Automated Support of the Modelling Process:

A view based on experiments with expert information engineers[1]

G.M. Wijers[2,3] *and H. Heijes*[2]

[2]Delft University of Technology
Faculty of Technical Mathematics and Informatics
Department of Information Systems
P.O. Box 356, 2600 AJ Delft, The Netherlands

and

[3]SERC
P.O. Box 424, 3500 AK Utrecht, The Netherlands

Abstract

A trend can be discerned clearly indicating that current developments in methodologies concentrate on an ongoing structuring and integration of modelling techniques. Automated support is the prime mover of this trend. In this article it is argued that a lack of explicit knowledge about the process of model construction has resulted in problems with CASE-tools concerning adequate support in the form of verification and navigation. We present a view on modelling processes which has been applied in experiments with expert information engineers. The key concepts of this view are strategy and natural level of consistency. It is argued that adequate support can be realized for navigation and verification on the basis of (1) knowledge about tasks and decisions part of a strategy and (2) knowledge about the natural level of consistency of tasks. The experiments with expert information engineers have been performed as part of the realization of a knowledge acquisition approach for modelling knowledge. From each expert we have been able to extract detailed knowledge about the modelling process studied. The last part of this paper describes an architecture of modelling support systems, as well as a prototype that is capable of supporting the presented aspects of modelling processes.

[1]This research is partially funded by the Dutch User Group of Structured Development Methodologies (in Dutch the NGGO), BSO/Rotterdam AT, Hightech Automation, Nixdorf Computer and Unisys Nederland.

1. Introduction

Within the field of information systems no one doubts that it is irresponsible to develop systems on an ad-hoc basis. While many reasons exist for the interest in methodologies systematizing the development process, most of these are concerned with issues of quality of the information system and productivity of the development process (see also Blokdijk (1987) and Sol (1985)).

A trend can be discerned clearly indicating that today's developments in methodologies concentrate on a further structuring and formalizing of the necessary development products (Martin (1986)). The ultimate goal of these so-called engineering-like methodologies is to offer a coherent, integrated set of techniques that covers the entire development process.

Increased complexity of applications, improved verification possibilities, documentation, communication and standardization are among the explanations used for the above mentioned trend. In our view automated support is the prime mover of the ongoing formalization. Martin (1986) states that engineering-like methodologies could not have even existed before 1983 because they depend on automated tools.

According to Butler Cox (1987) and Yourdon (1986) experience with the use of structured techniques such as entity-relationship diagrams, dataflow diagrams and structured English has shown that them to be tedious and time-consuming, and even impractical if they are not supported by automated tools. It is our assumption that automated support will facilitate and increase the use of structured techniques in the first phases of the information systems development process.

Automated support tends to pay less attention to the modelling process than to the modelling techniques (Knuth (1986), Lockemann (1986), Potts (1989)). In particular (but not exclusively) in the first phases of the development process, modelling is a complex problem solving process in which iteration, incompleteness, heuristics, etc. are important characteristics. Today's tools do not support these characteristics explicitly, and for this reason they are merely used as drawing and reporting tools instead of as modelling support tools (NGGO (1989)).

A similar observation can be made on methodologies in general: although most methodologies prescribe tasks to be performed and the sequence of these tasks, very few methodologies offer detailed guidelines how to perform the various tasks and, more important, how to determine the quality of the development products, i.e. the models. Quality is dependent on the competence of the information engineer (Bubenko (1986)).

In our opinion, quality will not be achieved only by specifying the ultimate products. In particular, knowing where to start, how to continue, what to look for, and when to finish - in other words a clear strategy - will largely contribute to the quality of the information system and the productivity of the development process. In this article

we report on our study of the modelling process of experienced information engineers as part of an approach to increase existing modelling knowledge. A new architecture of automated tools is presented that reflects both the characteristics of the modelling process and the products of the modelling process.

The main purpose of this architecture is to reduce the dependency of the quality of information engineers, and to introduce more of their capabilities into a modelling support system. It is our philosophy that automated support should be a support environment in the sense that it supports the user in his or her modelling process. Support is achieved by offering guidance based on an explicit strategy as extracted from expert information engineers and by giving advice on (probably) correct or incorrect model structures.

The remainder of this article is organized as follows: In section 2 we introduce a framework for methodologies that we use to describe an approach for information systems development by distinguishing between a way of thinking, a way of modelling, a way of working, a way of control and a way of support. In section 3 we will use this framework to describe some of the problems occurring in the automated support of the way of modelling and the way of working. It is argued that these problems can be solved by a better understanding of the modelling process. Section 4 is related to our model of the modelling process that we have used in the analysis of the experiments performed to gain more insight in the process of model construction. Section 5 highlights some aspects of the experiments as they are performed with expert information engineers and in section 6 some major results concerning the modelling process are presented.

Section 7 includes the description of the new architecture of modelling support systems for which we are developing a prototype by now. At a global level the distinction between inference engine and modelling knowledge is important. Also, in section 7 we will describe the mechanism of the workbench shell in more detail. Finally, section 8 contains our conclusions, as well as some directions for future research.

2. Understanding information systems development methodologies

In the development of information systems, a major problem is that development is too complex to be handled straightforwardly. We need some means to handle this complexity. Constructing models of the problem constitutes such means. By the use of a model it is possible to focus on certain aspects of the development process. Modelling is a prerequisite for the information engineer to understand the area under consideration by making certain aspects of that area explicit in a simplified and understandable way. In the development of information systems there are many different aspects and interrelated (problem) areas which have to be dealt with. Thus, a series of models is usually needed to represent all these aspects within the

development process. This series of models is defined to be the way of modelling (i.e. the models which have to be constructed and their interrelationships).

