POSSIBILITIES AND LIMITATIONS OF REUSING ENTERPRISE MODELS
(New Requirements for Enterprise Engineering Tools)

P. BERNUS*, L. NEMES**, R. MORRIS***

* School of Comp. and Info.Techn., Griffith Univ., Nathan Qld 4111 Australia (bernus@cit.gu.edu.au)
** Division of Manuf. Technology, CSIRO, Preston VIC., 3072 Australia (lnm@mlb.dmt.csiro.au)
*** Dept. Computer Science, Florida Inst. of Technology, USA (morris@cs.fit.edu)

Abstract. An account is given of the possibilities and limitations of reusing Enterprise Models (EM). Measures are identified which ensure that models are interpreted as intended, thereby controlling the quality of the processes using enterprise models – such as enterprise engineering. A definition of model completeness is presented, based on a pragmatic theory of meaning and theory of communication. New requirements for Enterprise Engineering CASE tools are discussed.

Keywords. Enterprise Modelling, Enterprise Integration, Theory of Meaning, Concurrent Engineering, Enterprise Engineering

1 INTRODUCTION

1.1 Why Enterprise Modelling? Goals.

Enterprise Engineering is based on the belief that an enterprise, as any other complex system can be designed or improved in an orderly fashion thus giving a better overall result than ad hoc organisation and design. Enterprise Engineering is a large-scale design effort usually carried out by a co-operating team of designers, analysts and managers.

Enterprise Modelling (EM) encompasses all those modelling tasks which arise in the process of enterprise engineering. EM uses various languages, methods and tools to achieve its goals. These vary according to the life-cycle of the enterprise. Such life cycles are captured in generic models, called Enterprise Reference Architecture (Williams *et al*, 1993).

In this article the nature of Enterprise Models is dealt with as used in Enterprise Reference Architectures, or life-cycle models. EM-s can serve a variety of purposes:

a. To express the design or re-design of the information- and material flow of the enterprise;
b. To achieve common understanding of the enterprise by participants (management, workers etc);
c. To control the enterprise based on the model.

1.2 Problems

1.2.1 Need to reuse models. Although the return from producing high quality enterprise models for enterprise engineering can be significant model building from scratch is unacceptably expensive for a large part of industry (esp. small and medium scale), and there is a recognised need to share and reuse previously produced models. There are two types of models which lend themselves for reuse: *generic models* and *paradigmatic models* (where a typical, particular case is captured and is subsequently modified to suit the new situation[1]).

1.2.2 Sharing and reusing models. Given the need of reuse it is important to investigate if such models can be really shared. And, if they can, to what extent. Notably, what is it that ensures that the information a model was intended to carry is not distorted in the process of reuse? Are there any guarantees that models are correctly interpreted in a new context?

1.2.3 Completeness and consistency of enterprise models. Those enterprises who purchase models for reuse would like to have guarantees that the models contain the information necessary for a successful reuse. This need is in stark contrast with reality: it is known to practitioners that EMs are almost never complete in the sense of complete formal specification, like that of a computer algorithm. What then, is a useful definition of completeness and how can it be achieved?

The main purpose of this article is to give a practically applicable criterion for completeness of enterprise models.

Firstly, two important features of EMs need to be understood:

- The range of phenomena addressed by enterprise modelling stretches multiple disciplines. Multiple modelling languages and practices are used, and there is no single person/profession who would be able to guarantee consistency;

[1] The two cases roughly correspond to top-down design and case based design respectively.

- Models play various roles in the life cycle of the enterprise, and only some of them are executable. From those which are, only a part is machine executable.

Models also tend to be big, hard to maintain, complex to analyse, and even worse (Nemes-Bernus, 1994) – from the incompleteness of enterprise models it follows that there is no formal way of deciding whether the model is only incomplete or it is inconsistent.

Thus to guarantee consistency (and completeness as it will later be defined), users need organisational measures (special functions built into the enterprise engineering process to establish model consistency), or institutional guarantees (use of standard models).

1.3 The structure of this article

Section 2 investigates where the meaning is in Enterprise Models.

The analysis reveals that enterprise models are first of all a means of communication between people to ensure a common understanding of the present or planned enterprise. Only a part of EM-s are used to control business processes, and even that executable part is made possible only because of the first essential function: mediating common understanding between those who design, engineer and operate the enterprise.

