Feasibility of Flexible Information Modelling Support

T.F. Verhoef[1] and A.H.M. ter Hofstede[2]

[1]ID Research
Kastanjelaan 4
NL-3833 AN Leusden
The Netherlands
e-mail: DVerhoef@idr.iaf.nl

[2]Department of Information Systems
University of Nijmegen
Toernooiveld 1
NL-6525 ED Nijmegen
The Netherlands
e-mail: arthur@zeus.cs.kun.nl

Abstract

The necessity of CASE tools for system development is beyond dispute. The current generation of CASE tools, however, is too inflexible to provide adequate modelling support. One of the proposed solutions to this problem is the development of so-called *CASE-shells*. A CASE shell is a method independent CASE tool, which may be instantiated with a specific method to become a CASE tool supporting that method. As such, a CASE shell provides complete flexibility. This paper does not address the benefits of CASE shells, as they are completely clear, but focuses on the *feasibility* of this concept from a theoretical as well as a practical point of view.

1 Introduction

CASE tools are currently considered to be an indispensable part of the systems engineer's toolkit. Justification of CASE tools is no longer subject of debate. and [Flo86]). Naturally, the range of the required facilities should be thoroughly understood before using automated tools, see also [BS87]. In this reference an environment (automated or not) is proposed supporting the *practising information engineer* in the use of suitable techniques, depending on the current situation. CASE tools, however, have the problem that the view of the information systems development life cycle to be supported has been *hard-coded* in these tools, and therefore cannot be changed or customised to include also knowledge that is based upon information engineers' practical experience. By consequence, information engineers are left with the problem of finding a way of applying these rigid tools in their information engineering practice.

From the mid-1980s onwards, research has focused on this problem. It is claimed that automated tools are preferably built according to a *CASE shell* architecture.

Such an architecture allows for the modification and extension of the tool's behaviour as the tool includes explicit and adaptable method knowledge. As a consequence, information engineers are able to adapt support tools to their working styles instead of the other way around. Crucial for the development of a CASE shell is the availability of a suitable and formally defined technique for the representation of method knowledge. Such a technique is referred to as a *meta-modelling technique*. Method knowledge represented in a CASE shell according to such a technique is called a *meta-model*.

The concept of a CASE shell is not new. Commercial products such as Toolbuilder of IPSYS Software, Virtual Software Factory of Systematica and MetaDesign of Meta Software Corporation or academic products such as RAMATIC [BBD+89], Metaview [STM88] and MetaPlex [CN89], claim to generate CASE tools tailored to specific methods and organisations. Even a tool exists (MetaEdit [SLTM91]), that supports the modification of meta-models. However, all these shells focus on the support of modelling techniques and hardly pay attention to the modelling process (the importance of which is stressed in [KDH86], [Pot89], [WH90], and [LM86], among others). Furthermore, the degree of support of modelling techniques which they offer is limited, due to the low expressive power of the meta-modelling techniques used.

The focus of this paper is on the *feasibility* of flexible support of information modelling in the early phases and as such on the feasibility of CASE shells. Flexible support is of course considered to be feasible if its benefits outweigh its realisation effort. As the benefits are clear, this feasibility study focuses on the effort needed to realise adequate flexible support. This effort is determined by the complexity of the modelling knowledge to be captured in general, and the diversity needed to support *individual* information engineers. As a result, it is important to know (1) the complexity of information modelling in the early phases and (2) the extent to which flexibility is needed in these phases.

With respect to the first research question, it can be remarked that the early phases of systems development are still poorly understood (cf. [GC88]). Activities in these stages are characterised by incompleteness and vagueness [Bel85]. Terminology is often fuzzy and not standardised. Therefore, a prerequisite for dealing with the first research question is a language in which information modelling concepts can be adequately expressed, i.e. an adequate meta-modelling technique. State-of-the-art meta-modelling techniques, as described in [AC92], [HÖ92], [Bri90], and [SLTM91], are not fully suited for this purpose. They do not have sufficient expressive power to capture information modelling concepts and relations between these concepts, and tend to neglect the modelling process. In addition to that they rarely have a formal semantics and therefore only tend to add to the current confusion with respect to information modelling (see also [HW92]). In section 2, focus is on the various aspects of information modelling and their relations, essential for flexible support. In section 3, techniques are described that are capable of formally describing the aspects described in section 2. Focus in this section is not on the techniques as such, but rather on the *inherent* complexity of an adequate meta-modelling technique.

With respect to the second research question, it can be remarked that relatively little is known about the diversity of information modelling processes in practice and the corresponding degree of flexibility needed for adequate support. Empirical studies reported in the literature (e.g. [Gui90a], [Gui90b], [Bal87], [Wij91] and [BB93]) show

that information modelling knowledge as applied by *experienced* information engineers turns out to deviate from modelling knowledge described in textbooks, regarding both modelling concepts and the way models using these concepts are constructed. These studies served as a starting point for the experiments described in section 4, which focused more closely on the precise behaviour of information modelling experts and the degree of flexibility needed for adequate support.

In section 5 the findings of the previous sections are summarised and the feasibility of adequate flexible support of information modelling processes in the early phases is addressed. The question arises whether the concept of a CASE shell is a realistic goal, given the inherent complexity of an adequate meta-modelling technique and the desired degree of flexibility.

2 A View on Information Modelling

Information modelling processes can be looked upon from many different perspectives, depending on the underlying goal. From a management point of view, resources, deadlines, and quality requirements are important. From a collaboration perspective, the focus will be on communication between individuals in groups. Given our goal, the investigation of the feasibility of flexible support, *knowledge* about the information modelling processes to be supported is important. This knowledge has to be reflected in flexible information modelling support environments. Therefore, this section addresses our view on information modelling by exploring the structure of the repository of a CASE shell.