2.1. Way of modelling

In all methodologies models are used: models of (part of) reality, models of the organization or business system, models of the information system (or parts of it) in its final form or in intermediate stages, etc. Some models are rather sketchy and informal, others are highly formalized. Some models may be introduced only occasion while little reference is made to them; others are used extensively. Some models are rather isolated, others are intricately related.
The way of modelling of a methodology describes the network of its models, i.e. the models, their interrelationships and, if present, a detailed description of the model components and their relationships. More specifically the way of modelling:

- describes the models used
- describes the components and their relationships within the models used
- describes the relationships between the models used

There are methodologies that introduce models with little instruction, other methodologies offer algorithms, or at least explicit procedures, to construct a specific model or to verify it. Yet other methodologies give informal but practical suggestions to obtain a model. Therefore it is sensible to distinguish between the 'way of modelling' and the 'way of working' of a methodology.

2.2. Way of working

The way models are being constructed is put into practice in the way of working. The way of working includes the means a methodology offers to create, handle and use models at an operational level. The way of working describes how the work is structured. More specifically, it:

- offers tasks and task definitions at the operational level
- offers a task structure
- includes procedures
- includes informal suggestions
- defines its relationship with the way of modelling

2.3. Way of control

The way of control is in essence related to the control of time, costs and quality of the I.S. development process and its products. It comprises the means a methodology offers to handle models at a managerial level. Typical tasks of controlling a project are:

- setting up a project organization
- splitting up the project into well-defined parts
- defining decision points and control points

It will be clear that the way of control on the one hand and the way of working and the way of modelling on the other hand mutually interact and are dependent on each other. This is the reason why we present them together in figure 1. The way of working constitutes the process-oriented operational aspect of a methodology. The way of modelling constitutes the product-oriented operational aspect of a methodology. The way of control is concerned with managerial aspects.

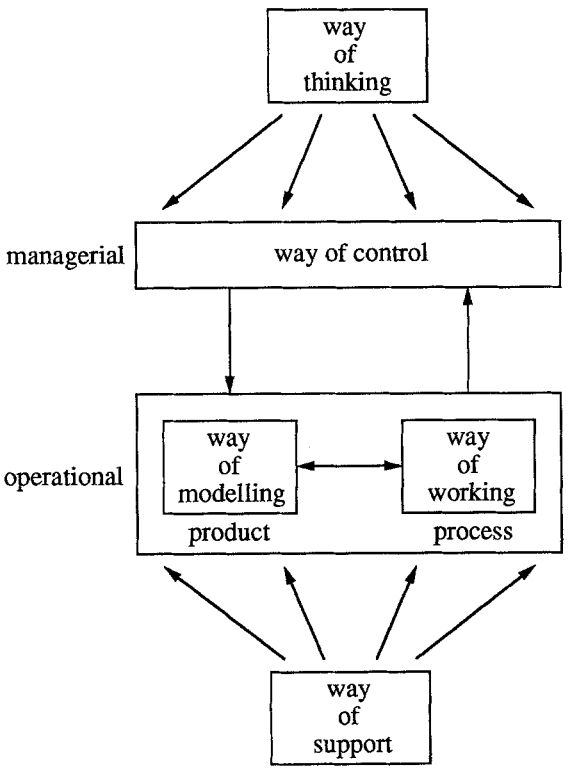

Figure 1. Framework for methodologies

It appeared, see Seligmann (1989), that methodologies can be described quite satisfactory by distinguishing between a way of modelling, a way of working and a way of control. However, for really understanding a methodology it is necessary to know its underlying philosophy or way of thinking used to look at organizations and information systems (Sol (1983), Kensing (1984)). The importance of the way of thinking is reflected in figure 1.

2.4. Way of thinking

The philosophy or Weltanschauung behind a methodology is often overshadowed by or implicit in the techniques and methods embedded in that methodology. These hidden assumptions, however, are of great influence on the ultimate appropriateness of methodologies to certain information systems development problems. In fact, features which are considered to be important to the process of constructing models and to the models themselves, depend on the underlying way of thinking.

The major characteristics of a methodology (be it that they surface only implicitly, e.g. through the character of the network of models, or also explicitly through expository writing), are handled in the 'way of thinking'. In the way of thinking we present basic (implicit or explicit) assumptions and viewpoints of a methodology as regards information systems, their relationship with the environment and their components.

In the way of thinking we describe the major perspectives of a methodology on the following subjects:

- what is an information system
- what is the function of an information system (see environment)
- what are the components of an information system
- what constitutes the environment of the system
- what are the components of the environment
- what are the major characteristics of the components of both the information system and the environment (i.e. a larger system)

2.5. Way of support

In a complete methodology one can argue that complete knowledge exists about which models have to be constructed and in which sequence they have to be constructed. Even then, the development of information systems has to be supported by tools that facilitate the development process. Only a few years ago typical tools were word processors, editors, compilers, stencils, wipe-boards, etc. Today CASE-tools are considered to be an important component of the way of support.

3. Automated support

As stated in section 2 the way of support includes the so-called CASE-tools. These tools should support the way of modelling, the way of working and the way of control in the best possible way. This means that we believe that the usefulness of tools is directly related to the level of integration with the other "ways" of a methodology. If a tool is not truly integrated with the other "ways", it will never

become anything more than a documentation aid, see Wijers (1987). A mismatch between modelling process and CASE-tool constitutes an inefficient and ineffective support.

CASE-tools offer editors to support the specification of models. All models are stored in a central dictionary, an encyclopedia, a design database or whatever it may be called. If a model has been fully specified it can be verified by the application of verification rules. Most tools offer the possibility to decompose diagrams; some offer the possibility to integrate models as well. Automatic transformation from a given model to successive models is possible in only some cases.

More general characteristics of current CASE-tools can be found in Bubenko (1988) and NGGO (1988). We will focus on problems that arise when CASE-tools are applied while there is a lack of knowledge about the process of model construction.

3.1. Problems in verification

A problem in offering verification possibilities in CASE-tools is the choice between "a priori" verification and "a posteriori" verification. A priori verification is realized by offering support in which it is impossible to specify incorrect models. For example, IEW only records relationships between entity types in which both roles are named.

A posteriori verification is realized by offering support in which all specifications are allowable and in which only on request verification rules are applied. In general, a posteriori verification is favorable because a priori verification might conflict with the desired way of working of an analyst. However, in the specification of a relationship between entity types, for example, nobody notices the restriction that each relationship has to be drawn between two entity types.

