- Cover page -

USING ONTOLOGIES FOR ADVANCED INFORMATION ACCESS

Michael Sintek, Bidjan Tschaitschian,
Andreas Abecker, Ansgar Bernardi
German Research Center for Artificial Intelligence (DFKI GmbH)
Knowledge Management Group
P.O. Box 2080, D-67608 Kaiserslautern, Germany
Tel ++49 631 205 3582 Fax ++49 631 205 3210
{sintek|tschaits|aabecker|bernardi}@dfki.uni-kl.de

Heinz-Jürgen Müller
T-Nova Deutsche Telekom Innovationsgesellschaft mbH
Technologiezentrum Darmstadt FE14k, D-64307 Darmstadt, Germany
Tel ++49 6151 83 5682 Fax ++49 6151 83 4124
Heinz-Juergen.Mueller@telekom.de

Contact Author: Ansgar Bernardi

Keywords: enterprise knowledge management; coporate memory; ontologies; information access and retrieval; information integration; applications

Abstract: Accessing heterogeneous information sources poses interesting problems not only with respect to technical integration, but also with regard to conceptually sound and ergonomically adequate user interaction. Suitable ontologies offer the means to structure the domain on various levels and with appropriately choosen terms, concepts, and interrelations. Using a recent application study as an example, we demonstrate how such ontologies can support the interaction with the user on adequate but varying levels of detail. We illustrate the contribution of the ontology to the automatic configuration of a graphical user interface and argue that the consideration of ergonomical questions will lead to a transgression from the descriptive ontology towards an integrated navigational epistomology. The results facilitate the easy access to complex and weakly-structured information sources. This is applied to information retrieval and experience reuse in the software development department of a large telecommunications company.

USING ONTOLOGIES FOR ADVANCED INFORMATION ACCESS

Michael Sintek, Bidjan Tschaitschian,
Andreas Abecker, Ansgar Bernardi
German Research Center for Artificial Intelligence (DFKI GmbH)
Knowledge Management Group
P.O. Box 2080, D-67608 Kaiserslautern, Germany
Tel ++49 631 205 3582 Fax ++49 631 205 3210
{sintek|tschaits|aabecker|bernardi}@dfki.uni-kl.de

Heinz-Jürgen Müller
T-Nova Deutsche Telekom Innovationsgesellschaft mbH
Technologiezentrum Darmstadt FE14k, D-64307 Darmstadt, Germany
Tel ++49 6151 83 5682 Fax ++49 6151 83 4124
Heinz-Juergen.Mueller@telekom.de

Abstract: Accessing heterogeneous information sources poses interesting problems not only with respect to technical integration, but also with regard to conceptually sound and ergonomically adequate user interaction. Suitable ontologies offer the means to structure the domain on various levels and with appropriately choosen terms, concepts, and interrelations. Using a recent application study as an example, we demonstrate how such ontologies can support the interaction with the user on adequate but varying levels of detail. We illustrate the contribution of the ontology to the automatic configuration of a graphical user interface and argue that the consideration of ergonomical questions will lead to a transgression from the descriptive ontology towards an integrated navigational epistomology. The results facilitate the easy access to complex and weakly-structured information sources. This is applied to information retrieval and experience reuse in the software development department of a large telecommunications company.

1. Introduction

Accessing heterogeneous information sources poses interesting problems not only with respect to technical integration. Wrapper / mediator architectures (Wiederhold and Genesereth, 1997), intelligent information agents (Knoblock and Ambite, 1997), and ontology-based Internet retrieval (Decker *et al.*, 1999; Luke *et al.*, 1997) lifted the technical / "syntactic" level of integration to the conceptual / "semantic" level taking into account that important information search and delivery problems are often situated in a technically distributed, and conceptually heterogeneously modeled world of information sources.
The AI idea of an ontology as the "common denominator" for, or a "meta-level bridge" between several dispersed information systems has been adopted in the context of federated databases and also been transferred to the idea of Internet information retrieval (Duschka *et al.*, 1998; Fensel *et al.*, 1999). Ontologies as the explicit account of the conceptualization

underlying a specific information system allow to find appropriate information sources dealing with a given topic, to define mappings between different conceptualizations, and thus merging of information from different sources.

In many scientific "toy examples" or academic applications (model a scientific community, or build a digital library for scientific literature), there seem to be not very much disagreements on what ontologies for knowledge retrieval are, where they come from, and how they can (and should) be used. Rather it is presupposed that there exist ontologies, formal models of meta and background knowledge to enable network retrieval and to support precise-content retrieval in difficult query situations.

However, we have the impression that there are not yet very much fielded applications employing these ideas in industrial environments where ontological engineering has to face cost-benefit analyses and "normal users" have to understand the interfaces provided and to maintain the underlying models. Further, we seldom discover a study offering other, more innovative or more sophisticated uses of ontological background knowledge than just exploiting a concept taxonomy for query reformulation (query extension, refinement, or relaxation, usually only the latter) (McGuiness, 1998). We do not know an empirically more broad study about user acceptance and users' estimations about the usefulness of ontology-based retrieval mechanisms. We neither detected a paper discussing whether the use of ontologies for supporting information access and retrieval imposes new requirements on ontology content, representation, building, etc.

It is always assumed that the "classical" concept of an ontology as a stable, well-understood, as formal as possible representation (usually in a given standard representation language like Ontolingua, description logic, or conceptual graphs) of what is agreed upon by the user community in a well-organized, collaborative process, and reusable for a whole bunch of applications, is sufficient and good also for building "knowledge portals" for users accessing a complex information space in a company, or even exploring the Internet.

In this paper, we do not want to question all the above "classical" ontology issues, but we would like to consciously think about such questions as:

1. *Where can ontologies for information access in industrial environments practically come from and what information should they contain?*
2. *Which kinds of ontological information is adequate to be handled by a "normal user" aiming at a comfortable and easy to understand system interface?*
3. *How can this ontology content be exploited by retrieval and access services to make optimal use for advanced retrieval results and better user adequacy? And how can knowledge portals be designed presenting these services with conceptually sound and ergonomically adequate modes of user interaction?*
4. *Last, are there specific software architecture issues to be considered when building systems along the ideas to be outlined following the above questions?*

We do not have answers to all these questions. But we faced most of them in an industrial case study to be described in the following Section 2. In Subsection 2.1, we describe the application context, namely information retrieval and experience reuse in the software development department of a large telecommunications company. We sketch the basic functionality of a first demonstrator prototype which has recently been deployed and will be further extended in the near future. Subsection 2.2 gives an idea of the content and potential usage of ontologies in this system, the implementation ideas and some more technical details of which are discussed in Subsections 2.3 through 2.5. Since this system prototype has been

designed for a very specific application and needs to be further developed, we make more general considerations about underlying principles and related work in Section 3. Finally, we shortly summarize with Section 4.

2. The Case Study: Software Configuration

2.1. Application Scenario & System Services

In a software development department of a large company there are already three relational databases, built up in different groups of the company, which describe different aspects of previous system development projects and installed solutions from different points of view. Currently, one important goal is to unify this distributed expertise in a central database and provide a user-convenient access which stipulates use of knowledge already gathered in the company and reuse of solutions already developed.

In this paper we can consider all three databases as one large information system although merging and fusioning of database schemata and information stored is not a trivial problem in the general case. Nevertheless, our focus in this paper is the question how to design a comfortable user interface which supports easy and precise finding and retrieval of relevant information. Basically, we talk about a comfortable way for querying one large database table, where each old problem case (i.e. one previously built system, or project performed) is represented by one row specifying the attribute values for a huge number of characterizing attributes.