Essentially, three orthogonal dimensions are recognised within the repository. In its most elementary form, the structure of the repository of a CASE shell can thus be represented as a 2×2×2 cube, see figure 1. These dimensions are discussed subsequently.

It is important to realise that the *combination* of these, as such relatively well-known, dimensions poses significant demands on the, by consequence inherent, complexity of meta-modelling techniques necessary for adequate information modelling support. Section 3 may be interpreted as a justification for this statement.

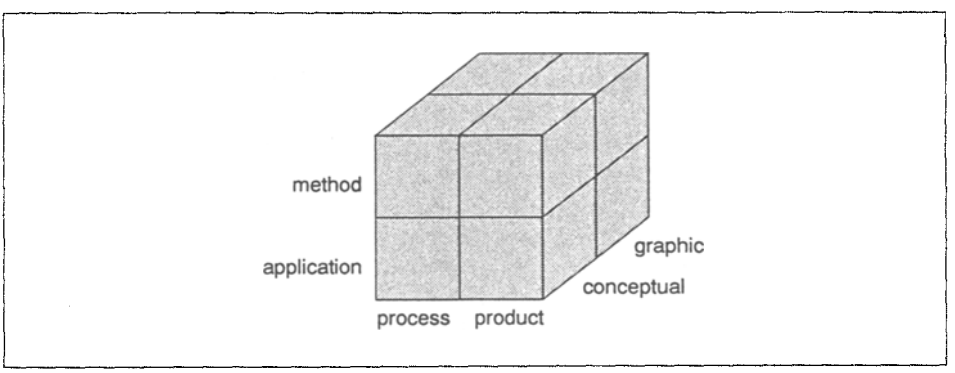

Figure 1: Information modelling dimensions

The first dichotomy is that of *method* level versus *application* level, also referred to as *types* versus *instances*. The method level is concerned with knowledge which may be *used* by information engineers. The method level controls the ways how information modelling processes may be performed, and defines which products may result from those information modelling processes. The application level is concerned with information which *results from* projects for specific organisations and applications by a specific group of information engineers. The application level is an *instantiation* of the method level.

The second dimension is that of *process* versus *product*: in order to provide information modelling support, it is necessary to have knowledge about the (intermediary) products and their relationships on the one hand, and about the underlying modelling process on the other hand. In other words, both questions "*what* should be produced?" and "*how* should it be produced?" should be answered. In [Wij91], the process side is referred to as the *way of working*, while the product side is referred to as the *way of modelling*.

Knowledge about information modelling processes is structured by several key concepts. It is necessary to know which *tasks* may be performed as part of an information modelling process. Tasks can be large tasks: "Perform the Business Area Analysis" within the Information Engineering method, and can be minor tasks: "Add a total role constraint to an Information Structure Diagram" within the NIAM method ([NH89]). These examples show that *decomposition* is a key concept too: tasks may be decomposed into subtasks. Knowledge about information modelling processes also concerns the flow of control: which tasks may be performed next?

Knowledge about information modelling products shows the structure of and the relationships between information modelling products. Examples of information modelling products are a "list of requirements", a "CRUD matrix", a "cardinality constraint", and an "organisation hierarchy". Examples of structure and relationships: "attribute types belong to entity types" and "organisation hierarchies consist of organisation units".

This completes the discussion of the second dimension. It should be clear that the two dimensions are orthogonal: both knowledge about information modelling processes and knowledge about information modelling products exist at method level and at application level. To clarify this, it may be specified, at method level, that the following tasks are to be performed: (i) "Select manager for interview session", (ii) "Interview manager", and as a result (iii) "Refine organisation model". These three tasks may be succeeded by the decision (iv) "Is the organisation model at the desired level of detail?", which triggers task (i) if the outcome is negative, and which leads to continuation if the outcome is positive. Correspondingly, execution of these tasks in a specific project, at application level, may lead to dozens of specific interviews and specific model refinements. Analogously, a notion such as "entity type" on the method level, may lead to many instances on the application level, e.g. "Customer" and "Article".

The third dichotomy concerns the difference between *conceptual* and *graphical* knowledge. Evidently, models must be represented in one way or another: diagrams, matrices, tables, lists, and program specifications are examples. A clear distinction should be made between the *modelling concepts* and their *external notation*. In [SBL89] it

is argued that some methods allow alternative equivalent notations for one and the same modelling concept, but that on the other hand similar graphical and textual topologies can represent different types of modelling concepts.

A similar argumentation is valid for the process oriented view on information modelling. If one looks at some of the commercially available CASE tools, one observes different ways of model manipulation, for example, how entities can be created in entity relationship diagrams. In IEW one action within the ERD window suffices to create an entity. In Excelerator a menu selection has to be performed first, after which one can point at the location preferred.

This third distinction is particularly important for CASE shells. In some way or another, it has to be specified how models appear on the screen and how actions can be performed on these represented models. Furthermore, the specification of graphical knowledge allows information engineers to change the user interface of tools to their own preferences.

Again, it should be clear that this third axis is orthogonal in relation to the two previous ones. Both knowledge about information modelling processes and information modelling products have graphical counterparts. Modelling concepts such as data flows and organisation units are related to graphical notions such as arrows and boxes. Conceptual tasks such as additions of model components lead to graphical interaction patterns such as menu selections, object clicking and dragging, and so on.

3 Complexity of Meta-Modelling

A *meta-modelling technique* is a technique in which modelling knowledge can be expressed. As such, a meta-modelling technique should at least be capable of capturing the various perspectives on information modelling as described in the previous section. This implies that a meta-modelling technique should have sufficient *expressive power*. There are, however, other requirements that meta-modelling techniques have to fulfil.

As a meta-model should not be ambiguous, a meta-modelling technique should be *formally defined* (both syntax and semantics, see also [HW92]). It has to *abstract from implementation details*. Meta-models often need to be validated with modelling experts, therefore a meta-modelling technique should support the construction of *comprehensible* meta-models (e.g. offer graphical representations, decomposition mechanisms etc.). Finally, for the development of CASE-shells meta-models should be *executable*.

