Access to the Repository using Natural Language Knowledge

Harriet Dahlgren, SISU-Gothenburg, harriet@sisug.sisu.se
Jan Ljungberg, SISU-Gothenburg, jan@sisug.sisu.se
Sten-Erik Öhlund, SISU-Stockholm, ster@sisu.se

Abstract:

A repository stores deliverables from the different phases of the software development process as well as information about those deliverables. The contents could be specifications from the very early phases (Business Engineering) as strategic or goal oriented models as well as data models, process models or code-modules. It is easy to get lost in the repository since the concept you are looking for might have another name than you expect.To navigate properly in such a repository a navigation support will be needed. We will argue that a conventional browser is not enough for this purpose and that a tool based on knowledge about user language is required. A prototype of such a navigation support system which mainly deals with conceptual models or data models is sketched in this paper.

1. Introduction

A repository might store data about both the business system and the information systems supporting the business system. Generally a repository will store deliverables from different phases of the software development process as well as information about the deliverables. This could include specifications from the very early phases as strategic or goal oriented models as well as data models, process models, code-modules and even executable objects. Project managment data and other cross-lifecycle information will also find its place in the repository. There exists a variety of different definitions, views and opinions of what a repository really is. Three sample definitions collected from (Forte 1989) are:

i) A *'mechanism for defining, storing and managing all the information needed to develop, maintain and execute a corporation's software systems'* .
ii) Systems *'that store project data and design data non-redundantly, and maintain all the links between them.'*
iii) A *'collection of information which supports business and data processing kinds of activities...'*.

The repository should provide the services for keeping the information consistent and nonredundant. A repository might be based mainly on enterprise information or project information and should be defined on the basis of a well founded meta-modell. This meta-model should not depend on a specific method, but it should be rich enough semantically to be used by different tools and supporting different methods and modelling languages. Focus will thus be given to the meaning of the data stored in the repository.

Different tools will be available for interpretation and operation on the repository contents. What requirements one could put on the tools for accessing and maintaining the repository depends on the main role(s) that a repository will play. Will it mainly keep the information required during different phases of the lifecycle keeping this information consistent and supporting the transformation of a specification from one representation to another. Will reusability of models, concepts and specifications make sense ? In that case how do we get to the right items to reuse ? Our view of the repository in this paper is mainly that of a central corporate resource dealing with all aspects of enterprise and business concepts. Due to phenomena as *terminological*, *cultural* and *sociological* plurality of a large organisation the navigation among centrally defined concepts will pose a large problem. In this paper a tool will be sketched for navigation among models and concepts in a repository of corporate data.

2. Accessing the Repository

The hypothesis of this paper is that a tool that bridges the gap between the user´s view of his problem and the actual representation in the repository should be very useful to achieve, among others, the task of reuse. It is likely to believe that the user´s view is reflected in his use of 'natural language'. Thus the functionality of such a support tool should be to guide or navigate the user through the repository, by bridging the gap between the user´s language and the business terminology as it is actually defined in the repository. This approach means that we adopt a view where the repository is seen as a base for defining the semantics of a business´ terminology (at least concerning the earlier phases).

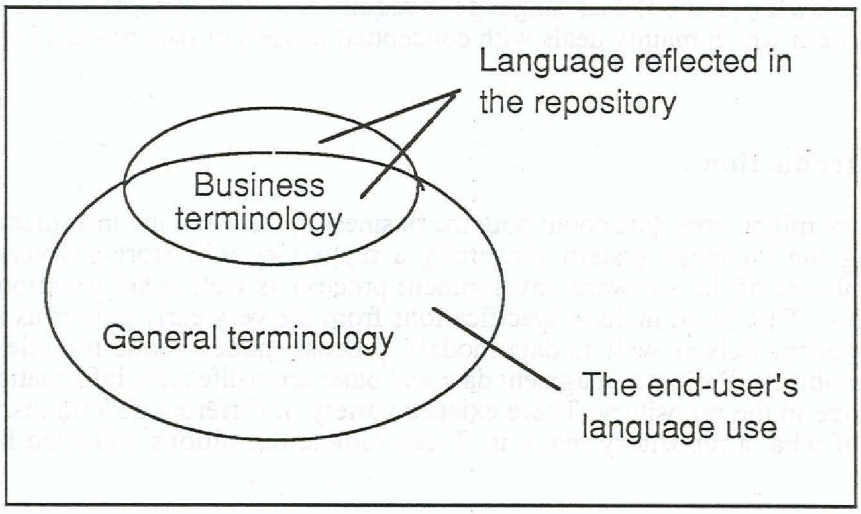

Language reflected in the repository
Business terminology
General terminology
The end-user's language use

Fig. 1. User-language visavi business language.

To overcome the great difficulty in overviewing the data and information present in a repository or a number of conceptual models, the graphical means of orientation and navigation have to be extended with alternative ways of accessing information. Using a conceptual approach to navigation normally puts demands on the user´s knowledge of business terminology. The user could be assisted in his/her use of concepts for retrieving and searching in a repository or a specific model. One fundamental part in a tool for arriving at the right business concepts is the use of knowledge about mening relations at the word level inherent in language, as for example synonyms, i.e. semantically close concepts and their namings.

2.1. A Modelling Perspective

The conceptualization process is hardly an easy process for end-users. To identify the important parts of a business using main concepts in the process of business modelling generally takes a lot of effort from domain experts together with modelling experts. This is a creative process involving different kinds of abstraction mechanisms.