Too strong a priori verification burdens the designers creativity and exploration possibilities. Only a posteriori verification might result in illogical and unsound models because too little guidance was offered in course of the modelling process. A correct trade-off between a priori and a posteriori verification is defined as supporting a natural level of consistency.

3.2. Problems in navigation between various models

Some models have to be constructed before others. For example, an action diagram is made for an already specified process in a dataflow diagram. Other models might be developed in parallel or in an iterative way, such as dataflow diagrams and entity relationship diagrams. Also, models of the same model type are specified in a certain order. One could think of a top-down or a bottom-up approach.

Many CASE-tools, such as SDW, Blues and Excelerator, are very modular in the sense that they offer separate editors for each type of model. By consequence, these tools do not offer any navigation between model types. Other tools, such as IEF and IEW, are more integrated towards the phases of the life cycle. In these tools navigation between model types is realized in some way or another.

In this section we have focussed on problems that arise because there is a lack of knowledge of the process of model construction. Other possibilities to improve the quality of support might be found in the introduction of domain knowledge in CASE-tools, see Falkenberg (1988), Loucopoulos (1989) and Puncello (1988). In our opinion the introduction of domain knowledge will not solve the problem that CASE-tools offer support that does not correspond with a natural way of working. For this reason, we find that priority should be given to process support above domain knowledge support.

4. The process of modelling

In section 2 we have defined the process of modelling to be the way of working. The way of working represents the operational process-oriented aspect of a methodology. We have defined it in terms of tasks that have a sequence which can be coordinated by decisions, whereas each task can be decomposed into sub-tasks and decisions, see also Bots (1989). In this section we will interrelate the way of modelling and the way of working by relating models, model components, tasks and decisions. The basis of this integration of the way of modelling and the way of working is illustrated in the base model of figure 2.

If we look at the way of modelling we can distinguish models, model components, and relationships between model components. Models can in turn consist of sub-models and model components can be represented in more detail by a model. We have abstracted this into the view as presented in figure 2 in which a modelling concept, i.e. a model or model component, is related to other modelling concepts, and might be further specialized into more specific modelling concepts.

If we look at tasks in the process of modelling, we assume that information engineers work with certain (interrelated) concepts in a task, i.e. create instances of these concepts, and that this task is performed at a given natural level of consistency. In other words, using the terminology of section 3, each task has a corresponding set of a priori and a posteriori rules. On the one hand certain conventions (a priori rules) have to be respected within a task, on the other hand a result has to be realized within a task (a posteriori rules). As part of achieving the desired result, the task can of course be further refined into sub-tasks and decisions. A decision is made in order to define the subsequent course of action. The likelihood of one of the alternative courses is based upon one or more decision rules. If no alternatives exist the termination of one task results in the start of the subsequent one.

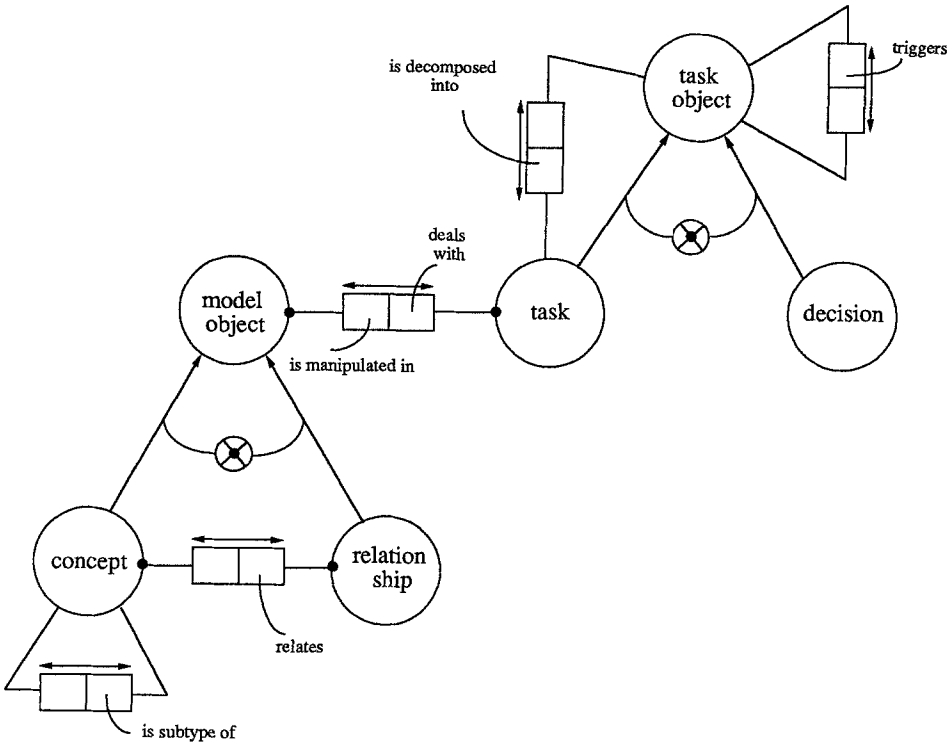

Figure 2. Base model for the modelling process

In order not to complicate figure 2, we have not included explicitly the notions of a priori rule, a posteriori rule and decision rule. Furthermore we have not included that we distinguish simple tasks (or procedures) from composed tasks, see section 7, and that specific information sets can be defined in order to accomplish explicit information passing.

With regard to the coordination of tasks we stress the importance of the relationship "triggers" between task objects, because it enables specification of a sequence of tasks, a choice between tasks and parallelism between tasks, see figure 3. Iteration is accomplished by a decision through which one can return to a previous task.

This so-called task structure is a description of the possible flow of tasks in the way of working. It represents the strategy and direction in the modelling process. Decisions are mostly made based on incomplete knowledge and there always is some doubt as to which alternative should be selected, see Bots (1989). Therefore a choice made at some point in time for one of the alternatives sometimes may have to be reconsidered at some later instant. The task structure only represents the primary flow of tasks.

