expressed in terms of the enterprises' product models. Of course, a useful product domain ontology will also define associated concepts like suppliers, buyers etc., and
2. *contextual structure:* meta-content like the document-creation context or possible application areas are stated in terms of the enterprise ontology, the main part of which are business process models and organisational models.

Using these formal structures for indexing knowledge items has the advantage that already existing formalizations can be reused. Compared to conventional IR approaches, we consider the context dimension very important. [Celentano *et al.*, 1995] show how rich knowledge about business processes, started process instances, and dependencies between documents in different business process activities can be employed for powerful search and retrieval of office letters. We adopt this view, but extend it from office letters to all information sources used in a business process.

While the conceptual and contextual structure mainly helps *finding* the appropriate sources (*what can be found?*), the representational ontology helps to extract and access knowledge from an information source and to find the appropriate piece of knowledge within a document (*how is it found?*). Though standard in office-letter processing, this idea must also be lifted to arbitrary information sources (e.g., a database has no layout and typically a logical structure identical with the conceptual one).

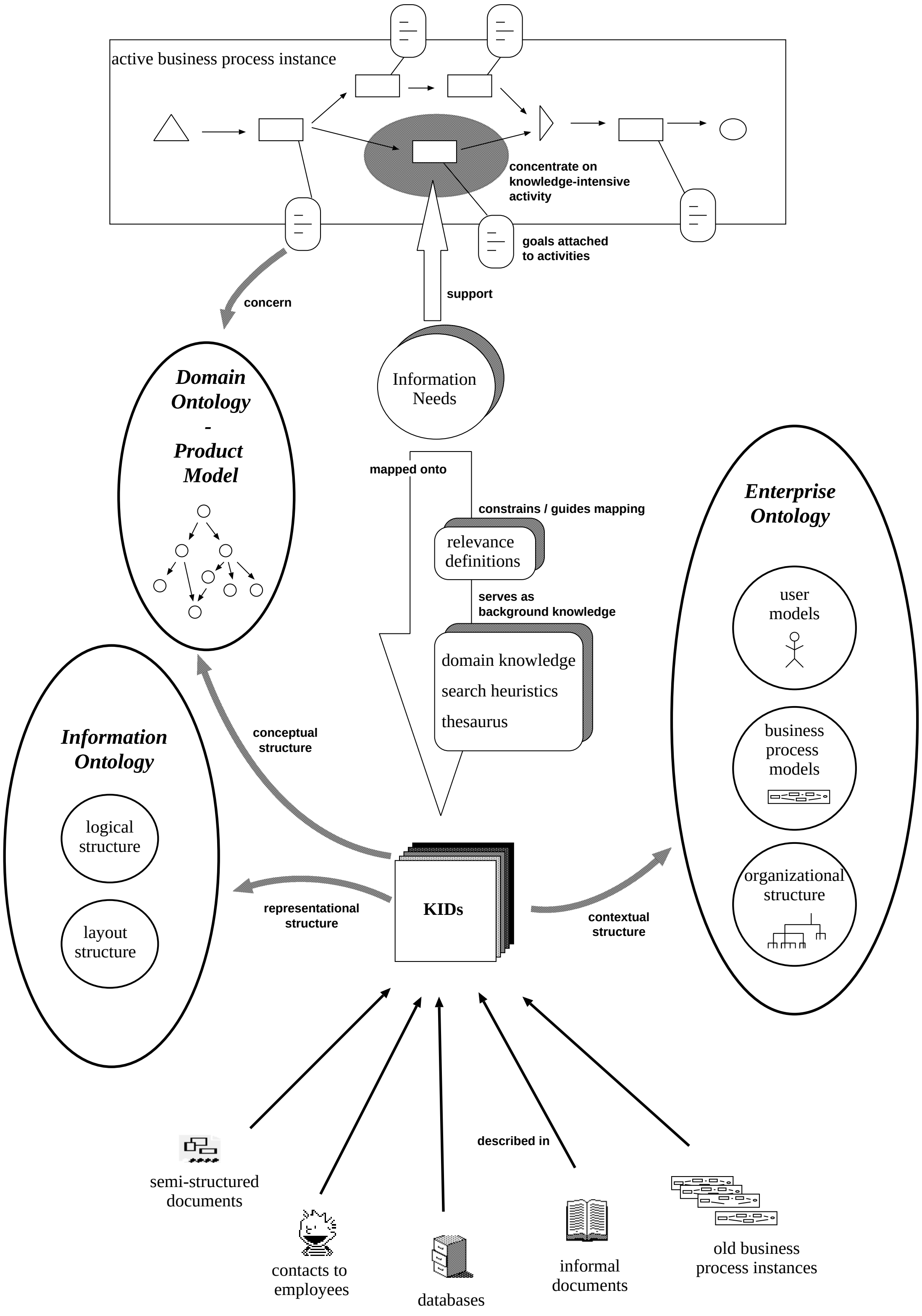


Figure 2.1: Overview of an Organizational Memory

2.2 Designing Knowledge-Description Formalisms

2.2.1 Requirements

The respective information modeling dimensions introduced above now put their specific demands on the representational constructs of our formalism.

Layout Structure

In an OM, we assume that the overwhelming part of documents to be managed is available as electronic documents where layout issues are of little interest. Moreover, automatically generated queries to the OM will likely not refer so extensively to layout properties as manually generated queries could do which are heavily depending on the way a human user remembers documents. Layout issues are, of course, treated in detail in the DFKI document analysis projects [Dengel and Hinkelmann, 1996, Baumann *et al.*, 1997]. Consequently, in KnowMore, we focus more on the conceptual and less on the representational level. Nevertheless, the expressiveness provided to adequately represent the other document dimensions should also suffice to capture considerable parts of the layout structure.

Logical Structure

Modeling logical structure of documents is a common technique in *document analysis* [Baumann *et al.*, 1997, Bläsius *et al.*, 1997, Meghini *et al.*, 1991]. There, knowledge about types of possible documents and their generic logical building blocks spans and constrains the search space for interpreting scanned documents.

Ongoing research in our department [Junker, 1998, Junker and Abecker, 1996] also investigates how document structure can be exploited for improving *learning text categorization* and learning of *information extraction* patterns from text.

Structured-document retrieval as a recent branch of knowledge-based information retrieval [Rölleke and Fuhr, 1996, Fuhr, 1995] deals with document structure for a number of reasons. First, it allows passage retrieval, i.e. delivering exactly the part of a document which really contains the desired information, instead of a large document coping with a multitude of additional, irrelevant topics. Such a more fine-grained description of documents is also the basis for combining relevance factors of document parts in order to find the most appropriate aggregation level (a paragraph, a section, or a book) to present to the user. Second, the growing interest in network and hypermedia retrieval makes it necessary, e.g., to follow links in hypermedia documents and to appropriately propagate information about interestingness of document parts along such links. Such a mechanism is of special interest when dealing with multimedia documents which consist of aggregates of multimedia document elements. Third, users may want to exploit document structure when querying for documents they know and remember partly (e.g.: *The textbook with the phrase "Projekt PROKON" in its subtitle*). Fourth, in the presence of multiple information sources with varying media types, modeling the logical structure of information sources helps to map from conceptual structures to access paths. [Fuhr, 1994] argues that differentiating between conceptual structure and logical structure can make information retrieval more effective. [Christophides *et al.*, 1994] present retrieval models which take into account the structure of documents and provide the possibility to query for paths which lead to the relevant part of a document.

Since all these purposes—understanding, categorization, and high-precision retrieval of multimedia documents—are of utmost importance also in the enterprise-knowledge management setting, we have to provide the usual mechanisms for describing information source structure. In detail, we should be able to cope with:

- **Document types:** A corporate memory contains manifold types of document sources (books, memos, databases etc.) which can be arranged in an is-a hierarchy (such as: an offer is an office letter, an invoice is an office letter etc.).
- **Document parts:** Complex documents are composed of simpler parts. For instance, a scientific article consists of title, authors, abstract, some sections, and references.
- **Order of parts:** Imagine a document archive where complex documents are split into their basic building blocks (e.g., paragraphs), these elementary building blocks are directly stored in the archive, and complex documents are only represented by the links to their parts. If retrieval now evaluates a more aggregated document part the most suitable for answering the query (e.g., a section consisting of several paragraphs), the original order of document parts must be recovered, of course.

  Thinking some steps ahead, we can imagine an OM's document base as the knowledge server for intranet knowledge services like personally tailorized tutorials on demand. If we want to engineer such a multimedia instructional sequence within an electronic tutorial system from building blocks like examples, figures, introductory texts etc., the order of the presentation is certainly highly relevant.
- **Links in hypermedia documents:** As a further generalization of the previous, tree-like document model which applies to sequential, paper-based documents, hypermedia information sources introduce arbitrary links between document parts. These can be exploited for query formulation (e.g.: *Show me all web pages dealing with project descriptions which can be reached starting at the DFKI homepage and following at most four navigation steps !*). Links are computationally dangerous because they may introduce cyclic relationships. Furthermore, it is not a priori clear how relevance of documents for a given topic is inherited by other documents which can be reached via a hyperlink.

Conceptual Structure

The most interesting point of each document modeling approach is certainly how to model document content (called the document's *conceptual structure* by [Meghini *et al.*, 1991]). The possibilities range from pure keyword-based representations up to complete formalizations of the semantic content in some expressive knowledge representation formalism. We will briefly review the range of possibilities discussed in the literature ordered according to increasing complexity:

- **Keyword-based content description:** The standard approach in conventional Information Retrieval [Salton and McGill, 1983, Knorz, 1996] represents a document as a vector of words characterising what the document talks about. Keywords can be weighted in order to reflect the importance of terms. Weights are usually produced by automated indexing techniques. The keyword vocabulary can be free or controlled, i.e., predefined in a classification system or an indexing thesaurus. Index terms are not necessarily explicitly contained in the document.

Index terms may be organized, e.g., ordered by explicit dependency structures; for example, *information about the need for technology* and *the need for information technologies* could be represented as

`(information :- need :- technology)`

or

`(need :- technology :- information)`

respectively. They can also be subdivided in `main headings` and `additional qualifiers`. Structured indexing allows to establish given relations (role indicators) between main headings and additional qualifiers which determine how to interpret relationships between them which can not be disambigued by dependency structures (regard: *solution in water* versus *solution with water*).

- **Concept-based content description:** With the advent of multimedia IR systems concept-based indexing started. Here, indexing cannot rely on terms occuring in a document; instead, there must be a model of the domain of discourse such that document content can be characterised with respect to this model. Since it is nearby to use well-known domain modeling techniques and languages from knowledge-based systems to build up such a concept base, there have been considerable efforts especially in building domain models with the help of description logics. This opens possibilities for formal inferences within the domain model which support retrieval. The most typical example is to exploit the subsumption hierarchy to reformulate the given query if retrieval is too specific, or not specific enough, respectively. One step further is done in the DEDAL system [Baudin *et al.*, 1995] where it is allowed to explicitly formulate domain-specific search heuristics as second-order statements over the given domain model.
- **Precoordinated domain concepts:** While the simple concept-based approach is essentially quite similar to keyword vector indexing—with the difference that index terms are taken from an explicit domain models which can be used for formal inferences—the Condorcet project at Twente University [van Bakel *et al.*, 1996] investigates more detailed content modeling by precoordinating index concepts (e.g.: `cures(Aspirin, Headache)`). This mimics ideas from the above mentioned structured organization of keyword indices and allows powerful queries like, e.g.,

  Show me all documents telling what Aspirin is good for!
  ≡ `? :- cures(Aspirin,X)`

  or

  Show me all documents concerning some remedy for heart diseases!
  ≡ `? :- cures(X,heart_disease)`

  Recently, Schmiedel *et al.* [Schmiedel and Volle, 1996] proposed to imitate the compositionality of topic indexes of books by a similar approach in description logics introducing precoordination operators as primitive concepts and roles for semantic cases of their arguments. This allows also nested (composite) descriptions, e.g.,

```
( Comparison
    of  ( Application
             of Description Logics
             to Configuration )
    and ( Application
             of Description Logics
             to the WWW ) )
```

- **Aiming at a complete formalization of document content:** There are also approaches which try to formalize document content to a larger extent. For instance, Zarri *et al.* propose to translate natural-language documents into formal meta-documents which represent the semantic content in some formalized lingua franca [Zarri and Azzam, 1997] which provides an ontology with basic templates for narrative events that are instantiated by objects describing real-world ojects or events that are in turn instances of some domain ontology.

Discussion. If we want to allow for intelligent retrieval mechanisms, we need at least some concept-based indexing with topic descriptors taken from some domain (or topic) ontology. Of course, the more detailed the content description is, the more sophisticated queries and precise answers are possible. However, query generation as well as query evaluation become more complex. Furthermore, correct and consistent indexing gets more complicated, especially for automated approaches. Nevertheless, especially in a company environment, there is need for sophisticated retrieval machinery tackling the problem of information overload; thus, we might find better conditions for sophisticated indexing than it is the case in "less controlled" scenarios, as, e.g., in a research community that shall consistently index scientific papers. One advantageous factor could be that some domains of interest could be formally captured to a large extent if it is important for the company's business (consider, e.g., the chemical industry which has massive knowledge management requirements). Some of this work can also be done for a whole branch of industry by the development of sharable ontologies as it is demonstrated, e.g.. in the Plinius project [van der Vet *et al.*, 1994]. Another important factor is that for all documents which are created within the company (and this holds especially true for such important documents as best practice reports) appropriate tools and editors can promote consistent, sophisticated indexing.

