A Linguistic Approach to Conceptual Modelling

Jeroen Hoppenbrouwers

Infolab/
Institute for Language Technology
and Artificial Intelligence

Tilburg University
PO Box 90153
NL-5000 LE TILBURG
The Netherlands

hoppie@kub.nl

Table of Contents

Abstract

Some information system development methods concentrate on the acquisition of domain (user) knowledge, and others on the formalisation of the already acquired knowledge. But no matter what approach is followed, the process of knowledge acquisition always includes extensive communication between information system specialists and application field specialists. This communication, however essential, often is considered to be a handicap. The various parties need to cooperate, but they have little language in common and their goals are quite different as well.

Our approach in information system development integrates the acquisition process with the conceptual modelling. We try to find a way in which a computerised lexicon, that stores natural language words and sentences in a formalised way, can help information analysts and domain specialists to communicate better. The lexicon will also provide checking and simulation facilities for the information system under development.

Since the method hinges around a lexicon that stores natural language, it seems appropriate to store not only system development information in the lexicon, but more general, organisational information as well. The common natural language that is being used extensively throughout the method opens up direct communication possibilities between parties that earlier on kept away from system development, and it might even lead to a better understanding of the business process itself. As such, the method can be seen as a powerful tool to support business process remodelling and simulation.

Project Context

This is one of the subprojects in the LIKE (Linguistic Instruments in Knowledge Engineering) research project, a collaboration of the universities of Amsterdam, Maastricht, Twente and Tilburg (all in the Netherlands), bringing various disciplines together. Computer science, linguistics, organisation science, and information science are the four cornerstones of the LIKE project.

LIKE currently is one of the final competing projects to get funding from the Dutch national research organisation NWO and we are currently waiting for the final decision which is expected early May 1994. If accepted, the project budget will be about $5M of which a considerable part will be provided by commercial institutions and companies.

1. Introduction

Many information system classifications divide the whole analysis and development process into several distinct phases. Especially the *(informal) requirements determination* and the *(formal) conceptual modelling* are explicitly separated. As [Wieringa 1993] says:

> *"A specification of Universe of Discourse (UoD) models consists of two parts, an informal part and a formal part. [...] The informal specification is needed to be able to understand what the formal specification says. It acts as documentation for readers of the formal specification. In addition, it is also needed as a stepping stone to the formal specification."*

This two-step approach, although certainly workable, has some disadvantages.

Firstly, it assumes that the informal natural language (NL) specifications are translated into formal specifications by a *specialist*. For the later phases of the system development process, specialist translation from one language to another is no problem, since the design and implementation languages are supposed to be complete and deterministic. In the first two steps, however, such a translation is far more sensitive to problems since the NL requirements might still have inconsistencies and other errors that will only come out during the translation to a formal language. This translation process thus will be a cyclic action with much communication between the people who (co)produced the informal specifications and the translator(s). Furthermore any update in one of the two descriptions inevitably must lead to a corresponding update in the other, which calls for a second translation and gives room for errors and inconsistencies, not to mention the burden of maintaining two separate documents.

Secondly, using natural language is not as informal as people might think. Language is always used in a well-defined context in which all communicating agents know the meaning of the used concepts. If this underlying knowledge is absent or wrong, NL communication is doomed to fail. If we start on a common knowledge level and base our descriptions on this common level, working our way up to tighter descriptions, we can leave out many formal ambiguity resolvers since the discussion takes place in a context that already resolves the ambiguity for us.

To avoid the disadvantages of separating the first two phases, we propose to use a method that *integrates* step one (requirements collection) and step two (conceptual modelling) by using a pre-existing formal *Lexicon* that contains many concepts that are used in normal, day-to-day communications. With this Lexicon, people can formally express the NL sentences they now write down as informal strings of characters.

Specialised versions of the Lexicon can then be constructed to cover specialised contexts, such as specific domains. These Lexicons will define specialised concepts in terms of more generic, common concepts, and thus enable people to quickly familiarise themselves with the domain. They also facilitate the development of cooperating organisations and information systems.

Note that we do not propose to *parse* NL specifications, since this would imply complete knowledge of NL structures, which we do not have at this moment. We want to use a very convenient tool to let people *interactively construct* their specifications with immediate feedback in NL rather than writing down informal NL and formalising it later.