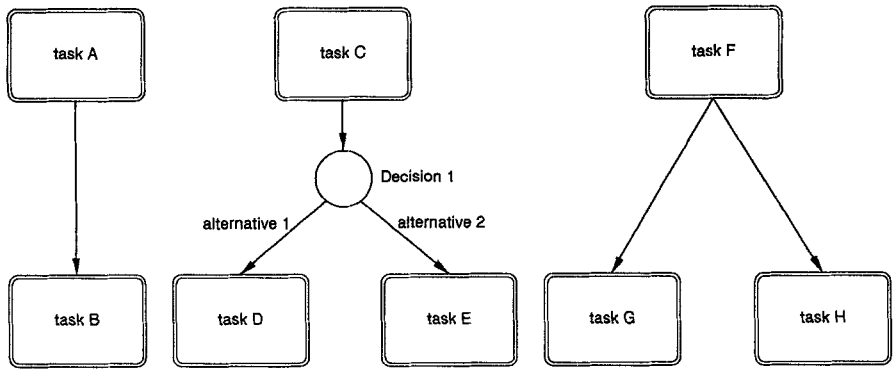

Figure 3. Sequence, choice and parallelism in the way of working

5. Experiments with expert information engineers

Over the past 18 months we have performed three experiments with expert information engineers. As we put forward in section 1, quality depends on the competence of the information engineer. Quality cannot be achieved only by having knowledge about the ultimate products. Knowing where to start, how to continue, what to look for and when to finish does contribute to the quality of the ultimate information system. In the experiments we were particularly interested in the extraction of a detailed understanding about (1) the flow of tasks, (2) the kind of concepts used within and across each task, and (3) the natural level of consistency for each task. As part of the experiments we applied our view on modelling as presented in section 4. Being able to extract detailed modelling knowledge provides a starting point in integrating this knowledge into automated tools, see section 7.

We have investigated for each expert which tasks he performs, with which concepts he works within a specific task, how each task is decomposed, which decisions are taken as part of a task, which decision rules can be distinguished, and which goals he expects to achieve in each task. As for the modelling concepts, we have investigated further specializations of concepts, applicable "a priori" rules and relationships between modelling concepts, in particular those across task boundaries.

5.1. Experiment construction

In Kidd (1985), Hart (1986) and Slatter (1987) various elicitation and formalization techniques are explained together with their useability concerning different types of knowledge.

Since we were particularly interested in the task structure, the concepts an expert uses within the tasks, and the decisions he or she makes to go to other tasks, we have chosen to use verbal protocols, structured interviews and focussed interviews, see for more detail Ledderhof (1989).

Experts have been selected with respect to the two following criteria: (1) the expert should have been working for more than 10 years in the field of information analysis or information planning, and (2) the expert should be considered a leading authority in his or her company. After some introductory conversations with the expert, an experiment has been performed in which he had to construct a global information model consisting of a functional decomposition, one or more dataflow diagrams, and one or more entity-relationship models.

Figure 4 shows a photograph of one of the experiments. An important reason to use an experiment environment as shown in this figure is that the expert is separated from the user by a sound-proof wall in order to allow the expert to think aloud as much as possible.

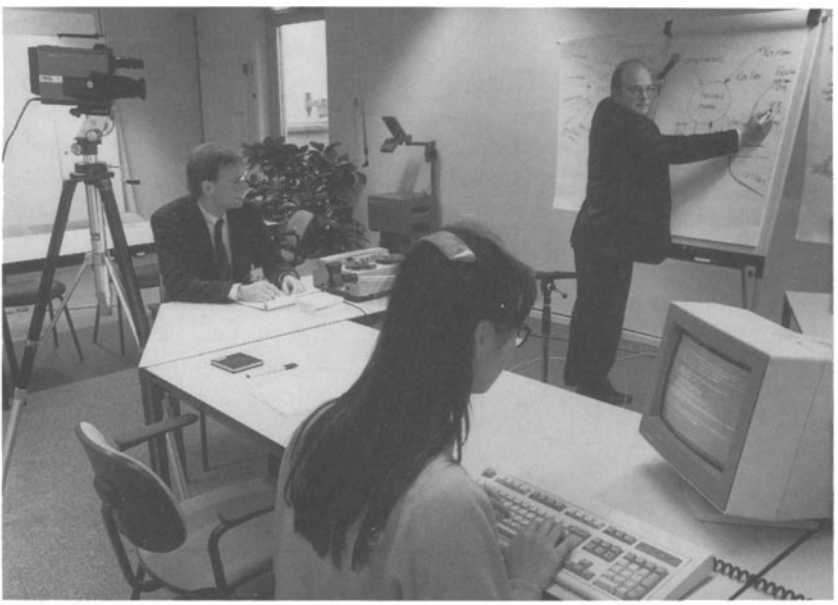

Figure 4. A photograph of one of the experiments

Communication with the user is realized by computer and video connections. Details concerning the above experiment environment can be found in Ledderhof (1989).

5.2. Experiment analysis

Analysis has been performed in three steps: (1) making transcripts of the sessions, (2) clustering the transcripts into tasks and defining the concepts within each task,

and (3) formalizing the task clusters and concept lists into the notions as presented in section 4. As part of this formalization step we have extensively used two diagramming techniques, namely task structure diagrams, see Bots (1989), to represent tasks, decisions, their trigger relationship and task decompositions, and NIAM models, see Nijssen (1989), to represent the modelling concepts, their specializations, their relationships and some of the most important rules.

6. Results of experiments

In view of this paper, we will only highlight some of the conclusions based on our experiments. The conclusions are all related to the overall conclusion that by using the view on modelling as presented in section 4 together with the knowledge acquisition approach as described in section 5, a detailed understanding of a modelling process can be achieved. In this paper we will focus on those results that are related to the extracted strategies and their inherent decisions, the natural level of consistency concerning tasks and the specializations of modelling concepts found.

6.1. Strategy in task structures

In section 4 we have mentioned the coordination between tasks. All three experts appeared to have quite a different strategy. In paragraph 5.1 the overall task of the experts was described as the construction of a global information model consisting of a function decomposition, ER-models and dataflow diagrams. Each expert appeared to have his own preferred technique. The first focussed on ER-models, the second on dataflow diagramming and the third on functional decomposition. The most intriguing fact was that all three used various arguments and practical examples to explain their preference. Another aspect of strategy is that of flexibility. Although there was a preferred strategy all experts had alternative paths in their strategy. Figure 5 shows a strategy of one of the experts.