The problem of topics. [Schmiedel and Volle, 1996] pointed out that most Intelligent Information Integration (I^3) projects (like, e.g., the Information Manifold [Kirk, 1996]) which also rely on a formal model of information sources and their semantic content expressed in terms of a domain model do not explicitly address the problem of topics. Though Schmiedel & Volle do not explain their critique in detail, we also noticed that most I^3 projects frequently claim that their approaches would be applicable in the same way to formal information sources (such as databases), semistructured ones, and unstructured ones (like web pages). However, they usually discuss only the case of structured information, i.e. databases. This presupposes that the semantic content of a natural-language document could simply be represented by a one-to-one match against a formal domain model what is obviously not the case.
Rather, we have to expect a quite rough topic description which might be accompanied by some special notions like the above mentioned precoordination operators. Although it seems straightforward to employ the same representation formalisms for modeling the sources and the domain (the topics), Welty [Welty, 1996, Welty, 1998] discussed in much detail the ontological nature of subject taxonomies and concluded

that the topic model should be separated from the rest of the information model and handled in a special way. As an example for representation problems within the standard object-oriented or description logic formalisms take the range of the "about" attribute of a document. If we represent topics as classes/concepts, they are in the ususal representation schemes not allowed as attribute fillers. On the other hand, if topics are instances/individuals, natural subtopic relationships cannot be expressed by the standard means of subclass/superclass relations. Moreover, if we work around the problem with a complex modeling approach, we cannot easily profit from the tuned inferences in such object-oriented systems which are usually concentrated on subclass relations, sometimes also supporting part-of relationships. Further, as sketched above, the topic hierarchies in classification systems of legacy information systems (libraries, digital libraries) use to be pragmatically designed for convenience of human access and not with a semantically clearly motivated rationale underlying. It is unclear how well logic-oriented knowledge modeling approaches fit here. [van der Vet and Mars, 1996] also noticed the not yet completely clear semantics of precoordinated index terms. If `cures(Aspirin,Headache)` is the index of a document, this is certainly not an assertion because the statement is only the topic of the document.

Contextual Structure

As argued above, under contextual structure we subsume all document meta information which is not directly contained in the document:

1. Standard attributes (like author or creation date) do not impose new requirements on the representation formalism (we need only some factual assertional formalism).

2. For documents generated within the company, their creation *context* in terms of modeled business processes requires attribute values which can be references to entities defined in other parts of the knowledge base (namely the enterprise ontologies).

3. Steier *et al.* [Steier *et al.*, 1995] pointed out that besides the factors characterizing the content of searched information in a business application, knowledge delivery services have also to regard a number of serach constraints. These concern document source and document meta information. [Steier *et al.*, 1995] propose three categories of not content-related document meta-information, namely *form* meta-information, *quality* meta-information, and *resource* meta-information. These denote, e.g., information about medium, indexing and ease of access, expected answer time for a given query, or expected costs required to produce the answer. We see virtually all this information as properties of document *sources* rather than properties of single document instances. One important observation is that complex statements about search constraints to be regarded can be formulated as expressions over the notion of context as defined in KnowMore via the concept of extended workflow notions.

2.2.2 Design

To sum up, the specific requirements coming from the several document modeling dimensions concern some basic object-oriented modeling facilities plus a comfortable means for defining and annotating attributes (e.g., if we model structure, we have to provide some part-of relationship and must be able to specify how relevance statements are inherited via part-of links; if we would like to model layout, we must be able to import some notion of spatial relationships and must be able to specify how these can be exploited in our inference). Furthermore, there are some general requirements coming from our goal of producing solutions which are oriented towards real-world industrial environments. These concern particularly that it should be possible to easily embed our software in a conventional IR infrastructure, that our solutions should be scalable if the OM grows, and that our notions should be intuitive and easy to understand, as far as possible.

These aims led to our general design approach: the knowledge description formalism is designed as modular and extensible as possible. On the conceptual level, this is achieved by an ontology-based, modular architecture. As sketched in Figure 2.1, knowledge-item descriptions comprise descriptions of content in terms of a domain ontology and / or topic hierarchy, descriptions of document representation in terms of an information ontology, and descriptions of context in terms of an enterprise ontology / business process model. All these dimensions could be realized by links into external tools, but are currently also represented in our KnowMore representation language. Each sub-ontology could be extended by linking additional ontology modules providing appropriate notions for additional properties to be modeled. From an ontological view, knowledge-item descriptions are instances of concepts of an *Information Ontology* in the broader sense, an application ontology in the sense of van Heijst *et al.* [van Heijst *et al.*, 1996a]. The several parts—information ontology (in the narrower sense), domain ontology, and enterprise ontology—can be seen as domain ontologies in the sense of van Heijst *et al.*, which in turn might be implemented incorporating generic ontologies as, e.g., Allen's spatial relations for talking about document layout [Allen, 1984]. On the implementation level, we defined a rigorous object-oriented formalism which is by nature easily configurable and extensible. This basic representation formalism (essentially implementing a representational ontology in the sense of Gruber's Frame Ontology) is presented in Section 2.3 and provides the basis for defining all other parts. For reasons of efficiency and integratability, the formalism is as close as possible to approaches in the deductive database community. In order to be also highly flexible with respect to inferences exploiting special relationships between knowledge items, we provided a powerful annotation mechanism. The complete knowledge-description formalism is then derived from the repesentational means by configuring it with the appropriate ontologies. Some simplified examples how this can be achieved are presented in Section 2.4.

2.3 Language Specification

The KnowMore representation language is an object-oriented and relational representation language. The integration of these two paradigms was accomplished mainly by *unifying* the following concepts found in these and other programming paradigms:

- *classes* as found in object-oriented programming languages;
- *relations* as found in logic programming languages and in databases;
- *types* as found in functional and conventional imperative programming languages.

As a result, the KnowMore representation language inherits the concepts classes, inheritance, objects, and methods from OOP and set orientation from relational databases. In the following, the basic concepts of the KnowMore representation language are discussed.

2.3.1 Class Declarations

The KnowMore representation language is a strictly typed formalism where types may be either class names or complex type specifications. All types and thus all classes must first be declared. For a simple top-level class, the declaration looks like this:

$$\mathit{class}\ (\mathit{attribute}_1 : \mathit{type}_1, \ldots, \mathit{attribute}_n : \mathit{type}_n)$$

Here, type_i may be either a class name or a complex type specification which will be introduced in the next section.

Example:

```
human(
  name   : string,   // string and number are
  age    : number,   // built-in classes
  father : human,
  mother : human
)
```

If a class class_1 extends a class class_2, the following format is used:

$$\mathit{class}_1 : \mathit{class}_2\ (\mathit{attribute}_1 : \mathit{type}_1, \ldots, \mathit{attribute}_n : \mathit{type}_n)$$

Here, $\mathit{attribute}_i$, $1 \leq i \leq n$, are additional attributes for class_1 (which must not already occur in class_2) while inheriting all attributes from class_2.

Example:

```
man : human()
woman : human(
  maidenName : string
)
```

2.3.2 Complex Type Specifications

Although—from a theoretical point of view—it is sufficient to use only built-in and user-defined classes as types[1], meaning that the corresponding attribute may take a single object as value, additional type specifications are introduced: sets and annotations.

---

[1] A class representing a set of humans could be defined as follows: `setOfHumans(first : human, rest : setOfHumans)`. Since not all attributes must have a value for a certain object, lists are terminated by an object of class `setOfHumans` with attribute `rest` left unset.

Sets

Set-valued type specifications have the form $\{class\}$ meaning that the corresponding attribute takes a set of objects as value.
An alternative representation of the parent-children relationship in class `human` is to use the attribute `children` instead of `father` and `mother`:

```
human(
  name     : string,
  age      : number,
  children : {human}
)
```

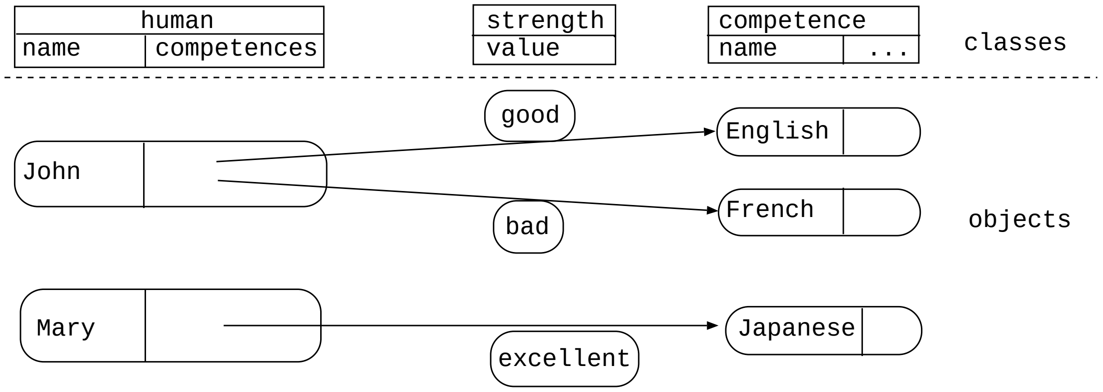
Note that a set of objects is equivalent to a relation or a class; it may either be constant (i.e. simply stored in a database) or computed by a method (see section 2.3.4).

Annotations

In the KnowMore representation language, attributes are used to model semantic nets. In order to allow edges in these nets to be labeled with objects, a new complex type constructor was introduced, namely annotations. Annotated attributes are declared as follows:

$$attribute : class_1 \ / \ class_2$$
or (for set-valued attributes)
$$attribute : \{class_1\} \ / \ class_2$$

In both cases, $class_2$ is the annotation class. Objects of this class can be used to annotate values of the corresponding attribute, thus labeling the edge between an object of $class_1$ and an object of $class_2$.
In the following example, edges between humans and competences can be labeled with objects of class `strength`:



```
human(
  name : string,
  ...,
  competences : {competence} / strength
)
competence(name : string, ...)
ann() // the top class of all annotations
strength : ann(value : string)  // e.g. "good", "medium", "bad"
```

In addition to user-defined annotation classes, the KnowMore representation language defines some builtin annotation classes which are interpreted by the inference mechanism. These builtin annotation classes fall into several categories, e.g. connections (like part-of), uncertainty and probabilistic annotations.

2.3.3 Objects

Objects (instances) have a textual represention which allows them to be loaded from and stored to simple text files (similar to a set of facts in PROLOG). An object is notated as follows:

$$\mathit{class}(\mathit{attribute}_1 = \mathit{value}_1, \ldots)$$

If an object is used as value in other objects, it either is embedded or referenced by a name (thus allowing one object to be referenced in more than one place). Objects are given a name (which can be viewed as a user-provided object identifier) like this:

$$\mathit{name} : \mathit{class}(\mathit{attribute}_1 = \mathit{value}_1, \ldots)$$

Example:

```
woman(
  name = "Mary Smith",
  age = 27,
  competences = {competence(name = "japanese language") /
                    strength(value = "excellent")}
)
man(
  name = "John Doe",
  age = 42,
  competences = {english / good, french / bad}
)
english : competence(name = "english language", ...)
french : competence(name = "french language", ...)
good : strength(value = "good")
bad : strength(value = "bad")
```

Note that in the KnowMore representation language predicate symbols and function symbols are not distuingished—they are both class names.

2.3.4 Methods

As was mentioned in section 2.3.2, set-valued attributes may not only be constant but also be computed by methods.

Since a set of objects is equivalent to a (database or PROLOG) relation (which is a set of tuples), methods may either be defined by expressions of an extended relational algebra (called OCRA—object-centered relational algebra) or by PROLOG-like rules.[2]

Parameterless Methods

Example using an OCRA expression to calculate the set of children:

```
human(
  name   : string,
```

---

[2] As in CycL ([Lenat *et al.*, 1988]), we unify single values and singleton sets thus allowing methods for single-valued attributes.

```
  age    : number,
  father : human,
  mother : human,
  children : {human} =
    union(
      sel[father == this](human),
      sel[mother == this](human)
    )
)
```

Here, `union` and `sel` denote set union and selection as used in relational algebras. The relational operators were extended to work on sets of objects instead of ordinary database relations. This mainly means that an expression like `sel[...](class)` ranges over all objects in `class` **and** all its subclasses.

Example using PROLOG-like rules:

```
human(
  name   : string,
  age    : number,
  father : human,
  mother : human,
  children : {human} = {
    Child:human() :- Child:human(father = this).
    Child:human() :- Child:human(mother = this).
  }
)
```

Here, expressions of type *variable*:*literal* are used to identify objects (c.f. the syntax of named objects in section 2.3.3). This is necessary because of the copy-semantics of PROLOG which is not appropriate for an object-oriented language. A rule like `human(name = N) :- human(name = N, father = this).` creates new objects of class `human` where only the attribute `name` has any values, whereas a rule like `Child:human() :- Child:human(mother = this).` creates a subset of `human` which is identical to the second selection in the previous example, i.e. `sel[mother == this](human)`.

Methods With Parameters

Methods may also have arguments and are then notated as parameterized attributes:

$$class_1(method(A:type_1,B:type_2) : \{class_2\} = \ldots\ A\ \ldots\ B\ \ldots)$$

Example:

```
human(
  name   : string,
  age    : number,
  father : human,
  mother : human,
  childrenOlderThan(Age:number) : {human} = {
    Child:human() :- Child:human(father = this, age = A), A > Age.
    Child:human() :- Child:human(mother = this, age = A), A > Age.
  }
)
```

2.4 Configuring the Generic Description Formalism

In this section, a possible usage of the KnowMore representation language in the KnowMore scenario is demonstrated. The following ontologies are considered:

- information ontology
- enterprise ontology
- domain ontology

Figure 2.2 shows the relationships between these ontologies.