Partly identifying *interesting* parts in business documents, handling rules, entities, relations, goals etc., definitely depend on the aims and methods for modelling. One way to support this process is to actively use the repository. We may want to check that the concepts and models we are creating are not violating any constraints of the repository before we put them there. It might be possible to reuse some concepts if we are just able to find them etc. There are many aspects of the problems involved in reusing specifications or models, finding analogies between the problem at hand and old solutions for example (Suitcliffe and Maiden 1990). When we talk about reuse here we mainly refer to reuse of ontological phenomena as objects, relations, terms etc. rather than whole specifications.

Consider the case when a system analyst is making a data model for a new application somewhere in a large organisation. The analyst would like to know if there already exist similar concepts that he intends to use in the repository from applications in similar domains created by other parts of the organisation. For example if he is working with an application for registering facts about cars, ownership etc, he would be interested in finding applications that use the entity concept *car* or want to look at how *ownership* has been defined in other applications. There might even exist a collection of base models for different definitions used in the company that he would like to look into.

The analyst might not know anything about the other applications, nor about their data base schemas. For example he doesn´t know that in other applications the entity to denote a car has the name "auto" and ownership is named possession. With the help of different lexicons, morphological analysis, classification hierarchies etc., it would be possible to guide the analyst through the repository in order to find models that are interesting and relevant for him. If merely string search was used only concepts with the same spelling could be found.

Typically the navigation support will respond to the request for information about *car* with answers like:

There is no such *object* as *car*, but *car* is an *attribute* of object X in model Y. Furthermore the system will tell the user that the object *auto* is defined in model Z. The user wants to look closer at the model with *auto* in it, passes the model to a CASE-tool for further exploration. The communication between the navigation-support and the CASE-tool could be performed either through bridges or via the repository.

Another obvious use of this functionality is to support different kinds of view-integration.

2.2. Viewing the Repository through Documents

Another way of supporting parts of the modelling process is to use text understanding with research traditions from AI and Computational Linguistics. State of the art in this area use templates to focus on relevant (predefined) matters in combination with partial text parsing (Jacobs 1990), (Reimer 1990). A similar approach is used by (Loucopoulus and Champion 1989) to capture domain specific facts in a tool for requirements engineering. A semiotic approach is used by (Kersten, Weigand et al. 1986). Also (Cauvet, Proix et al. 1988) use a natural language interface for french in their advanced modeling environment.

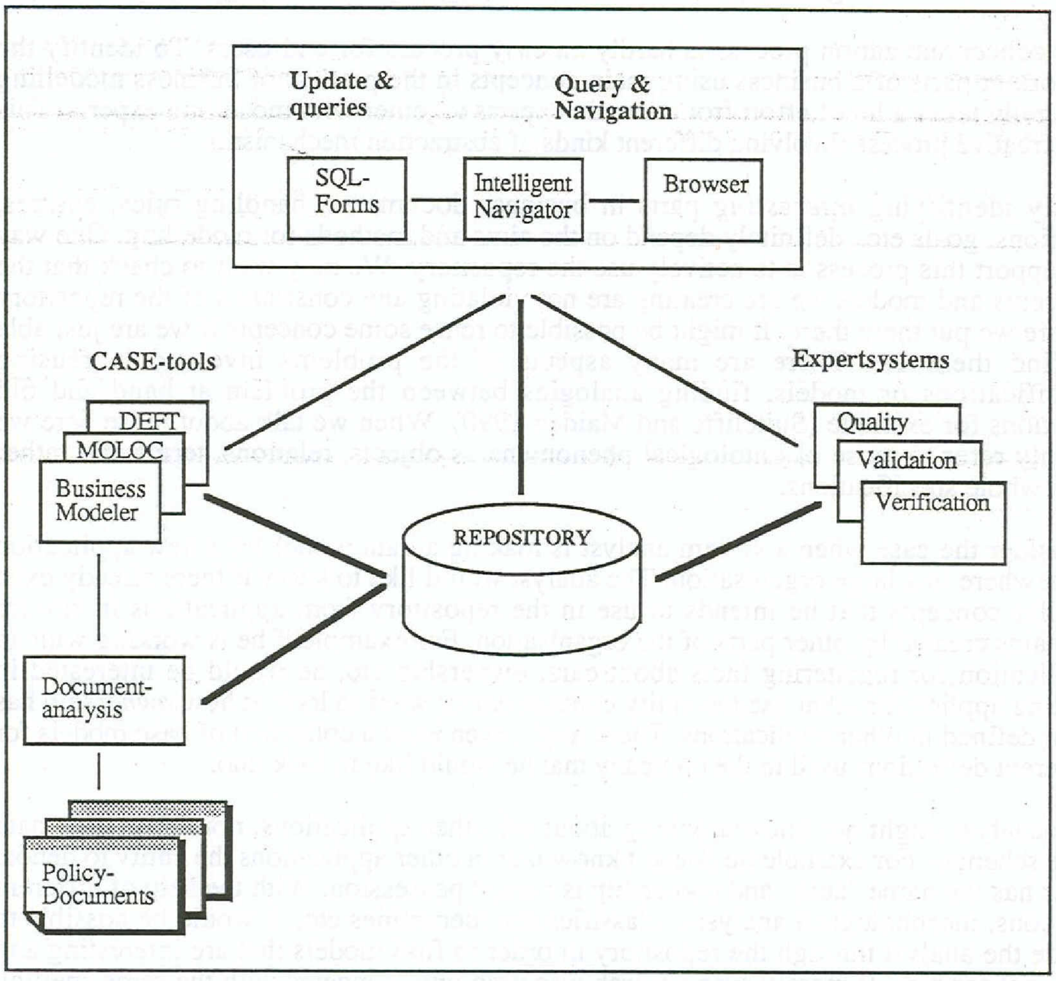


Fig. 2. An integrated environment. Communication is handled through bridges or directly via the repository.