From the same analysis completeness criterion is derived, applicable to enterprise models.

Section 3 draws the consequences on the enterprise engineering process, presents limitations of reuse, and suggests ways to ensure successful reuse. Section 4 deals with the implications to enterprise engineering tools and environments (e.g. CASE tools).

2 WHERE IS THE MEANING IN EM?

The meaning of a model can be defined in more then one legitimate way, thus substantial confusion arises when one talks about the meaning of a model. Three important meanings are investigated: the model theoretic, the denotational, and the situated meaning. It will be obvious to the reader that all of these contribute to the perception of what a model means, thus these theories of meaning are complementing one another.

2.1 Model theoretic meaning: An illustration

In model theory meaning is a mathematical structure representing an interplay of syntax and semantics. This subsection gives an illustration of model theory (skipping some technical details with reference to the literature (Lloyd, 1994)).

Figure 1 presents an Extended Entity Relationship diagram typically used when enterprise data have to be modelled. The figure intends to say that:

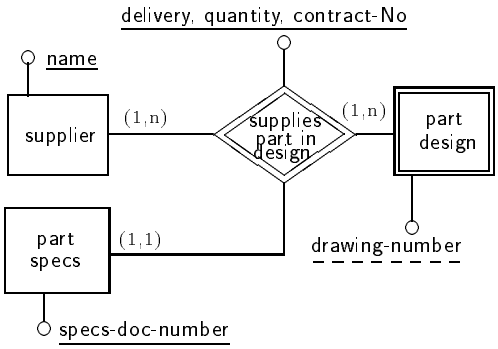

Fig. 1. ER Schema Describing a Sample Universe of Discourse

"Part specifications can be met by various designs offered by part suppliers at various delivery times in various quantities. Such delivery contracts, then are identified by the part specification (its document number), the name of the supplier, and the drawing number of the part. Furthermore to identify a design one has to know the supplier (because the drawing number is unique only within the numbering system of a given supplier). If a supplier supplied a part at a given time in a given quantity then that is registered in a contract with a unique contract number. A given part specification gives rise to a unique contract."

Such an Entity Relationship diagram is expressed in a formal graphical language. To emphasize this point it is possible to translate what the ER diagram says into a list of logical propositions. Figure 2 shows part of such a translation (expressed say in Prolog). (Similar translation can be done into the database programming language SQL.) [2].

If this formal representation is implemented as a Prolog database (as on Fig.2), and the database is filled with facts about individual entities, then queries from this database should return answers that correspond to reality. In other words, the database is in fact a *theory* that describes reality, and this theory has the *predictive power* to tell what would be the outcome of any question if tested on reality instead of testing it on the database!

However, since Fig. 3 represents the *same* ER diagram, it represents the same database as well. This means that the theories embodied in the two diagrams have the same predictive power — they model reality to the same depth.

The fact that entity types have meaningful names on Fig.1 (as opposed to Fig.2) does not influence the behaviour of the database derived from it.

In Model theory it is customary to define the meaning of a theory (i.e., here the meaning of the ER diagram) as the simplest, *minimal model* of the theory. It follows that the first ER diagram does not actually

[2] For the simplicity of demonstration the translation supposes that entities have immutable object identifiers. E.g. the entity set `supplier` has entities sup-1, sup-2, etc with `name` attributes Intel, DEC, etc...

Sample Translation of the ER Data Model:

entity types (entity sets): `et(t)` – *t is an entity set*
entities: `ei(t,o)` -- *entity o is an instance of type t*
relationship types: `rt(t)` -- *t is a relationship type*
relationships: `ri(t,r)` -- *relationship r is of type t*
attributes of entities/r'ships: `at(t,a,d)` -- *type t has attribute a with domain d*
subtyping: `st(sup,sub)` -- *sup is a supertype of type sub*
key attributes: `k(t,l)` -- *attribute list l is key of type t*
attributes of entities/r'ships: `e(o,a,v)` -- *entity o has attribute a with value v*
`etc...`
Axioms (integrity constraints):
"entities of a type t have a unique key value" =
$\forall t \exists k (\forall x_1, x_2\, ei(t, x_1) \wedge ei(t, x_2) \wedge (x1 = x2 \leftrightarrow e(x1, k, v) \wedge e(x_2, k, v)))$ `etc...`
Inference rules: *"inheritance of attributes"* =
$\forall T_{sub}, T_{sup}, A\, st(T_{sup}, T_{sub}), at(T_{sup}, A) \rightarrow at(T_{sub}, A)$.
`etc...`

