13
The meaning of an Enterprise Model

P. Bernus School of Comp. & IT, Griffith Univ.
L. Nemes, Division of Manuf. Technology, CSIRO
R. Morris Dept. Computer Science, Florida Inst. of Technology

Abstract

An account is given of the possibilities and limitations of reusing Enterprise Models (EM)[1]. Special difficulties are discussed which arise from the fact that stake-holders in the enterprise engineering process do not belong to a single homogeneous language community. Measures are proposed which ensure that models are interpreted as intended, thereby controlling the quality of the processes using enterprise models – such as enterprise engineering. A practical definition of model completeness is presented, based on a pragmatic theory of meaning and theory of communication.

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

1 INTRODUCTION

1.1 Enterprise Integration and Enterprise Engineering

Enterprise Integration is a process in which an enterprise is transformed into an agile and adaptable business system, capable of acting purposefully and coherently as a whole in the interest of its current and strategic business goals in an optimised manner. Enterprise integration is based on the belief that an enterprise, like 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. Thus the integration of the enterprise needs a strategic, large-scale design effort, called *enterprise engineering.*

This design process is usually carried out by a co-operating team of designers, analysts and managers, and can be visualised in a life-cycle model of the enterprise engineering[2] process. Several generic life-cycle models are available for enterprise engineering; and are referred to as Enterprise Reference Architectures (Williams *et al*, 1994).

[1]An earlier version of this article, without the model theoretic definitions, was presented at the 2nd IFIP/IFAC/IFORS Workshop on Intelligent Manufacturing Systems, Vienna 1994. pp11-16
[2]Or rather re-engineering, since the enterprise is never designed from scratch.

P. Bernus et al. (eds.), *Modelling and Methodologies for Enterprise Integration*

1.2 Why Enterprise Modelling? Goals.

Enterprise modelling (EM) encompasses all those modelling tasks which arise in the process of enterprise engineering. EM uses multiple languages, methods and tools to serve the enterprise engineering process Various phases of the enterprise engineering life-cycle use different sets of languages for modelling.

Enterprise models serve the following purposes:

a. EMs must capture the results of all design decisions made in the enterprise engineering process;
b. EMs must help designers to make design decisions, by serving the experimentation, analysis and simulation needs of the design process;
c. EMs should promote common understanding of the enterprise by all stake-holders (management, clients, business partners, engineers, workers using various facets of EMs as common reference, or terminology);
d. EMs need to allow the direct enactment, continuous monitoring, and control of the business processes – on all practical levels of decision making, including relevant aspects of timing, cost, and resource usage.

1.3 Problems

Need to reuse models

Although the economic return from producing high quality enterprise models in the process of enterprise engineering can be significant, model building from scratch remains 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* (common aspects of a type of enterprise) and *paradigmatic models* (where a typical, particular case is captured in model form and that model is subsequently modified to suit the new situation).

Sharing and reusing models

Given the need for reuse it is important to investigate if enterprise 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?

Completeness and consistency of enterprise models

Enterprises which 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 disci-

plines, and accordingly many modelling languages and practices are used. There is no single person/profession who would be able to guarantee the consistency of all models produced;

- 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, 1984) – from the incompleteness of enterprise models it follows that there often is no formal way of deciding whether the model is only incomplete or it is inconsistent.

To guarantee consistency (and completeness as it will later be defined), users often adopt organisational measures (special human functions built into the enterprise engineering process to establish model consistency), or institutional guarantees (through the use of standard models). For reasons shown in this article there are assumptions behind these measures that make them successful in a smaller domain, but which do not always hold in the wider domain of enterprise engineering.

1.4 The structure of this article

Section 2 introduces a formal treatment of enterprise modelling concepts based on model theory. Readers less familiar with the model theory of logic may skip this section.