The tasks presented in the strategy of figure 5 are all further decomposed into sub-tasks and decisions. At more detailed levels in the task structure we found two typical patterns.

Division

In the experiments we often found a strategy similar to the divide-and-conquer principle. For example, a trigger analysis is made for each external trigger, an ER-model is constructed for each dataflow diagram, a market potential is defined for each product group. In order to realize this strategy, the select procedure appeared to be very important. A selection is made in order to do something specifically for that selection.

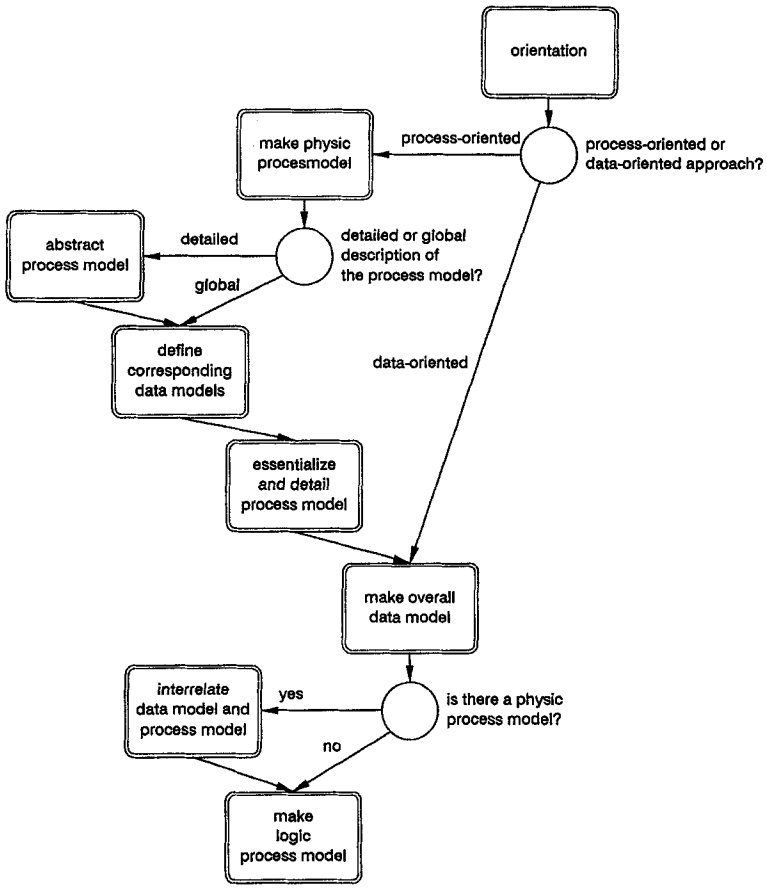

Figure 5. An example of a strategy

Integration

A consequence of division is that we also observed various integration tasks, such as: make a complete data model based on the existing partial models.

Particularly in relatively small tasks, decisions what to do next are heavily influenced by previous tasks. For example, it appeared to be most natural to define directly the relationships for the last entity and not to continue with the definition of new entities. So, the decision to add a relationship is heavily influenced by the creation or identification of an entity. In our formalism an information set "the last entity" is defined and on the basis of that information set a decision rule can increase the likelihood of the task "define a relationship".

Generally, information about already defined concepts is widely used across task boundaries. Sometimes an already defined concept can be directly used in new models (for example, in an integration step or in dataflow diagramming after functional decomposition), in other cases we were able to express heuristic decision

rules such as: "if there is a model construct X, then there is probably a model construct Y".

6.2. Natural level of consistency

In section 4 we introduced the concept of natural level of consistency of tasks. A correct trade-off between a priori rules and a posteriori rules for a task constitutes a natural level of consistency for that task. Both a priori and a posteriori rules appeared to be very important. Examples are thoughts such as "I think we also need something like a payment", "Do I have external parties?". In general, we see that a task is finished if the expert does not have explicit expectations anymore, i.e. if the natural level of consistency is reached, and - very important - if the user has nothing more to add. With regard to the natural level of consistency, refinement is very important. Refinements occur when additional concepts (new expectations) are used within a certain model or stronger verification rules are applied. For example, an entity-relationship model is further refined with respect to optionality of relationships. Another example in ER-modelling is the fact that finally all relationships have a cardinality, but not in the first version of the model.

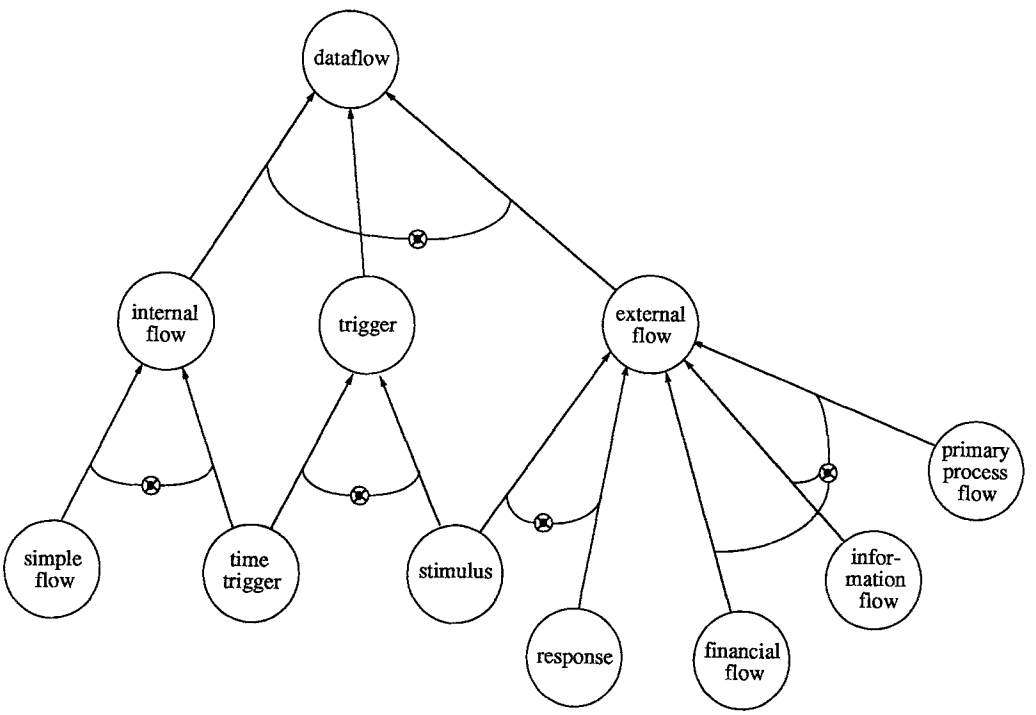

Figure 6. A specialization of the concept "dataflow"