Sample Translation of the ER Schema of Fig.1:

```
et(supplier).
ei(supplier,sup-1).
rt(suppliesPartInDesign).
ri(suppliesPartInDesign,r1).
at(supplier,name).
k(partDesign,[drawingNumber,name]).
e(sup-1,name,Intel).  etc...
```

Fig. 2. Translation of an ER Schema Into Logical Propositions and Rules.

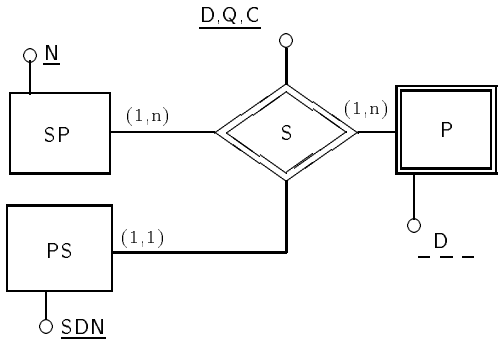

Fig. 3. ER Schema "Equivalent" to that of Fig.1

capture more of reality than the second. This is because the structure and interpretation of both are the same. Even though in Fig. 1 "part specs" intuitively refers to a Part Specification, the database can not tell anything more about the nature of Part Specification than an (identical) database derived from Fig.3!

It also follows, that all systems onto which Fig. 1 can be mapped are equally described by Fig. 3.

Formal software specifications have the same nature; the meaning of the specification is not effected if the names of symbols used in the specifications are changed. Such a "model" is complete, in the sense that no ambiguous interpretation is possible provided the formal specification language is chosen with care. It is "only" to be ensured that the unique model so attributed to the specification (with suitable 1:1 mapping of language tokens to real world entities and relationships) is the same as the real word model; and that the suitable renaming is an intended renaming.

If one compares the use of the two "models" in Fig 1 and Fig 3, it is obvious that some other type of meaning should also be attached to our enterprise models.

Note here that one of the reasons why meaningful names are needed is because users want guarantees that their universe of discourse (i.e. the relevant part of reality) is a model of the theory embodied in the "model" (here the ER diagram).

2.2 Denotational meaning

As illustrated in Fig. 1, terms of a model refer to *denotations* which are concrete or mental entities or relationships. Symbols of a language have denotations through language conventions and therefore the denotations are common to a given *language community*. In the case of Enterprise Models, this community should be that of a profession in which the enterprise does business. Encyclopaedias, dictionaries, technical textbooks offer descriptions of these standard meanings.

There are two ambiguities that spoil the simple denotational theory of meaning.

First, denotations are often context dependent, thus a useful theory of meaning must take into account the contextual elements.

Second, denotations are common to a *language community* rather then a language alone; the looser the connections in that community the less one can trust the denotational meaning.

It is unfortunate that the community of persons who interact with Enterprise Models do *not* form a single language community, so enterprise engineering processes need built-in assurances to filter out the consequences due to false interpretations which arise from incorrect denotations. E.g., the iterative procedure of author/reviewer cycle of IDEF-0 is interspersed with figures for "demonstration only" to establish this common interpretation of terminal symbols (Marca-McGowan, 1988). These "demonstrations only" are the bridge between language communities. Demonstrations can be pictures, drawings, movie frames, or even other models.

The reader might conclude that the model theoretic meaning together with the denotational meaning of EMs sufficiently explains the nature of meaning communicated through EMs. By the combination of the two above theories of meaning a vast set of model theoretically possible models can be excluded from being possible interpretations.

Users are not very interested in which other models may exist for the same theory if the symbols in the diagram were grounded in an *unintended way*. When using EMs they implicitly *assume* that this mapping-back to reality of meaningful symbols remains as intended. This in turn allows EM-s to be less detailed than they should be if only the model theoretic meaning were used. As the next subsection shows even less detail is enough tolerable without jeopardising uniform interpretation.