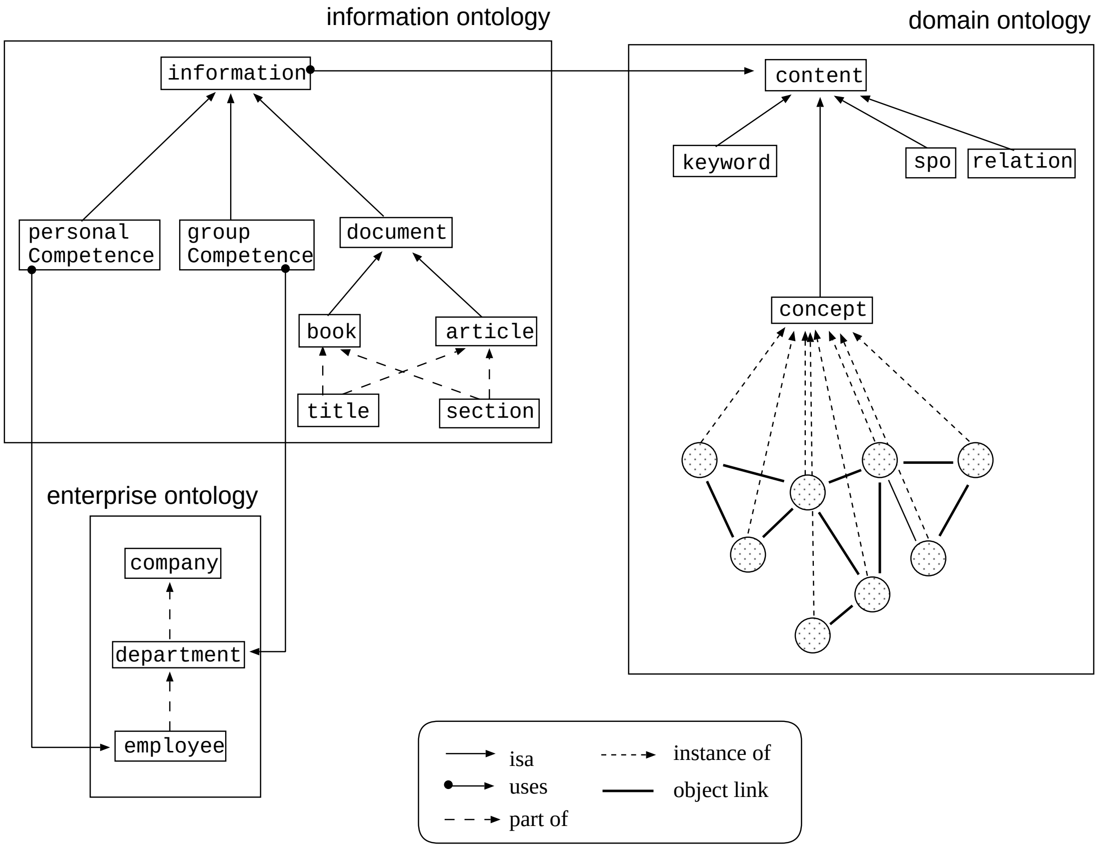


Figure 2.2: The KnowMore Ontologies

Information Ontology In the information ontology, any piece of information has a location (given as a URL) and a content, which is given as a set of content descriptions. Information sources may be documents, data, and rules, as well as references to personal and group competences.

Any association of some piece of information with a content description may be annotated with a `strength` object, which for documents may be some uncertainty value and for personal competences a capability specification.

```
information(  // information source
  name : string,
  url : string,
  content : {content} / strength
)

strength : ann()
uncertainty : strength(value : string)
capability : strength(value : string)

personalCompetence : information(
  employee : employee
)

document : information()

faq : document(
  title : string
)
```

```
article : document(
  title : string,
  authors : {person},
  abstract : abstract,
  sections : {section},
  references : {reference}
)

abstract : document()

section : document(
  title : string
)

memo : document(
  context : contextSpec
)
```

Enterprise Ontology In the enterprise ontology, the organizational structure of a company is modeled, consisting of departments, employees, and their respective roles. In KnowMore, the enterprise ontology is used mainly for two reasons: the employees are actors in business processes, and they have competences and are therefore modeled as information sources in the information ontology.

```
company(
  name : string,
  address : string,
  // ...
  departments : {department}
)

department(
  name : string,
  employees : {employee} / role
)

role : ann(name : string)
```

```
employee(
  person : person,
  phone : string,
  eMail : string
)

// general person database
person(
  lastName : string,
  firstName : string,
  dateOfBirth : string
)
```

Content Descriptions Content descriptions as used in the information ontology may either be unstructured, like keywords or concepts of the domain ontology, or they may be complex, structured terms, like a sentence of the form *subject predicate object*.

```
content()  // abstract super class

keyword : content(
  word : string
)

concept : content(
  name : string,
  ld : {ld},
  links : {concept} / conceptLink
)

conceptLink : ann(...)
isco : conceptLink() // is subconcept of

ld( // ld = linguistic description
  lang : string, // German, English, ...
  name : string,
  abbr : string
  // linguistic attributes ...
)
```

```
// complex content descriptions

spo : concept(
  subject : concept,
  predicate : concept,
  object : concept
)

relation : concept(
  relationship : concept,
  objects : {concept}
)
```

Domain Ontology Since the domain ontology is a complex semantic net, all primitive concepts are instances of class `concept` where the interrelationships are modeled with the `links` attribute and some annotation objects like `sub` (= subconcept of).

```
science : concept(
  name = "science",
  ld = {ld(lang="English", name = "science"),
        ld(lang="German", name = "Wissenschaft")
       }
)

cs : concept(
  name = "computer science",
  ld = {ld(lang="English", name = "computer science"),
        ld(lang="German", name = "Informatik")
       },
  links = {science / sub}
)

sub : isco()  // standard subconcept link

hw : concept(name = "hardware", ld = ..., links = {cs/sub})
gc : concept(name = "graphics card", ld = ..., links = {hw/sub})


// complex concept:
// "the millennium graphics card is compatible with pc345"

mcp : relation(
  relationship = compatibility,
  objects = {millennium,pc345}
)
```

```
compatibility : concept(
  name = "compatibility,
  ld = ...
  links = ...
)

millennium : concept(
  name = "millennium graphics card",
  ld = ...
  links = {gc / sub}
)

// a concrete pc:
pc345 : concept(
  name = ...
  ...
)
```

2.5 Implementation

The KnowMore representation language prototype was implemented in JAVA [JAVA, 1998] whichs allows it to be used on all JAVA-enabled platforms like UNIX, Macintosh, and Windows without any porting efforts.
The core part of the implementation maps the basic language constructs to conventional relational databases which are coupled with JAVA via JDBC [JDBC, 1998].
In particular, classes are mapped to relations and objects to tuples, while embedded objects are represented by their object identifiers (a concept similar to primary keys in relational databases).
Set-valued attributes can be mapped in various ways; in our first prototype, we use blobs (binary large objects) or memo fields (depending on the expressiveness of the underlying database system) which allow values of arbitray size. An alternative mapping would introduce an additional binary relation for every set-valued attribute.

Example:

```
// classes
human(name : string, age : number, children : {human})
man : human()
woman : human(maidenName : string)

// objects
jd : man(name = "John Doe", age = 42, children = {md,pd})
md : woman(name = "Mary Doe")
pd : man(name = "Peter Doe")
```

These classes and objects are mapped to the following relations:

human	oid	name	age	children

man	oid	name	age	children
	jd pd	John Doe Peter Dow	42	md,pd

woman	oid	name	age	children	maidenName
	md	Mary Doe			

In our current implementation, all database relations are loaded into main memory (from text files or from a relational database) which is quite efficient for small databases (up to about 5 MB). For larger databases, only small relations will be processed in this way—large relations like those in many legacy databases will be left in the external database.
Inferences, i.e. the execution of methods, are processed in several steps: PROLOG-like queries are transformed into OCRA queries, which are than mapped into internal relational queries for relations in main memory or into SQL queries for external relations.

2.6 Relevant Work from Others

Logic-Based Information Retrieval

In the IR literature, two modeling approaches are predominant at the moment; one is based on description logics, the other recurring to the logical datamodel (Datalog).

The use of description logics for IR is attractive because object-centered knowledge representation is a natural way for describing many interesting issues in information modeling; moreover, it is quite plausible to understand retrieval as a classification process which tries to classify the formalized information need with respect to the subsumption hierarchy describing document topics and document structure. Other reasons are the sound formal semantics of the reasoning process, the ability to cope with incomplete knowledge, and the possibility to automatically check domain and document models with respect to the subsumption hierarchy. Some authors also mention the uniform formalism for document descriptions, queries, and answers as an advantage; another reason is the fact that the terminological reasoner can treat all kinds of occurring knowledge, ontological as well as concrete document knowledge together with the domain knowledge in a single inference.

Our formalism also provides comfortable means for object-centered knowledge representation with a simple, intuitive semantics. We will develop reasoning methods based on inferences for standard first-order semantics, similar as it is done in the Frame-Logic approach. We have to deal with incomplete information as well as with local scopes of statements (the statement a document gives is an assertion true within the local context of this document, as proposed by [Rölleke and Fuhr, 1996]).

On the other hand, there have been several proposals for extending description logics for document modeling:

- Lambrix & Padgham [Lambrix and Padgham, 1995, Lambrix and Padgham, 1996, Lambrix and Padgham, 1997] introduced an extension for handling part-of relations and order of parts for dealing with document logic;

- Lambrix *et al.* [Lambrix *et al.*, 1997, Wahllöf, 1996] showed how default reasoning at assert time and at query time can ease dealing with incomplete document descriptions and improve precision of answers;

- Several authors [Sebastiani, 1994, Meghini and Straccia, 1996] combined terminological with probabilistic and relevance logic, respectively, in order to reflect the uncertain nature of indexing and retrieval processes.

These numerous extensions indicate that there are at least some limitations concerning the appropriateness of description logics for document modelling. Within the Condorcet project at Twente University there were also principal doubts that current description logic systems could meet the performance requirements raising from real-world intelligent IR applications [Speel *et al.*, 1995, Speel, 1995]. Although some progress has been reported recently in the area of building high-performance terminologcial reasoners [Hendler *et al.*, 1996, Hendler *et al.*, 1997a], these efforts are oriented towards efficient exploitation of standard terminological inferences, and it is not to foresee how these efficiency improvements could be transferred to an extended expressiveness as required by comprehensive document modeling.

Essentially, the same holds true for intelligent IR approaches relying on the logical data model. Object-centered modeling facilities were proposed as "syntactic sugar" layers on top of conventional Datalog in order to provide a comfortable means for a declarative description of the several document modeling dimensions [Rölleke and Fuhr, 1997]. Inheritance and other reasoning services need to be expressed by rules [Hendler *et al.*, 1997b]. Probabilistic extensions were proposed as conservative extensions which basically preserve the Datalog data and inference structures but extend them by some mechanisms for dealing with uncertainty which is expressed within the Datalog model and processed a posteriori after retrieving possibly relevant facts [Rölleke and Fuhr, 1997]. The modeling approach is essentially the same as with terminological logic (see [Fuhr, 1995] for a comparison of both modeling approaches). Although there were already encouraging results for sample document bases of non-trivial size (the TREC AP newswire data containing about 28000 documents, however, modeled without considering structure), the quite conventional DDB approach does not make beneficial use of ontological structures for query optimization. A more seamlessly integrated approach combines the structured object model NF^2 with some notion of uncertainty [Fuhr and Rölleke, 1996]. However, there exist neither advanced reasoning facilities nor an efficient implementation of this idea.

Hence, our research aimed at designing a formalism which provides basically the same representational facilities as pNF^2 but together with an efficient implementation on the basis of the ideas underlying the HySpirit system which relies on "flat" Datalog. To this end, it is necessary to define a machinery which deeply integrates the object-oriented flavour into the reasoning process.

In the area of intelligent IR there are mainly conventional DBMSs in use; it is nevertheless nearby to examine approaches to deductive object-oriented databases with respect to their usability for our purpose. Recently, [Neven and van den Bussche, 1997] pointed out that structured-document retrieval could be beneficially supported by means of deductive object-oriented databases. As [Vila *et al.*, 1996] argued, such an object-oriented data model would already suffice to represent several frequent phenomena of imprecision and uncertainty in data modeling.

However, there is a broad range of possible combinations of relational / deductive and object-oriented features and it is far from being commonly agreed upon how a preferable solution should look like. [Fernandes *et al.*, 1992] analyzed a number of different ways of designing deductive object-oriented databases and concluded that the most promising approach is what they call the "Stony Brook" approach with Frame-Logic, or F-Logic, respectively, as the most prominent representative [Kifer *et al.*, 1995]. Basically, idea and notation of our formalism can be seen in the spirit of F-Logic. However, F-Logic provides too much expressiveness which is not needed in the document modeling scenario (e.g., integrated reasoning over schema and instances) such that it would make sense to create a special-purpose formalism with exactly the required expressiveness especially tuned towards efficiency. Our resulting formalism comes close to the Rule-Based Object Language (ROL) [Liu, 1998b, Liu, 1998a], a pragmatic but feasible integration of rules and objects. ROL can also be seen to be inspired by F-Logic, but downsized to a tractable expressiveness, with the two aims of elegant, deep paradigm integration and clear, intuitive syntax and semantics.

Knowledge-Based Access and Integration of Internet Information

Recently, there has been a growing number of research efforts investigating the use of formal logic and ontology based methods to finding and integrating Internet information sources. Most of these approaches do not extensively discuss questions of complex document structure, capturing informal content or even document context, but rather focus on the integration of databases with different schemata (see, e.g., [Ambite and Knoblock, 1997, Kirk *et al.*, 1995]. Although it is an interesting question how our KnowMore approach fits into their conceptual frameworks, these approaches miss some points important in the organizational memory context, especially at the intersection of formal and informal knowledge. Some projects, like SHOE at the University of Maryland [Luke *et al.*, 1997] or Ontobroker at the University of Karlsruhe, Germany [Fensel *et al.*, 1998], go one step further; they impose structure on informal documents by the addition of semantical tags to HTML documents. The tags are taken from an ontology structuring the domain of discourse the HTML documents are talking about; so, the content of these WWW pages can be queried in a similar way as formal knowledge bases. If one would gather all these semantic tags in a condensed description and represent the original file content separately, accessible by a link within the description file, the functionality were exactly the same as provided by the KnowMore approach. However, both SHOE and Ontobroker do only care about HTML sources and do not discuss how to integrate other kinds of information sources. Moreover, at the implementation level, the systems rely on description logics or the Frame-Logic approach, respectively; as argued above, we propose a specialized representation and inference engine which can be optimized to the special needs of information retrieval. The most important point which will be attacked in KnowMore but is not mentioned in the other approaches is the representation of uncertain relationships. This is mainly due to the fact that in the applications these approaches are also aligned to the "hard facts" hidden in informal information sources in order to formally reason about them for retrieval. Except for the Untangle project at Vassar College [Welty, 1994, Welty, 1996, Welty, 1998] they do not discuss the "topic" problem which leads to the question how to conjointly process factual knowledge about document meta content or domain model together with vague content representations expressed over rough classification systems. However, this problem is still unsolved.