6.3. Finer categorization of concepts

During the analysis of the experiments it appeared to be very well possible to further specialize important concepts part of the diagramming techniques used within the experiments. For each expert the preferred technique appeared to be the most specialized one. In figure 6 we show an example of a finer categorization of the concept "dataflow".

7. An architecture of modelling support systems

In the previous sections we have presented some typical results of our experiments. In all, these results have indicated that our approach indeed leads to a better and detailed understanding of the modelling processes analyzed. In this section we will present our developments up till now concerning automated support of modelling processes as observed in the experiments. For the moment a prototype is available supporting the concept of task agenda, see paragraph 7.3. Intelligent support, see paragraph 7.4, is for the moment restricted to goal information and simple rules concerning cardinality and optionality restrictions.

In this article we prefer to use the notion of modelling support system, because it has never been our goal to fully automate the modelling process. It is our assumption, though, that the modelling process can be supported in a better way than is achieved at the moment by introducing more knowledge of the modelling process in CASE-tools, see section 3.

7.1. Global architecture

A modelling support system itself consists of three main components, see figure 7.

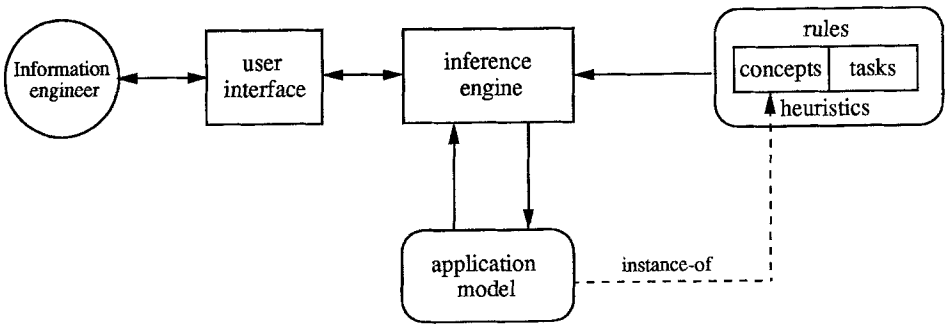

Figure 7. The global architecture of a MSS

First of all, the knowledge of the modelling process of a methodology should be available. Modelling knowledge is structured into:

- Knowledge of tasks
- Knowledge of concepts and relationships used in tasks
- Rules governing the applicability of tasks and concept structures

Secondly, there is the inference engine which operates according to the modelling knowledge specified. The inference engine is a kind of mechanism that presents the information engineer with allowable tasks at a certain moment in time. User interface problems are not considered to be part of the inference engine or the modelling knowledge but have of course to be taken care of in a modelling support system.

The third component is called the application model. The application model consists of all the information specified by the information engineer about a specific application. The items stored in the application model are instances of the concepts and relationships part of the modelling knowledge.

The inference engine, as stated above, operates on knowledge about a specific modelling process. This knowledge is specified using a language called *Methodology Representation Language* (MRL).

This language offers the possibility to specify:

- a task structure
- a concept structure
- decision rules
- a priori rules
- a posteriori rules
- unstructured information for goal facilities

A specific program generates a third generation language source file (Object Oriented Turbo Pascal 5.5) out of a MRL-specification. Linking this source file to the source code of the inference mechanism and the general user interface routines will result in a modelling support system for a specific modelling process.

7.2. Three levels of abstraction

In order to fully understand our architecture as described above, it is important to see that three levels of abstraction can be discerned when an information engineer is working with a specific modelling support system, see figure 8.

At the first level (the application level) case-specific data is specified. Case-specific data is, for example, an organization called *"my_Company"*, which has a *sales department*, a *production unit*, etc.

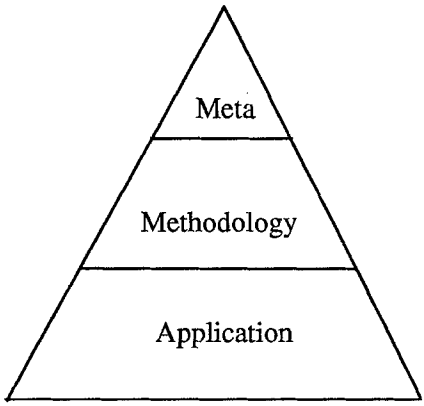

Figure 8. Three levels of abstraction

At the second level (the methodology level) models and model components are defined which are used to represent the case-specific data in a more or less structured way. A possible type of model is a *Functional Decomposition Diagram* ("my_Company") describing *Business Functions* ("Sales", "Financial management", "Production", etc.).

The third level (the meta level) is concerned with structuring the methodology level in task structures, concept structures and additional rules. For example, the analysis of the construction of a functional decomposition results in the *concepts* ("FD-Diagram", "Business Function", "Business Process") and the *task* ("Make a Functional Decomposition Diagram") and the *rule* ("Each Business Function has to be part of a FD-Diagram").

When an information engineer is working with a modelling support system, he is defining models at the application level. These models are instances of the kind of concepts defined at the methodology level. When we are developing a specific modelling support system for a specific methodology, we define a methodology using MRL. MRL can be placed at the meta level.