2.3 Situated Meaning

2.3.1 Efficiency and Completeness. To create an all encompassing model of an enterprise is seemingly a daunting job, not only because of the complexity of the task, but more importantly, because the ever changing "organic" nature of business makes the enterprise a moving target for the modeller. Any modelling tasks that is to be done in a particular enterprise integration project must be done in a short period of time. It is thus imperative to capture enterprise models in a manner as efficient as possible. To achieve this efficiency it is necessary to define what the necessary level of completeness for enterprise models is, because this criterion has major influence to efficiency.

Furthermore, EM is not practical if the size of models to be encountered by any one person is beyond the capacity (time, prerequisite knowledge, tool support etc) of the individual.

Efficiency and completeness are defined here in the context of the *use* of these models:

Def.1. *Efficiency* An enterprise model is efficient if it conveys the intended meaning concisely between the parties who produce or use the model.[3]

Def.2 *Completeness of enterprise models* An enterprise model is complete *relative* to the processes using the model if the processes can create (and behave according to) the intended interpretation of the model.

Three important consequences of the definitions are

- If there is no process that uses a model, there is no need for that model. For example, the quest for an integrated corporate database schema of the 1980's (Smith *et al*, 1981) was flawed because there was no need for the complete schema. Only those parts of database schemata – the federated schemata – should be produced that integrate data needed by some meaningful business process (Shet-Larson, 1990).
- It is not necessary that the EM be a true model (or even a theory) at all! The only requirement is that the EM *constrain* its user in such a way that only the intended interpretation is created in, or by the user. This point is explained in detail in section 2.3.2
- Notice, that even the interpretation of the model need not be fully made explicit by any one user – the only pragmatic requirement is that when a "model" is used the user should be able to create that *part* of the intended interpretation which is pragmatically useful for the user's actions.

It is thus apparent that enterprise models need to maintain the efficiency of natural language (including written and spoken word, figures, etc) – while adding accuracy and formality. Efficiency is a key factor.

How is this possible? Semantic theories of natural language (Barwise, 1988) and (Barwise and Perry 1983) can be used to define a third meaning of enterprise models which explain efficiency.

[3] Note the use of the word "conveys" instead of "contains."

2.3.2 Situated Theory of Meaning. Below is a short exposition of the situated theory of meaning, or situation semantics for short. Situation semantics analyses meaning in the context of language use, such as a conversation or a cooperative action which involves language use.

Utterances and described situations. Participants of a conversation pass on pieces of information to other participants in form of spoken or written word, drawing, or other accepted modality of communication. Any such piece of information is called an "utterance." Utterances are produced with the intention e to add to, or modify, the recipient's model of a topic, or "described situation." Interpretation is the process by which the recipient uses the utterance to build or modify its own model of the described situation.

Since the recipient normally has an extensive set of models available about a range of situations (past experience, common models of a field of expertise), the recipient needs only a small set of utterances to build the intended model of the described situation. At least the set of utterances can be small compared to the model.

Utterance situation. Any utterance is uttered in an "utterance situation," which includes the speaker, the listener, some agreement about the goal of the conversation, and possibly other circumstantial elements. This utterance situation is either directly perceived by the participants or is of some standard form (e.g. the producer of utterances can anticipate the situation in which the recipient will interpret the utterances). Note that the same utterance may be interpreted in very different ways if the utterance situation changes.

Meaning as a relation. The situated meaning of an utterance is the relationship that the utterance establishes between the utterance situation and the described situation. This relationship enables the recipient to restrict the set of possible described situations and thus helps build the internal model of the described situation. The conversation is successful if the recipient is able to re-create the intended interpretation.

It follows that there are three elements which can be controlled for the act of communication to succeed:

- Utterances;
- Utterance situation;
- Described situation.

2.3.3 Situated Meaning of Enterprise Models. An enterprise model will be thought of as a set of utterances intended to convey in a precise and efficient manner some information about the enterprise. The goal of enterprise modelling is to achieve a target situation (some new, improved state of the enterprise). This target situation is the *described* situation of EMs. If that described situation is constrained from the outset then the efficiency of EM is improved.