In this section, (partial) meta-modelling techniques for the various perspectives on information modelling are outlined. As stated before, the goal of this section is to stress the inherent complexity of meta-modelling rather than to provide an in-depth treatment of the various techniques. This section reflects the view on information modelling presented in the previous section. Section 3.1 concerns the representation of product oriented knowledge, section 3.2 concerns the representation of process oriented knowledge. Both these sections exclude the representation of graphical knowledge, so they are restricted to conceptual knowledge.

3.1 Representing a Way of Modelling

Modelling techniques in general contain concepts with complex structures and their models usually have to satisfy quite complex rules. To capture these structures and rules, a powerful data modelling technique is required, together with a powerful constraint modelling technique. In this section the data modelling technique PSM and the constraint modelling language LISA-D are highlighted. The Predicator Set Modelling technique (PSM) has been specifically defined with the representation of complex structures, often needed for meta-modelling, in mind. PSM is defined in [HW93], and LISA-D in [HPW93].

The elementary notions of object type, relationship, and role are assumed to be well-known. For their graphical representation, the NIAM style has been adopted. First, the necessity of *complex* objects and object inheritance is illustrated by several meta-modelling problems. Subsequently, attention is paid to constraints to represent *complex* rules in product knowledge.

3.1.1 Object Composition

Knowledge about information modelling products can be characterised as structured in a complex way. For example, the information modelling product "entity-relationship diagram" consists of a large variety of information model components. To describe these composition relationships between modelling concepts, PSM offers three representation mechanisms for object composition: set types, sequence types, and schema types.

An instance of a *set type* is a set of instances of its *element type*. As a simple example of the use of set types in the context of meta-modelling, consider the total role constraint in NIAM. An example of such a constraint is depicted in figure 2.

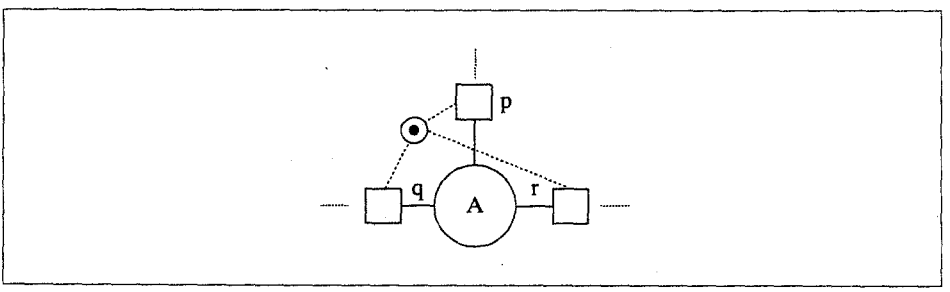

Figure 2: A sample total role constraint in NIAM

In this figure the total role constraint requires every instance of entity type A to participate in at least one of the roles p, q and r. Syntactically, a total role constraint is nothing more (or less) than a *set* of roles. Total role constraints have no other identification than their constituing roles. In a meta-model of NIAM, the total role constraint should therefore be modelled as a set type having an object type "Role" as its element type (see figure 3).

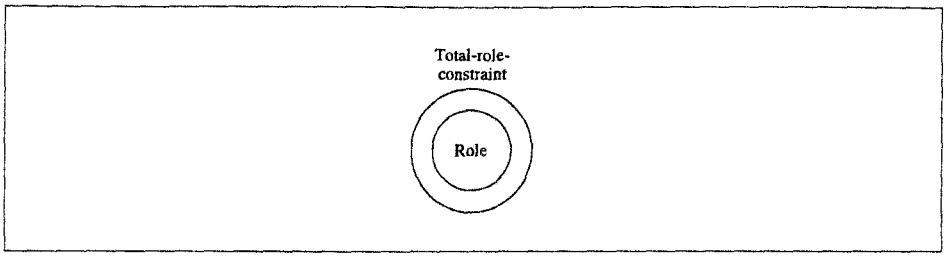

Figure 3: An example of a set type in the context of a meta-model of NIAM

Sequence types can be compared to set types. The differences are that, in the case of sequence types, the ordering of elements is important and elements may occur more than once. An instance of a sequence type is a sequence (tuple) of instances of its element type. A clarifing example of the use of sequence types is given in [Hof93] where a meta-model of JSD entity structure diagrams is explored. This meta-model captures the fact that an action can be decomposed into a *sequence* of other actions.

The third and most complex representation mechanism within PSM is schema objectification. Schema objectification allows to define part of a schema as an object type (referred to as *schema type*). Schema objectification can thus be seen as a decomposition mechanism. An instance of a schema type is an instantiation of the associated schema part. Schema types are particularly important for meta-modelling as they allow for a natural representation of decomposition constructs in modelling techniques. As an example of a schema type, consider the meta-model of Activity graphs as shown in figure 4.

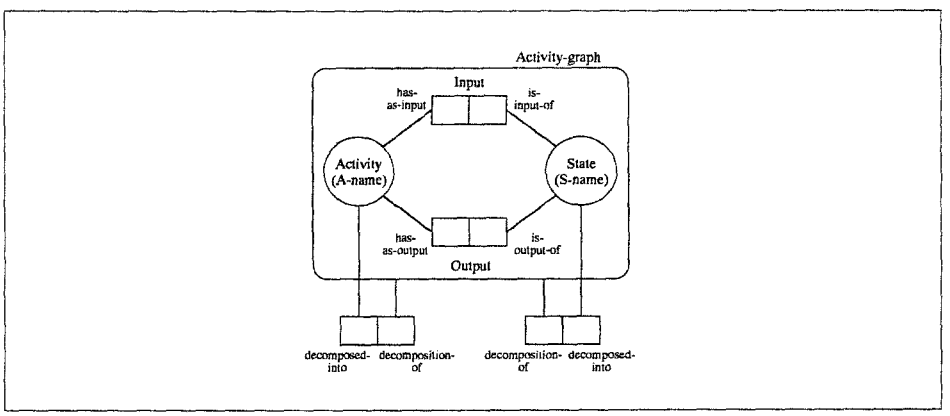

Figure 4: An example of a schema type in the context of a meta-model of activity graphs

Activity graphs are bipartite directed graphs consisting of activities and states. States can be input and output for activities and can be compared to flows in DFDs. Both states and activities can be decomposed into other activity graphs. Figure 4 shows the use of the concept of Schema type to represent the meta-model of Activity graphs. "Activity graph" is a schema type, the decomposition relation is reflected by the binary relationships to "Activity" and "State".