As an important side-effect of having NL descriptions, many development and design documents that formerly were only readable and/or understandable by information specialists can now at least be read by non-specialists who are knowledgeable about the domain. NL is not always the most efficient means of communication, but given enough generic context, it might be the best way to communicate between non-specialist groups.

2. Project Assumptions, Definitions, and Goals

Our project is based on a few assumptions.

First, we use natural language (NL) extensively throughout the method without ever parsing NL descriptions. In this sense, our way of using NL is neither AI-based nor informal. We use NL as a communication vehicle between humans and machines (in all permutations), but it is always controlled and restricted, and accompanied by the formal description out of which it is generated.

Second, our view of the role that information systems play in organisations concentrates on the communication between human beings who work with that system. The system itself could possibly, in the most extreme case, have no autonomous behaviour at all. In this way, it could be possible to develop an information system that gradually can provide more and more automation as the context is better understood.

2.1 Human language and our world

People know about their world, in which they are used to work with real-life entities and common concepts that everyone in their environment agrees upon. Humans can only operate in our world because our mental model of the environment is both close enough to the physical reality to make valid assumptions about behaviour of objects in that physical reality, and because most humans in a certain environment share a very large part of the same mental model.

When humans communicate, they exchange physical phenomena (sound, writing) that are intended to partially modify the receiver's model of the world to bring it in accordance with the sender's model. This clearly requires both models to be already very similar before the communication takes place. Viewed as such, communication only is the exchange of deltas on mental models (see also [Sperber 1986]).

The communication between humans about their world takes place in a language that is known as common natural language. This language is a very complicated and powerful system of concepts, syntax, and words. But since it only carries deltas on a mental model, it is heavily dependent upon the *context* of an utterance. An utterance that takes place in a football match context may be complete nonsense in a restaurant context.

The extreme context-sensitivity of natural language often leads to the conclusion that natural language in itself is ambiguous, and therefore unfitted for any formal or technical communication. But if natural language (NL) indeed was ambiguous, human beings would encounter extreme problems in communicating with each other. However, daily practice shows that the vast majority of all communication succeeds, which strongly suggests that NL is not ambiguous at all.

2.2 Human language in current methods

Information engineering methods usually aim at a formal description of the relevant domain, so that a mechanised system can be built that conforms with that domain. When a project to build such a system starts, it often is not yet clear how the domain should be formally modeled (with the possible exception of technical/physical domains that are already analytically solved, and completely formalised bureaucracies). Conceptual modelling methods provide a way to find such a formal model.

Among the most often used conceptual modelling methods are entity-relationship (ER) modelling [Chen 1976, Batini 1992], binary/*n*-ary object-role (OR) modelling such as NIAM [Nijssen and Halpin 1989] and NORM [De Troyer 1991], and object-oriented modelling [Rumbaugh 1991, Coad and Yourdon 1990, Shlaer and Mellor 1988/1992]. These models share a rather large common part, with models such as NORM explicitly bridging the gap between OR and OO.

Of these models, especially NIAM has a considerable NL part that enables the user of the method to (partially) model the domain in the domain specialist's own familiar jargon. Nijssen's recent Dutch publication [Nijssen 1993] and Dietz book on DEMO [Dietz 1992] both stress the importance of using NL and the domain specialist's own terminology in conceptual modelling. We share this view.

Still, even these methods that specifically use NL during the information requirements collection do not use NL during the actual conceptual modelling, which results in many technical schematics. Reading these schematics always requires a fair knowledge of the method, and any remaining linguistic information is effectively erased at the drawing board. Much effort could be saved if the conceptual models could retain the linguistic information that had been present in the earlier stages.

On the other hand, specialised information requirement collection methods such as ISAC's analysis of change [Lundeberg 1981] and many business-oriented approaches [Rockart 1979, Porter 1980/1985]] have almost no formal component and their descriptions are completely in natural language, possibly with the addition of simple diagrams to show some relationships.

[Wieringa 1993] divides requirements specification documents into three parts:

1. A *statement of the purpose* of the required system;
2. A *functional requirements specification*, or the necessary behaviour;
3. A *non-functional requirements specification*, or additional constraints on the implementation of the system that improve the usefulness of the system.