The interpretation of an EM is embedded in "enterprise engineering situations." This includes experts,

reference materials (previous knowledge of paradigmatic cases, standard models, experience), and most importantly the methodology which is followed in the process of enterprise engineering.

When modelling-experts produce a model and communicate it to some recipient group, the intention of this communication is already fixed, the participants are known, and the supposed prerequisite knowledge of the participants may also be defined. All of the above elements form part of the *utterance situation* in which the EM is to be interpreted.

EMs, as a collection of utterances, thus contain information only inasmuch as they constrain the user sufficiently so that only the intended interpretation is created. Enterprise models do not necessarily contain the information that they are usually supposed to carry!

Def.3 *The situated meaning of enterprise models* is the relationship between the situation in which the EM is communicated and the situation about which the EM is stating something.

This *pragmatic* treatment of meaning allows for a hugely increased efficiency in EM communication, and may be implemented as a new way of using enterprise models – provided one can have good control over the "utterance situations" and over the initial state of the "described situation." The same treatment also allows for completeness to be defined relative to the process of use of EMs rather then completeness as an intrinsic property of EMs.

Definition 2 gave completeness as a pragmatic property of enterprise models. It is now clear that this pragmatic completeness coincides with the EM unambiguously and sufficiently carrying its situated meanings. In contrast with def. 2, an absolute measure of completeness is the extent to which enterprise models are theories (in the sense of model theory) – together with some denotational assignment – as opposed to being utterances allowing the recipient to create these theories.

One would think that if an EM is complete in the absolute sense then it is also complete in the relativistic sense. However, this is not so: the recipient of the model may not possess the requisite abilities, or tools to correctly interpret a model which may be judged complete in the absolute sense by an omniscient external observer.

3 LIMITATIONS AND POSSIBILITIES OF REUSE

3.1 Possibilities of reuse

Clearly the important condition of successful model reuse is that models be pragmatically complete (Def.2). Figure 4 shows at a glance the factors which together form the enterprise engineering situation. The "described situation" here is the interpretation

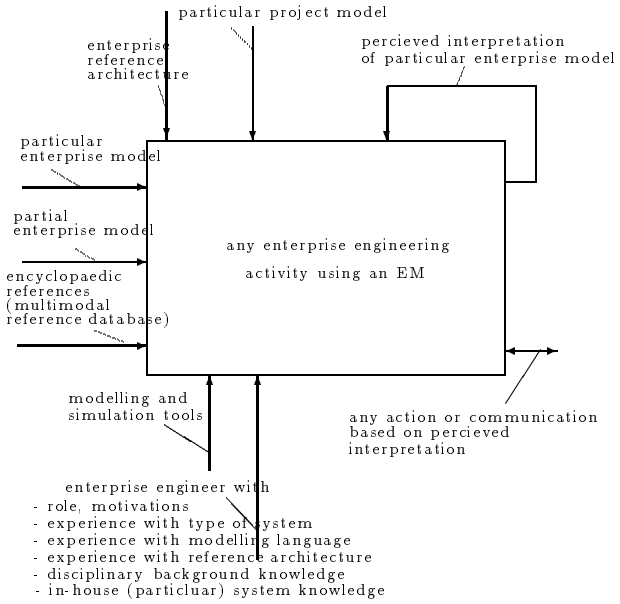

Fig. 4. Factors effecting the pragmatic interpretation of enterprise models

of EM-s as perceived by the enterprise engineer.[4]

For an EM to be (re)usable as intended, it is thus necessary to specify what is supposed about the use of the model in terms of all these factors (as listed on Fig.4). These factors of successful reuse are to be reproduced in the "reusing" process:

- Qualities of the enterprise engineer
- Reference to the type of enterprise engineering situation in which the EM is to be used. (E.g. by reference to an *enterprise reference architecture*.[5]
- Ability to explicitly view the current model of the enterprise engineering process.
- Quick access by the enterprise engineer to the wide range of reference material which may have been used in the production of the EM.
- Links in the design database between the partial and particular models and the enterprise engineering process.
- Capturing of the design history.

The complexity of the enterprise engineering process (and the models produced in it) can be significantly reduced by standardising many of the above components, as:

- Enterprise Reference Architectures;
- Ontologies / Partial Models.

[4] The term enterprise engineer is used as a shorthand in place of naming all those persons who are involved in interpreting enterprise models.