Chapter 3

Constructing an Integrated Ontology/Thesaurus

3.1 Motivation

In [Kühn and Abecker, 1997] and [Abecker *et al.*, 1997a] the conceptualization for a generic computer-based organizational memory system was presented based on experiences from industrial case studies. Several crucial prerequisites for the practical realization of such a system were identified and a tentative architecture was suggested.

One of the core requirements for the development of an organizational memory is to limit the effort for up-front knowledge engineering. Expert time is generally scarce and the costs for manual knowledge acquisition for building a knowledge base of sufficient size to be useful are for most companies prohibitive. The construction of an organizational memory has thus to be based on easily available information sources, mostly in the form of natural language documents.

On the other hand, an organizational memory should provide intelligent assistance to knowledge-intensive tasks in complex work-processes, which goes beyond the functionality of a mere information retrieval system. Therefore, an organizational memory has to include some formally represented knowledge. Furthermore, an organizational memory has to be tightly integrated with application software used in the various tasks as well as with all kinds of useful information sources such as databases or electronic document repositories.

For structuring, accessing and maintaining large amounts of heterogeneous information, appropriate meta-level descriptions are needed which specify the structure, content, and potential usage of the object-level knowledge.

Such meta-descriptions are provided for data in the form of data models, and for formal knowledge as ontologies [Gruber, 1994]. As pointed out in [Schreiber *et al.*, 1996], ontologies can be seen as enhanced data models, or data models as simplified ontologies. They serve to describe both the structure and the contents of databases or knowledge bases respectively. The construction of an ontology is a rather difficult and time-consuming process which may only be alleviated if previously constructed ontologies exist which may be reused [Schreiber and Birmingham, 1996].

In electronic libraries and document management systems, document schemata describe the structure of documents whereas thesauri provide relevant terms and term-relations for organizing the collection of documents and accessing their contents. Contrary to ontologies, domain-specific thesauri can be constructed automatically from a given document corpus [Grefenstette, 1994].

Ontologies and thesauri have so far been developed and used in different research communities so that the potential benefits of their integration remain largely unexplored. A suggestion to apply linguistic text analysis techniques to ontology-building for knowledge-based applications only has been recently made by Guarino [Guarino, 1996], who also observed that suitable tools and techniques are still missing.

An investigation of tools and techniques for an integration of ontologies and thesauri is the principal objective of the current paper. For building an organizational memory, the question to what extent automatic thesaurus generation methods can be exploited to reduce the costs for up-front knowledge engineering is of particular importance.

The next section gives a brief comparison of ontologies and thesauri, which so far have been considered mostly in isolation. The subsequent section presents the automatic thesaurus generation tools TRex, which includes a variety of generation methods described in the literature. It is then shown, how this tool can be applied for the conjoint construction and evolution of an integrated Ontology/Thesaurus and the respective knowledge base. Finally, some first results and conclusions are presented together with open questions for future investigation.

3.2 Ontologies and Thesauri

3.2.1 Ontologies in knowledge-based systems

Ontologies as explicit specifications of the conceptualization underlying a knowledge-based system were broadly advocated by the *ARPA Knowledge Sharing Initiative* [Neches *et al.*, 1991], and have since become a focus of many research activities. Explicit ontologies are both a prerequisite for enabling knowledge sharing and reuse, and may be shared and reused themselves to guide and facilitate the development of knowledge-based systems [Gruber, 1994]. For this purpose, a language for representing ontologies was developed as an extension to KIF [Gruber, 1991] and a basic library of sharable ontologies was provided. Its use has, however, been more or less limited to some research initiatives [Schreiber and Birmingham, 1996] so that the practical utility of reusing an ontology is rather doubtful.

In the context of an Organizational Memory, a new domain ontology will have to be developed for each organizational unit, with little opportunities for reuse. Rather than being reusable, domain ontologies should be easy to develop and adaptable to accommodate new knowledge.

In the *KADS approach to knowledge engineering* [Schreiber *et al.*, 1993], several kinds of ontologies were distinguished depending on the type of ontological commitment they contain which in turn determines their potential for sharing and reuse. The most important distinction is that between *domain ontology* and *representation ontology* [Wielinga *et al.*, 1992]. The former defines the terms and schemata which occur in the domain model, i.e. the representation of the domain knowledge, whereas the latter is a kind of meta-theory defining the terms and structures by which the domain ontology is specified. The construction of a domain ontology obviously has to be repeated for each domain, whereby the representation ontology may be used as a kind of representation language or template.

To summarize, KADS advocates a task-oriented, top-down knowledge engineering approach, which may be well suited for the development of task-specific expert systems, but is hardly applicable in the development of an organizational memory which has to capture frequently changing knowledge to be reused for several tasks. The approach of re-using a generic representation ontology to support the development of a domain ontology is nevertheless useful. The latter should, however, be built primarily with bottom-up methods, e.g. text analysis.

In the *KACTUS project*, the relationship between ontologies, data models, and design patterns employed in software engineering was explored, and detailed guidelines for designing domain ontologies were developed [Schreiber *et al.*, 1996, Ostermayer *et al.*, 1996]. Similar to KADS, of which KACTUS is a successor project, different types of ontological commitments are distinguished, and a tool-set for the development and translation of ontologies between different representation formalisms (ONTOLINGUA, CML, and EXPRESS) is provided.

In a recent special issue of the International Journal of Human-Computer Studies on "Using explicit ontologies in KBS development" [van Heijst *et al.*, 1996a], a current review of many theoretical and practical issues concerning ontologies is presented. In this paper, we will follow in particular the suggestions made by Guarino [Guarino, 1996], who advocated that domain analysis and task analysis should be given equal attention and that linguistic analysis techniques and thesauri should be employed to support the construction of domain ontologies.

3.2.2 Thesauri and thesaurus generation methods

Thesauri are best known as dictionaries providing synonyms, antonyms and related terms for commonly used words and expressions. Besides these general linguistic thesauri, special purpose thesauri have been used for decades for documentation, catagorization and retrieval in libraries, archives and databases. National and international guidelines for the development of such thesauri have been established [DIN Deutsches Institut für Normung, 1987], and numerous computer tools for the manual creation and management of such thesauri are available [Milstead, 1990].

Thesauri play an important role in solving searchers' vocabulary problem during information retrieval. *Intelligent information systems* such as CODER, I3S, or UMLS (for an overview see [Chen *et al.*, 1993]), attempt to capture experts' domain knowledge for information retrieval in thesaurus-like knowledge bases constructed with traditional knowledge acquisition methods. These systems assist users articulating their queries and perform replacement of search terms to increase retrieval precision and recall.

To support the *generation of thesauri*, several algorithmic methods have been suggested, which are virtually all based on the statistical co-occurrence of words in texts [Salton, 1972, Grefenstette, 1994]. Of the various relations between terms found in a manually constructed thesaurus (e.g. abstraction and partition hierarchies, definitions, and antonyms), these techniques can only identify an unspecific association or similarity between terms together with common expressions and word families in which individual terms occur.

Nevertheless, these knowledge-poor techniques have proved useful in generating thesauri for information retrieval in Electronic Community Systems in several biological and medical research areas [Chen *et al.*, 1995]. Some results presented by [Grefenstette, 1995] bear considerable resemblance to hand-built thesauri, even though only lexical syntactic analysis with no domain-knowledge and minimal morphological analysis is employed.

Latent semantic indexing (LSI) has been suggested as a completely automatic method to determine synonymy between terms which need not co-occur in the same documents [Derweester *et al.*, 1990]. It is based on a singular-value decomposition of the term-document matrix, from which only a limited number of orthogonal dimensions (typically 100 to 300 from several thousands) are retained. Whereas the authors claim that LSI has outperformed standard vector methods and other variants in almost every case [Dumais, 1995], independent evaluations have shown no improvements in retrieval precision and recall for other document corpora. Whether LSI can be profitably applied to identify synonyms and related terms in automatic thesaurus generation remains an open question.

The NSF-project "Building a Concept Space for an Electronic Community System" at the University of Arizona employs a three step procedure for automatic thesaurus generation consisting of object filtering, automatic indexing, and cluster analysis [Chen *et al.*, 1995]. In the first step, domain-specific knowledge is supplied in the form of domain-specific keywords which are used to identify important concepts in documents. They also observe that such a list of keywords can usually be obtained easily from available sources. In our approach, we will build on this approach and will further investigate how *algorithmic thesaurus generation methods may be enhanced by supplying domain knowledge* not merely in form of a keyword list, but as an initial domain ontology providing concept definitions together with abstraction and part-of hierarchies.

3.2.3 A Brief Comparison of Ontologies and Thesauri

Ontologies and thesauri have many interesting commonalities and differences which should be examined before a suggestion for an integration is made.

Common features of ontologies and thesauri:

- both include formal relations between concepts such as abstraction hierarchies, partonomies, and definitions,
- both are designed to convey the intended meaning of concepts to humans not just for usage by computers,
- both are used to facilitate retrieval of needed information from large amounts of available information,
- both are essential for translating between different conceptualizations and languages.

Differences between ontologies and thesauri:

- ontologies are usually developed a priori, i.e. they specify assumptions and commitments to which domain theories must comply, whereas thesauri are developed a posteriori by analyzing how language terms are being used in a domain,
- ontologies include concept definitions and integrity constraints, whereas thesauri focus on explicating the relation between concepts and natural language terms (e.g. homonyms, synonyms, abbreviations),
- thesauri emphasize unspecific association relations between concepts (enumeration of related or similar concepts), which are usually not found in ontologies,
- relations between concepts in a thesaurus may be weighted and context-dependent (e.g. "program abortion" may be associated with "program failure" by 0.8 in client reports but only by 0.3 in troubleshooting guidelines provided by software developers),
- ontologies have to be constructed manually, whereas some useful methods for (semi-)automatic thesaurus generation exist.

3.2.4 Sketch of an Integrated Ontology/Thesaurus

As this comparison of ontologies and thesauri shows, their integration offers multiple advantages for the conjoint management of formal and informal knowledge in an Organizational Memory and for limiting the efforts to be invested in up-front knowledge engineering.

An integrated Ontology/Thesaurus , which combines the advantages of ontologies and thesauri, should have the following three characteristics:

1. it includes both formally defined concepts as well as natural language terms,
2. it includes both formal relations with a well defined semantics as well as unspecific, weighted associations,
3. it is developed and evolved in parallel with the knowledge base and the document collection included in the Organizational Memory, profiting from automatic thesaurus generation methods.

Such an integrated Ontology/Thesaurus can be represented with a knowledge description formalism, as suggested in [Sintek and Abecker, 1998]. The construction and evolution of the Ontology/Thesaurus will be described in the subsequent sections of this paper.

In the Organizational Memory, the integrated Ontology/Thesaurus will support knowledge acquisition, knowledge utilization and user interaction. The following uses are of particular importance:

- performing query expansion for document and database retrieval,
- supporting the categorization and description of new knowledge items by helping to automatically fill the schemata as described in the previous section,
- helping the users to formulate queries and entering new knowledge to the Organizational Memory,
- explaining to users why a particular piece of information was provided in particular task context,
- supporting the collection and interpretation of relevance feedback from users in order to improve the quality of information which is supplied to the users by the Organisational Memory.

3.3 The Automatic Thesaurus Generation Tool TRex

TRex is a flexible tool for automatic thesaurus generation which includes a variety of state-of-the-art methods. TRex has been originally developed to generate a German thesaurus for helpdesk support in a large software company, and is now being enhanced to support the construction and evolution of an integrated Ontology/Thesaurus for organizational memories. TRex has been implemented in C++, and on a standard PC with sufficient memory it can handle tens of thousands of input documents, generating a thesaurus with tens of thousands of terms.

Besides efficiency, the design requirements for TRex included support for incremental thesaurus updates and a high degree of flexibility so that different combinations of methods can be tested with various parameters. Furthermore, TRex can exploit

additional knowledge about the structure of documents, specified relations between documents (e.g. customer queries and associated solution notes), and lists of trivial or particularly important words occurring in documents.
An overview of the Trex system is given in Figure 3.1. The documents to be processed are first parsed according to the supplied document schemata, and irrelevant text sections are eliminated. Terms are then generated from selected text segments based on a morphological analysis, an elimination of stopwords and supplied lists of important terms, which are preferentially treated. The generated terms are then stored in a term-context matrix which keeps track how often each term occurred in a particular context. Based on the co-occurrence of terms in the various contexts, term similarities are computed which are stored in a similarity thesaurus.
All these processing steps may be controlled by numerous parameters. The inputs, major processing steps and the obtained results will now be described in more detail.