The Repository will be the kernel of an open integrated tool set including CASE-tools, 4gls, expert systems etc (fig. 2). Interaction with the repository might be done directly from the CASE-tools or different kinds of interfaces as forms, browsers etc. The tool sketched in this paper will be one such integrated tool specifically designed for navigation in the repository. When the user arrives at the target (a specific model or concept), the control should be overhanded to whatever tool is needed for further manipulation of the requested item.

A graphical interface might be appropriate to a certain degree, but we will argue that for a large and complex repository it will not be enough for the navigation purposes. When addressing ad-hoc queries, vendors of repositories and integrated tools mainly refer to this matter by means of SQL- or form based querying. A utility of making really free ad-hoc queries which the system responds to in a *cooperative* way is the essence of our proposed tool. The really important question is not what the user sees, if he interacts with a mainly graphical interface, a NL-interface[1], a forms-based interface or something else. The main issue is what is going on behind the surface of the dialogue. The front to the user might be a mixture of graphics, natural language, forms etc and the best presentation of answers and messages should be chosen according to the situation.

[1] Natural Language Interface

A large amount of linguistic knowledge is needed for such an application together with general, domain specific and modelling knowledge. Either tools render their knowledge from substantial amounts of user interaction, preferably this could be done with techniques for learning from examples or learning by being told. This is an alternative to building new large knowledge bases in addition to building an application for abstracting concepts and conceptual information etc from text. Another alternative could be to use existing models and repositories as knowledge bases to guide further knowledge acquisition.

A source for finding central concepts from the business is documents that usually in large numbers belong to a business, often containing business policies, goals, rules, recourses, organisation etc. Our idea is to let this kind of documents serve as an interface to the repository in combination with the query-interface. By starting from documents describing the business, it is possible to find interesting terms, concepts and relations between concepts in a semi-automatic way. This terms could be used to access and even update the repository. Consider the following scenarios of this facility:

i) *Build an initial model fragment* based on a document. The user of the system wants to make a fast initial model based on a document. He highlights interesting terms or phrases in the document and lets the tool analyze them. The internal representation is sent via a bridge to a CASE-tool for presentation of the model on the screen and possibly further manipulation and refinement. The highlighted concepts might also be put right into the repository with the tool.

ii) *Validate a document* based on the contents in the repository. Highlighting parts of the text, the user could request the system to search the repository for appropriate concepts and structures. The system will present the relevant concepts or fragments of models from the repository and let the user compare them. The actual presentation of the retrieved repository contents will always be a choice of textual presentation, form based presentation or graphical presentation.

iii) *Validate a model* in the repository based on the contents of a relevant document, in the same manner as above. Analyze how certain concepts from the repository is used in the document.

Retrieving concepts from text documents and using them to retrieve models or parts of models from the repository gives an opportunity to compare different views of the relation between concepts and different ontological phenomena as they are represented in the documents and in the repository as in scenario *i)* and *ii)*.

3. Linguistic Knowledge

The sketched tool uses different kinds of knowledge about language (linguistic knowledge). This does not neccesarily means that we want to build a traditional NL-interface as TEAM (Grosz, Appelt et al. 1987) or HSQL (Amble, Knudsen et al. 1989) by means of letting the user state queries totally free in his own language. The point is to use knowledge about language to *answer* the queries (or navigation requests) not only mapping the queries to an internal query language. Nevertheless a restricted NL-interface of NL-menu type (Tennant 1986) might be useful for stating more advanced navigation requests and more complex queries. However we will initially focus on the use of linguistic knowledge at the word level.

3.1. Lexical Knowledge

A lexicon is a list of words where each word (entry) is associated with its syntactic properties as syntactic category and morphological information. Word semantics is generally found in the lexicon as well.

Most certainly there is a gap between the concepts used individually among personnel and the business concepts possibly centrallly defined. This could be a major problem in information retrieving from repositories and the like. An approach to filling this gap partially is by exploiting closely related words and synonyms. Language use is individual and one way to get acceptance for a standardised business terminology is by accepting each users language but still guiding users to business concepts, motivating the importance on eased communication with a common language. End users should thus be able to build individual lexicons, relating their own language to business terminology.

A navigation system using natural language knowledge therefore should make use of machine-readable lexical information. Application and domain specific lexica could reflect the business concepts and the authorized synonyms connecting closely related concepts and words. Usually conceptual hierarchies are used in models defining concepts partially, relating them to their closest abstraction category higher in the hierarchy (genus proximum) and their specifying attributes (differentia specifica).

The domain specific lexica should be accessed for update only by authorized personnel. This is important as these terminological lexicas should reflect vocabulary that is standardised within the business.

General lexica should capture surface forms of concepts and general vocabulary to different applications and domains. This could serve as a lexicon during parsing of texts together with application specific lexica. Syntactic and semantic information are necessary in these lexica for language analysis. Information present in the lexicon must at least be *word stem*, *inflectional class*, *syntactic properties* and *semantic specifications*.

Word Structure

The part of linguistics dealing with word-stucture is *morphology*. Knowledge of this kind can be very useful in using keywords for retrieval of data. Most information-retrieving systems are very sensitive to the absolute matching of letters instead of a more flexible search, e.g. not finding information related to *salesman* when the user has used the word *salesmen*.

If we use text as a source for concepts it requires the use of natural language analysis techniques. One of the major problems in this application area is to identify the main variety of surface forms that can disguise a concept of interest. This is important in order to be able to collect concepts that are related to one and the same concept. What the expressions refer to must be found. Different word-forms will represent the same concept. In the following example the concept takes different morphological forms which must be recognized when identifying a concept in a text.