3.1.2 Object Inheritance

PSM offers two representation mechanisms for the representation of inheritance of properties between modelling concepts: specialisation and generalisation.

Specialisation, also referred to as subtyping, is a mechanism for representing one or more (possibly overlapping) subtypes of an object type. Intuitively a specialisation relation between a subtype and a supertype implies that the instances of the subtype are also instances of the supertype. Specialisation relations are organised in so-called specialisation "hierarchies". The top of a specialisation hierarchy is referred to as the *pater familias*. Identification of subtypes is derived from their supertypes, as object types inherit all properties from their ancestors in the specialisation hierarchy. Graphically, each specialisation relation is represented as an arrow. Specialisation is useful in the context of meta-modelling as it allows the definition of specific subsets of instances of certain object types for which only specific relations are important.

Generalisation is a mechanism that allows for the creation of new object types by uniting existing object types. For generalisation it is typically required that the generalised object type is covered by its constituent object types (or *specifiers*). Furthermore, properties are inherited "upward" in a generalisation hierarchy instead of "downward", which is the case for specialisation. This also implies that the identification of a generalised object type depends on the identification of its specifiers. Generalisation allows for the specification of recursive structures. In the context of meta-modelling this is important as e.g. document structures often are of a recursive nature. Graphically, generalisation is represented by means of dashed arrows.

3.1.3 Constraints

PSM offers a number of graphical constraint types for the representation of rules which hold for modelling products. Examples can be found in the meta-model of Yourdon DFDs presented in figure 5. This meta-model also demonstrates the need for the many type construction mechanisms in PSM. First the DFD concepts which appear in this meta-model are clarified.

According to [You89], a DFD pictures a system as a network of functional processes. The main components of a DFD are processes, flows, data stores, and terminators. A process transforms input into output. Processes have a process specification or are decomposed into a DFD. Each process has a number. Control processes are a special kind of process. A control process does not process data, but coordinates other processes. The operation of a control process is modelled by means of a state transition diagram. Terminators represent external processes communicating with the system under consideration. Data stores model collections of data "at rest". Data stores may be external, which means that they are used for communication with the outside world.

Flows represent data "in motion". Several types of flows exist. A simple flow has a source and a destination. Processes, data stores and terminators can be source or

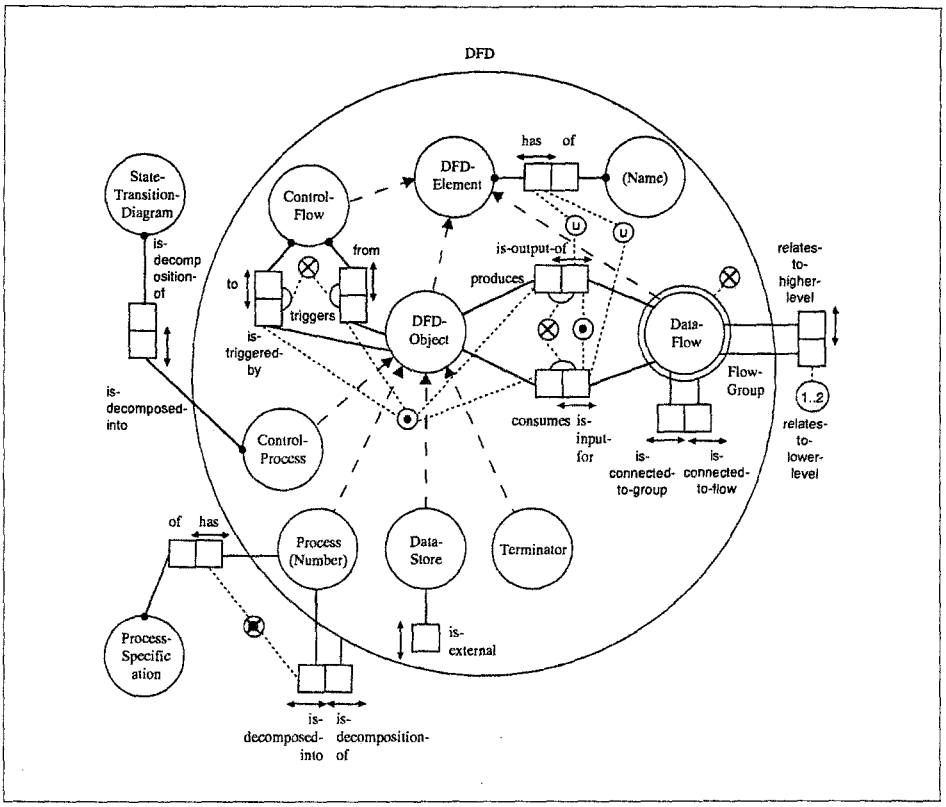

Figure 5: Meta-model of Yourdon DFDs

destination of simple flows. A complex flow consists of a set of flows converging to one other flow or a flow diverging into a set of other flows. Control flows represent triggers, i.e. signals or interrupts.