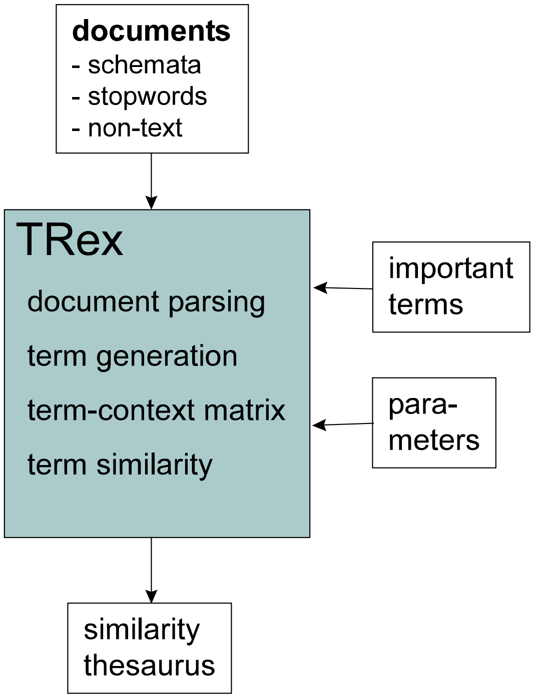


Figure 3.1: Overview of the thesaurus generation tool TRex

3.3.1 Input documents and related information

TRex has been designed to handle ordinary documents written by customers and employees of a company in order to record and exchange information. Such documents are far less homogeneous and of poorer quality than documents usually employed for automatic thesaurus generation, e.g. scientific journal articles, encyclopedias, or works from literature. The comparatively poor quality of ordinary documents such as business letters, e-mails or product documentations manifests itself not only in an abundance of typos, idiosyncratic abbreviations, arbitrary punctuation and grammatically incorrect sentences, but most importantly in a mixture of important and unimportant information which makes the statistical extraction of relevant term relations very difficult.
In order to deal with this problem, TRex allows the user to specify additional information about the sets of documents to be processed. In particular, the following kinds of

information may be specified:

1. *Document slots indicated by tags:* If the documents have a fixed structure consisting of a number of slots separated by tags (e.g. title, abstract, keywords and normal text), this structure may be specified so that text occurring in individual slots may be ignored for thesaurus generation or given a higher weight.

2. *Document-specific stopwords:* If the documents are known to include frequently occurring words or strings which are not related to the contents, these words may be specified so that they can be ignored in the subsequent thesaurus generation. This is also useful for conveniently removing formatting tags from documents (e.g. html or Tex) without preprocessing.

3. *Strings indicating irrelevant text (non-text):* If documents contain a mixture of relevant and irrelevant text passages, the user may specify strings which indicate the presence of an irrelevant text passage, together with the action to be taken, when such a string is encountered. The possible actions include:

    (a) ignore the current line, section, or document slot,
    (b) ignore the rest of the current line, section, or document slot,
    (c) ignore the preceding text in the current line, section, or document slot.

    By specifying appropriate strings and actions, frequently but inconsistently occurring text passages such as company headers, copyright remarks, or interspersed programming code may be removed in a convenient way.

3.3.2 Lists of important terms

When constructing a thesaurus from a document collection, a domain specific list of important terms which should be included in the thesaurus is often known in advance. TRex allows the specification of several important term lists, so that different kinds of important terms (e.g. products, person names, technical terms occurring in product descriptions, error codes) may be grouped together.

For each term-list the user may furthermore specify a weighting factor together with processing directives (e.g. product names are to be included both as single-words terms and as phrases with subsequent unknown words). More details will be given in section 3.3.5.

It should be noted here that the individual members in each list of terms may be considered to be instances of some class (e.g. 'product' or 'person'). Obviously, such a term-list could also be used to specify the subclasses or the components (partonomy) of some concept. This indicates, how an initial ontology may be exploited for thesaurus generation, even though TRex does currently not allow to specify hierarchical term-relations. This topic will be discussed further in section 3.4.

3.3.3 Generation parameters

TRex allows the specification of numerous parameters which affect the individual processing steps from document parsing to similarity computation. TRex can thus be easily tuned to particular kinds of input documents, as well as to kinds of desired results. The effects of some of these processing parameters will be discussed in the following sections.

Default values are provided for most parameters. They will usually yield a good result, which can be further optimized by parameter tuning. Since thesaurus generation in TRex is performed in a sequence of processing steps, the user may inspect the result of each step and opt to repeat it with a different set of parameters in order to check whether more satisfactory results can be obtained.

3.3.4 Document parsing

Documents are parsed in a sequence of three steps, when a fixed document structure has been specified; otherwise only the last two of the following steps are performed:

1. **Document slots:** When document slots indicated by tag-words have been specified, theses slots are filled with the corresponding text when reading the documents. When a specified slot-tag is not found, the processing of the current document is aborted and an error message is printed. Each slot of a correctly read document is then parsed further in the subsequent steps. When no document structure was specified, the document is considered to contain just one slot to which the entire text is assigned.

2. **Document sections:** The lines of text read into individual document slots are grouped into document sections, based on empty lines, indented lines, and differences in line length of subsequent lines. The employed procedure is just a simplified version of more elaborate text structure recognition approaches used in document analysis.

   Line boundaries, and hyphens occurring at the end of lines are then removed so that each document section consists of a single string of characters.

   Each section is then scanned for strings indicating irrelevant text, and appropriate actions are taken, as described in section 3.3.1.

3. **Word tokens:** The character string for each text-section is scanned character by character and cut into word tokens based on processing parameters indicating which characters may occur within a word and which characters are word delimiters.

   Digits as well as characters preceding and following digits are given special treatment so that numbers and combinations of numbers with letters and special characters, which often occur in product descriptions or technical documents, will be parsed correctly.

   Punctuation characters are used to delimit sentences and clauses, and are then removed together with other special characters.

   A normalization with respect to umlauts and upper/lower-case is performed, and the presence of such characters is indicated by flags stored with the respective word-token.

The result of this processing step is a sequence of normalized word-tokens annotated by flags. These flags indicate which normalizations were performed and whether the token occurred at the end of a clause, sentence or section. Preserving this information is important in order to enable a disambiguation in subsequent processing steps. On the other hand, a normalization of word tokens is performed at this early stage in order to enhance the efficiency of the next step.

3.3.5 Term generation

Each word token is subjected to a *morphological analysis* which, if successful, yields the word stem, the grammatical word category (e.g. noun, verb, adjective) and the word components. Often enough, the word-token is classified as unknown, or the word category is ambiguous. In the latter case, a disambiguation is attempted based on the original spelling flags and the word types of neighbouring words (e.g. if something might be a noun or an adjective and is preceded by an article and begins with an uppercase, assume it is a noun).

Each word-token and word-stem is then looked up in the list of *stopwords*, and the various lists of *important words*. Occurrence in each of these lists is tagged by setting respective flags.

The *term-selection and phrase-generation* is performed according to rules which may be specified by the user. In the current implementation of TRex there are four methods which determine, whether a word-token should be included as a single-word term, whether it should be used as the starting point for a phrase, whether it may continue a phrase, whether it should be ignored within a phrase. The latter category was included so that different combinations of words can be mapped onto the same phrase.

Phrase generation is also limited by end-of-clause marks and by a maximum-phrase-length parameter. The latter had to be included due to the poor quality of the input texts: some document sections contained almost no punctuation marks so that appropriate clause boundaries could not be determined.

Whereas in the current implementation of TRex, term-selection and phrase-generation rules can only be specified by modifying the respective program code, one could also provide the user a list of options, which may be selected or deselected on a graphical interface.

One important question is, whether to use liberal or restrictive term-selection and phrase generation rules. Too restrictive rules may discard important information, whereas too liberal rules may generate more terms than can be handled. We currently opt for a liberal approach, keeping as many terms and phrases as possible and eliminating some of them later if we run out of space during the construction of the term-context matrix.

To summarize, the result of this processing step is a sequence of terms, which may be either single-word terms or phrases. Each term carries quite a lot of additional information, which has been accumulated in the processing steps. This information comprises the spelling prior to normalization, the word-token prior to stem reduction, information about grammatical categories and word components obtained from the morphology, information about occurrence in the list of stopwords, and in the lists of important words.

An example for the processing results obtained up to this step is given in figure 3.2. As can be seen, a considerable condensation of information has been performed, by reducing an entire document to just two terms.

3.3.6 Construction of the term-context matrix

The generated terms are added to the term-context matrix which records how often each term occurred in each context. Possible contexts which have been suggested for automatic thesaurus generation are:

1. the entire document (in this most common case the term-context matrix is a term-document matrix),

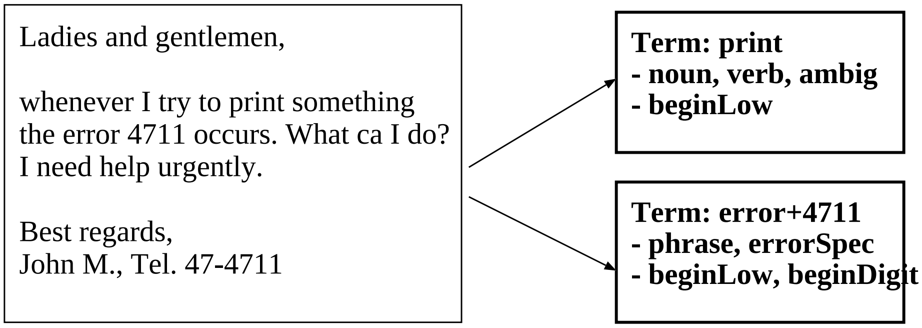


Figure 3.2: Terms generated from an input document

2. individual document sections,
3. a moving text window of size n surrounding each term,
4. categories determined by lexical-syntactic analysis [Grefenstette, 1994].

TRex currently offers the options one and three. Option two could be easily added if needed, whereas option 4 would require a deeper linguistic analysis of the documents. In particular, a proper identification of subject, predicate and object of individual sentences would be needed. This may prove to be even more difficult in German than in English, and almost impossible or at least practically useless for poor quality documents.

The decision on whether to use a document context or a text-window context should be based on the contents and the number of documents to be processed as well as on efficiency considerations. When only a small number of large documents with variable but continuously evolving contents are available (e.g. the bible or a novel) the text-window approach will extract more valuable information. For a large number of small, homogeneous documents (e.g. customer queries collected in a database) using a document-context is more appropriate.

Even though term-context matrices are very sparse, they require a lot of computer memory, in particular in combination with a liberal term-generator. In order to overcome these problems, TRex offers the option to remove rare terms at regular intervals. For instance, one may specify that after processing n documents (or after adding n terms), all terms or phrases which have so far occurred only k times should be removed. Since term frequencies are exponentially distributed an about half of all terms occur only once, even a small k leads to a considerable reduction.

TRex also allows to build several term-context matrices in parallel (e.g. one from each document slot), as well as to store, load, and join individual term-context matrices. This is not only very useful for overcoming memory limitations, but also for performing thesaurus updates and for computing mappings between terms from different matrices, as will be explained in later sections.

3.3.7 Computation of term similarities

In automatic thesaurus generation, two terms are assumed to be similar if they frequently co-occur in the same contexts. For computing a similarity value, different kinds of term contexts, weighting schemes, and similarity measures have been suggested in the literature [Salton, 1989, Grefenstette, 1994].

TRex allows an arbitrary combination of contexts, weighting schemes, and similarity measures, even though not all combinations make sense or provide useful results for information retrieval. It is an open question, however, to what extent a particular combination might yield a specific kind of similarity information which might be useful for constructing or extending an ontology (see 3.6).

Term contexts

Besides "document", "section", and "text-window" contexts which have to be considered already when building a term-context matrix, one may also compute new contexts from a given term-context matrix.

One method, which has become known as *Latent Semantic Analysis* [Derweester *et al.*, 1990], reduces the dimension (and the number of contexts) of the term-context matrix by computing a small number of new orthogonal dimensions, which account for a maximum of the observed variability in the original term-context matrix. The mathematical technique used to compute these dimensions is called *Singular value decomposition*, and is nowadays feasible even for large term-context matrices [Berry *et al.*, 1996]. Several arguments can be given for the theoretical justification of this method:

1. It is reasonable to assume that the observed frequencies in the term-context matrix are caused both by true term associations and more or less random term occurrences. By reducing the number of dimensions while retaining a maximum of the observed variability, the random error is reduced and the true term-associations become more prominent.

2. Orthogonal dimensions provide a clear justification for computing term-similarities.

3. By reducing the number of dimensions over the whole pattern of observed term-frequencies, even indirect term co-occurrences are taken into account, e.g. if term a and term b never occur together, but both frequently co-occur with a third term c.

TRex offers the option to perform a latent semantic analysis of any term-context matrix (based on "document", "section", and "text window"contexts), in combination with different weighting schemes. Term similarities may then be computed from the reduced "term-abstract dimensions matrix'.

Weighting schemes

As previously mentioned, the term-context matrix records, how often each term occurred in each context. This observed *term frequency (TF)* can be used directly for subsequent analyses.

One may question, however, whether the absolute term frequency really is a good indicator. For instance, when a document is about topic A, an associated term might occur only once or be mentioned a large number of times, without affecting the degree to which the document deals with topic A. Therefore, a *binary weight (B)* may be preferable, which only indicates whether a term did or did not occur in a context.

In document retrieval, it has also been suggested to assign a higher weight to terms occurring rarely in the corpus. The so-called *inverse document frequency (IDF)* has been defined as the logarithm of the total number of documents minus the logarithm of

the number of documents with term t plus one [Salton, 1989]. The product of the term frequency (TF) and IDF is one of the the most frequently applied weighting schemes in document retrieval, and it is also available in TRex, labeled as TF*ICF (ICF = inverse context frequency), since other contexts may be used besides documents [1].
Obviously, the weighting schemes B and TF*ICF can be easily computed from TF. TREX thus uses TF when constructing the term-context matrix and converts it to B or TF*ICF later, if requested by the user. Thus the effect of different weighting schemes may be easily tested in combination with different similarity measures.

Similarity measures

For computing the document similarity in document retrieval, the following three measures are most widely used [Salton, 1989]:

1. the **scalar product** of two document vectors,
2. the **cosine** between two document vectors, which is simply the scalar product divided by the vector lengths,
3. the **Jaccard score** which in the case of a binary weighting scheme is simply the intersection divided by the union.