SELL - verb	SALES - noun		SALESMAN - noun	
sell				
sells	sale	sale's	salesman	salesman's
sold	sales	sales'	salesmen	salesmen's

Fig. 3. Word forms

A Swedish example of the same main concept´s surface forms will illustrate the amount morphological varieties that can appear for one relation (the verb sell) only: sälja - säljas - säljer - säljs - sålde - såldes - sålt - sålts.

Concepts are generally more static representations than their surface representations. Verbs are often turned into nouns in the conceptualization process. Also the difference in grammatical category between conceptually related words in text can be confusing.

Word Semantics

There exists a lot of meaning relations at the word level, we will not list them all but some well known phenomena that are related to the problem is (Lyons 1977) :

Synonymy could be distinguished in a stricter and a looser interpretation. By the stricter interpretation two items are synonymous if they have exactly the same sense. It is an open issue if there even exist synonyms by the strict definition. The looser interpretation is what is found in thesauruses, a word we could use instead of another word with a similar sense. For the navigation purposes we will stick to the loose interpretation. Exploiting the use of synonyms could serve as conceptual interfaces within and between collections of lexical and conceptual imformation.

Homonymy is when the same word form can also represent different concepts (homonyms giving lexical ambiguity). When identifying concepts, homonyms must also be handled. A substantial amount of world and application knowledge can be used to disambiguate situations like these. Forming an ontology that handles the domain should preferably not be done in a detailed manner unless there are obvious applications for re-use. A more general approach is preferred as the modelling part in itself could be seen as a process of knowledge acquisition.

Hyperonymy is the phenomenon in language of specialisation and generalisation of terms. It is equivalent to the super/subclasses or types so widely used in computer science and information engineering. Semantic categorization of words and concept hierarchies should be used. The navigation system could present concepts adjacent (at a higher, lower or equal level) in the hierarchy to help users find the right keywords.

Complex lexemes are built by forming a more complex stem from a simple stem by derivation. The suffix *-ly* may for example be attached to certain noun-stems in English to produce the stems of the corresponding derived or complex adjectives (*man -manly, friend - friendly*).

Compound lexemes are very common in concept construction and modelling. How they should be processed in an automatic analysis is not obvious. Should the compound noun be in the lexicon as an entry ? Should it be derived from its parts ? How is the meaning combined ?

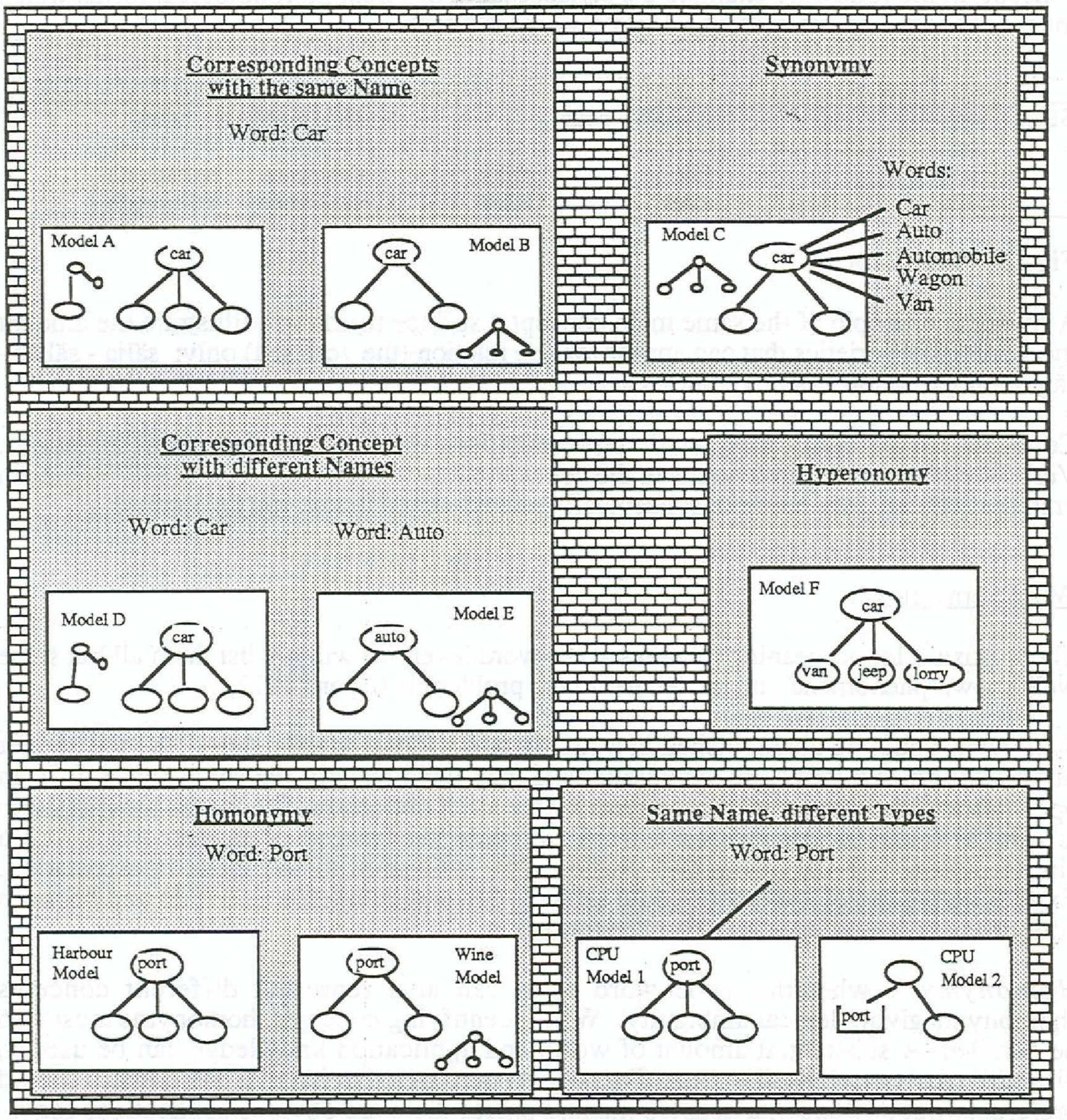


Fig. 4. Relations between words and concepts.