From the user point of view, when a new system has to be designed and layouted, at several decision points (concerning, e.g., questions like which database management system to use, which hardware platform, etc.) the new to-be-built assistant system should be consulted in order to find out whether a similar system configuration already occurred in the past such that specific decisions, interesting experiences, or whole parts of the system design could be transferred to the actual situation.

Usually, when starting to design a system solution, there are only very few attributes known to be filled in. What the engineer wants when laying out the system to be built is an easy-to-use information access which helps in incrementally specifying the relevant system parts and parameters step by step, identifying interesting information in a given partial configuration situation, showing up similar old designs in similar or partly identical situations, and showing the effects of changes in the current decision tableau.

To achieve such a functionality, a demonstrator prototype has been deployed the user interface of which is shown in Figure 1.

In the upper left panel, we see a folder-like representation of a concept tree visualizing the structure of characteristics and system parts describing a given IT system specification. The user broswes through this structure identifying the properties or classes of properties he or she is currently interested in. Further unfolding the folder hierarchy leads to more and more specific system specification criteria until it ends with the level of concrete database attributes representing specific characteristics of the stored solutions, like the kind of DBMS employed. If the user knows more or less exactly which system parameters he or she wants to investigate, it is also possible to input their names or similar terms into the text input field at the top of the user input mask. Then the system unfolds the attribute hierarchy as deep as necessary and directly jumps to the respective concrete attribute or attribute class. Selecting positively (or negatively) a given concrete attribute by a mouse-click means that the user is interested in stored solutions which possess an (or possess no, in the case of a negative selection) attribute value for this specific attribute; which just means that this attribute is interesting. If a selection

is made at a more abstract level in the attribute hierarchy, this means that at least one of the characteristics below is of interest (or, all characteristics below are of no interest). Coming to the level of concrete DB attributes, the upper right panel shows for a selected attribute the range of possible attribute values. The user can also specify here at the value level that the solutions to be found should have a specific attribute value, or should not have a specific attribute value. Since also the attribute value level is structured in an object-oriented manner, one mouse-click can again represent a more complex logical selection condition.
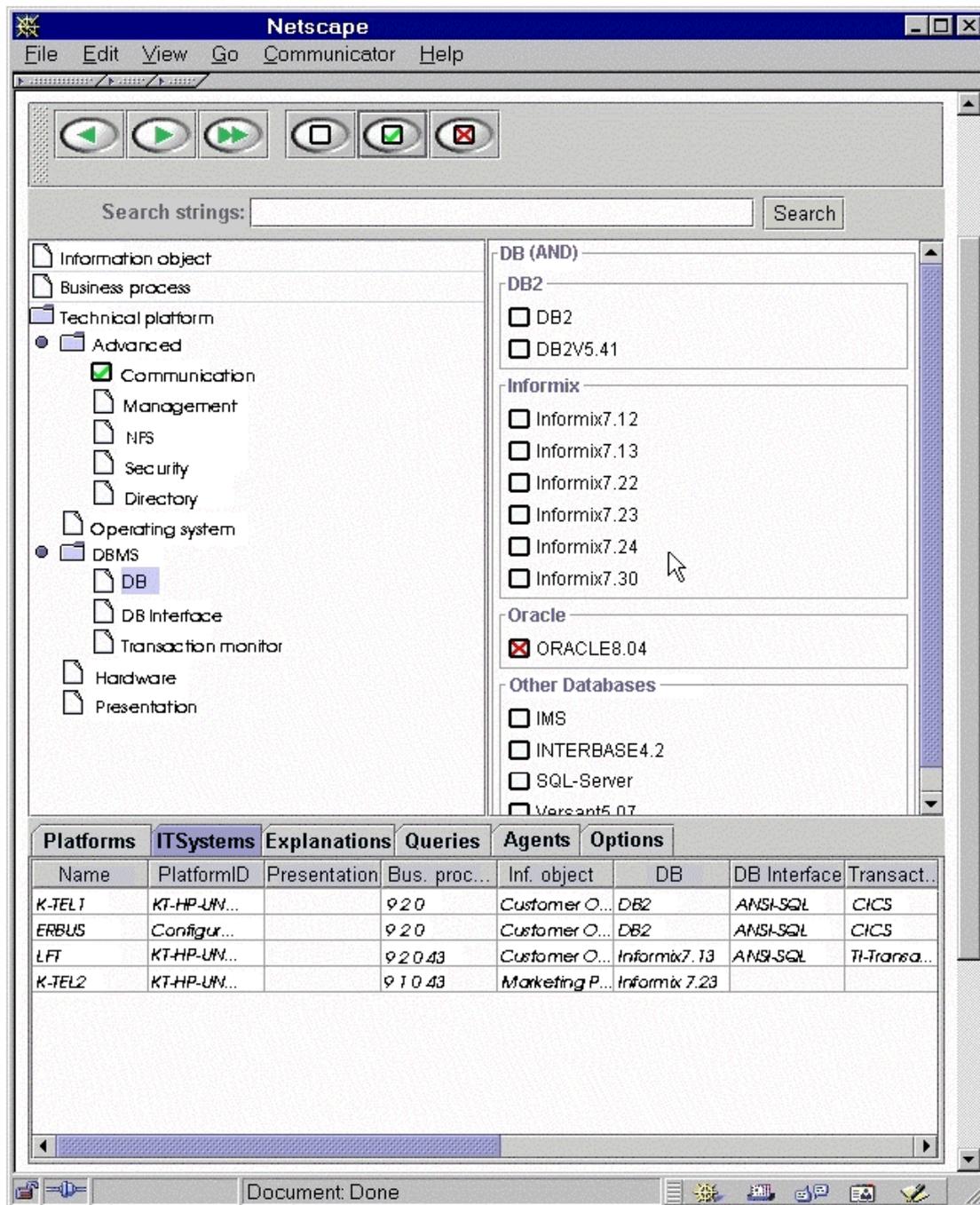

Figure 1: The User Interface.

While the user is such specifying more or less concretely what he or she is interested in (or not), the lower part of the GUI always shows the current result set in a tabular form, i.e. the set of all DB entries fulfilling the actually valid search conditions (system specifications). This result panel is incrementally updated, each selection or deselection of the user immediately causes changes in the result set which gives a comfortable combined browse-and-search interface for playing around with several system configurations to find out what effects certain design decisions have on the retrieval behaviour on the stored solutions DB. If there are too many possible answers, the user can directly try to enforce search constraints and immediately sees the effects of his/her actions on the result set. If there is no solution found, the user can try to weaken search conditions and try out which combinations of selections cause an empty answer set and which query relaxation steps succeed in finding similar cases.

Since we assume that in the large information space to be managed by the users and with the manifold possibilities of automatic query relaxation or similarity assessment of situations, system behaviour will not be very easy to understand, in the lower panel the system can also be asked for explanations of retrieval results if possible, or for a trace of the agent communication taking place when the several system parts collaborate for retrieval.

In the following we will try to show which intelligent retrieval support can be given, and how this is realized on the basis of ontologies.

2.2. Use of Ontologies in the Case Study

In our field study the following prototypical application scenarios for advanced modeling and inference services on the basis of a more sophisticated domain ontology were identified.

2.2.1. Abstract from Database Attributes and Values

Since the database schema, as it is, is too flat and too broad for a user convenient access, we insert new, abstract concepts into the schema. Essentially the goal is to bridge the gap between existing data structures and the information needs of employees expressed in user terms in order to provide an easier and more intuitive information access.