These three similarity measures can also be applied to assess the similarity of term-vectors across contexts. For instance, with the Jaccard score and a binary weighting scheme two terms would be judged the more similar, the more often they occur together and the less often only one of the two occurs in a context.
Besides the above-mentioned similarity measures from document retrieval, special similarity measures have been suggested for thesaurus generation, together with their own weighting schemes. TRex includes two of these measures besides the three standard measures from document retrieval:

1. the **log-entropy score** suggested by Viegener [Viegener, 1997] is a Jaccard score in combination with a local weight based on the logarithm of TF and global weights based on the entropy of the context.
2. the **cluster-weight score** was suggested by the working group on electronic community systems at the University of Arizona [Chen *et al.*, 1993]. Contrary to all other similarity scores, it is asymmetric, i.e. the similarity between term a and term b is not necessarily equal to the similarity between term b and term a.

3.3.8 Similarity thesaurus

The output produced by TRex is a set of terms (single words or phrases) extracted from the documents together with similarity relations between these terms computed by different methods. For each term and each similarity measure selected by the user, the N most similar terms are computed and stored in the thesaurus. As already mentioned, each term is annotated by various kinds of information which were accumulated in the thesaurus generation process.

---

[1] It may be questioned whether TF*ICF which has proved useful in document retrieval where similarities between documents are assessed, should also be applied in thesaurus generation where similarities between terms are sought. One might even suggest to apply instead an analogous TF*ITF weight, where ITF is defined as the logarithm of the total number of terms minus the logarithm of the number of terms in document d. Thereby the occurrence of a term in a shorter document would be given a higher weight than its occurrence in a document with many terms.

Furthermore, additional information such as other terms containing the term as a substring or phrases containing the term can be easily computed from the term-context matrices.

An example of the amount of information for each term contained in the similarity thesaurus is given in table 1. The shown information is to be read as follows: The term ’restore’ occurred 133 times in the corpus, in 70 different contexts (documents). It occurred both with an uppercase and with a lowercase as first character and belongs to the word types noun, verb, and composite noun. It also occurred in the specified lists of important words. One time it occurred in the form ’restores’. The term ’restore’ is a substring of the term ’brrestore’, and a component of the phrases ’restore db’ and ’restore recovery’ which are also part of the thesaurus.

The lower half of the table shows the most similar terms to ’restore’, computed by eight different methods: ’JB’ is the Jaccard score based on a binary weighting scheme, ’CB’ the respective cosine; ’JF’ and ’CF’ are the Jaccard and cosine scores for raw term frequencies , whereas ’JI’ and ’CI’ are based on TF*IDF weights. The last two columns labeled ’LE’ and ’CW’ are the log-entropy and cluster-weight scores mentioned in the previous section.

The numbers indicate the rank-order of similarity scores computed according to the various methods. Only the 10 most similar terms were computed for each method and rank-orders higher than 10 are indicated by a dot. The numbers indicated to the right of the similar terms, show the frequency (F=) and the number of contexts (C=) for these terms.

```
Term: ‘‘restore’’
  Frequency: 133
  Contexts: 70
  Flags: BEGINUPPER, BEGINLOWER, NOUN, VERB, COMPNOUN, IMPORTANT
  Forms: 132 ’restore’, 1 ’restores’
  Components: ’re’ ’store’
  Substring of: ’brrestore’
  Phrases: ’restore db’, ’restore recovery’

Similar terms (rank):
JB CB  JF CF  JI CI  LE CW
 1  1   1  1   1  1   1  1  ’restore’ F=133, C=70
 2  4   2  4   2  5   2  2  ’recovery’ F=110, C=58
 3  2   6  2   5  2   3  5  ’restore recovery’ F=9, C=7
 5  6   3  6   3  7   5  4  ’brrestore’ F=44, C=18
 .  .   5  .   6  .   .  .  ’d11’, F=35, C=13
 4  .   5  .   .  .   4  9  ’recover’, F=25, C=16
 6  3   .  3  10  3   6  8  ’restore db’, F=7, C=6
10  7   .  7   .  4   .  .  ’psapclu’ F=7, C=6
 .  .   .  .   .  .  10  3  ’sapdba’ F=550, C=241
 8  8   4  7   4 10   7  6  ’adsm’, F=47, C=23
 .  .   .  9   .  6   .  .  ’cold’, F=7, C=6
 .  .   8  .   7  .   .  .  ’oninit’ F=19, C=6
 7  9   7  .   8  .   8  7  ’tape’ F=104, C=56
 .  .   .  8   .  9   .  .  ’log-file’ F=17, C=16
```

Table 1: Information about the term ’restore’ provided by the similarity thesaurus.

As can be seen from the example, most of the listed similar terms can be readily recognized as related to the target term by an average computer-literate human. ’restore’ and ’recovery’ are pretty much synonyms, data are usually restored from ’tape’, etc.

The terms such as 'brrestore', 'psapclu', and 'sapdba' are technical terms, for which more expertise is required to judge their actual relation with the term 'restore'.

The example also shows that the different similarity computation methods agree pretty well as to what the most similar terms are, even though there are some striking differences. Whereas all methods basically agree about the three most similar terms, 'JF' and 'JI' consider the term 'd11' to be rather similar, which is far off according to the other methods.

Experiments have shown that the face validity of the above results can be improved, if more than the 10 most similar terms are computed for each method, which effectively 'pushes down' terms which are seen as similar only by some methods. Judging term-similarity by the average rank from a combination of various methods, also seems to outperform any single method. The amount of consensus between the different methods can be used as an indicator of the validity of the found term associations. For the last issues, however, a more thorough evaluation is needed.

3.4 Constructing and Evolving and Integrated Ontology/Thesaurus

As shown in the last section, the thesaurus generator TRex is a versatile tool for extracting many interesting relations between terms occurring in a document corpus. By just looking at the most frequent nontrivial terms, TRex can also quickly identify the major topics which are talked about in the given documents.

A major limitation of TRex is that the identified term relations can only be interpreted as 'has-probably-something-to-do-with' relations. This may be good enough for manual or automatic query expansion in information retrieval, where TRex showed some impressive results, but such a similarity-thesaurus is far away from a hand-built thesaurus let alone an ontology, where semantic relations are required.

Since for the time being we see no promising way of improving TRex so that it can identify semantic relations, the semantic classification of the similarity relations obtained from TRex is performed manually. Even so, a considerable amount of time and work can be saved as compared to a purely manual ontology construction in which interviews with domain experts are conducted and documents are scanned by hand.

The example given in the preceding section also shows that many similarity relations identified by TRex contain information which should be put into the knowledge-base rather than into the ontology. For instance, the relation between 'restore' and 'tape' is due to the fact that "A restore of data is frequently done from tape." Such a piece of knowledge would normally be stored in the knowledge base rather than in the ontology [2].

3.4.1 Generic procedure

A general schema of how TRex can be applied for constructing and updating an integrated ontology/thesaurus together with a knowledge base is shown in figure 3.3.

When starting to build an organizational memory, the knowledge engineer will collect relevant documents about the application domain and will construct an initial ontology containing some important terms and their relations. The terms from this ontology

---

[2] One might, however, represent in the ontology that 'tape' is one possible source (together with 'hard-disk', 'CD-ROM', or a mirror site) for performing a restore of data.

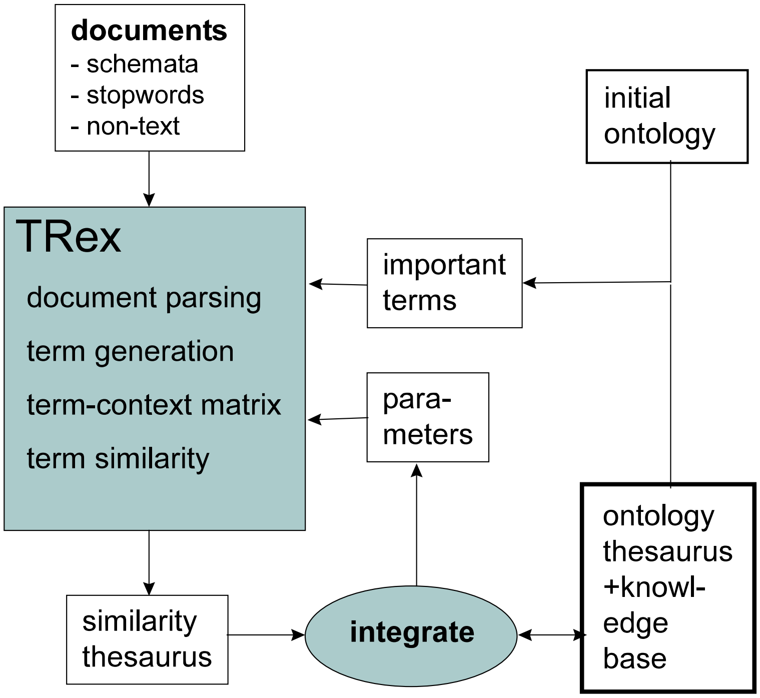


Figure 3.3: Using TRex for constructing and updating an integrated Ontology/Thesaurus

together with the documents can then be fed into TRex which produces a similarity thesaurus.

The knowledge engineer may then inspect the similarity thesaurus focusing on the terms from the initial ontology and the most frequent nontrivial terms of the document collection identified by TRex. For each similarity relation identified by TRex one of the following actions can be taken:

1. It is ignored, since it reflects a semantic relation already explicitly or implicitly known in the ontology.
2. It is classified semantically and added to the ontology.
3. It is formalized and added to the knowledge base.
4. It is attached to a concept in the ontology/thesaurus as a weighted "has-to-do-with" relation which is not specified further.
5. It is used to update the weight of an already known "has-to-do-with" relation.
6. It is considered to be spurious and ignored.

Obviously, actions 1 and 5 can be easily automated, if for determining implicitly known relations only a limited number of deduction steps are performed. Action 6 can at least be partially automated by remembering rejected term-associations so that they will not be reproduced in future runs of TRex.

Of course, such a manual inspection and classification can only be performed for a limited number of similarity-relations. Interesting terms will be chosen by the knowledge engineer based on the current ontology, observed term frequencies, and weights of the newly found term-similarities. This could be supported by an interactive tool to which the knowledge engineers specifies parameter thresholds, and which then suggests appropriate term-associations for a semantic classification.

For the rest of the associations in the similarity thesaurus, the knowledge engineer may then decide either to ignore them, or to add them automatically to the current ontology/thesaurus as weighted "has-to-do-with" relations which are not specified further.

Figure 3.3 also indicates that the knowledge engineer may repeat the generation of the similarity thesaurus with different parameters, if the results obtained in the previous run are deemed unsatisfactory. The described procedure can also be repeated with new documents so that an integrated Ontology/Thesaurus can be built and updated incrementally.

3.4.2 Example

Figure 3.4 gives an example of how information about a target term **t** in an integrated Ontology/Thesaurus is updated from a similarity thesaurus, which was automatically generated by TRex.

The left box in the figure lists the similar terms to **t** together with the similarity weights determined by TRex from a set of documents. The right box indicates, how the newly found similarity relations can be explained with known relations for term **t** in the Ontology/Thesaurus .

If it is already known that "t isa a", the similarity of 0.65 between t and a found by TRex is accounted for, and no further action need to be taken. Similarly, the found similarity of 0.47 between t and b can be explained by the ontology, since T has a part p which itself has a part b.

newly found similar terms for target term t
t 0.65 a
t 0.47 b
t 0.33 c
t 0.31 d
t 0.26 e
t 0.19 f
t 0.17 g
t 0.16 h

explanation by known relations for t from the ontology/thesaurus
t isa a
t haspart p, p haspart b
t antonym c
t 0.25 d
?
t isa u, f isa u
t haspart v, v synonym g
?

performed updates	
t 0.28 d,	suggest d
t 0.26 e,	suggest e
t 0.10 h	

Figure 3.4: Updating information about a target term **t** in an integrated Ontology/Thesaurus

As figure 3.4 shows, only the found similarity between t and e and between t and h cannot be accounted for by the current Ontology/Thesaurus . Furthermore, there is a discrepancy in the weight for the unspecified 'has-to-do-with' relation between t and d. The suggested updates for the given example are shown in the bottom box of the figure: the weight for the association between t and d is updated, and since it exceeds a given threshold (e.g. 0.25) it is suggested to the knowledge engineer for a semantic classification. The same would be the case for the newly found similarity between t and e. Since the found similarity between t and h is below this threshold, it is either ignored or added automatically to the Ontology/Thesaurus as an unspecific “has-to-do-with” relation.

3.5 First results from a practical experiment

In order to assess the practicality of the suggested procedure for constructing an integrated Ontology/Thesaurus, an experiment was performed within the knowledge management group of the DFKI Kaiserslautern.

We started to build an organizational memory, which should contain various kinds of information about the group, in particular the competencies and interests of individual members. The resulting OM is to be used primarily for determining whom to ask about a particular topic but other uses should not be precluded. In particular, the OM should also provide information about current and previous projects and the various topics which were investigated by the knowledge management group.

3.5.1 Procedure

In the first step, a rudimentary ontology was built by asking individual members what competencies they considered important.

In parallel, group members were asked to name electronically available documents which might contain relevant information about their competencies. This collection of documents was enhanced by scanning publication lists and homepages of group members, which were easily accessible from the group's intranet. The so obtained document collection included among others: interest profiles which had been compiled for an internal workshop, short project descriptions available on the intranet, regular project reports, project proposals sent to customers, diploma theses written by students, etc.

A collection of 67 documents comprising 3.5 MB of ASCII text could thus be composed without much effort. The document collection was restricted to German texts, since TRex currently only has a German morphology. Only minimal preprocessing was done to the documents, which was basically restricted to removing LaTex and HTML-tags. These documents were then fed into TRex together with a list of important terms extracted from the ontology, supplemented by lists of project names and person names.

3.5.2 Results

Whereas the results obtained from a first run of TRex (in which the same parameters were used as in the helpdesk application) were rather discouraging, they could be considerably improved by adjusting various generation parameters and by supplying more complete lists of stopwords and of important terms.

Due to the small number and the heterogeneity of documents, using document contexts for building the term-context matrix proved grossly inadequate. Text windows yielded much superior results, with a rather large window size of about 50-100 neigbouring terms being optimal. This can be explained by the fact that more contexts simply provide more data for estimating term similarities, which is particularly important when due to a small text corpus also rarely occurring terms have to be retained[3].