Section 3 investigates what forms of meaning can be attributed to Enterprise Models. The analysis reveals that enterprise models are first and foremost a means of communication between people, and should ensure common understanding of the present or planned enterprise by them. Only a subset of EM-s are used to control business processes, and even that executable part is made possible only because of the first essential function: communicating ideas to achieve common understanding between those who design, engineer and operate the enterprise. From the same analysis a completeness criterion is derived, applicable to enterprise models.

Section 4 draws on the practical consequences of the theory, suggesting ways to avoid misinterpretations and to ensure successful reuse through the control of enterprise modelling situations as interpretive discourse between participants.

Section 6 deals with the implications to enterprise engineering tools and environments (e.g. CASE tools).

2 WHAT IS AN ENTERPRISE MODEL? – MODEL THEORETIC DEFINITIONS

The reader is cautioned that what follows is an abstraction that will be used to point out the weakness of using model theory *only* for the understanding of enterprise models We actually believe that enterprise models are a means to enable acts of communication for enterprise engineering (and business process execution). The followijg treatment culminates in the conclusion that a fundamental requirement (that of sharing the same herbrand universe is not met in enterprise engineering and modelling, and thereby results

in a need for other types of support for the pragmatic interpretation process of enterprise models). With this *caveat*, here are the model theoretic definitions for enterprise models:

Def. [Preliminary definition] Enterprise Theory::= ET ::= {entities, attributes, propositions}
An ET is a set of propositions formulated in a logic L about some set of entities and their attributes. The language L has a set of logical connectives, inference rules IR_L, and axioms in the usual way.
Propositions are (well formed) formulae of L and have truth values associated with them. An ET can be used to deduce new propositions from existing ones, using the inference rules, thus it is possible to make deductions, or predictions based on an enterprise theory.

Def. Enterprise Theory Tokens::= ETT::= {constants, function symbols, predicate symbols}
These are symbols of ET, where

- Constants represent entities;
- Function symbols represent attributes; and
- Predicate symbols represent propositions about entities and attributes. Predicate symbols have an arity. Truth values of L are represented as a set of constant predicate symbols.

Def. Herbrand pre-interpretation of Enterprise Theory::= HPI ::= {names }
The HPI of an ET is a set of names, such that for each constant and function symbol in ETT there is a name in HPI. In other words a HPI defines a name for each entity and for each attribute present in an ET.

Def. Herbrand Interpretation of Enterprise Theory::= HI ::= {names, relationship types defined on these names, names for truth values, and an evaluation procedure which can assign truth values to relationships with known types in this HI }
The following properties must hold:

- HI is a HPI; and
- For each truth value in ET there is a name in HI; and
- For each proposition that is true in ET there is a relationship in HI such that the truth value of the proposition in ET maps to the corresponding truth value name in HI[3].

The significance of the HI is that for every ET there can be at least one HI and it can be constructed symbolically (on paper or in a computer) Thus we can represent and manipulate the interpretation of a theory rather then having to resort to the real-world interpretation of the theory.

Def. Herbrand Model of Enterprise Theory::= HM ::= { HI and all possible relationships that are generated by the ET as interpreted in HI }
A HM consist of a HI and of all the possible consequences of an ET, given a HI. The consequences, taking the form of propositions, can be interpreted as relationships in HI.

[3]This circumscribed description of truth value mapping is due to the wish to be independent of the set of truth values defined in L, so that our treatment allows various first order logics not only classical FOL.

By this we will have generated a symbolic representation of all possible consequences of the Enterprise Theory ET, thus its symbolic model.

Def. Herbrand Universe of ET ::= HET ::= {the language to which the symbols in HI belong}

Def. Enterprise State ::= EST ::=
An enterprise state is a set of real world entities[4] and their (attributive and relational) properties at an instant of time.
As such, an enterprise state has an infinite number of properties. The change of any property of any entity creates out of an enterprise state another enterprise state .