Some of the graphical constraints in this meta-model deserve some further explanation. We only explain some of the graphical constraints in figure 5. The *total role* constraint on the role named "has" attached to the object type "DFD-Element" expresses that each instance of this object type has to have a "Name". The two *exclusion* constraints attached to binary relationship types express that the source and the destination of a "Data-Flow" are different and that the source and the destination of a "Control-Flow" are different. The two *uniqueness* constraints each over two relationship types, express that no two "Data-Flows" with the same "Name" have the same "DFD-Object" as destination and that no two "Data-Flows" with the same "Name" have the same "DFD-Object" as source. The *occurrence frequency* constraint on the role with role name "relates-to-lower-level" expresses that a "Data-Flow" is related to at most two other "Data-Flows" on a lower decomposition level. The exclusion constraint attached to the set type "Flow-Group" states that a "Data-Flow" does not occur in more than one "Flow-Group".

Of course, there are many other constraints that have to be fulfilled. These constraints, however, are too complex to be expressed using the graphical constraint types offered

by PSM. It should even be noticed that the strive for expressing the most complex constraints graphically might decrease the comprehensibility of the meta-model under consideration. Figure 5 provides a good example of a meta-model which cannot be grasped at once, even in spite of the fact that only a minority of the constraints applicable are represented graphically.

The language LISA-D has been introduced for the representation of constraints that cannot be graphically expressed in PSM. LISA-D expressions exploit the natural language basis of PSM to improve comprehensibility. In meta-models, complex constraints often occur. We do not treat the language LISA-D within the scope of this paper, but refer to [Hof93].

3.2 Representing a Way of Working

This section addresses several constructs for the representation of a way of working. As stated in section 2, a way of modelling and a way of working are closely related. Therefore, attention is also paid to the representation of relationships between a way of modelling and a way of working.

To represent knowledge about information modelling processes adequately, constructs are needed that allow for the description of moments of choice, sequence, parallelism, synchronisation, and iteration. *Task structures* (formally defined in [HN93]) contain constructs for expressing these task dependencies. In figure 6, the main concepts of task structures are graphically represented. They are discussed subsequently.

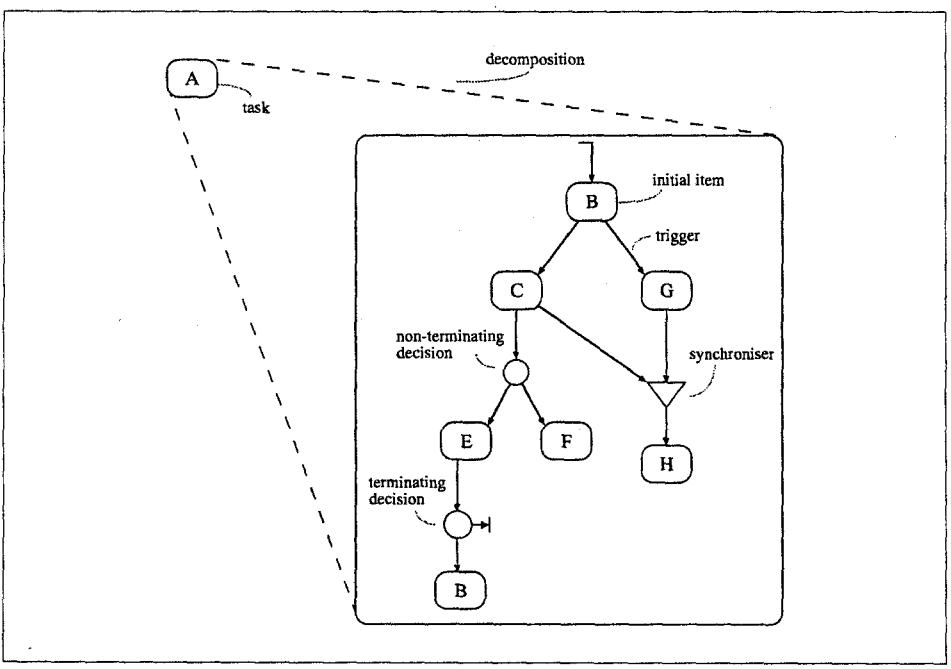

Figure 6: Graphical representation of task structure concepts

The central notion in task structures is the notion of a *task*. A task is defined as something that has to be performed in order to achieve a certain goal: the realisation of (part of) some information modelling product. A task can be defined in terms of other tasks, referred to as its *subtasks*. This *decomposition* may be performed repeatedly until a desired level of detail has been reached. Tasks with the same name have the same decomposition, e.g. the tasks named B in figure 6. Performing a task may involve choices between subtasks, *decisions* represent these moments of choice. Decisions coordinate the execution of tasks. Two kinds of decisions are distinguished, *terminating* and *non-terminating* decisions. A decision that is terminating, may lead to termination of the execution path of that decision. If this execution path is the only active execution path of the supertask, the supertask terminates as well.

Triggers, graphically represented as arrows, model sequential order. In figure 6 the task with name G can start after termination of the top task named B. *Initial items* are those tasks or decisions, that have to be performed first as part of the execution of a task that has a decomposition. Due to iterative structures, it may not always be clear which task objects are initial. Therefore, this has to be indicated explicitly. Finally, *synchronisers* deal with explicit synchronisation. In figure 6 the task named H can only start when the tasks with names C and G have terminated.

3.3 Conclusions

This section has given a flavour of the complexity of the representation of modelling knowledge, according to the view on information modelling of section 2. It is shown that modelling knowledge is very complex to be represented in its full variety. Mechanisms are needed to represent complex object structures, to represent complex rules, to represent tasks within information modelling processes in all their orderings and decompositions, and to represent graphical representations and interaction patterns. Furthermore, many interrelationships exist between these items.