The presented relations between words and concepts can be seen as relations between form of expression and form of meaning. A model could be seen as a knowledge base and also as a representation of meaning. For the cases in figure 4 we can view relations between meaning and expression in the context of working with and retrieving information from a repository.

a) Where the same concept has the same name throughout the repository is the best case. Retrieving the different uses of a concept or corresponding concepts either in different models or other representations shows a concept from different perspectives and is essential in work with integrating models, parts of models etc. Identification of corresponding concepts is simple in this case, e. g. when searching for *car* all the instances are dispalyed.

b) Different names for one and the same concept is not very easy to handle as the naming often serves as the identification for the described phenomena. In different groups the terminology unfortunately differs and also time affects language-use and the naming of concepts. A way to handle this is with synonyms that can even be authorized within a universe of discourse.

The retrieval of concepts with synonyms can serve as an adaption with a change of business´ terminology. It could be very fruitful letting the users access with their own language, e. g. the user can only think of the naming *auto* when interacting with the repository. That name might not even be in the repository but using the information about synonyms to *auto* leads to that *car* is found which is present in the repository and graphically displayed with a context from a model.

c) In different models and applications different names could be used for what could be seen as corresponding concepts, with similar structure and meaning. This is a major problem for integration. With the use of synonyms distribution of a concept like this could be discovered. When working with quality of models the naming of concepts is checked as a criteria for better quality, where discovering a corresponding concept with different names could be useful.

d) Viewing related concepts is essential when dealing with large information stores and e. g. different models. The hyperonomy relation is hierarchic between concepts, and it could in combination with synonym handling display some implicit relations between concepts. Searching for *Jeep* could lead to existing information about *vans* and *wagons* for example.

e) Homonyms, i. e. when the namings of different concepts appear to be the same, lead to difficulties in identification and retrieval especially when dealing with many models. This is important to recognize so that concepts are not confused, especially when integrating models etc.

f) The same naming of different phenomena that don't have the same categorization could be an example of homonymy but it need not be. Nevertheless the different uses of the same name should be identified , displayed and organised.

3.2. Syntax

Syntax deals with the relation between words, their categories and linearisation but also hierarchical structures. A grammar could be seen as a kind of knowledge representation of certain aspects of what we know about language and are able to explicitly express in a formal way to be understood by a machine (Gazdar and Mellish 1989). An automated syntactic analyser(parser) is an algorithm that takes a grammar and a lexicon together with a string and returns the syntactic structure of that string.

People seldom speak in a syntactically correct[2] manner neither do they write correct grammatically. Trying to parse a text from a newspaper or write a grammar to handle the sentences in a normal conversation is a very hard problem. Nevertheless people should have to violate the grammatical rules badly to not understand each other. The phenomenon of long ill-structured sentences are sometimes called *noisy sentences*. Since the structure of noisy sentences are hard to describe in syntactical rules we instead have to rely heavily on semantics. This is the case when trying to extract something from a text as well as in the communicative situation human-computer where we have to involve a great deal of pragmatics. Still the grammar is important to resolve ambiguities for instance.

Syntactic ambiguity gives rise to different meanings for the same string where different grammar rules could be applied for one and the same substring. An example of this is the attachment of prepositional phrases as either an attribute or an adverbial, as in *John drove*

[2] *Correct* is used here according to a normative viewpoint.

cars from the garage (either the cars are from the garage or the garage is the starring-point for John´s driving)

In NL-applications, syntactic ambiguity could be reduced or even eliminated by restricting the possible language constructions that can be expressed. One way to do this is by composing a query by selecting subconstructions from dynamically generated menues of what is alloud to express syntactically (or semantically) as in NL-Menues (Tennant 1986). Parse trees as an output of syntactic analysis is used for further analysis of semantics etc. Syntactic and semantic parsing could also be performed in parallell. To further enhance the functionality and ease of use of a navigator for repositories a interface of NL-menu style should be implemented for composing advanced queries about the repository. Actually this could be an interface for several services around the repository.

3.3. Semantics

Semantic information could be structured in many ways. Selectional restrictions can be used for defining word sence but even for disambiguating a sentence. AI-techniques, such as semantic nets, primarily were used for defining word sence making with hiararchic structures using the relations **is** and **has** (Quillian and Collins 1969) and has much inspired conceptual models. A conceptual model could be seen as a semantic representation of a domain. In a NL-interface the semantics of the possible queries and phrases in natural language must be mapped to the semantics of the application. To do this mapping we must use the meta-model together with some additional knowledge. Thus the meta-model will be represented as part of a knowledge base for this purpose.

3.4. Pragmatics

Language use and linguistic- and non-linguistic context are concerns of pragmatics. The language that a group of people within a business most certainly differs from language used by other groups outside and even within the business. Certain terms have a special meaning depending on context. Background knowledge in general but also about a domain, about language and the speaker is essential when understanding language.