7.3. Mechanism of the inference engine

Knowing the difference between the application level and the methodology level, it will be clear that the inference engine offers the information engineer a task structure allowing him to enter specifications at the application level. The task structure is offered by presenting a ***task agenda,*** see Bots (1989).

In the task agenda, see figure 9, all the tasks presented are part of a higher level task or of the top task in the hierarchy of tasks. A task in the task agenda is either a most likely task, an executable task, or a non-executable task, depending on the current state of the modelling process.

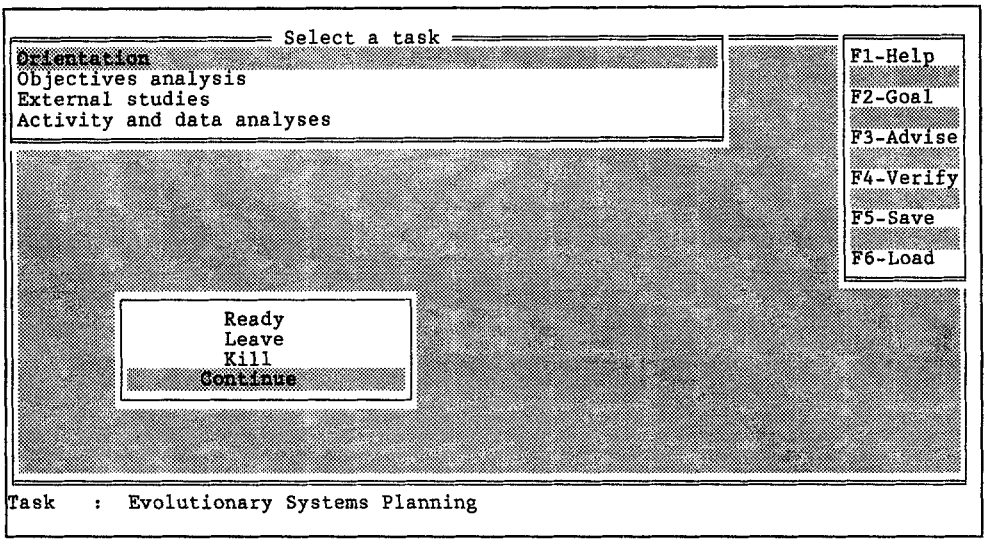

Figure 9. Screen dump of an example of a task agenda

Executing a most likely or executable task will result either in a new task agenda or in the execution of a simple task (procedure). Manipulation of concepts or relationships is realized by procedures. There are four types of standard procedures: create, select, modify and delete. Procedures always behave in accordance with the "a priori" rules of the corresponding task. In addition it should be possible to specify non-standard procedures, the so called *User-Defined-Procedures (UDP's)*. An example of an UDP is the procedure "move". In the current prototype UDP's cannot be defined.

7.4. Intelligent support

At each moment in the modelling process the information engineer might be supported by the modelling support system. Three ways of support are realized: goal information, advice, and verification.

Goal information

The information engineer may ask for information about the goal of a (simple) task. The reason why a task has to be performed and what its result should be is explained, see figure 10.

Advice

Based on (heuristic) decision rules, the inference engine is able to deduce likely or unlikely model structures concerning the model the information engineer is working on. Part of these rules are suggestions for various ways to implement the likely model structures or to correct the unlikely model structures. If the information engineer decides to follow up a specific advise, the inference engine will execute the appropriate task.

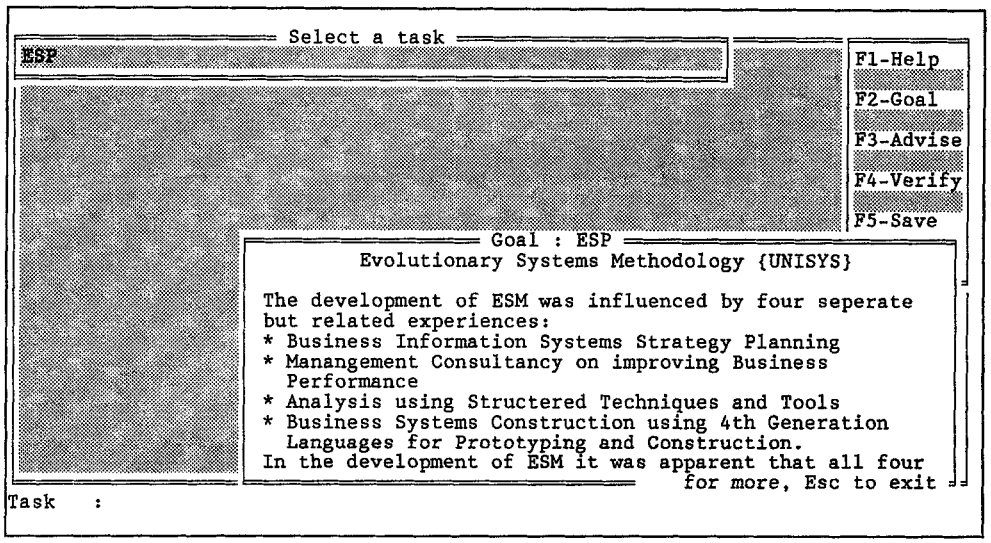

Figure 10. Screen dump of an example of goal information

Verification

Verification checks the "a posteriori" rules. Besides explicit request for verification, verification is automatically performed when the information engineer decides to finish a task. A task is only considered to be finished if the task result is consistent with the "a posteriori" rules.

8. Conclusions and future research

In this article we have argued that a lack of knowledge about models and modelling processes has resulted in problems with CASE-tools concerning verification and navigation. A more process-oriented view on modelling is presented in which the notions of strategy and natural level of consistency are important. A knowledge acquisition approach has been formulated in order to extract detailed modelling knowledge. Protocol analysis is an important part of this approach. This approach has been applied to three expert information engineers. These experiments with expert information engineers have resulted in a better understanding of the modelling processes analyzed.

Diagrams constructed by experts appeared to contain more concepts than that there could be represented externally or explicitly. In other words, the known modelling concepts could be specialized into more specific concepts. In general, the modelling process gradually changed from informal to formal and we noticed that the natural level of consistency changed from a heuristic nature into a more strict nature. In order to maintain a limited span of control, division and integration appeared to be important notions in the way of working.