Of these three parts, the functional requirements specification will be developed into the conceptual model. To our knowledge, only NIAM gives a reasonably complete method to perform this translation from NL to a formalism (for the static part), and no method offers true integration of the requirements collection and the conceptual modelling phases. Some other methods have been described [Auramäki 1988, De Antonellis 1984] that *do* use natural language to conceptualise an environment, however they are not so well-suited to the current system development practices.

Other approaches try to capture as much of the context of a certain situation as needed to weed out ambiguities, and express this knowledge in formal terms, often based on logic. Basically these methods break down the information contained in natural language into the smallest possible parts that have completely defined semantics. In many cases this leads to an extensive collection of common knowledge, mostly needed to represent the common part of most people's mental model. However, because of the vast amount of knowledge that we accumulate during our life, we think these projects are doomed to fail unless we can build a machine that autonomously learns and stores such knowledge.

2.3 Human language as information carrier

If we try to model an organisation of human beings, we could decide to leave out as much as possible of the context-sensitive knowledge and concentrate on the *communication* between humans. Communication (the NL utterances, or written presentation of the equivalent) as argued above does neither contain any knowledge nor context information. Communication in itself is not a carrier of knowledge. It is, however, a carrier of information *about* knowledge [Dietz 1992]. Such information added to existing knowledge can generate new knowledge. Reasoned the other way, it is impossible to know what a given example of communication is about unless the major part of the necessary background knowledge is already available, an accepted view in the Artificial Intelligence and Machine Learning world.

By placing the necessary background knowledge in the communicating agents and taking it completely out of the communicated utterances, we simplify the representation problem. If we have a way of monitoring and storing the messages between agents in a certain context, we capture the whole information flow in that context. The catch is that this information will only be meaningful if it is replayed, as it were, in exactly the same context.

This is not too different from our real-world experiences in which we recognise phenomena such as jargon. In any context that hosts a fixed group of people who work closely together, a sublanguage of the common NL will appear that after a while becomes so specialised that people outside of the context have difficulties to understand it.

We argue that building a monitor and storage system for context-sensitive communication is a more generic, and in some cases better approach than either the formalised common knowledge projects (the context-free approach, trying to describe the world in the smallest possible primitives) or the artificial generic context projects (using a formalism tuned to system development). Common knowledge is too vast to capture successfully without fully automated learning machines, and artificial context approaches often fail because such a formal context either misses the point by being not powerful enough to capture the domain or is not well enough understood or too complicated to use by (some of) the communicating agents (usually the domain specialists).

This is not to say that the common knowledge principle is not useful; capturing the background knowledge together with the communications has the enormous advantage of giving the whole integrated world model. For some tasks the availability of such a model, which can be used to reason about the context and to plan actions to control the future, is critical. But if we cannot build such a complete model, and in many cases we indeed cannot, keeping the world knowledge out of the system and only using the system for facilitating the communication between agents in that system may prove very valuable as well. If we have a working system to support the communication between agents, we can relatively easy formalise and automate isolated parts of the system. The communication system then functions as a base layer to build information systems on.

Viewed from this point, the current practice in organisation theory and information system development is to educate the consultant or analyst to fully understand the client's domain and to formalise it to the greatest extent possible. An analyst is like a naval pilot who knows his environment inside out, but needs re-training for every new harbour. But an analyst is not a genius who can learn all things quickly and efficiently, and not all systems are completely deterministic in nature. We propose a method in which the analyst does *not* need to become a domain specialist for every new project. We want to keep the great bulk of domain knowledge where it already is, namely with the domain specialists, and transform the role of the analyst into a scout or cartographer who guides domain specialists through the forest of their own world, drawing understandable maps for them on the fly, but not rebuilding their familiar world.

2.4 Communication as an organisational tool

If we can produce a *communication chart* of an organisation, someone who is knowledgable about this organisation and a part of its context can understand the messages that flow between the humans in that organisation. This gives a good overview of the business processes in that organisation and may lead to insight in possible problems and improvements.

The base of a communication chart is a grid of communicating agents, which can be individuals or functional units. In this respect, the fundamentals of communication charts might be organisation charts. However, the links between units are not representing the lines of command or the reporting structures, but the routes messages between the units follow. Naturally the original command and report lines will be part of the chart, but many more links may be present to account for all the communication between all units.