To deal with these aspects a system for ultimate usefulness must be cooperative. A system is cooperative when it tries to actively understand the users´ intentions and help to solve their problems. This includes to allow as free formulation of queries as needed and high acceptance of user´s requests besides proper response to every situation that might occur. The ability to create a discourse evolving as the query session goes on is a crucial feature which simplifies and enhances the usability of the system. Especially in a navigation situation where a continuous dialogue might occur until the user arrives at the target. Another very important aspect of cooperation is the presentation of the answer that should be ultimately done in an appropriate medium and format. This demands for a close integration of different medias, especially between graphics and text.

4. A Repository-Navigator based on Word Knowledge

Our prototype is built on the platform Macintosh, Prolog and Oracle relational database. This platform might be changed as the project goes on to workstation/unix or PC/OS2.

4.1. Query-model

What queries are interesting to state to a repository ? We have to build a model of queries that we like to handle. At this first stage we will focus on the lexical issues. Queries about conceptual relations like synonymy and hyperonymy will be maintained. Our plan is then to extend these queries and involve some kind of NL-menu facility which will handle more free queries and also be able perform a dialogue in cooperation with the user.

The query model defines the syntax and semantics of valid questions to the repository. Through mapping concepts in the meta-model to natural language concepts it is possible to have a more developed semantic control of meaningful queries to the repository before any query is made. This will decrease the number of failed queries and also make queries more semantically correct thereby increasing the acceptance and credibility of the navigator. For example it has been defined that attribute types in the meta-model should always be verbs if in the query model , it would be meaningless to pose a question like "Which are the synonyms to the attribute type car". If it is possible to know, through extracting knowledge from the meta-model , that the only types of concepts that have synonyms are entities and attribute types, then a query like "Which are the synonyms for the relationship ownership?", would be meaningless.

4.2. The Meta-Model

The meta-model used is based on work at SISU and the Swedish Telecom in defining a reference model for repositories. This reference model includes different model types as: goal model, strategic model, concept model, process model and relational data model. We will focus on the concept model in the first version of the prototype and later extend it to the other model types.

In our case the meta-model itself includes phenomena like synonyms and corresponding concepts.

A model consists of entities, properties, domains and relations. Entities might have sub-entities and a domain could be composed of sub-domains. For each entity, property, domain or relation a set of synonyms could be defined in that model. An entity must have an administrative role, i.e. occur in a model in the repository.

In the meta-model you have defined the type of concepts that are valid, the relationships between these concepts, attributes to concepts, the mapping between concept types, concepts and their attributes, etc. The meta-model specifies what kind of information the repository could contain and is used together with additional knowledge to link queries to the repository in the navigator.The benefits of using knowledge about the meta-model is that it becomes possible to make the navigator more independent and adaptive to the syntax of the database schema. Using a meta-model also makes it possible to hide the details of a particular application for the user.

4.3. Systems architecture

In the lexicon based navigator we have developed an interface based on dialogue-boxes and menues. This part of the navigator is based on queries on single words using different types of lexicons.