Example: The project leader of a new IS project wants to assess alternative realization platforms. To this end, she has to take into account the existing IS environment. There are already ideas about the database management system of the new project. She wants to specify these ideas about DBMS in order to find existing cases in the IS planning tool database, and is thus searching for the appropriate DB attributes dealing with such factors.

In the IS planning tool, the database attributes *database, database interface*, and *transaction monitor* are already in use. Attribute values are concrete products like, e.g., *Informix 7.13, Informix 7.24, ORACLE 8.04, DB2v5.41, Sybase.*

In order to support the user navigating through the database attributes for specifying database details, we build a *navigation ontology* which comprises parts of the original database schema plus additional meaningful abstractions above them. Using the term navigation ontology implies, for instance, that we take into account ergonomic criteria like depth and width of a hierarchy which can be dealt with by a human user, or integration of end user terminology. So, in contrast to the very basic meaning of the word ontology, our aim here is not so much to describe things as the are (the ontological dimension), but rather in a way which can be

understood and used (rather sort of epistemological dimensions, or so to speak, an application ontology for the task of user navigation and access).

Figure 2 shows a part of the ontology constructed so far, the white boxes indicating original database attributes, the grey ones indicating newly introduced concepts.

Please note that, up to now, we did not mention anything about the semantics of the "parent" relationship used to organize the concepts. Having a closer look at how we used it, we can distinguish at least three different uses of this parent relationship:
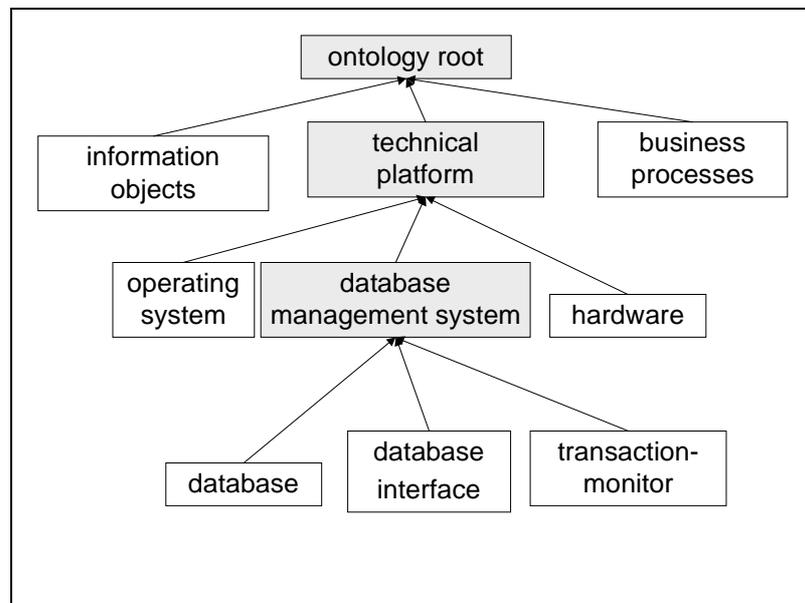

Figure 2: "Parent" Hierarchy.

1. Just as a structuring tool like a folder in Windows (bag-of), e.g., to bundle things which are in a close content relationship, like *operating system, database management system,* and *hardware*, which can be comprised under *technical platform.*
2. To express aggregation (part-of), as it is used to indicate that the *database*, the DB *interface*, and the *transaction monitor* together constitute the *DBMS*.
3. Of course, there are also generalization/specialization relationships (is-a/instance-of) which can be seen below,in particular when extending the upper levels described so far towards the database values which can be used for the several attributes.

Example (cont'd): Now the user wants to specify that he is interested only in relational database systems. Although there are indeed a number of attribute values used in the DB instance (like *Informix 7.13*, *Informix 7.24*), there is no abstract value to specify this concept, but all occuring relational databases would have to be specified explicitly in a disjuction.

In our navigation ontology this could be reflected by a number of abstract concepts inserted into the concept lattice between the attribute *database* to be further described and the set of specific occuring attribute values. This can be done as it is shown in Figure 3, the "parent-relationship" representing here is-a links in the upper part of the picture and instance-of links between concepts and attribute values in the lower part.
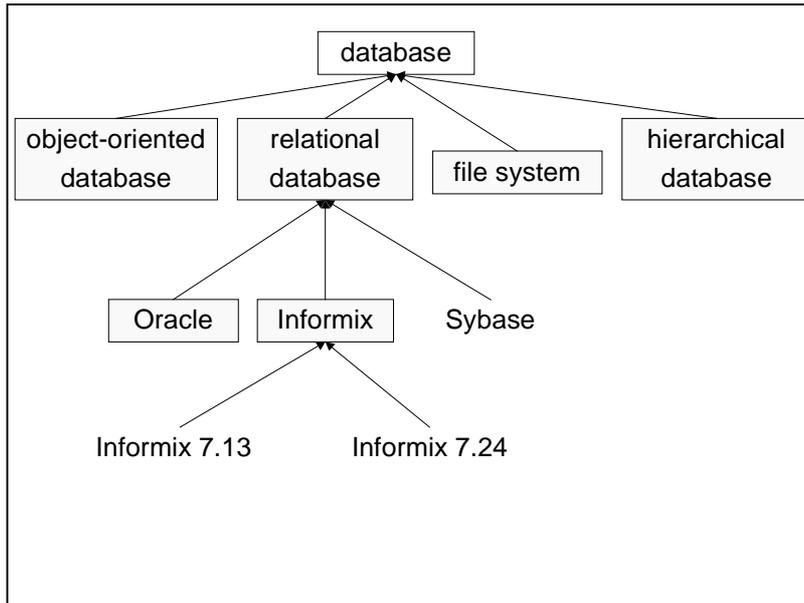
Figure 3: Modeling Concepts Between Attributes And Attribute Values.

2.2.2. Exploit Knowledge about Incompatibility and Similarity

Example (cont'd): Now assume the user determines to employ the database system Informix 7.24, makes some other (uncritical) specifications, and then chooses Windows 3.11 as operating system. Since Informix systems in general are not compatible with Windows platforms, such a query will not produce any result. In contrast to other, meaningful selections which produce no result because there was no such system configuration in the past, this selection makes no sense at all, which is known at modeling time and could be represented in the system. This would not only allow to explain the empty answer set, but it would even be possible to forbid such a selection already at the user interface.

In our system we can deal with such phenomena introducing a new, symmetrical "incompatible" relationship in our ontology modeling formalism as shown in Figure 4.

Example (cont'd): Suppose the user decides to use a *Sybase* database instead of the former decision for *Informix*. Basically, this is no problem since the database also works under *Windows 3.11*. Nevertheless it might happen that no information can be found in the system because this solution was never realized before. However, similar solutions under *WindowsNT* were implemented several times. In this case, the user could be provided with the information concerning similar previous cases if there was a way to express this similarity in the ontology.