While normal organisation charts only show the *formal* communication, the communication chart also shows the *informal* communication. This implies that we need a way to discover informal communication structures, which traditionally has been one of the hardest tasks for any organisation consultant. We assume that the major reason for the difficulties encountered during this kind of investigation lies in the fact that informal communication can only be pinned down if the analyst knows the organisation inside out, which in turn is only possible if (s)he has been working in it for a long time.

Our approach lets the people who work inside the organisation *themselves* chart the communication lines. Although this does not guarantee that we also find (all) the informal links, as many people may tend to idealise their world and write down only the communication that they think *should* be present instead of what actually *is* present, it probably works better than if a complete stranger has to model it. Typically external consultants talk more to upper and middle management than to people at the work floor, and therefore they may miss essential informal communication much easier.

Of course, a bold enterprise like this can only succeed if we have very easy to use tools at hand to give to the people. Ideas for these tools will be described further on in this paper.

2.5 Communication as a base for information technology

In the same sense as with organisations, communication charts are powerful platforms to build information systems on. Such an information system would not be a complete (deterministic and formal) *monitoring and controlling system*, but a far less formal enhanced *communication support system.*

This division in two types of systems produces interesting results.

For those environments where we actually can build a complete semantical model, we can construct information systems that *replace* humans and do their tasks better, faster, and cheaper, but also with less flexibility. These systems could be called *quantitative* systems, and they are typically found in technical environments where humans either are too slow or too error-prone to control a certain process, or in completely formalised bureaucracies.

When we cannot or do not want to implement a fully formal model, and therefore keep humans in the system to take the final decisions, we can build a communication system that only *supports human communication.* Such a system could be called *qualitative*.

Note that we do not want information illiterates to describe their daily tasks in any kind of formal or graphical language. Our long-term goal is to capture their mutual *communications*, which are mostly in natural language. With this knowledge of communications, we could create a *communication support system* that at first would not add any value to the organisation except for the formalised communication *channels* (which is not formalised *communication*). For some parts of the organisation, such a system would be valuable exactly *because* the communication is now captured and can be archived and (re)distributed more easily than e.g. telephone calls. The current *groupware systems* often fall in this category.

After creating such a communication support system, it is relatively easy to start formalising and partially mechanising the behaviour of *isolated agent pairs* whose agents mutually exchange messages. Since these pairs are isolated, replacing their communication link by a completely deterministic algorithmic procedure will not in any way disturb the communication between all other agents in the whole organisation. The two communication agents will not disappear, they will only replace a tiny routine part of their activity by an information system, and therefore have more time to react to non-routine problems.

Since in this way it is possible to automate the best-described or most-routine parts of an organisation first and later on start work on the less deterministic parts, possibly skipping seldom used communication links and leaving them to humans, we think that such an approach can prevent many of the current problems in information systems development and organisational implementation. After all, only a very small number of organisations can work without any human intervention.

2.6 Natural Language and information systems development

As stated above, natural language is the common glue of all communication. For specialised contexts, other forms of communication such as graphics or symbolics certainly are useful as well, but generally speaking the major carrier of information is always natural language. It is widely believed that *any* piece of information can be expressed in natural language, even if that would not be the most *efficient* way of expressing it [Dietz 1992].

Since we argued that we do not need to understand NL utterances in order to build a useful communication tool, the usage of NL as a major communication vehicle in information systems development becomes possible. If we succeed in capturing the *type frames* of NL utterances that are communicated between the agents in a certain context, we can create a blueprint of most communications.

Normally, people speak in terms of *examples* while information analysts speak in terms of *families of examples*. During their work, information analysts create frames in which the users' examples fit. If a communication and information system has all the frames available to cover all the users' daily examples, we can consider the system to be complete. It then will be able to carry and store all communication between the users of the system.

We do not claim that such a complete communication support system can be built with the currently available tools. Especially informal documents like strategy policy studies cannot yet be generalised to a type *strategy policy study* with certain attributes that need to be filled in to create all possible studies. However, an important part of the communication in organisations is of a far less complicated type, and it is this kind of information that we want to describe.

The central repository for these natural language expressions we call the *Lexicon*. Besides entity definitions, it can contain *frames* (sentence types) to describe the information system, and it can generate *examples* out of these frames to validate them. It does not, however, store sentences as plain strings, nor as symbolics. All information in the Lexicon is stored as relational information between concepts, with syntax and morphology generated as needed.