At the same time, we have deliberately used the term "flavour". To offer one coherent toolkit for an adequate representation of modelling knowledge, even more representation mechanisms are required, such as those described in [Hof93] and [HV94]. This reference restricts itself to the formal definitions for the representation mechanisms *within* the conceptual part of the cube. A first attempt to a formal definition of the representation mechanisms within the graphical product oriented part of the cube is given in [HVNW92].

4 Diversity of Information Modelling

This section focuses on the second research question which is addressed in this paper. Since this paper deals with flexible support of information modelling processes, it is, of course, necessary to know how much flexibility is needed, in other words, to know in which manner and to which extent information modelling processes deviate from each other. As stated before, little attention has been paid in the literature to differences and similarities between information modelling processes. This section discusses the

insight gained while comparing information modelling processes, both in theory and in practice. For a detailed discussion of both the approach and the results, see [Ver93].

4.1 Results

This section discusses the insights gained by representing the modelling knowledge as it is *prescribed* in the Yourdon method handbook of [You89], and by representing the modelling knowledge of the Yourdon method as it has been actually *applied* by three expert information engineers in two project situations. The representation of prescribed modelling knowledge and applied modelling knowledge has resulted in seven meta-models. These meta-models have enabled us to compare the individual experts' actual behaviour. Comparison has taken place, not only to grasp the diversity of information modelling processes, but also to compare the practice of information modelling processes to the so-called stick of reference. Methodological considerations about the experimental setting chosen and its justification are given in [Ver91] and in [Ver93].

Given our focus on the early stages, emphasis has been on *data-flow diagrams* (DFDs) and *entity-relationship diagrams* (ERDs) while using a product oriented view on modelling knowledge in Yourdon, and on *constructing the essential model* while viewing Yourdon's modelling knowledge from a process oriented perspective.

4.1.1 A Product Oriented Perspective

Focusing on a *way of modelling* in Yourdon, the main modelling concepts are similar over the model type variants. For example, every ERD consists at least of entity types and relationships. DFDs always consist of processes and data stores, with flows between them. Although the main modelling concepts are similar, we observed that at the same time each model type variant has its own modelling concepts. Comparing the prescribed model type variants to the applied ones, we observed that some prescribed modelling concepts, such as complex data flow and associative object type, are not applied at all. At the same time, the experienced information engineers used *more refined modelling concepts*. Examples are customer and supplier rather than external party, and planning, control, preparation, transformation, and termination processes, rather than just processes. Finally, the applied model type variants contain more concepts which serve communication purposes (e.g. sample value) or which provide quantitative information (e.g. frequency and volume). In addition to ERDs and DFDs, several *other modelling concepts* were used by the experts as well, in particular to create a (sometimes only mental) model of organisational aspects during the problem analysis stage. These non-diagramming concepts are found only in the applied ways of modelling. Some typical examples are: problem cause, organisation unit, information need, and requirement. We observed that several *different graphical notations* are used to denote one modelling concept. Three external notations for the modelling concept relationship within ERDs were seen. One of the information engineers even used two different graphical notations during one knowledge acquisition session. Clearly, the choice of a fixed set of graphical notations is not considered to be a matter of relevance during the problem analysis stage.

4.1.2 A Process Oriented Perspective

Consecutive modelling tasks gradually lead to more structured models, both in the prescribed *way of working* and in the applied ways of working. In the course of modelling processes, more, and more refined modelling concepts are used, and the intermediate models have to satisfy a growing number of verification rules. The nature of modelling tasks changes from free to structured.

The applied ways of working differ from the reference book to a large extent with regard to the *order* in which modelling tasks are performed. The prescribed way of working is characterised by an almost strictly linear order of modelling tasks. The actual application shows an opportunistic order, which is determined by characteristics of the problem domain and of the problem at hand, as well as by the expert's preferences. The information engineers reformulated their approach several times during the course of the knowledge acquisition sessions. In some cases, they even scheduled a number of tasks to be performed in advance. In most cases, however, they only stated that they preferred to pay attention to a specific part of the problem domain, usually to fill clear lacunae in their insights in the problem domain. Their momentary needs strongly influenced the order in which the several modelling techniques were used. Modelling techniques were used as a means to increase insight or to communicate insights, be it in the problem domain or in a specific solution scenario.

The experts showed *individual* ways of working. This is clearly demonstrated by the relative dominance of data modelling and process modelling. One of the applied ways of working can be characterised as data driven, one as process driven, whereas the third shows an equilibrium between the two.

Various *process modelling strategies* have been applied: input driven process modelling, output driven process modelling, and data driven process modelling. From an input driven point of view, processes handle events, and lead to other processes. From an output driven point of view, processes result in fulfilling information needs, and other processes are necessary to deliver the input for these processes. From a data driven point of view, processes manipulate data, i.e., create, read, update, and delete instances of entity types, relationships, and attributes.

Various *data modelling strategies* have been applied too: noun driven data modelling, object driven data modelling, and process driven data modelling. In the noun driven strategy, each noun in the description is considered to be a candidate entity. In the object driven strategy, objects in the real world are related to each other. Each object is questioned for the necessity of storing information on it. The process driven strategy investigates each operating process for entity types, and integrates the resulting partial data models.

As a final observation, the experts incorporated *user participation* as an essential ingredient in their ways of working. They often validated their results with respect to correctness and completeness. They focussed on comprehensibility of intermediate information models, by adding sample values or quantitative data.

4.2 Conclusions

This section summarises the findings and insights with regard to the diversity of information modelling processes in practice. It can be concluded from the individual meta-models that the information engineers differ from each other to a considerable extent, both in their way of modelling and in their way of working. This is a remarkable conclusion since the information engineers were expected to work according to the same underlying information systems development method.