Technically, this can be provided for in the same way as it was done for incompatibility knowledge: introduce a new, symmetric relationship "similar" stating that two concepts are similar with respect to retrieval issues.
In a further extension, weighted similarity could be introduced in order to express, e.g., that *Windows 95* is closer to *Windows 98* than to *Windows NT4.0.*
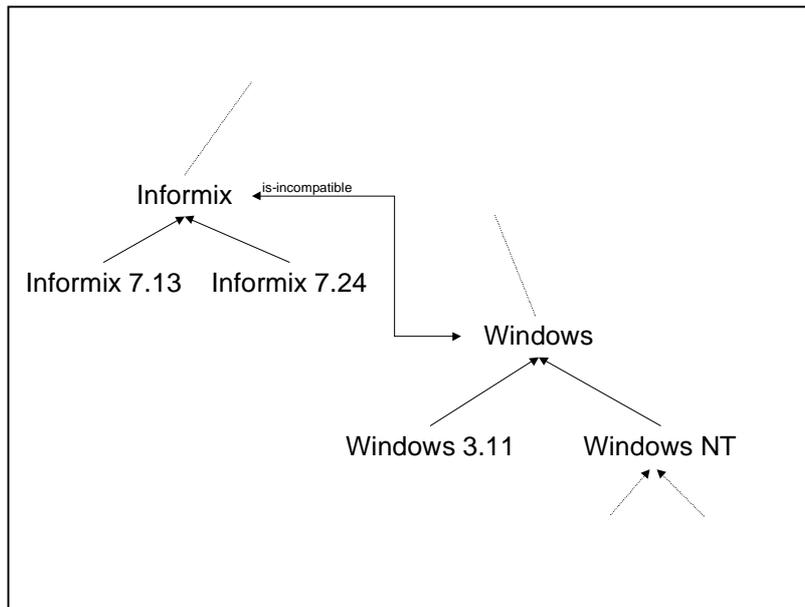
Figure 4: Sample Use of The "Incompatible" Relationship.

Example (cont'd): Consider an application which requires a broad hierarchy of concepts and is typically run by unexperienced users. In this case, navigating in large ontologies might not be the appropriate way of working. Instead, a query-like approach like in Internet search engines might be more useful.

To support such applications, we can extend our ontological structures by two kinds of linguistic data. First, we can attach to each ontology concept a list of synonyms which could be used by the end user to express her actual information need (such a list could either be explicitly attached to the ontology representation, or an online thesaurus like WordNet could be employed at query evaluation time to link natural-language expression to formal ontology concepts). Second, we can attach to each concept a list of keywords, indicators which by their occurence in a text give a strong hint that a given concept is discussed.

For instance, we could extend the representation of the *Windows* concept by synonyms such as "Win", "MS Windows", or "Microsoft Windows", and could attach as keywords terms like "Bill Gates" or "Microsoft".

2.3. System Architecture

Since we focus here on ontological issues, we will only briefly sketch the system architecture, insofar as it is useful for understanding the rest of the paper. The prototype is designed with a three-layered architecture (see Figure 5) and implemented as a client/server application. A **presentation layer** manages all user interactions. As client software for the user interface, we require only a Java-enabled Internet browser. Application logic and data management run on—if required, different—servers. The **application logic** hosts the ontology as well as a number of agents for query preprocessing. Its main task is to serve as an "intelligent mediator" between the users' abstract information needs and detailed database queries in one direction,

and between detailed database result sets and user adequate results presentations. The **data management layer** provides the basic data storage and manipulation services.

In order to prepare for later extensions of the system, the system is designed as an open, agent-based application. In the current, minimal version, the following parts are already there:
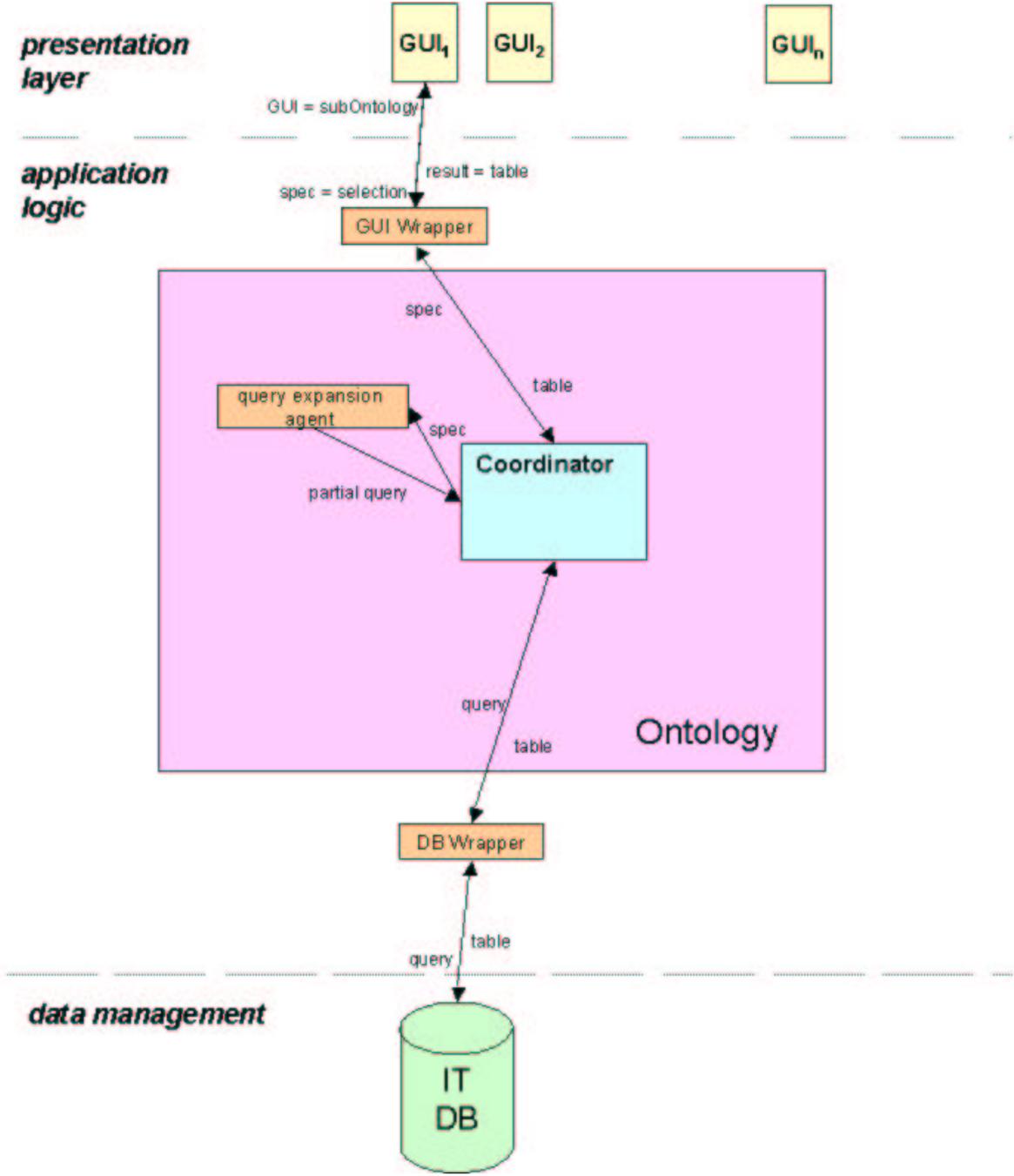
Figure 5: System Architecture.

- **GUI:** Starting the system each user is provided with an instance of the graphical user interface. It knows how to present the given ontology to the user and allows to make

selections and text input. In the software configuration context, the GUI allows the user to specify which aspects of IT systems he or she is interested in.