3. The Lexicon

Our Lexicon is the central repository of all terminology and related linguistic elements that we need to describe communication in and about our world. As such, it is much more than a list of terms. The Lexicon contains the concepts that make up our communications, and it defines the relationships between these concepts.

We assume that the Lexicon contains enough generic concepts to express the specialised domain concepts. We specifically do not want to go "down to the bare bones of semantics" and express every concept in the smallest possible primitives. As argued above, most if not all humans share a large part of the knowledge they have about the world, and our Lexicon should not go "deeper" than the most specialised knowledge level that still is shared by all the users. If necessary, extensions can always be added, and in the most extreme case we indeed find some primitives that can be used to express the whole Lexicon in. But for the time being, only the concepts just below the "jargon" level are necessary, and they will not be defined themselves.

3.1 The architecture of the Lexicon

All entries in the Lexicon, entities as well as relations, are placed in one ISA hierarchy and therefore there is one common root object called *Entity*. The base entity object has some properties that make it suitable to represent most real-world entities that are *things*.

A subclass of this *Entity* root class is the class *Relationship*. Relationships are specialised entities that can hold pointers to one or more entity classes.

The *Entity* (and thus also the *Relationship)* classes do not have names or other meaningful references. They have only a system ID that can be used to construct links between classes. If we want to give an entity class a name, we have to link it to an entity subclass called a *Lexical*. Lexicals are specialised objects (*Lexical* class instances) that have a small sense of morphology. The *Lexical* class has two subclasses in the standard Lexicon: *English noun* and *English verb*. As the names imply, the Lexicon is multi-lingual in nature: we can link various lexicals to the same entity.

If we want to give a name to an entity class, we link it to an object of the type *Noun* (or *Verb* if we have a relationship at hand). From then on, the Lexicon is able to use a linguistic expression to represent the entity or relationship to the outside world. Only by means of the linked lexicals we can communicate with the Lexicon.

In this way, we explicitly decouple linguistic information from the stored concepts, and open up the possibility to attach more lexicals to the same concept (synonymy) or more concepts to the same lexical (homonymy). Also note that we do not in any way store semantical information except for the fact that relationships have specific entity classes to work on.

There have been many different storage hierarchies proposed to date, and it seems that no hierarchy can fulfil all wishes of all users. Therefore we designed the Lexicon in such a way that it does not imply a specific hierarchy, and that it supports many hierarchies at the same time (using multiple inheritance) so that an entity can be in more ontology trees.

Additionally, we propose at least one "base" hierarchy that is solely based on economics: a strict ISA structure with entities deeper in the tree having more relations attached to them as their parents. The general idea is that if an entity can be used in a certain relation, all the entity's children can be used as well.

3.2 Translations to Natural Language

In order to represent the Lexicon's contents to the outside world, it needs to be able to produce a linguistic expression for each object stored into it. Each object in the Lexicon knows (by means of its implementation type, and only indirectly by its place in the hierarchy) how to properly express itself in one or more languages. For example, if we send the object that represents the *horse* concept the message that we want an English sentence, it will follow its link to the lexical (English noun) "horse" and produce the sentence "There are horses", implying that the *horse* class can possibly have many instances.

Relationship objects have a verb object attached, and if the object is viewed without any attached entities (a zero-place relationship, so to say), it will produce e.g. "To transport" without any clue as to its meaning. If we have attached one entity class, the sentence turns into "Horses can be transported"—the word *can* implies a possibility, not the necessity that horses are always transported. We use passive voice to indicate that this relationship logically is a *property* of the horse. Naturally many properties can better be represented by adjectives, so for most one-place relationships it would be preferable to attach an adjective lexical to it instead of a verb.

Two-place relationships produce sentences of the kind "People can ride on horses", with the passive voice variant "Horses can be ridden on by people". In many cases this passive voice may seem artificial, since many verbs have a special "passive voice" verb attached (*to work for* versus *to employ*, or even more salient cases like *to buy* versus *to sell*). When a generated passive voice expression seems inappropriate, the "reverse" root form can be included in the verb lexical just as the "normal" root.