The poor quality of the first results was most conspicuous by the large number of trivial terms and phrases which appeared among the computed term associations. A domain-specific list of stopwords, which included a few dozen frequent but content-neutral terms such as 'paper', 'section', 'report', 'example', 'year', etc. was manually compiled with little effort[4].

A few adjustments to the term generator were made so that all strings containing numbers, which had been rather important in the original helpdesk application, were excluded. After re-running Trex, which only took about 30 minutes, the observed improvements were quite dramatical. This clearly demonstrates the importance of adjusting generation parameters and of providing appropriate domain-specific information together with a collection of documents.

An example of the information which was extracted for the term 'Unternehmensgedächtnis' (in English 'corporate memory') is shown in table 2, which is to be read in the same way as table 1, given in section 3.3.8. Since a text window of size 100 was used, given term frequencies, which are frequencies of terms in contexts, are to be divided by a factor of about 201 to obtain true term frequencies[5].

```
Term:'unternehmensgedaechtnis'
  Flags: BEGINUPPER MORELOWER HASUMLAUT NOUN COMPNOUN TECHTERM
  Words: 'unternehmensgedaechtnisses' 'unternehmensgedaechtnis'
  Components: 201 'unternehmens' 201 'gedaechtnis'
 Frequency: 1688 Contetxs: 592 IDF: 3.35652 Vectorlength: 139.564

 JS CS  JF CF  JI CI  LE CW
  1  1   1  1   1  2   1  1  'unternehmensgedaechtnis' F=1688 C=592
 12 15  12  9  16  .  11  8  'wissen+unternehmen' F=1661 C=476
  5  5   2  7   2  .   5  .  'informationsueberflutung' F=1037 C=487
  7  7   3  2   .  .   9  3  'wissensverarbeitung' F=3443 C=1150
  . 19   .  4   .  .   . 19  'mitarbeiter' F=19214 C=3175
  4  4   .  3   . 13   2  .  'informationstechnische' F=888 C=497
  3  3  13  .   .  .   3 10  'arbeitsablauf' F=1673 C=806
  8 10  20 10   .  .   7  .  'arbeitsprozess' F=1141 C=603
  2  2   .  .   .  .   4  2  'ug' F=3441 C=332
  .  .   . 14   .  .   .  .  'wissen' F=127401 C=8186
  .  .   .  5   .  .   .  .  'unternehmen' F=21236 C=2782
  . 13   .  8   .  .   .  5  'wissensmanagement' F=10645 C=1630
  6  6   .  .   .  .  12 12  'aktualisierung' F=2211 C=947
  .  .   7  .   3 11   . 16  'informationstechnologische+infrastruktur' F=1206 C=305
 16 12   . 11   .  4   .  .  'wissensstrukturierung' F=449 C=280
 11  9   .  .   .  .  20  4  'geschaeftsprozess' F=5242 C=1203
  .  .  10 13   8 14   .  .  'unternehmensweit' F=813 C=362
 19  .   5  .   5 19   .  .  'information+wissen+unternehmen' F=1206 C=349
```

Table 2: Term associations obtained for 'Unternehmensgedächtnis' from a small number of documents using as text window of width 100.

---

[3] In this experiment, all terms occurring at least twice were retained, whereas in the original helpdesk application where thousands of documents were available a minimal term frequency of 10 was used.

[4] Using observed term frequencies to construct a domain specific stopword list automatically, would als have excluded many frequent but interesting terms, which we wanted to include in the Ontology/Thesaurus.

[5] With a text window of 100, the maximum number of contexts for one occurrence of a term is 201. This number can, however, not be fully exhausted, if the entire document contains less than 201 valid context terms, which was the case for some short documents.

It is interesting to note that the abbreviation 'ug' of the target term, which constitutes a kind of synonym, is indeed found at the 9th place in the list of similar terms.
It is not surprising that the quality of term associations obtained from a small number of documents which were selected based on availability cannot match that from a large number of homogeneous documents, as was the case in the helpdesk application. Nevertheless, this experiment highlights that many interesting term relations can be extracted automatically even from a small number of hastily collected documents.
The quality of the obtained term associations also differs greatly for different types of target terms. Whereas for project names and for specific technical terms the quality of found terms associations is often surprisingly good, the same is not true for rather general terms and in particular for person names, where mostly trivial associations are provided. This is due to fact that specific terms and project names occur only in a limited number of thematically related contexts, whereas person names and general terms occur in a variety of contexts which share mostly trivial terms.
Even though the automatically constructed similarity thesaurus thus cannot be used directly for obtaining information about the competencies of a person, additional filters which exploit information from the ontology might help to overcome this problem. Such filters are currently being implemented so that queries such as "Which terms of category c (e.g. competency or person) are most closely related to target term t?" can be directly answered from the similarity thesaurus.

3.6 Conclusions and open questions

Reducing the costs for up-front knowledge engineering and continuous maintenance of an organizational memory requires the exploitation of easily available information sources and the utilization of automatic knowledge acquisition tools. Furthermore, only a limited amount of the information to be managed with the OM can be affordably formalized so that an integration of formal and informal knowledge together with appropriate meta-level descriptions is required.
An integrated Ontology/Thesaurus which contains both formally defined concepts and informal term associations constitutes an important prerequisite for supporting knowledge structuring and retrieval in an OM. Such an Ontology/Thesaurus can be constructed and maintained with the help of an automatic thesaurus generation tool, which extracts relevant terms and similarity relations from a given corpus of documents.
Since currently only similarity relations can be extracted automatically, a manual semantic classification of identified similarity relations is required for constructing an Ontology/Thesaurus. The goal of reducing the cost for up-front knowledge engineering can thus only be achieved partially, since domain experts and knowledge engineers still have to be consulted.
The proposed approach for constructing an integrated Ontology/Thesaurus with the help of the thesaurus generation tool TRex is nevertheless very useful, since a large number of interesting terms and relations can be automatically extracted from ordinary documents which are routinely created in industrial practice. TRex thus taps an easily available and plentiful information source and discovers similarity relations which can be exploited both for initial knowledge acquisition and continuous knowledge maintenance. It remains to be explored, whether TRex is not even more useful for domain-knowledge acquisition and knowledge discovery in text bases than for ontology construction.
It also remains to be explored, which combinations of term contexts, weighting schemes and similarity measures yield optimal results, and to what extent the semantic classification of identified term relations can then be supported by automatically applied

heuristics. An integration of TRex with other tools for knowledge acquisition from text, such as KARAT [Tschaitschian *et al.*, 1997], might yield additional benefits.

A final open question concerns the usefulness of the weighted 'has-to-do-with' relations in the integrated Ontology/Thesaurus as compared to the formalized ontological relations. The observed practical benefits of having formalized vs. informal knowledge will ultimately determine how much knowledge engineering effort will be invested in building and updating an organizational memory.

Bibliography

[Abecker *et al.*, 1997a] A. Abecker, A. Bernardi, K. Hinkelmann, O. Kühn, and M. Sintek. Towards a well-founded technology for organizational memories. In *[Gaines* et al.*, 1997]*, 1997.

[Abecker *et al.*, 1997b] A. Abecker, S. Decker, K. Hinkelmann, and U. Reimer. Proceedings: Knowledge-Based Systems for Knowledge Management in Enterprises—Workshop held at the 21st Annual German Conference on AI (KI-97). Document D–97–03, DFKI GmbH, 1997.

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

[Allen, 1984] J. Allen. Towards a general theory of action and time. *Artificial Intelligence*, (23):123–154, 1984.

[Ambite and Knoblock, 1997] J. L. Ambite and C. A. Knoblock. Agents for information gathering. *IEEE Expert – Intelligent Systems and Their Applications*, 12(5), 1997.

[Baudin *et al.*, 1992] C. Baudin, J. Gevins, V. Baya, and A. Mabogunje. DEDAL: using domain concepts to index engineering design information. In *Proc. of the Meeting of the Cognitive Science Society, Bloomington, Indiana*, 1992.

[Baudin *et al.*, 1995] C. Baudin, S. Kedar, and B. Pell. Increasing levels of assistance in refinement of knowledge-based retrieval systems. In G. Tecuci and Y. Kodratoff, editors, *Machine Learning and Knowledge Acquisition – Integrated Approaches.* Academic Press, 1995.

[Baumann *et al.*, 1997] S. Baumann, M. Malburg, H. Meyer auf'm Hofe, and C. Wenzel. From paper to a corporate memory — a first step. In *[Abecker* et al.*, 1997b]*, 1997.

[Bernardi, 1998] A. Bernardi. Modeling knowledge-intensive tasks as a basis for active user support by context-oriented information retrieval. KnowMore Internal Report. DFKI Kaiserslautern. Knowledge Management Group. Also Part I of thsi Document, February 1998.

[Berry *et al.*, 1996] M. Berry, T. Do, G. O'Brien, V. Krishna, and S. Varadhan. Svdpackc (version 1.0) users's guide, revised march 1996. CS-93-194, Department of Computer Science, University of Tennessee, 1996.

[Bläsius *et al.*, 1997] K.-H. Bläsius, B. Grawemeyer, I. John, and N. Kuhn. Knowledge-based interpretation of business letters. In *[Abecker* et al.*, 1997b]*, 1997.

[Celentano *et al.*, 1995] A. Celentano, M. Fugini, and S. Pozzi. Knowledge-based document retrieval in office environments: The Kabiria system. *ACM Transactions on Information Systems*, 13(3), 1995.

[Chen *et al.*, 1993] H. Chen, B. Schatz, J. Martinez, and T. Ng. Automatic thesaurus generation for flybase. *Proceedings of SIGIR*, 1993.

[Chen *et al.*, 1995] H. Chen, T. Yim, D. Fye, and B. Schatz. Automatic thesaurus generation for an electronic community system. *Journal of the American Society for Information Science*, 46(3):175–193, 1995.

[Christophides *et al.*, 1994] V. Christophides, S. Abiteboul, and S. Cluet. Retrieval and database systems. In *Proceedings of the 1994 ACM-SIGMOD. Int'l Conference on Management of Data.* Springer Verlag, 1994.

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

[Daniel *et al.*, 1997] M. Daniel, S. Decker, A. Domanetzki, E. Heimbrodt-Habermann, F. Höhn, A. Hoffmann, H. Röstel, R. Smit, R. Studer, and R. Wegner. Erbus - towards a knowledge management system for designers. In *[Abecker* et al.*, 1997b]*, 1997.

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

[Decker *et al.*, 1997] S. Decker, M. Daniel, M. Erdmann, and R. Studer. An Enterprise Reference Scheme for Integrating Model Based Knowledge Engineering and Enterprise Modelling. In *10th European Workshop on Knowledge Acquisition, Modelling, and Management*, Sant Feliu de Guixols, Catalonia, Spain, October 1997. Springer Verlag.

[Dengel and Hinkelmann, 1996] A. Dengel and K. Hinkelmann. The specialist board – a technology workbench for document analysis and understanding. In *Integrated Design and Process Technology, Proceedings of the Second World Congress*, pages 36–47, Austin, Texas, 1996. Society for Desing and Process Science.

[Derweester *et al.*, 1990] S. Derweester, S. Dumais, G. Furnas, T. Landauer, and R.Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6), 1990.

[DIN Deutsches Institut für Normung, 1987] DIN Deutsches Institut für Normung. Erstellung und weiterentwicklung von thesauri: Einsprachige thesauri. Beuth Verlag GmbH, Berlin, November 1987. Zusammnehang mit ISO 2788 - 1986.

[Dumais, 1995] S. Dumais. Using lsi for information filtering: Trec-3 experiments. In D. harman, editor, *The third Text Retrieval Conference.* NIST special publication, 1995.

[Fensel *et al.*, 1998] D. Fensel, S. Decker, M. Erdmann, and R. Studer. Ontobroker: How to make the WWW intelligent. Technical report, Institute AIFB, University of Karlsruhe, Germany, April 1998. Also in: Proc. of the 11th Banff Knowledge Acquisition for Knowledge-Based System Workshop (KAW98), Banff, Canada.

[Fernandes *et al.*, 1992] A. A. Fernandes, N. W. Paton, M. H. Williams, and A. Bowles. Approaches to deductive object-oriented databases. *Information and Software Technology*, 34(12), December 1992.

[Frei *et al.*, 1996] H.-P. Frei, D. Harman, P. Schäuble, and R. Wilkinson, editors. *Proc. of the 19th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval.* August 1996.

[Fuhr and Rölleke, 1996] N. Fuhr and T. Rölleke. A probabilistic nf2 relational algebra for integrated information retrieval and database systems. In M. M. Tanik, F. B. Bastani, D. Gibson, and P. J. Fielding, editors, *Integrated Design and Process Technology, Proceedings of the Second World Conference*, pages 17–30, Austin, Texas, USA, December 1996. Society for Design and Process Science.

[Fuhr, 1994] N. Fuhr. Logical and conceptual models for the integration of information retrieval and database systems. In *East/West Database Workshop, Klagenfurt.* Springer Verlag, 1994.

[Fuhr, 1995] N. Fuhr. Modelling hypermedia retrieval in datalog. In *Proceedings HIM'95.* Universitätsverlag Konstanz, April 1995.

[Gaines *et al.*, 1997] B. R. Gaines, M. A. Musen, R. Uthurusamy, R. Dieng, G. van Heijst, D. Lukose, and F. Maurer, editors. *AAAI Spring Symposium Artificial Intelligence in Knowledge Management.* AAAI, March 1997.

[Gordon and Domeshek, 1995] A. Gordon and E. Domeshek. Conceptual indexing for video retrieval. In M. Maybury, editor, *Intelligent Multimedia Information Retrieval, Working notes of the IJCAI-95 Workshop, Montreal, Quebec*, August 1995.

[Grefenstette, 1994] G. Grefenstette. *Explorations in Automatic Thesaurus Discovery.* Kluwer Academic Publishers, Boston, 1994.