- **GUI-Wrapper**: The GUI wrapper realizes the interface between GUI (presentation logic) and coordinator (the central contact point for the application logic). So, it is the gateway to the agent platform which is the basis of the application logic realization. The GUI wrapper translates GUI selections into a symbolic representation and transmits them to the coordinator. Vice versa, it delivers query results from the application logic for presentation to the GUI.
- **Coordinator:** The Coordinator manages the steps and responsible agents for all tasks required to:
  a) generate meaningful database queries from user selections using meta knowledge (the ontology)
  b) create an appropriate presentation of results, again with the help of meta knowledge
  c) ensure an efficient query and result management caching prior queries and the respective results, taking care about consistent result presentation in the case of multiple queries processed concurrently and restraining incoming results which are not longer relevant due to selections changed in the meanwhile.
- **Query Expansion Agent:** The expansion agent generates one or more abstract database queries (in a notion similar to SQL) from the given specification of the user's interest (i.e. the set of all his/her GUI selections). To this end, the agent interprets a set of transformation rules which are described in more detail in subsection 2.4.
- **Database Wrapper:** The database wrapper realizes the interface between application logic and database management system. It translates abstract database queries (in the above mentioned generic notion) into the concrete syntax of the underlying DBMS. Vice versa, it receives a set of result tables from the DBMS and transmits them to the application logic to be further processed.
- **IT DB:** the concrete database to be queried by the system. In the current prototype version we have just a small XML database storing IT systems and technical platform information.

2.4. Transforming User Interactions into Database Queries

The parent relationships modeled in the ontology, appropriately visualized (e.g., in a "Treeview") in the GUI, can enable browsing through the criteria described in the IT planning tool. It is important to note that the parent hierarchy not only serves for browsing the (partly abstract) database attributes, but also for specifying the user's information needs. The user chooses in the hierarchy exactly those aspects he or she is interested in, or which shall definitely not occur in a result (maybe several at a time, in different branches of the tree). When an abstract concept has been chosen (for instance, *database management system*) this means that aspects of this concept must be considered in a possible result which means in turn that a possible result contains database entries for attributes which are modeled below the abstract concept in quest. So, the abstract concept selection has to be rewritten in a *disjunction* of partial queries testing for database entries in the concrete attributes covered by the abstract concept. In the above example this would mean that the user gets only IT systems as results which have some entry in at least one of the attributes *database, database interface*, or *transaction monitor*.

In the same way as for using abstract attributes, also abstract attribute values can be supported in the GUI. When the user selects an abstract attribute value, all concrete values below can be considered as selected, i.e. the abstract query again can be rewritten into a *disjunction* of

partial selection constraints in the ultimate database query. In the above example, selection of *relational database* leads to a rewritten query to the IT planning tool which searches for entries containing one of the attribute values *Informix 7.13, Informix 7.24,* or *Sybase.* This mechanism also allows to hide levels of detail which are definitely not appropriate for the intended user. In this case, the user no more manipulates the actual database values, but only deals with abstract concepts above them.

Please note that the rewrite rules for mapping user GUI actions to concrete database queries is not completely trivial, e.g., dealing with negative selections. Below, in Table 1, we shortly sketch the transformation rules currently used.

Since, in our experience interface design is a crucial point, and also virtually simple modes of interaction were not understood / adopted by "normal" users, details of the user interaction – query transformation rules should be determined in collaboration with the intended end users. In our prototype, we implemented the rules below. For denoting them, we use "!" to indicate the NOT operator for formulating query conditions, and "|" to describe attribute values of set-valued (multiple values) database attributes.

GUI action	DB query part
select value *value$_1$* of a single-valued DB attribute *attribute$_1$*	`(= attribute`$_1$` value`$_1$`)`
negative selection of value *value$_1$* of a single-valued DB attribute *attribute$_1$*	`(!= attribute`$_1$` value`$_1$`)`
selection of value *value$_1$* of a set-valued DB attribute *attribute$_1$*	`(CONTAINS attribute`$_1$` |value`$_1$`|)`
selection of DB attribute *attribute$_1$*	`(!= attribute`$_1$` null)`
negative selection of DB attribute *attribute$_1$*	`(= attribute`$_1$` null)`

selection of an abstract concept *concept$_1$* such that *attribute$_1$*, ..., *attribute$_n$* are all DB attributes which have *concept$_1$* as a superconcept	`(:or`
	`    (!= attribute`$_1$` null)`
	`    (!= attribute`$_2$` null)`
	`    ...`
	`    (!= attribute`$_n$` null))`

negative selection of concept *concept$_1$* such that ...	`(:and`
	`    (= attribute`$_1$` null)`
	`    (= attribute`$_2$` null)`
	`    ...`
	`    (= attribute`$_n$` null))`

selection of several values *value$_1$*, ..., *value$_n$* of a single-valued DB attribute *attribute$_1$*	`(:or`
	`    (= attribute`$_1$` value`$_1$`)`
	`    ...`
	`    (= attribute`$_1$` value`$_n$`))`

negative selection of several values *value$_1$*, ..., *value$_n$* of a single-valued DB attribute *attribute$_1$*	`(:and`
	`    (!= attribute`$_1$` value`$_1$`)`
	`    ...`
	`    (!= attribute`$_1$` value`$_n$`))`

selection of an abstract concept *concept$_1$* below the DB attribute *attribute$_1$* and above its values *value$_1$*, ..., *value$_n$*	`(:or`
	`    (= attribute`$_1$` value`$_1$`)`
	`    ...`
	`    (= attribute`$_1$` value`$_n$`))`

negative selection of an abstract *concept$_1$* below ...	`(:and`
	`    (!= attribute`$_1$` value`$_1$`)`
	`    ...`
	`    (!= attribute`$_1$` value`$_n$`))`

selection of abstract concepts *concept$_1$* and ~~*concept$_2$* below the single-valued DB~~	`(:or` `    (:or`

concept₂ below the single-valued DB attribute *attribute₁*where *concept₁* is above the attribute values *value₁, ..., valueₙ* and *concept₂* is above the attribute values *valueₙ₊₁, ..., valueₙ₊ₘ*	`(= attribute₁ value₁)`
	`...`
	`(= attribute₁ valueₙ))`
	`(:or`
	`(= attribute₁ valueₙ₊₁)`
	`...`
	`(= attribute₁ valueₙ₊ₘ)))`

Table 1: Transformation Rules for GUI Interactions.

All other cases with multiple selections can be translated into the conjunction of the partial queries resulting from translating their parts. In the case of set-valued attributes, the semantics of user interactions should be negotiated with the system users.

2.5. Query Processing with Ontological Background Knowledge

In our system, the different parts of the ontology can be used for different purposes to support the users' information access. What we already mentioned is the use of the ontology for structuring the GUI; this enables easy and intuitive specification of user interests in a mixed browsing/query-like mode of interaction with the retrieval results continuously updated depending on the user interactions refining or relaxing the query in order to see how the result set changes accordingly. Moreover, several other uses to support better retrieval can be envisioned. Though, they are not yet implemented, we sketch some nearby extensions to the current state of affairs:

Thesaurus information helps interpret manual text input: in a coming version of the system the user will be allowed to fill in search terms in a text box. Exploiting the synonym and indicator keyword lists in the ontology, the system will then try to identify which formal concepts of the ontology he or she is interested in and automatically mark the appropriate concepts in the GUI presentation of the ontology. In particular in large ontologies, this also allows to find a quick entry to the system before further browsing the information space via graphical interaction and observation of the incrementally updated result sets. This also eases jumping between different branches of a large ontology dealing with many different aspects. Besides this function as a "shortcut" when navigating through a large ontology, the text-to-concept mapping of course also helps unexperienced users to find appropriate database concepts without detailed knowledge of the specific terms.