[5] Enterprise Reference Architectures are models, accompanied by methodologies, of the enterprise engineering process – encompassing the entire life cycle of the enterprise.

3.2 Limitations of reuse

The lack of the same factors which enable the intended interpretation to take place can limit the possibility of reuse. To prevent problems it is necessary to review the prerequisites of successful interpretation at the planning stage of any enterprise engineering process.

The lack of adherence to an enterprise reference architecture can have a significant negative effect on the success of reuse. When models are produced they tend to be precise with respect to the intended use and ignore details not necessary for that use. Failure to understand the original intended usage is a *misuse* of EMs and is a major cause of misinterpretation.

The lack of prompt access to adequate encyclopedic reference material. Both the access and the promptness of this access are necessary; since failure to use the reference when it would be necessary introduces an undue (and maybe unnoticed) backtracking point to the design process.

CASE tools of a new type (see section 4) can alleviate the denotational and some of the situational misinterpretations, while simulation tools can help investigate the implicit properties of (executable) formal specifications. Below a set of functions are outlined that enterprise engineering CASE tools should perform to ensure that the enterprise engineering process puts each participant in a position (situation) where correct interpretation is possible.

4 IMPLICATIONS TO ENTERPRISE ENGINEERING TOOLS AND ENVIRONMENTS

Traditional CASE tools allow the designer to create a model in a chosen modelling language. They do not help, however, other designers to understand that model. Below is a shortlist of requirements that enterprise engineering CASE tools need to satisfy. Only those factors are listed here which are not usually part of a state-of-the-art CASE tool.

- Offer links to reference material and cross references to other EMs;
- Be permissive – allow multiple modelling languages
- Translate between various representations of the same EM through algorithms or common reference;
- Make the enterprise engineering process explicit and up-to-date (reference models and particular model);
- EMs are to be communicated between people and mutual understanding is to be an observable occasion in this process (support negotiation, discussion, and in general, cooperative group work)[6].
- Ability to navigate in the EM via queries and links;
- Ability to discover implicit properties by simulation;

The "Intelligent CAD" community has argued that both the representation and automatic control of design data and design processes is needed to support design. Indeed enterprise engineering tools with the above qualities will have a kind of intelligence in this respect; namely that they enhance the intelligence of their users although not necessarily displaing some "inherent" form of intelligence.

Networked information discovery tools seem very appropriate for the provision of the extensive reference functions with intelligent querying facilities to locate the references.

5 CONCLUSION

Reasons of incompleteness of enterprise models were analysed and a definition of pragmatic completeness was given which can, and should, be achieved in enterprise modelling. It was demonstrated how enterprise models carry meaning. This resulted in requirements for the enterprise engineering process, which – if not met – can limit the viability of the process. The analysis of the same factors resulted in requirements for improved enterprise engineering CASE tools.

[6] Theories of conversation (Dorval, 1990) demonstrate that to achieve mutual understanding the participants need to have an agreement on what constitutes the present design situation, and have access to the same common references for denotational (e.g. experimental) purposes.

REFERENCES

Barwise,J. (1988) "The Situation in Logic," Stanford,USA, CSLI ,1988

Barwise,J., Perry,J. (1983) "Situations and Attitudes," Cambridge,MA, MIT Press.

Dorval,B., (1990) "Conversational organization and its development," (B.Dorval ed.), Ablex, Norwood.

Lloyd,J.W. "Foundations of Logic Programming," Springer, Berlin,1984

Marca,D.A., McGowan,C.L (1988) "SADT," McGraw Hill, New York

Nemes, L., Bernus, P. (1984) "An Incomplete Manufacturing Model Needs Matching Design Tools," Proc. 16th CIRP Int. Sem. on Manuf. Sys. Tokyo, pp.26-43

Sheth,A.P., Larson,J.A. (1990), "Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases," ACM Comp. Surv. **22***3*, 183-236

Smith,J.M., et al (1981), " Multibase – Integrating Heterogenous Distributed Database Systems," Proc. AFIPS, **50**,*5* 487-499

T.J. Williams P *et al*, (1993) "Architectures for Integrating Manufacturing Activities and Enterprises," Prepr. IFAC'93 W.C., Sydney, **X**, 273-283