Higher-place relationships are also included, but they are not described in this paper.

3.3 Covering static and dynamic descriptions

Use of fixed entities and relationships is not enough to fully describe a system. It might very well be possible that people can ride on horses, but normally only one person can be on the back of a certain horse at the same time, and usually a person cannot be on more than one horse at the same time.

We want to restrict the frame *people can ride on horses* so that the actual communications we store in this frame (such as *person Karl rides on horse Adam*) are restricted in the sense above. This means that first the system would not allow both *person Karl rides on horse Adam* and *person Karl rides on horse Strat* at the same time, and second that this constraint is made explicit and expressible in natural language as well. To accomplish this, the simple frames could be turned into *Each person can ride on only one horse at the same time* coupled with the passive voice *Each horse can be ridden on by one person at the same time* (or an equivalent with a different verb lexical). We use a subclass of *Relationship* for this, called *Event*. For the origin of the term *Event*, see [Weigand 1989].

What cannot be described with these static structures is the dynamic behaviour of the world. We argued that for many systems, formalising of this dynamic behaviour is not possible or desirable, but if we want to replace certain human actions with algorithms, we need dynamic constraints. Note that these constraints do not need to be present in the initial stages of the model—only when we want to implement behaviour. In the next paragraph the dynamic constraints and triggers will be explained.

3.4 COLOR-X and its translation

[Dignum 1989] has developed a conceptual language, founded on Dik's Functional Grammar [Dik 1989] and several logics, that seems especially appropriate to describe various constraints and triggers. CPL itself uses a rather complex and extensive syntax, so it is not useful for direct usage by users or even analysts. But since CPL is based on Functional Grammar, it is straightforward to translate CPL expressions into natural language. Therefore it is relatively easy to develop an interactive CPL editor that gives immediate feedback in NL as to what exactly is described by its syntax. We use CPL to capture the formal contents of "dynamic" linguistic utterances.

In our examples, we will use a dialect of CPL called COLOR-X, developed at the Vrije Universiteit Amsterdam in the context of the LIKE project [Burg and van de Riet 1994].

An example sentence and its COLOR-X translation (with a slight syntax modification):

If a user does not return a book after borrowing it for one week, the library sends him a reminder

NECESSARY: ACTION:
send(**agent**=L in library) (**patient**=R in reminder)
    (**receiver**=U in user) (**temp**=T2 in time)
    (**id**:T2=T1+1*week)
    (**situation: ACTION**: not return(**agent**=U in user)
        (**patient**=B in book)(**temp**=T1 in time))

In a real-world library, the people working in it will have the above rule laying dormant in their context model. Although the rule probably is explicitly stated somewhere, in normal operations nobody would browse through the Library Rule List to see if there was a rule applicable to a book being lent for more than one week. We therefore can safely assume that in *normal* operations the rule itself is never communicated between people. What *is* communicated is the fact that a certain student has borrowed a book for more than one week, which triggers some action from the "overdue" department.

The communication system should be able to store a sentence frame of the type *People can borrow books on dates*, to capture the beginning of a borrow period. An instance of this sentence frame is generated when someone borrows a book (*student Smith borrows book DG554-A on April 5th*). The triggered sentence frame would be something like *Libraries send reminders to people*. In the simpler communication-only systems, this is enough, since with such systems the agents will initiate the second message themselves without system intervention. Should we want to automate the trigger, we need to formalise the event (which is fairly simple in this case).

Our target is to couple the simple sentence frames so that the second frame is automatically instantiated after the instance of the first is one week old and there is no fitting instance of *People return books on dates*. For this trigger action, it is not required to know the semantics of either of the two frames. Only the existence of an instantiation of the first frame and the fact that one week has passed since then is enough to trigger an instance of the second frame. Note that we do not want to say that COLOR-X *does* require the semantics of both sentences to be known, because it doesn't.

We can create another relation in the Lexicon that connects both frames and that reads *IF a person borrows a book on a date AND NOT if a person returns a book on a date AND one week has passed since date THEN the library sends a reminder to a person*. Naturally we need to make sure that the same *person* is used in all three sentences, that the dates correspond etc. COLOR-X is expressive enough for many, if not all, of these common constraints and triggers.