Concept hierarchies help relaxing too restrictive queries: In the case that a user selection does not produce any result (i.e. no prior case is found matching exactly the desired criteria) the idea of query relaxation, well-known from information retrieval, can be done on the basis of ontological background knowledge. Weakening search constraints or generalizing search concepts may help finding old cases which are not exactly what is wished, but sufficiently similar. The question is how to relax a query.

The most simple realization of this idea is to use the "parent"-relation to replace specific terms by more general ones, or to replace specific attribute values by more abstract value sets. For instance, having a look at Figure 4, we could replace *Windows 3.11* by *Windows.* Having in mind that another extension to the existing system should be an explanation module explaining to the user why a specific case was retrieved, such relaxation steps via the concept hierarchy would be an easy to understand system functionality.

If the underlying ontology contains also explicit "similar" relationships, these can also be used for query relaxation, replacing given concepts by similar ones. It requires some further consideration to decide whether "similar" relationships should be used transitively for query relaxation and whether exploitation of "similar" relationships and "parent" hierarchy should be done in an intertwined manner.

Search heuristics allow for complex navigation in ontologies: In contrast to most publications about exploitation of ontological background knowledge for query relaxation, which do not discuss in detail how far or how deep an ontology should be traversed before stopping relaxation because the possible result is likely to be "too far away" from what the user is really looking for, we expect a flexible control of relaxation strategies important, especially when more relationships than just "parent" links are prevalent. It is beyond the scope of this paper, but we refer to (Liao *et al.*, 1999) for a specification formalism to describe search heuristics as simple navigation instructions in ontologies which determine which steps to make in the ontological concept network to come to a relaxation concept when the current query did not produce a result.

In the above mentioned case of an ontology containing "parent" and "similar" relationships, an informal description of such a stepwise relaxation strategy could look as follows:

1. Take a concept c from the set of (positive) user selections and search for similar concepts traversing "similar" links in the domain ontology at most two steps forward.
2. If Step 1 does not produce a result, try a generalization step, starting from c and following one "parent" link.
3. If Step 2 fails, too, make one step to a parent p of c, then search for "similar" concepts to p.
4. ...

In a system with sophisticated relaxation facilities, a future extension could allow the user to prioritize several parts of the query to further control relaxation.

Incompatibility of concepts helps constraining search: While all remarks above concern query relaxation in order to find information despite an initially empty result set, we can also use the "incompatible" links to inform the user about contradictory concept selections before sending a query to the underlying databases, or to explain the user why a selection did not produce a result.

Informally, the (slightly simplified) strategy looks as follows: for each two concepts taken from the set of user selected concepts, transitively follow all "parent" links upward, and test all "incompartible" links going out from such a "parent", "grandparent", "ancestor" concept whether the linked, incompatible concept is above of (i.e., again, an "ancestor") or identical with the other start concept. If such two concepts can be found, the user selection is contradictory.

3. General Considerations & Related Work

Coming back to the questions arisen in Section 1, we would like to make some general remarks about design decisions and insights of the software configuration case study.

Starting with *the question where ontologies come from in practical applications* it should be taken for granted that in real-world Knowledge Management and Organizational Memory applications it is almost always the case that the problem to face is not building a completely new, amazingly intelligent system from scratch. Instead, we have to build sort of a "meta information system" on top of manifold existing information and knowledge systems internal and external to the company, elements of which are combined and fusioned for specific application purposes (Abecker *et al*., 1998). All these legacy systems already come with their own ontological commitments, either explicit in a formal ontology in the narrower sense, or implicit, as it was the case with the database schemata of the IT information systems to be accessed in this case study.

So, an easy answer would be to integrate / merge the existing database schemata into one "ontology seed", enrich this seed with additional semantical information, and streamline conceptual differences between the different input schemata a bit. In the current prototype, this easy solution seems to work, and it even seems needless to employ the complex agentware describe in Figure 5 if the customer would have used a good object-oriented database together with a well thought out DB schema. Hower, besides the fact that we have to live with one or more legacy systems in the form of relational DBs, and also besides the fact that in other applications there might appear a (or more) document archive(s) instead of databases coming with their individual catalogues which might not easily be mapped into an object-oriented DB schema, our main focus was on user adequacy. So, we come back to the questions 2 and 3 of the introductory section, dealing with the topic of *conceptually sound and ergonomically adequate modes of user interaction.*

First of all, it should be noted that schemata, models, and ontologies in the way computer scientists and system designers build them are by no means sufficient for communication with "normal users". Already in the almost trivial case dealt with in our project—only one source of information, though bound together from three prior databases, nevertheless described by one conceptual schema; all information for determining the relevance of a stored solution directly contained in the attribute values; the first user group to be supported experts in the field—it nevertheless turned out that the existing relational schema was not clear, not handy, and not comfortable enough to build a direct manipulation interface on top of it. Even if we would have had the opportunity to reengineer the relational schema into an object-oriented one which seems much more adequate to deal with for users and which covers many of the aspects described in Subsection 2.2.1 (introduction of additional abstract concepts) it nevertheless seems to be the case that a model of the world *as it is* (and this is the very meaning of "ontology") is not the most appropriate tool here.

We would rather propose to define a **navigation ontology**, or a **presentation ontology**, an additional layer of abstraction between the information system's view (the relational schemata in our study, the legacy systems' given ontologies in the general case) and the user. Such a user-oriented knowledge organization and access model will ususally be as simple as possible to achieve the purposes aimed at, it maybe formally inconsistent or contain smaller formal errors in favour of user adequacy, and it might be a bit more "fuzzy" than an *"explicit account of a conceptualization ..."*. Such a navigation ontology should reflect the users' conceptual views and mental models thus being the appropriate starting point for defining user interfaces.

We are not sure whether the notion of "ontology" is the most appropriate word here, since we are not so much interested in how the world *is*, but rather in how the user *sees it* or in which concepts he *thinks about it*. So, we merely deal with the epistemological or the semiotic dimension than with the ontological one. However, what we are sure is that this is the

appropriate level of consideration when designing knowledge portals and user adequate information systems.
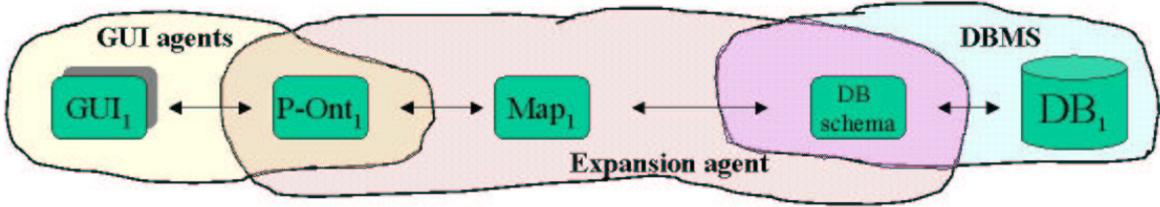
(a) The simplest scenario: one user class
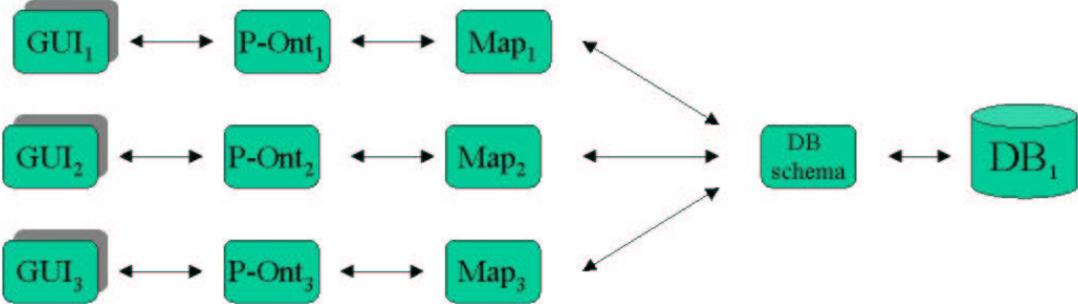
(b) First upgrade: multiple user types

Figure 6: Development Stages of The Complete Scenario (I).