Furthermore, it can be concluded that detailed insights in similarities and differences between information engineers have been gained, but at the expense of a time-intensive approach. The six elicitation sessions led to voluminous text protocols, each including about 150 pages of text and about 30 diagrams, some of which went through several stages. Due to the bulky text protocols, the interpretation task and the conceptualisation task have been time-intensive for the knowledge engineer. This observation is even reinforced by the fact that the representations of the information engineers' way of working do not have a high level of granularity: within the context of creating a diagram, the representations indicate that the diagrams went through several stages, but they do not reach the level of manipulating individual objects.

5 Epilogue

As stressed in the introduction of this paper feasibility of flexible information modelling support is dependent on (1) the complexity of information modelling in the early phases and (2) the extent to which flexibility is needed in these phases. The more complex information modelling is, and the more diverse information modelling processes are in practice, the more effort is needed to realise flexible information modelling support and the less feasible this goal is.

Section 3 dealt with the first research question and demonstrated the inherent *complexity* of a meta-modelling technique capable of describing all the relevant aspects (as defined in section 2) of information modelling methods. Information modelling products are in general quite complex due to decomposition mechanisms, complex structures and complex rules. In addition to the rules that information modelling products (syntax) have to satisfy, their formal meaning (semantics) must be also described. In case of a data model this means that the meta-model should capture which instantiations satisfy the constraints specified and in case of a process model, this means that all possible process executions have to be defined on the meta-level. Information modelling processes may be quite complex if modelling tasks may be performed in parallel. Furthermore, a formal and complete description of the precise effect information modelling processes may have on the various products (and vice versa) turns out to be difficult. Finally, both information modelling processes and products not only have to be approached from a conceptual point of view, but also from a representational point of view. Information modelling products may have complex associated representations and information modelling processes may have complex associated graphical interactions. This relation between the conceptual part of a meta-model and its representational part is essential for flexible support, but has hardly been investigated.

Section 4 dealt with the second research question and demonstrated the inherent *diversity* of information modelling in the early phases. The ways of modelling *and* the ways of working applied by the observed experienced information engineers differ to a large extent. Each information engineer uses its own rules, heuristics, graphical representations and so on. This means that for adequate flexible support, meta-models have to be constructed for each *individual* information engineer. Clearly, this is not feasible, especially since capturing the method followed by an experienced information engineer turns out to be a very time-consuming and difficult task.

These problems may be partially solved if one is less ambitious. To achieve flexible support, it is necessary to find an adequate way to decrease the level of ambition whilst approaching this area. To be more precise, it is necessary to diminish the specification effort effectively. The easiest (and least satisfactory) approach is to neglect aspects of information modelling knowledge. For example, by *not* paying attention to the modelling process or by *not* paying attention to representational aspects. This approach has been used in the development of all "state-of-the-art" meta-modelling techniques mentioned in section 1. None of these techniques address the modelling process. In [VHW91], the modelling process *has* been addressed. This reference, however, neglects the representational aspects of information modelling. A more promising approach would be to exchange complete freedom for "controlled flexibility". Those specifying the knowledge base are then provided with a (pre-specified) generic meta-model, which may be adapted to one's needs by the application of a number of pre-defined meta-model transformations. Another promising approach to reduce the large specification effort is triggered by our observation that a detailed way of working of information engineers is difficult to acquire and requires a lot of effort to describe. Perhaps, it is best not to try to support this level in all details. Alternative support could then be achieved by offering a number of predefined operations. We consider such a *building block* approach to be an interesting issue for future research.

Summarising, it is clear that unrestricted, adequate, flexible support of information modelling is, practically speaking, impossible to achieve. Sometimes, however, restrictions (e.g. when complete graphical support is not needed) may be perfectly acceptable. To balance benefits and efforts of flexible information modelling support, a research agenda has been presented, centered around the level of ambition to be realised.

References

[AC92] T. Araujo and R. Carapuça. Issues for a Future CASE. In B. Theodoulidis and A. Sutcliffe, editors, *Proceedings of the Third Workshop on the Next Generation of CASE tools*, pages 225–243, Manchester, United Kingdom, 1992.

[Bal87] J.M. Ballay. An experimental view of the design process. In W.B. Rouse and K.R. Boff, editors, *System Design: Behavioral Perspectives on Designers, Tools and Organizations*, pages 65–82, Amsterdam, The Netherlands, 1987. North-Holland.

[BB93] J.P. Bansler and K. Bødker. A Reappraisal of Structured Analysis: Design in an Organizational Context. *ACM Transactions on Information Systems*, 11(2):165–193, 1993.

[BBD+89] P. Bergsten, J.A. Bubenko, R. Dahl, M. Gustafsson, and L-Å. Johansson. RAMATIC - a CASE shell for implementation of specific CASE tools. Technical report, SISU, Stockholm, Sweden, 1989. First draft of a contribution to section 4.4 of the TEMPORA T6.1 report.

[Bel85] L. Belady. MCC: Planning the Revolution in Software. *IEEE Software*, 2(6), 1985.

[Bri90] S. Brinkkemper. *Formalisation of Information Systems Modelling.* PhD thesis, University of Nijmegen, Nijmegen, The Netherlands, 1990.

[BS87] D. Benyon and S. Skidmore. Towards a Tool Kit for the Systems Analyst. *The Computer Journal*, 30(1):2–7, 1987.

[CN89] M. Chen and J.F. Nunamaker Jr. MetaPlex: An integrated environment for organization and information systems development. In J.I. DeGross, J.C. Henderson, and B.R. Konsynski, editors, *Proceedings of the Tenth International Conference on Information Systems*, pages 141–151, Boston, Massachusetts, December 1989.

[Flo86] C. Floyd. A Comparative Evaluation of System Development Methods. In T.W. Olle, H.G. Sol, and A.A. Verrijn-Stuart, editors, *Information Systems Design Methodologies: Improving the Practice*, pages 19–54. North-Holland, Amsterdam, The Netherlands, 1986.

[GC88] R. Guindon and B. Curtis. Control of Cognitive Processes during Software Design: What Tools are Needed? In *Proceedings of the Conference on Human Factors in Computing Systems CHI'88*, pages 263–268, 1988.