In the architecture of modelling support systems we have implemented as much as possible our notion of "intelligent" support, which means support based on explicit knowledge of the modelling process. We have introduced the notion of task agenda. Based upon this notion it is possible to offer a strategy in a flexible way, to specify models at the application level in terms of procedures, to implement a functionality for the finer categorization of concepts, to divide and integrate models, to perform iterations, to re-use concepts across task boundaries, and to fire rules for advice and verification. A prototype is available supporting the concept of task agenda including intelligent support by goal information and simple rules based on cardinality and optionality restrictions. Future developments with regard to the prototype will, among other things such as more complex rules, be concerned with improvement of the user-interface.

In the SOCRATES project, see Hofstede (1989), further research is performed concerning knowledge representation of modelling knowledge, modelling knowledge acquisition and modelling support systems. The SOCRATES project will deliver a completely implemented MSS architecture as it is presented in this article.

References

Blokdijk, A. and P. Blokdijk, *Planning and Design of Information Systems*, Academic Press, London, England, 1987.

Bots, P.W.G., *An Environment to Support Problem Solving*, Ph.D. Thesis, Delft University of Technology, Delft, The Netherlands, 1989.

Bubenko jr, J.A., "Information System Methodologies - A Research View", in T.W. Olle, H.G. Sol and A.A. Verrijn-Stuart (Eds.), *Information Systems Design Methodologies: Improving the practice*, North-Holland, Amsterdam, The Netherlands, 1986, pp.289-318.

Bubenko jr, J.A., *Selecting a Strategy for Computer-Aided Software Engineering (CASE)*, Report No.59, SYSLAB, University of Stockholm, Stockholm, Sweden, 1988.

Butler Cox, *Using System Development Methods*, Research Report 57, Butler Cox Foundation, London, England, 1987.

Falkenberg, E.D., H. van Kempen and N. Nimpen, "Knowledge-Based Information Analysis Support", in R. Meersman and C.H. Kung (Eds.), *Proceedings of the WG2.6/WG8.1 Working Conference: The role of A.I. in databases and information systems*, Guangzhou, China, 1988.

Hart, A., *Knowledge Acquisition for expert systems*, Kogan Page, London, England, 1986.

Hofstede, A.H.M. ter, T.F. Verhoef, S. Brinkkemper and G.M. Wijers, *Expert-based support of Information Modelling: A Survey*, Report RP/soc-89/7, SERC, Utrecht, The Netherlands, 1989.

Kensing, F., "Towards Evaluation of Methods for Property Determination", in The.M.A. Bemelmans (Ed.), *Beyond Productivity: Information Systems Development for Organizational Effectiveness*, North-Holland, Amsterdam, The Netherlands, 1984, pp.325-338.

Kidd, A.L., *Knowledge Acquisition for Expert Systems: A Practical Handbook*, Plenum Press, New York, New York, 1987.

Knuth, E., J. Demetrovics and A. Hernadi, "Information System Design: On conceptual foundations", in H.J. Kugler (Ed.), *Information Processing 86*, North-Holland, Amsterdam, The Netherlands, 1986, pp. 635-640.

Ledderhof, H.J.A., *Structured Analysis in de praktijk: Hoe gebruikt een expert het?*, Master's Thesis, Delft University of Technology, Delft, The Netherlands, 1989.

Lockemann, P.C. and H.C. Mayr, "Information system design: Techniques and software support", in H.J. Kugler (Ed.), *Information Processing 86*, North-Holland, Amsterdam, The Netherlands, 1986, pp. 617-634.

Loucopoulos, P. and R.E.M. Champion, "Knowledge-Based Support for Requirements Engineering", in *Proceedings of the 1st Nordic Conference on Advanced Systems Engineering*, Kista, Sweden, 1989.

Martin, J., *Information Engineering Volume 1: Introduction to Information Engineering*, Savant Research Studies, England, 1986.

NGGO, *Computer Ondersteuning van Gestructureerde Ontwikkelingsmethoden, een inventarisatie van tools*, NGGO, Amsterdam, The Netherlands, 1988.

NGGO, *Ervaringen met tools*, NGGO, Amsterdam, The Netherlands, 1989.

Nijssen, G.M. and T.A. Halpin, *Conceptual Schema and Relational Database Design: A fact oriented approach*, Prentice-Hall, Englewood Cliffs, New Jersey, 1989.

Potts, C., "A generic model for representing design methods", in *Proc. of the 11th Int. Conf. on Software Engineering*, Pittsburgh, Pennsylvania, 1989, pp.217-226.

Puncello, P.P., P. Torrigiani, F. Pietri, R. Burlon, B. Cardile and M. Conti, "ASPIS: A Knowledge-Based CASE Environment", *IEEE Software*, March 1988, pp.58-65.

Seligmann, P.S., G.M. Wijers and H.G. Sol, "Analyzing the structure of I.S. methodologies, an alternative approach", in *Proceedings of the First Dutch Conference on Information Systems*, Amersfoort, The Netherlands, 1989.

Slatter, P.E., *Building expert systems, cognitive emulation*, Ellis Horwood, Chicester, England, 1987.

Sol, H.G., "A Feature Analysis of Information Systems Design Methodologies: Methodological Considerations", in T.W. Olle, H.G. Sol and C.J. Tully (Eds.), *Information Systems Design Methodologies: A Feature Analysis*, North-Holland, Amsterdam, The Netherlands, 1983.

Sol, H.G., "Kennis en ervaring rond het ontwerpen van informatiesystemen", *Informatie*, Vol.27, No.3 (1985).

Wijers, G.M. and H.G. Sol, *Intelligent development environments for information systems*, Report 87-05, Faculty of Technical Mathematics and Informatics, Delft University of Technology, Delft, The Netherlands, 1987.

Wintraecken, J.J.V.R., *Informatie-analyse volgens NIAM*, Academic Service, Den Haag, The Netherlands, 1985.

Yourdon, E., "What ever happened to structured analysis", *Datamation*, June 1986, pp.133-138.