Although, in our current application, the presentation ontology is a proper superset of the relational DB schema, in the general case, it might be an arbitrarily redesigned model, e.g., using other terminology, partly more coarse-grained, partly more fine-grained conceptual structures, orthogonal structuring dimensions, or less and less difficult to understand different link types between concepts. In Figure 6 (a) we sketch the current application with some emphasis on the transformation and mediation processes taking place, and the knowledge sources used. The goal of the system is to map a GUI user interaction to a query to the DB management system, and, vice versa, to feed back a DBMS answer table to a meaningful answer presentation to the user. To this end, the user interaction—which means manipulation (select/deselect) of presentation ontology parts—is mapped by the query expansion agent into DB queries. Therefore, it exploits the mapping rules described in Subsection 2.4. In more complex ontology mappings than the ones described above, the transformation of results into user terms may also be nontrivial, in the case that the mapping rules are not easily revertable.

Figure 6 (b) and Figure 7 (c) show two further extensions which are already provided for in the software architecture and which are conceptually consequent developments from the above scenario, which, however, were not yet tested at our user site.

Figure 6 (b) generalizes the scenario by adding several presentation ontologies to define interfaces to the same underlying system for different user types (novices vs. experts, managers vs. customers, etc.). This is simply done by adding new presentation ontologies with their respective mapping rule sets.

Figure 7 (c) introduces additional information sources to be queried simultaneously which makes the mapping processes more complex, now splitting queries into different query parts. In the case that the systems to be queried together are themselves differently conceptualized, we need additional mediators dealing with those semantic heterogeneities.
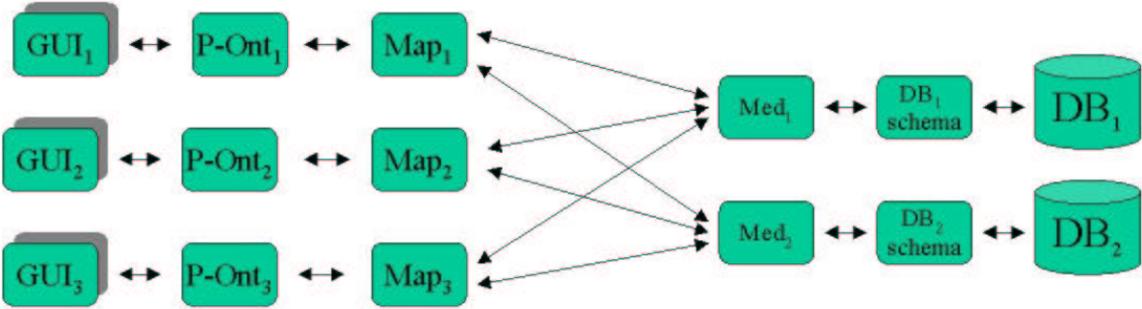
(c) Full version: multiple user types and multiple databases
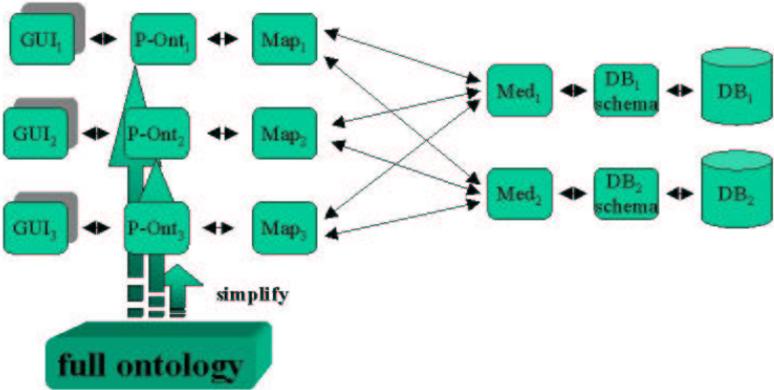
(d) Envisioned: presentation ontologies derived from „full" ontology

Figure 7: Development Stages of The Complete Scenario (II).

Finally, Figure 7 (d) reflects the fact that it often makes sense to have a powerful ontology with, e.g., fine-grained structures or manifold link types to one's disposal which must be modeled as a proper superset of the given underlying schemata containing really "more" information than these schemata, not only "syntactic sugar" for user adequacy. Such added knowledge might, e.g., be employed to make complex similarity estimations for cases or to make inferences on background knowledge. But it is also to expect that, according to our main goal with the system, such a full-fledged ontology is not appropriate to be presented to the

users. So, it could make sense to have such a powerful "full ontology" to the system's internal disposal, but generate presentation ontologies for different user classes as simplifying or restructuring projections from this basic ontology.

Though it seems a bit contradictory that we first vote for simple and clear ontologies and then propose a rich "full ontology", there are nevertheless application experiences which show the usefulness of richer background knowledge. For instance, for enhanced retrieval of mechanical engineering design documents, (Baudin *et al*., 1995) formulated domain-dependent retrieval rules for finding documents possibly relevant to a query. These "proximity retrieval heuristics" employ generic background knowledge about the mechanical engineering domain and the design process as well as the concrete machine model of the technical artifact to be constructed. Similarly, in the domain of managing maintenance and fault diagnosis experiences for a very complex technical system, (Bernardi *et al*., 1998) proposed to use knowledge about electrical and hydraulic connections as well as functional similarity for propagating relevance and assessing similarity of experience logs. These relationships are gathered in the formal knowledge organization models and are accessible by the retrieval mechanisms, but were also considered much too complex to be shown directly in the user interface. In (Liao *et al*., 1999) we describe a simple specification formalism for writing down retrieval heuristics as ontology graph traversal rules for searching similar index concepts. A technically very much similar approach (marker passing in conceptual structures) was adopted by (Schmalhofer *et al*., 1997) to support simulation of creative inferences. In the case study presented here, apart from the "parent" relations which directly organize the user's information access, all other relations are either exploited by the retrieval mechanism, or directly employed by the GUI agent, e.g., to forbid obviously inconsistent selections.

The basic theme underlying this discussion is *which link types should be provided by ontology modeling formalisms*. In the Karat system (Tschaitschian *et al*., 1999) we employed (except for few fixed relationship between documents, e.g., for versioning) completely informal knowledge organization models which suggested to have semantics by their intuitive graphical presentation, but there was no formal semantics underlying to be interpreted by the machine. (Voss *et al*., 1999) argue in the same direction and propose to use only the idea of *textually grounded concepts* for text knowledge organization, sharing, and access, which are not formally modeled by hand and automatically interpretable by the machine. In their *ConceptIndex* system, they only use two relationships between concepts, namely "comprise" (the same as our informal "parent" relationship) and "associated" (similar to "similar", defined via textual co-occurrence, related to the "see also" in textbooks and catalogues). (Voss *et al*., 1999) also give some interesting pointers to different possible operating points in the formal-informal universe. Several authors already pointed out that the classification schemes and index taxonomies used for Internet access are usually by far less informal and more semantically unclear than the "crisp" deductive methods to work with them presuppose (Welty, 1998). Above, we mentioned that formal, complex concepts and relationships can be valuable, but should not be shown to the user which led to the idea of presentation ontologies. It seems an open issue to us whether it would be possible to have integrated in *one* model both "hard facts" as utilized, e.g., by the Ontobroker (Decker *et al*., 1999) to make integrated inference and retrieval possible, to draw conclusions from document-embedded facts and to do inference-driven retrieval, and "informal notions" as indicated above.