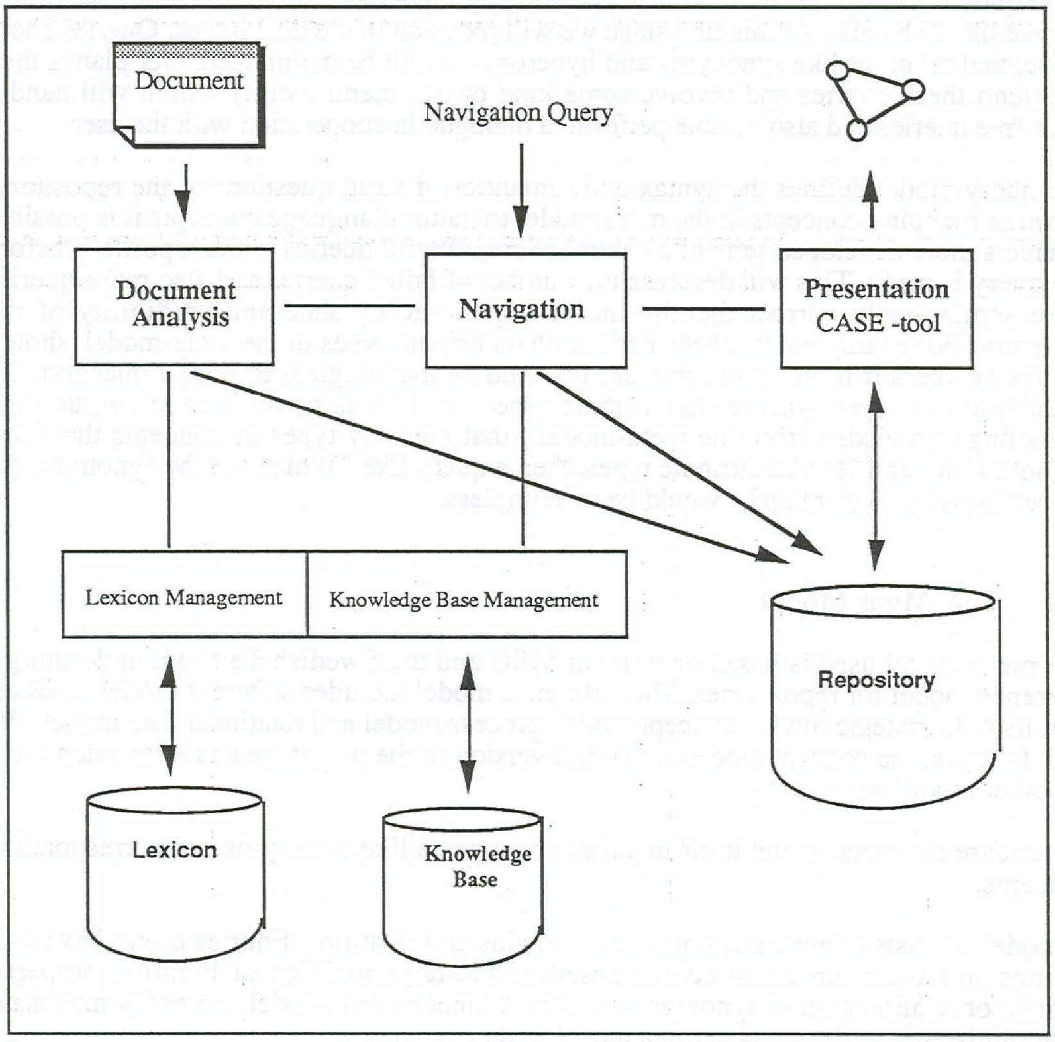


Fig. 5. The prototype architecture

Lexicon

A lexicon for general terms, a lexicon for domain-specific terms and a user-specific lexicon could be used in aggregation. The information stored in lexicon is word-forms, morphological and syntactic information, semantic information and synonyms. Morhological information is used for different word-forms and derivation of those. Syntactic and semantic information is helpful in deciding a conceptual correspondance in the repository-information. The synonyms serve as information-retrieval paths to the repository.

The domain-specific lexicon include terms related to the repository while a general lexicon is domain-independent. Swedish machine readable lexicons that could be used are The Swedish Academy Glossary (11th edition 120 000 head words), Lexical database (160 000 lexemes), Svensk ordbok (60 000 head words)(Gellerstam).

Lexicon management
The system-part handling lexicon management should be able to collect available synonyms and related concepts and pass on to a search through the repository for information. The user could make a choice if he does not want to use synonyms or where they should come from (the repository, his own lexicon or a general lexicon).

Also concepts higher in a concept hierarchy according to different sources should be used. In their turn synonyms could also be used etc. until the goal is reached.

Different lexicas in the prototype should be updated differently. The business specific lexica reflecting lexical information about concepts in the repository should only be updated by authorised persons. A general lexica could be accessed by several persons but control routines helping one or two responsible persons to look through recent updates should be available. Individual lexicas should be private for each user.

A special updating facility with a Lexicon Interface assists in adding the required information. The facility generates suggestions to spare the updating person unnecessary typing.

Knowledge Base
A knowledge base contains the repository organization but also some pragmatic knowledge about retrieval and cooperative behavior from the navigation system.

Document-Analysis
With the lexical, semantic and syntactic knowledge in the prototype, suggestions for concepts could be given starting from a text. Using a syntactic-semantic grammar for analysis could help in identifying structure and related concepts in the text. Using existing models assists the search for important concepts. The occurances of different terms used for one and the same concept could be identified in the document and high-lighted for comparison with a concept context from the repository, e. g. model fragments, mainly with the use of lexical and conceptual knowledge.

Repository
The repository is a result of a project in the Swedish Telecom company, dealing with information administration questions. The prototype repository is implemented as an Oracle relational database with a schema in accordance with the meta-model described earlier.

CASE-tool
Repository information could for example be presented as tables, in natural language or in graphics using a CASE-tool.

Navigation
The navigation makes use of linguistic (lexical) knowledge together with knowledge about the meta-model (and the repository) plus other knowledge from the knowledge base to access the repository according to the users´ requests.

4.4. Description of query processing

Query processing in the navigator is scetched in figure 6.

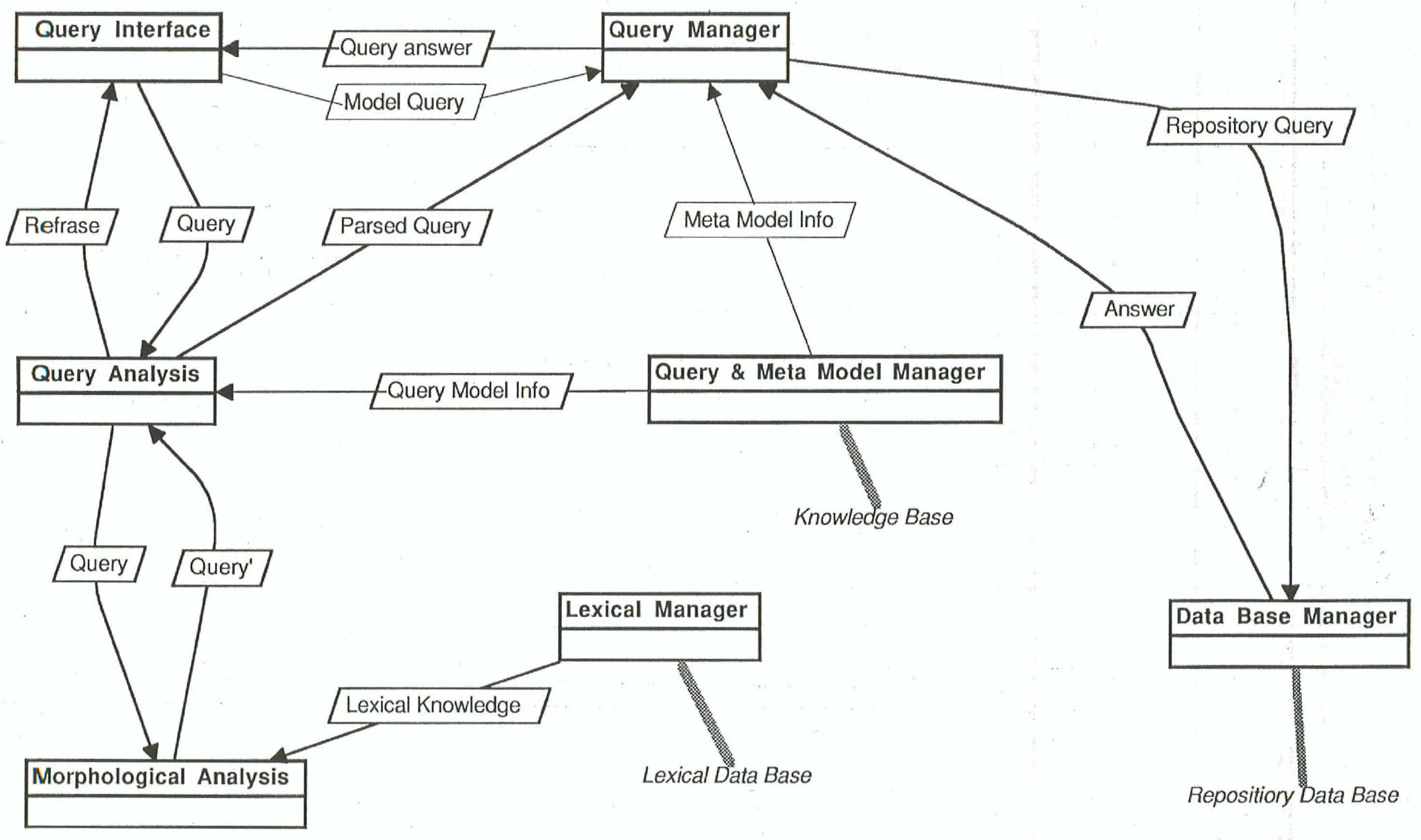
Query interface
In the query interface the user has several ways of initiating a query. One way is to use a document, for example a design document or a policy document. The user can load this document in a suitable format into the navigator. By clicking on words in the text the navigator could be invoked. Another way is to use the pull down menus to build up a query of some kind.