[Grefenstette, 1995] G. Grefenstette. *Automatic Thesaurus Generation from Raw Text using Knowledge-Poor Techniques.* Proceedings of SIGIR, 1995.

[Gruber, 1991] T. R. Gruber. Ontolingua: A mechanism to support portable ontologies. Technical Report KSL 91-66, Stanford University, 1991.

[Gruber, 1994] T. R. Gruber. Towards principles for the design of ontologies used for knowledge sharing. *International Journal of Human-Cumputer Studies*, 43(3/4):907–928, 1994.

[Guarino, 1996] N. Guarino. Understanding, building, and using ontologies. *International Journal of Human-Computer Studies/Knowledge Acquisition*, 1996. Special Issue on Using Explicit Ontologies.

[Hendler *et al.*, 1996] J. Hendler, K. Stoffel, and M. Taylor. Advances in high performance knowledge representation. Technical Report CS-TR-3672, University of Maryland Institute for Advanced Computer Studies Dept. of Computer Science, Univ. of Maryland, July 1996.

[Hendler *et al.*, 1997a] J. Hendler, K. Stoffel, M. Taylor, D. Rager, and B. Kettler. Parka-db: A scalable knowledge representation system. URL `http://www.cs.umd.edu/projects/plus/Parka/parka-db.html`, 1997.

[Hendler *et al.*, 1997b] J. Hendler, K. Stoffel, M. Taylor, D. Rager, and B. Kettler. Representing knowledge for multimedia retrieval in datalog. Submitted for publication, 1997.

[Hofer-Alfeis and Klabunde, 1996] J. Hofer-Alfeis and S. Klabunde. Approaches to managing the lessons learned cycle. In M. Wolf and U. Reimer, editors, *PAKM'96, Proc. First Int. Conference on Practical Aspects of Knowledge Management, Basel, Switzerland*, October 1996.

[JAVA, 1998] JAVA Homepage at SUN. `http://java.sun.com/`, 1998.

[JDBC, 1998] JDBC Homepage at SUN. `http://java.sun.com/products/jdbc/`, 1998.

[Junker and Abecker, 1996] M. Junker and A. Abecker. Einsatz Maschineller Lernverfahren für die Dokumentklassifikation. In W. Dilger, M. Schlosser, J. Zeidler, and A. Ittner, editors, *FGML-96: Fachgruppentreffen Maschinelles Lernen.* Chemnitzer Informatik-Berichte CSR-96-06, TU Chemnitz-Zwickau, August 1996.

[Junker, 1998] M. Junker. Symbolic learning of rules for text classification. Working Draft. DFKI Kaiserslautern. Information Management and Document Analysis Dept., 1998.

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

[Kindermann and Hoppe *et al.*, 1996] C. Kindermann and T. Hoppe *et al.* The MIHMA demonstrator application: bmt line. Technical report, Non Standard Logics S.A., Paris and TU Berlin, April 1996.

[Kirk *et al.*, 1995] T. Kirk, A. Y. Levy, Y. Sagiv, and D. Srivastava. The Information Manifold. In *AAAI Spring Symposium on Information Gathering in Distributed Heterogeneous Environments, Stanford, CA*, March 1995.

[Kirk, 1996] T. Kirk. Information Manifold: knowledge based access to information on the World Wide Web . In *WWW5 Workshop on Artificial Intelligence-based tools to help W3 users at the 5th Int'l WWW Conference, Paris, France*, 1996. URL `http://www.info.unicaen.fr/~serge/3wia/workshop/`.

[Knorz, 1996] G. Knorz. *Indexieren, Klassieren, Extrahieren. Vierte Ausgabe.* Saur Verlag, 1996.

[Kühn and Abecker, 1997] O. Kühn and A. Abecker. Corporate memories for knowledge management in industrial practice: Prospects and challenges. *Journal of Universal Computer Science*, 3(8):929–954, 1997. Special Issue on Information Technology for Knowledge Management .

[Lambrix and Padgham, 1995] P. Lambrix and L. Padgham. Analysis of part-of reasoning in description logics for use in a document management application. In *Proc. of the Int'l Symposium on Artificial Intelligence, Monterrey, Mexico*, 1995. A shorter version is in the Proc. of the Int'l Workshop on Description Logics (DL-95).

[Lambrix and Padgham, 1996] P. Lambrix and L. Padgham. A knowledge base for structured documents. In *Proceedings of the First Australian Document Computing Symposium*, March 1996.

[Lambrix and Padgham, 1997] P. Lambrix and L. Padgham. A description logic model for querying knowledge bases for structured documents. In *Proceedings of the International Symposium on Methodologies for Intelligent Systems - ISMIS-97*, number 1325 in LNAI. Springer Verlag, 1997.

[Lambrix *et al.*, 1997] P. Lambrix, N. Shahmehri, and N. Wahllöf. Dwebic: An intelligent search engine based on default description logics. In *Proceedings of the International Workshop on Description Logics - DL-97, Gif sur Yvette, France*, 1997.

[Lenat *et al.*, 1988] D. Lenat, R. Guha, and D. Wallace. The CycL Representation Language. Technical Report ACA-AI-302-88, MCC, 1988.

[Liu, 1998a] M. Liu. An overview of rule-based object language. *Journal of Intelligent Information Systems*, 10(1), 1998.

[Liu, 1998b] M. Liu. The ROL deductive object-oriented database system. *Informatica*, 1998. To appear.

[Luke *et al.*, 1997] S. Luke, L. Spector, D. Rager, and J. Hendler. Ontology-based web agents. In *Proceedings of First International Conference on Autonomous Agents, AA-97*, 1997.

[Meghini and Straccia, 1996] C. Meghini and U. Straccia. A relevance terminological logic for information retrieval. In *[Frei* et al.*, 1996]*, 1996.

[Meghini *et al.*, 1991] C. Meghini, F. Rabitti, and C. Thanos. Conceptual modeling of multimedia documents. *IEEE Computer*, October 1991.

[Milstead, 1990] J. Milstead. Thesaurus software packages. In *Proceedings of the 53rd Annual Meeting of the American Society for Information Science*, 1990.

[Neches *et al.*, 1991] R. Neches, R. Fikes, T. Finin, T. Gruber, R. Patil, T. Senator, and W. R. Swartout. Enabling technology for knowledge sharing. *AI Magazine*, Fall:35–36, 1991.

[Neven and van den Bussche, 1997] F. Neven and J. van den Bussche. On implementing structured document facilities on top of a dood. In *Proceedings of the 5th International Conference on Deductive and Object-Oriented Databases, Montreux*, December 1997.

[Nonaka and Takeuchi, 1995] I. Nonaka and H. Takeuchi. *The Knowledge-Creating Company.* Oxford University Press, 1995.

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

[Ostermayer *et al.*, 1996] R. Ostermayer, E. Meis, A. Bernaras, and I. Laresgoiti. Guidelines on domain ontology building. Technical Report Esprit Project 8145, Deliverable DO1c.2, KACTUS Consortium, 1996.

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

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

[Rölleke and Fuhr, 1996] T. Rölleke and N. Fuhr. Retrieval of complex objects using a four-valued logic. In *[Frei* et al.*, 1996]*, 1996.

[Rölleke and Fuhr, 1997] T. Rölleke and N. Fuhr. HySpirit—a Flexible System for Investigating Probabilistic Reasoning in Multimedia Information Retrieval. Technical report, University of Dortmund, Germany, 1997.

[Salton and McGill, 1983] G. Salton and M. McGill. *Introduction to Modern Information Retrieval.* McGraw-Hill, New York, 1983.

[Salton, 1972] G. Salton. Automatic thesaurus construction for information retrieval. *Information Processing*, 123:71–115, 1972.

[Salton, 1989] G. Salton. *Automatic Text Processing.* Addison-Wesley, Reading, MA, 1989.

[Schmiedel and Volle, 1996] A. Schmiedel and P. Volle. Using structured topics for managing generalized bookmarks. In *WWW5 Workshop on Artificial Intelligence-based tools to help W3 users at the 5th Int'l WWW Conference, Paris, France*, 1996. URL `http://www.info.unicaen.fr/~serge/3wia/workshop/`.

[Schreiber and Birmingham, 1996] A. T. Schreiber and W. Birmingham. The sisyphus vt initiative. *International Journal of Human-Computer Studies*, 44(3/4):275–280, 1996.

[Schreiber *et al.*, 1993] G. Schreiber, B. Wielinga, and J. Breuker. *KADS: A Principled Approach to Knowledge-Based System development.* Academic Press, London, 1993.

[Schreiber *et al.*, 1996] G. Schreiber, W. Jansweier, and B. Wielinga. Framework and formalism for expressing ontologies. ESPRIT Project 8145 KACTUS deliverable D01B2, University of Amsterdam, October 1996.

[Sebastiani, 1994] F. Sebastiani. A probabilistic terminological logic for modelling information retrieval. In *Proceedings of the 17th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 122–130. Springer Verlag, Heidelberg, 1994.

[Shum, 1997] S. B. Shum. Negotiating the construction and reconstruction of organisational memories. *Journal of Universal Computer Science*, 3(8):899–928, 1997. Special Issue on Information Technology for Knowledge Management .

[Sintek and Abecker, 1998] M. Sintek and A. Abecker. A generic knowledge-description formalism for multi-dimensional structuring of corporate knowledge. KnowMore Internal Report. DFKI Kaiserslautern. Knowledge Management Group. Also Part II of this Document, February 1998.

[Speel *et al.*, 1995] P.-H. Speel, F. van Raalte, P. E. van der Vet, and N. J. Mars. Scalability of the performance of knowledge representation systems. In N. Mars, editor, *Towards very large knowledge bases. Knowledge building and knowledge sharing.* IOS Press, Amsterdam, 1995.

[Speel, 1995] P.-H. Speel. *Selecting knowledge representation systems.* PhD thesis, University of Twente, Enschede, the Netherlands, 1995.

[Steier *et al.*, 1995] D. Steier, S. B. Huffman, and W. C. Hamscher. Meta-information for knowledge navigation and retrieval: What's in there, 1995.

[Sveiby, 1996] K. E. Sveiby. Tacit knowledge – an introduction. URL `http://www.sveiby.com.au/Polanyi.html`, 1996.

[The Workflow Management Coalition, 1996] The Workflow Management Coalition. Workflow management coalition terminology & glossary. Document number WFMC-TC-10011, June 1996.

[Tschaitschian *et al.*, 1997] B. Tschaitschian, A. Abecker, and F. Schmalhofer. Information Tuning with KARAT: Capitalizing on existing documents. In *10th European Workshop on Knowledge Acquisition, Modelling, and Management*, Sant Feliu de Guixols, Catalonia, Spain, October 1997. Springer Verlag.

[van Bakel *et al.*, 1996] B. van Bakel, R. Boon, N. Mars, J. Nijhuis, E. Oltmans, and P. van der Vet. Condorcet annual report. Technical Report UT-KBS-96-12, Knowledge-Based Systems Group, University of Twente, September 1996.

[van der Vet and Mars, 1996] P. E. van der Vet and N. J. Mars. Coordination recovered. In *Informatiewetenschap 1996*. Delft, 1996.

[van der Vet *et al.*, 1994] P. van der Vet, P. Speel, and N. Mars. The Plinius ontology of ceramic materials. In *Workshop notes ECAI94 workshop on Comparison of implemented ontologies, Amsterdam, The Netherlands*, pages 187–205, August 1994.

[van Heijst *et al.*, 1996a] G. van Heijst, A. Schreiber, and B. Wielinga. Using explicit ontologies in kbs development. *International Journal of Human -Computer Studies/Knowledge Acquisition*, 1996. Special Issue on Using Explicit Ontologies.

[van Heijst *et al.*, 1996b] G. van Heijst, R. van der Spek, and E. Kruizinga. Organizing corporate memories. In R. Dieng and J. Vanwelkenhuysen, editors, *KAW'96, Special Track on Corporate Memory and Enterprise Modeling*, November 1996.

[van Rijsbergen, 1989] C. van Rijsbergen. Towards an information logic. In *Proc. of the 12th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, 1989.

[Viegener, 1997] J. Viegener. *Inkrementelle, domänenunabhängige Thesauruserstellung in dokumentbasierten Informationssystemen durch Kombination von Konstruktionsverfahren.* DISDBIS, infix, 1997.

[Vila *et al.*, 1996] M. Vila, J. Cubero, J. Medina, and O. Pons. A conceptual approach for dealing with imprecision and uncertainty in object-based data models. *Int. Journal of Intelligent Systems*, 11, 1996.

[Wahllöf, 1996] N. Wahllöf. A default extension to description logics and its applications. Lic. thesis 591, Dept. of Computer and Information Science, Linköping University, 1996.

[Welty, 1994] C. Welty. Knowledge representation for intelligent information retrieval. In *Proceedings of the CAIA-94 Workshop on Intelligent Access to Digital Libraries*, March 1994.

[Welty, 1996] C. Welty. Intelligent assistance for navigating the web. In *Proceedings of The 1996 Florida AI Research Symposium*, 1996.

[Welty, 1998] C. Welty. The ontological nature of subject taxonomies. 1998. 1998 Int'l Conference on Formal Ontology in Information Systems (FOIS'98).

[Wielinga *et al.*, 1992] B. Wielinga, W. Van de Velde, G. Schreiber, and H. Akkermans. Towards a unification of knowledge modelling approaches. Proceedings of the 7th Banff Knowledge Acquisition for Knowledge -Based Systems Workshop 1992, Banff, Canada, 1992.

[Zarri and Azzam, 1997] G. P. Zarri and S. Azzam. Building Up and Making Use of Corporate Knowledge Repositories. In *10th European Workshop on Knowledge Acquisition, Modelling, and Management*, Sant Feliu de Guixols, Catalonia, Spain, October 1997. Springer Verlag.

Techniques for Organizational Memory Information Systems

Andreas Abecker, Ansgar Bernardi, Knut Hinkelmann,
Otto Kühn, Michael Sintek

D-98-0
Documen