Def. Enterprise ::= E ::= { An enterprise is an ordered collection of Enterprise States which satisfy certain condition C }.
In the opposite direction this definition suggests that an enterprise goes through a temporal sequence of states and that all these states satisfy the condition. If the condition ceases to be true, the resulting set of real world entities can not be said to form an enterprise state any more, and the enterprise ceases to exist. We can also say that $C(E)$ is true meaning that $E = \bigcup_i E_i \mid \forall_i C(E_i)$ Therefore ontologically an enterprise is a *natural kind* (Bunge, 1979) }
Imagine that an external omniscient and omnipotent observer observes an enterprise E by taking 100% accurate pictures of its states at sufficiently short time intervals to record any change in the state of the enterprise and putting these pictures in a table E as tuples. Any tuple e in E represents an EST at a given point in time. From here on will identify E with this imaginary table. E is not a representation in the sense that it is defined to be fully accurate therefore it can not be created in reality.

Def. Behaviour ::= B ::= { B is a view of an Enterprise E, defined by a function applied to E, i.e. $B = B(E)$ }
B is therefore also a table, each column of it is generated by the function $B(E)$, and corresponding to a column in E. For each relationship defined on B there is a corresponding relationship with the same truth value on E (the inverse is not true of course). From the point of view of the omniscient (and omnipotent) observer B is what is seen when concentrating on a "facet" of the enterprise.
E has several infinities: 1) infinite number of columns, 2) infinite number of tuples, 3) potentially infinite number of tuples between any two different tuples.
As opposed to this B is finite – at least from the second and third points of view. B is thus observable by a not omniscient and not omnipotent real observer.

Def. { The condition of being an enterprise::=
C where $\forall_i C(EST_i) \rightarrow E_i \in E$ }
The *condition* of "enterprise-hood" is a proposition that defines those properties which hold of each state of the enterprise. Pragmatically, since there is no element of such a condition which could not be expressed as a condition on a set of the behaviours of the enterprise an equivalent definition is:
{ The condition of being an enterprise::=
C where $\forall_i C(B_i(EST_i)) \rightarrow E_i \in E$ }
This second definition is more workable because it is a condition on a set of enterprise behaviours (having finite number of attributes) rather then a condition on E

Def. Enterprise Class ::= EC ::=

[4] or "substantial individuals" in ontological jargon.

{ An enterprise class EC is the class of all E_j for which a class condition holds }
In other words a class (or type) of enterprise is defined by a condition that holds for every enterprise in that class. Every enterprise has an "enterprise-hood" condition true of it and each of these conditions can be expressed as conditions on a set of behaviours of these enterprises. Therefore the condition of being en enterprise of a given class (type) is a condition expressed as a shared behavioural property. A class hierarchy of enterprise types can be defined in this way.

Def. Observed Behaviour or Enterprise ::= OB ::=
A real observer, within the precision of observability (limited by the observer's means of observation as well as physical laws) will represent $B(E)$ with some error relative to the real behaviour. While $B(E)$ and E are fictious entities, $OB(E)$ is not. $OB(E)$ is the result of real experimental observation.
$OB(E) \sim B(E)$ means that $OB(E)$ sufficiently approximates $B(E)$ where the level of tolerance is determined by the goal of the observation. E.g., statistical validation of a theory with a predetermined level of confidence. This approximation depends on the selection of means of observation as well as physical laws that limit observability.
The same approximations apply when comparing $B(HM)$ and $OB(E)$, i.e., $B(HM) \sim OB(E)$ means that $B(HM)$ sufficiently approximates $OB(E)$.
This approximation is a mathematical reflection of the fact that the predicted behaviour may be slightly different from the observed behaviour.

Def. [Full definition] Enterprise Theory::= ET ::= {entities, attributes, propositions true of some class of enterprise EC }
An enterprise theory can be about a single enterprise or a class of enterprises. In case the theory is of a particular enterprise the predictive power of the ET is limited to that particular enterprise. If the ET is of a class of enterprises (which is normally the case with enterprise theories) then ET's predictive power extends to all enterprises in that class. In other words an ET has its limits of validity for a class of enterprises that share certain behavioural properties.