Using the knowledge in the meta-model the user is presented with the types of concepts that exist in the application, and if wanted the instances for each type of object given a certain application, model or project. In the lexical navigator the query will consist of single words.The user can specify which types of lexicons that should be used and their names. These words are sent to the query analysis.

<u>The Query Analysis</u>
The Query Analysis makes an analysis of the query using morphological analysis and the query model to sort out queries that are not sematically valid. The morphological analysis uses lexical knowledge from the Lexical Manager. The Query & Meta Model Manager uses knowledge from the meta model and the mapping between language and meta model to make a semantic analysis of the query. The result of the query analysis is a parsed query to the query manager.

<u>Query Manager</u>
The Query Manager uses knowledge in the meta-model to construct a Data Base Query to the Data Base Manager. The Data Base Manager returns the answer of the query to the Query Manager that sends it to the Query interface where presentation of the answer to the user is prepared.

Process Model of the Lexical Navigator



5. Conclusions

We have sketched a tool that supports navigation in repositories as well as comparisons of different objects and models. Using linguistic knowledge in this way seems to be a rather new and fruitful approach to ease access of stored business information with individual considerations. A repository manager should be helpful for developers, administrators and people constructing and using models, when doing view integrations, checking for naming conventions and naming conflicts, reusing of models and definitions, etc.

The prototype that we are developing, using conceptual models for a start, must be constructed to meet the needs of these different categories of users and different purposes.

The first phase of prototyping will focus on the use of word semantics and lexical knowledge such as synonyms to achieve our goals. Further work will focus on including a larger query model with advanced queries stated in a natural language menu interface.

Further research is planned to adopt the same approach to a repository with different kinds of models as for example goal models, rule models and process models. The functionality and the ability to make more complicated navigation requests by means of a NL-menu interface will also be further investigated. More attention will also be paid to dialogue handling and presentation of answers.

Bibliography:

Amble, T., E. Knudsen, A. Lethola, J. Ljungberg and O. Ravnholt. (1989). Naturligt språk & grafik - nya vägar in i databaser. Statskontoret.1989:17.

Cauvet, C., C. Proix and C. Rolland. (1988). "Information systems design: An expert systems approach." IFIP working conference , The role of Artificial Intelligence in Databases and Information Systems. Guangzhou, China.

Forte, G. (1989). "A Mecca for CASE: All Roads Lead to the Repository." C/A/S/E outlook no 4.

Gazdar, G. and C. Mellish. (1989). Natural Language Processing in PROLOG - An Introduction to Computational Linguistics. Kent, Addison-Wesley.

Gellerstam, M. The language bank. Department of Computational Linguistics, Gothenburg.

Grosz, B. J., D. E. Appelt, P. A. Martin and F. C. N. Pereira. (1987). "TEAM: An experiment in the design of transportable natural-language interfaces." Artificial Intelligence. Artificial Intelligence, no. 32(2), pp.173-243.

Jacobs, P. S. (1990). "To parse or not to parse: Relation-driven Text Skimming." COLING. Helsinki.

Kersten, M. L., H. Weigand, F. Dignum and J. Boom. (1986). "A conceptual modeling expert system." 5th International Conference on ER-approach. Dijon, France.

Loucopoulus, P. and R. E. M. Champion. (1989). "Knowledge-Based Support for Requirements Engineering." CAISE 1989. Stockholm.

Lyons, J. (1977). Semantics. Cambridge, Canbridge University Press.

Quillian, M. R. and A. M. Collins. (1969). "Retrieval Time from Semantic Memory." Journal of Verbal Lerning and Verbal Behaviour,no. 8, pp. 240-247.

Reimer, U. (1990). "Automatic Acquisition of Terminological Knowledge from Texts." ECAI. Stockholm,:

Suitcliffe, A. and N. Maiden. (1990). "Assisting Requirements Analysis through Specification Resuse."

Tennant, H. R. (1986). "The commercial application of natural language interfaces." COLING-86, pp. 167.