Another requirement with respect to *ontology representation and content* which seems crucial to us is the linkage of (more or less) formal concepts to linguistic knowledge, textual representations, keywords, textual concept indicators, or thesaurus information. This was already a main innovation of our Karat system (Tschaitschian *et al*., 1997), has been further

considered in the KnowMore project for building active, context-driven access to Organizational Memories (Abecker *et al*., 2000), and recently gained growing interest in the area of Internet Information Retrieval (Lesser *et al.,* 1998; Guarino *et al*., 1999).

4. Summary

In this paper, we tried to approach more realistic scenarios for the use of ontologies for knowledge retrieval than just applications within the academic world. We identified the questions of ontology building, user-centered interfaces, and powerful uses of ontologies as interesting. Introducing a practical application study done for an industrial customer, we described an approach and architecture which is still a "small application", but generically designed with many possible extensions. The main design decision was to build an agent-based wrapper/mediator-architecture which introduces the notion of presentation, or navigation ontologies, respectively, as a further layer of abstraction mediating between the user interface and the system's internal knowledge. The use of linguistic knowledge for fast, direct jumps into the concept browser, the use of "parent" relationships and domain-specific knowledge for query relaxation, and the use of knowledge about similarity and incompatibility are provided for in the system, and partly realized in a prototype. Ontology-building was facilitated by defining the ontology as a conservative extension of the existing DB schema. Incremental update of answer sets for an intertwined browse-and-search-like user interaction, as well as technical scalability, and easy extensibility by additional intelligent services for similarity assessment or result fusion from different sources, are provided for by the Java-based agent platform at the heart of the system. Further steps concern gathering of application experience, investigation of uses of ontological background knowledge for case similarity assessment, generation of explanations of system answers, and use of several system components in other agent-based knowledge management projects aiming at a comfortable yet powerful and extensible implementation platform for Intranets and knowledge portals.

REFERENCES

Abecker, A., Bernardi, A., Hinkelmann, K., Kühn, O. and Sintek, M. (1998). Towards a Technology for Organizational Memories. *IEEE Intelligent Systems*, **13(3).**

Abecker, A., Bernardi, A., Hinkelmann, K., Kühn, O. and Sintek, M. (2000). The KnowMore Project: Knowledge Management for Learning Organizations. *International Journal on Information Systems Frontiers (ISF), Special Issue on Organizational Memory and Knowledge Management*. Kluwer Academic Publishers. To appear.

Baudin, C., Kedar, S. and Pell, B. (1995). Increasing levels of assistance in refinement of knowledge-based retrieval systems. In *Machine Learning and Knowledge Acquisition - Integrated Approaches* (Tecuci, G. and Kodratoff, Y. ed.s). Academic Press.

Bernardi, A., Hinkelmann, K. and Sintek, M. (1998). Model-Based Information Systems for Knowledge Management. In *IT & Knows, Conference on Information Technology and Knowledge Systems, a part of the 15th IFIP World Computer Congress, Vienna & Budapest* (Cuena, J., Tjoa, A.M. and Markus, A. eds.).

Decker, S., Erdmann, M., Fensel, D. and Studer, R. (1999). Ontobroker: Ontology Based Access to Distributed and Semi-Structured Information. In: *Database Semantics: Semantic Issues in Multimedia Systems, Proceedings TC2/WG 2.6 8th Working Conference on Database Semantics (DS-8), Rotorua, New Zealand* (R. Meersman *et al.* eds.). Kluwer Academic Publishers, Boston.

Duschka, O., Fensel, D., Lenzerini, M., Rousset, M.-C., Wache, H. (1998). *Second International and Interdisciplinary Workshop on Intelligent Information Integration*. Held together with the European Conference on Artificial Intelligence (ECAI'98), Brighton Centre, Brighton, UK.

Fensel, D., Knoblock, C., Kushmerick, N., Rousset, M.-C. (1999) *IJCAI-99 Workshop on Intelligent Information Integration.* Stockholm.

Guarino, N., Masolo, C. and Vetere, G. (1999). OntoSeek: Content-Based Access to the Web. *IEEE Intelligent Systems*, **14(3)**: 70-80.

Knoblock, C.A. and Ambite, J.L. (1997). Agents for Information Gathering. In *Software Agents* (Bradshaw, J. ed.). AAAI/MIT Press, Menlo Park, CA

Lesser, V., Horling, B., Klassner, F., Raja, A., Wagner, T. and Zhang, S.XQ (1998). A Next Generation Information Gathering Agent. In *Joint Proc. of the World Multiconference on Systemics, Cybernetics, and Informatics and the Int. Conf. on Information Systems Analysis and Synthesis.*

Liao, M., Hinkelmann, K., Abecker, A., and Sintek, M. (1999). A Competence Knowledge Base System for the Organizational Memory. In *XPS-99: Knowledge Based Systems - Survey and Future Directions, 5th Biannual German Conference on Knowledge Based Systems, Würzburg, Germany* (Puppe, F. ed.), Lecture Notes in Artificial Intelligence 1570, Springer-Verlag.

Luke, S, Spector, L., Rager, D. and Hendler, J. (1997) Ontology-based Web Agents. In *Proceedings of First International Conference on Autonomous Agents, AA-97.*

McGuiness, D.L. (1998) Ontological Issues for Knowledge-Enhanced Search. In *International Conference on Formal Ontology in Information Systems, FOIS'98* (Guarino, N. ed.), IOS Press.

Schmalhofer, F., Franken, L., and Schwerdtner, J. (1997). A Computer Tool for Constructing and Integrating Inferences into Text Representations. *Behavior Research Methods, Instruments, & Computers,* **29(2):** 204-209.

Tschaitschian, B., Abecker, A., Hackstein, J., Zakraoui, J. (1999). Internet Enabled Corporate Knowledge Sharing and Utilization. *In 2nd International Workshop on Innovative Internet Information Systems (IIIS-99) on Internet-Based Organizational Memory and Knowledge Management* (Schwartz, D., Divitini, M., and Brasethvik, T. eds.). Hershey (USA) & London (UK): Idea Group Publishing. To appear.

Tschaitschian, B., Abecker, A., and Schmalhofer, F. (1997). Information Tuning with KARAT: Capitalizing on Existing Documents. In *10th European Workshop on Knowledge Acquisition, Modeling, and Management (EKAW-97), Sant Feliu de Guíxols, Catalonia, Spain* (Plaza, E. and Benjamins, R. eds.). Springer Verlag, Lecture Notes in Artificial Intelligence 1319.

Voss, A., Nakata, K., and Juhnke, M. (1999).ConceptIndexes: Sharing Knowledge from Documents. *In 2nd International Workshop on Innovative Internet Information Systems (IIIS-99) on Internet-Based Organizational Memory and Knowledge Management* (Schwartz, D., Divitini, M., and Brasethvik, T. eds.). Hershey (USA) & London (UK): Idea Group Publishing. To appear.

Welty, C. (1998). The Ontological Nature of Subject Taxonomies. In *International Conference on Formal Ontology in Information Systems, FOIS'98* (Guarino, N. ed.), IOS Press.

Wiederhold, G. and Genesereth, M. (1997). The Conceptual Basis for Mediation Services. *IEEE Intelligent Systems*, **12(5)**.