Models and meanings

Let us generate the Herbrand Model HM of an Enterprise Theory ET.

HM will typically model a range of enterprise behaviours thus HM will be a "virtual enterprise". A behaviour of this virtual enterprise HM is defined as a view of HM, B(HM).

B(HM) is a table in the same way B(E) and OB(E) are. We say that an Enterprise Theory "models" an enterprise to a sufficient level of precision if

$$B(HM(E)) \sim OB(E) \sim B(E)$$

i.e., if the theory predicts the observable behaviour of the enterprise such that the observable behaviour is sufficiently close to the predicted behaviour.

Model reuse

Enterprise model reuse is the process in which we apply an ET previously developed for an enterprise (or a class of enterprises) and we want to apply this ET to an enterprise not known to be in the same class of enterprise to which the ET is known to be valid.

In other words the validity (or potential validity) of an ET should be ascertained in the process of reuse.

In the process of reuse we must be able to ascertain that the new enterprise meets the class-forming condition C of the enterprises for which ET was developed. This condition can be met by ensuring that the new enterprise shares behaviour with the mentioned class. Practically the result may be the partial reuse of an ET, meaning that during the process of validation ET is modified to suit the new particular modelling task.

Suppose, that ET is valid for a class of enterprise E' and we want to ensure that $ET(E')$ is valid for an enterprise E. What needs to be demonstrated is that E belongs to the class E'. Since ET is a symbolic construct, behaviour sharing is to be proved. Let us identify all behaviours contributing to $C_{E'}$[5] namely behaviours $B_i(HM(ET(E')))$. In the same way the corresponding observable behaviours of the existing enterprise (onto which $ET(E')$ is to be applied by re-use) are constructed as $OB_i(E)$.

The reusability of $ET(E')$ is ensured, if for each behaviour investigated

$$B_i(HM(ET(E'))) \sim OB_i(E).$$

Particular care must be taken when this similarity is examined, since tokens of ET, $(HM(ET(E'))$ and $B_i(HM(ET(E')))$ must be mapped to tokens of $OB_i(E)$. The mapping of tokens of $OB_i(E)$ is, however dependent on the observer, or rather the language community to which the observer belongs (the attunedeness of interpreting participants) The rest of this article argues that to avoid misinterpretation and mis-use of an enterprise theory (i.e. of "enterprise models") there is a need to reconstruct the way the developers of $ET(E')$ interpreted ET tokens by grounding them in observable behaviours and by controlling the interpretive situation in which the models are pragmatically used.

The interesting phenomenon is that the two Herbrand universes (the ones in which two different people interpret an enterprise theory) may not be the same (due to the lack of uniformly common language community) – therefore the problem of token mapping is a serious one not usually treated in model theory.

3 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.

3.1 Model theoretic meaning: An illustration

This subsection gives a simple illustration of model theoretic meaning.

Figure 1 presents an Extended Entity Relationship schema typically used when enterprise data have to be modelled. The schema captures the following facts about (some part of) the enterprise:

[5]From the class forming condition of E'.

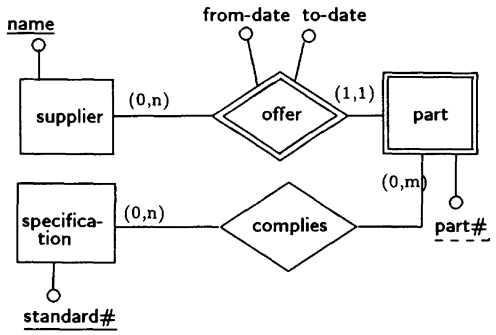

Figure 1 ER Schema Describing a Sample Universe of Discourse

"Parts are offered by suppliers between a from- and a to- date (meaning the availability of the part for order). Suppliers have a unique name. Part numbers (`part#`*) distinguish parts from the same supplier. For every part it is recorded if it complies with any standards. Standard specifications have a unique standard number (*`standard#`*)."*