[Gui90a] R. Guindon. Designing the Design Process: Exploiting Opportunistic Thoughts. *Human-Computer Interaction*, 5:305–344, 1990.

[Gui90b] R. Guindon. Knowledge exploited by experts during software system design. *International Journal of Man-Machine Studies*, 33:279–304, 1990.

[HN93] A.H.M. ter Hofstede and E.R. Nieuwland. Task structure semantics through process algebra. *Software Engineering Journal*, 8(1):14–20, January 1993.

[HÖ92] M. Heym and H. Österle. A Reference Model for Information Systems Development. In K.E. Kendell, K. Lyytinen, and J.I. DeGross, editors, *Proceedings of the IFIP WG 8.2 Working Conference on the Impact of Computer Supported Technologies on Information Systems Development*, pages 215–239, Minneapolis, 1992.

[Hof93] A.H.M. ter Hofstede. *Information Modelling in Data Intensive Domains.* PhD thesis, University of Nijmegen, Nijmegen, The Netherlands, 1993.

[HPW93] A.H.M. ter Hofstede, H.A. Proper, and Th.P. van der Weide. Formal definition of a conceptual language for the description and manipulation of information models. *Information Systems*, 18(7):489–523, 1993.

[HV94] A.H.M. ter Hofstede and T.F. Verhoef. Flexible Support of Information Modelling: Is the Game worth the Candle? Technical Report CSI-R9406, Computing Science Institute, University of Nijmegen, Nijmegen, The Netherlands, May 1994.

[HVNW92] A.H.M. ter Hofstede, T.F. Verhoef, E.R. Nieuwland, and G.M. Wijers. Integrated Specification of Method and Graphic Knowledge. In *Proceedings of the Fourth International Conference on Software Engineering and Knowledge Engineering*, pages 307–316, Capri, Italy, June 1992. IEEE Computer Society Press.

[HW92] A.H.M. ter Hofstede and Th.P. van der Weide. Formalisation of techniques: chopping down the methodology jungle. *Information and Software Technology*, 34(1):57–65, January 1992.

[HW93] A.H.M. ter Hofstede and Th.P. van der Weide. Expressiveness in conceptual data modelling. *Data & Knowledge Engineering*, 10(1):65–100, February 1993.

[KDH86] E. Knuth, J. Demetrovics, and A. Hernadi. Information System Design: On Conceptual Foundations. In H.J. Kugler, editor, *Information Processing 86*, pages 635–640, Amsterdam, The Netherlands, 1986. North-Holland.

[LM86] P.C. Lockemann and H.C. Mayr. Information System Design: Techniques and Software Support. In H.J. Kugler, editor, *Information Processing 86*, Amsterdam, The Netherlands, 1986. North-Holland.

[NH89] G.M. Nijssen and T.A. Halpin. *Conceptual Schema and Relational Database Design: a fact oriented approach.* Prentice-Hall, Sydney, Australia, 1989.

[Pot89] C. Potts. A Generic Model for Representing Design Methods. In *Proceedings of the 11th International Conference on Software Engineering*, pages 199–210, Pittsburgh, Pennsylvania, 1989.

[SBL89] A.G. Sutcliffe, W.J. Black, and P. Loucopoulos. System Specification Semantics: Defining the knowledge captured by structured system development methods in conceptual models. In E.D. Falkenberg and P. Lindgreen, editors, *Information Systems Concepts: an In-depth Analysis*, pages 53–77, Amsterdam, The Netherlands, 1989. North-Holland.

[SLTM91] K. Smolander, K. Lyytinen, V-P. Tahvanainen, and P. Marttiin. MetaEdit: A Flexible Graphical Environment for Methodology Modelling. In R. Andersen, J.A. Bubenko, and A. Sølvberg, editors, *Proceedings of the Third International Conference CAiSE'91 on Advanced Information Systems Engineering*, volume 498 of *Lecture Notes in Computer Science*, pages 168–193, Trondheim, Norway, May 1991. Springer-Verlag.

[STM88] P.G. Sorenson, J.-P. Tremblay, and A.J. McAllister. The Metaview System for Many Specification Environments. *IEEE Software*, pages 30–38, March 1988.

[Ver91] T.F. Verhoef. Structuring Yourdon's Modern Structured Analysis. In V.-P. Tahvanainen and K. Lyytinen, editors, *Proceedings of the Second Workshop on the Next Generation of CASE Tools*, pages 219–313, Trondheim, Norway, May 1991.

[Ver93] T.F. Verhoef. *Effective Information Modelling Support.* PhD thesis, Delft University of Technology, Delft, The Netherlands, 1993.

[VHW91] T.F. Verhoef, A.H.M. ter Hofstede, and G.M. Wijers. Structuring modelling knowledge for CASE shells. In R. Andersen, J.A. Bubenko, and A. Sølvberg, editors, *Proceedings of the Third International Conference CAiSE'91 on Advanced Information Systems Engineering*, volume 498 of *Lecture Notes in Computer Science*, pages 502–524, Trondheim, Norway, May 1991. Springer-Verlag.

[WH90] G.M. Wijers and H. Heijes. Automated Support of the Modelling Process: A view based on experiments with expert information engineers. In B. Steinholz, A. Sølvberg, and L. Bergman, editors, *Proceedings of the Second Nordic Conference CAiSE'90 on Advanced Information Systems Engineering*, volume 436 of *Lecture Notes in Computer Science*, pages 88–108, Stockholm, Sweden, 1990. Springer-Verlag.

[Wij91] G.M. Wijers. *Modelling Support in Information Systems Development.* PhD thesis, Delft University of Technology, Delft, The Netherlands, 1991.

[You89] E. Yourdon. *Modern Structured Analysis.* Prentice-Hall, Englewood Cliffs, New Jersey, 1989.