As can be seen, many COLOR-X primitives are directly useable in the Lexicon (such as necessity, action, and situation) while the connection to existing sentence frames provides us with enough linguistic material to express the whole trigger or constraint in natural language. Please note that our NL utterance consists of the original frames (sometimes made definite) coupled together by the constraint, while the original COLOR-X NL utterance has been tailored to the formal expression. Their meaning however is exactly the same.

A trigger of this kind may be implemented as a special case of a generic *Trigger* class, with a limited number of switches to configure it, and still giving direct NL feedback to the user.

3.5 The Lexicon Management System (LMS)

Our Lexicon can only be of any use if it is implemented on a computer system and can be easily accessed. We propose to split the Lexicon in at least two different parts that are separately implemented (possibly on different platforms) and communicate through a standardised protocol. This client/server architecture (the Lexicon being the server) opens the possibility to maintain a central Lexicon that can be accessed by various applications, even at the same time. It might be possible to distribute the Lexicon over more than one server, but our investigations did not yet explore these ideas.

A prototype system exists that can store most of the concepts proposed above. It uses a human-readable interface language that can easily be parsed and generated by machines as well, and that is expressive enough to embrace all Lexicon constructs. The syntax and semantics of this language, together with the design of the LMS, will be described in another paper.

References

Auramäki, E., E. Lehtinen, and K. Lyytinen (1988), A Speech-act based Office Modelling Approach, *ACM Transactions on Information Systems*, 6, 2

Batini, C., S. Ceri, and S.B. Navathe (1992), *Conceptual Database Design: An Entity-Relationship Approach*, Benjamin/Cummings

Burg, J.F.M. and R.P. van de Riet, (1994), *COLOR-X: Object Modeling profits from Linguistics*, working paper LIKE work group, Vrije Universiteit, Amsterdam

Chen, P. (1976), The entity-relationship model — toward a unified view of data, *ACM Transactions on Database Systems*, 1:9-36

Coad, P. and E. Yourdon (1990), *Object-oriented Analysis*, Yourdon Press/Prentice-Hall

De Antonellis, V. and A. Di Leva (1984), *DATAID: A Database Design Methodology*, IEEE Tutorial Notes 3

De Troyer, O. (1991), The OO-Binary Relationship Model: A Truly Object-Oriented Conceptual Model, in: *Lecture Notes in Computer Science 498: Advanced Information Systems Engineering,* eds. R. Andersen, J.A. Bubenko jr., and A. Sølvberg, Springer-Verlag

Dietz, J.L.G. (1992), *Leerboek Informatiekundige Analyse*, Kluwer Bedrijfswetenschappen, Deventer

Dignum, F.P.M (1989), *A Language for Modelling Knowledge Bases. Based on Linguistics, Founded in Logic.* PhD thesis, Vrije Universiteit, Amsterdam

Dik, S.C. (1989), *The Theory of Functional Grammar. Part 1: The Structure of the Clause*. Foris Publications, Dordrecht

Lundeberg, M., G. Goldkuhl, and A. Nilson (1981), *Information Systems Development — a systematic approach*, Prentice-Hall

Nijssen, G.M. (1993), *Universele Informatiekunde*, PNA Publishing, Beutenaken

Nijssen, G.M. and T.A. Halpin (1989), *Conceptual Schema and Relational Database Design*, Prentice-Hall, New York

Porter, M.E. (1980), *Competitive Strategy: Techniques for Analyzing Industries and Competitors*, The Free Press

Porter, M.E. (1985), *Competitive Advantage: Creating and Sustaining Superior Performance*, The Free Press

Sperber, D. and D. Wilson (1986), *Relevance, communication and cognition*, Basil Blackwell, Oxford

Riet, R.P. van de, and R.A. Meersman (eds) (1992), *Linguistic Instruments in Knowledge Engineering*, North-Holland, Amsterdam

Rockart, J.F. 191979), Chief Executives define their own data needs, *Harvard Business Review*, 57(2):81-93

Rumbaugh, J. et al. (1991) *Object-oriented modeling and design*, Prentice-Hall, New Yersey

Shlaer, S. and S.J. Mellor (1988), *Object-Oriented Systems Analysis: Modelling the World in Data*, Prentice-Hall

Shlaer, S. and S.J. Mellor (1992), *Object Lifecycles: Modelling the World in States*, Prentice-Hall