Such an Entity Relationship schema is expressed in a formal graphical language. To emphasize this formal nature, we can translate the ER diagram into a list of logical propositions. Figure 2 shows part of the translation (e.g., expressed in Prolog). [6]

If this formal representation is implemented as a Prolog database (as on Fig.2 a,b), and the database is filled with facts about individual entities (Fig.2 c), then queries from this database should return answers that correspond to reality. In other words, the ER schema (described by the database of Fig.2a together with Fig.2b) is in fact a *theory* about reality, and this theory has the *predictive power* to answer questions about the reality without having to test them in real life. (Example: it can be predicted from the ER schema that in *any state* of the enterprise for every part number there will be exactly one supplier.) Similarly, the complete database of Fig.2 (a,b and c together) will answer questions about the *current state* of the enterprise without having to test the questions in reality!

The model theoretic meaning of the ER schema is therefore the (minimal Herbrand) model of the theory embodied in the ER schema containing all possible consequences of the theory and no consequences that do not follow from the theory.

However, this minimal Herbrand model can not be accepted as the only meaning of the ER schema. Consider Fig. 3 that represents the *same* ER schema as Fig.1. Notice, that the schemata of Fig.1 and Fig.3 are the same, therefore the theories (with suitable renaming) will also be equivalent. Thus if the theories are equivalent the minimal Herbrand models will also be the same. This means that the theories embodied in the two schemata have the same meaning and predictive power — they model reality to the same depth. The fact that entity types have meaningful names on Fig.1 (as opposed to Fig.3) does not influence the behaviour of the database derived from it.

It follows that the first ER diagram does not actually capture more of reality than the second. Even though in Fig. 1 "specification" intuitively refers to a specification as described in a standard, the database can not tell anything more about the nature of

[6]Similar translation could have been done into the database definition and query language SQL.

a - Sample Translation of the ER Data Model:[7].

entity types: `et(t)` - *t is an entity type*
entities: `ei(t,o)` -- *entity o is an instance of entity type t*
relationship types: `rt(t)` -- *t is a relationship type*
relationships: `ri(t,r)` -- *relationship r is of relationship type t*
attributes of entity-/r'ship types: `at(t,a)` -- *entities of type t have attribute a*
subtyping: `st(sup,sub)` -- *entity type sup is a supertype of entity type sub*
key attributes: `key(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 any type t have a unique key attribute" =
$\forall t \exists k(\forall x_1, x_2\, et(t) \wedge ei(t, x_1) \wedge ei(t, x_2) \wedge \forall a \exists v_a(x1 = x2 \leftrightarrow a \in k \wedge e(x1, a, v_a) \wedge e(x_2, a, v_a)))$ `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...`

b - Translation of the ER Schema of Fig.1:

```
et(supplier).
et(part).
et(specification).
rt(offer).
rt(complies).
at(supplier,name).
at(part,part#).
at(specification,standard#).
at(offer,from-date).
at(offer,to-date).
key(part,[drawingNumber,name]).
key(supplier,[name]).
key(part,[standard#]).
```

c - Sample Database Instance Corresponding to the ER Schema of Fig.1:

```
ei(supplier,sup-1).
ei(supplier,sup-2).
ei(part,e-1).
ei(specification,sp-1)
ri(offer,o-1).
e(sup-1,name,Intel).
e(sup-1,name,DEC). etc...
```

Figure 2 Translation of an ER Schema Into Propositions and Rules.

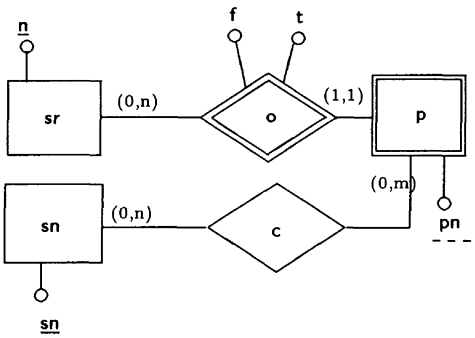

Figure 3 ER Schema "Equivalent" to that of Fig.1

this "specification" than an (identical) database (derived from Fig.3) tells about what is un-intuitively called "sn"! Conversely it also follows, that all systems modelled by the schema of Fig. 1 are also modelled by the schema of Fig. 3.

Formal software specifications have the same nature; the model theoretic meaning of the specification is not effected if the names of symbols used in the specification are changed. A formal model is complete, if no ambiguous interpretation is possible.

If one compares the usefulness of the schemas in Fig.1 and in Fig.3, it is obvious that some other type of meaning should also be attached to our enterprise models. For users to ensure that the EM is indeed a theory of the enterprise in question there must be uniformly accepted mappings of the theory's symbols to reality for all symbols of the Herbrand universe (the theory's language). The predictive power of the theory in Fig.3 is therefore not usable, because we do not know how to ask relevant questions (the theory is "locked in").

3.2 Denotational meaning

As illustrated in Fig. 1, terms of a model refer to *denotations* which are concrete or mental entities, or relationships defined on these. 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. Unfortunately the following observations spoil this simple denotational theory of meaning.

1) Denotations are often context dependent, thus a useful theory of meaning must take into account the contextual elements.

2) Not all symbols of the Herbrand universe are grounded in directly observable objects. Predicate symbols will typically mean[8] *relationships* in the real world, that those people who are day-to-day involved in the technological processes do not necessarily know. Such relationships must be abstracted and learned before respective theories can be interpreted.

3) Denotations are common to a *language community* rather then a language alone; the

[8] "Mean" means here *denote*

looser the connections in that community the less one can trust the denotational meaning. What is a normal assumption in the model theory of logic, i.e., that the Herbrand universe is the same for each participant interacting with the theory, does not hold in enterprise engineering.

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 additional assurances to filter out the consequences due to false interpretations. 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.

In a team of multiple disciplines the energy spent on negotiating and learning a common language can outweigh the energy spent on creating and refining the actual EM. Colleagues in the same profession (preferably educated at the same institution) will have the advantage of a shared language, while the problem becomes acute in multidisciplinary teams.

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, although to acquire common denotations may mean a long learning process for the involved parties.

Indeed, by the combination of the two above theories of meaning a vast set of model theoretically possible real world models can be excluded from being possible interpretations. This is because users are not very interested in which other interpretations may exist for the same theory – should the symbols of the theory be grounded in an *unintended way.*

4) Participants of enterprise engineering (i.e. humans) can not usually 'match' the enterprise models to some 'internal' representations, since as cognitive scientists have shown that expertise in an area does not require the possession of some reflective, explicit representation (Winograd, 1986). Agreed upon representations are gradually built up result of enterprise engineering, not something the participants can rely on from the beginning of that process.

3.3 Situated Meaning

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 are to be done in a particular enterprise integration project must be done in a short period of time.

It is thus imperative to create and use enterprise models in a manner as efficient as possible.The neccessary level of completeness for enterprise madels should be defined , because this criterion has major influence on efficiency.

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[9].

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 *federated* schemata should be produced that integrate data needed by some meaningful business process (Sheth-Larson, 1990).
- It is not necessary that all enterprise models be true models (or even theories) 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. (The intended interpretation may be a theory itself, without the EM being a theory.) This point is explained in detail in section 3.3
- Notice, that the interpretation of the EM need not be made fully explicit by any one user – the only pragmatic requirement is that 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.

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 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 (e.g., references accessible to all parties for observation and

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

experimentation). 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" [10] 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.

4 ENTERPRISE ENGINEERING SITUATIONS

Enterprise Engineering can be considered a large scale design effort of multiple agents. By building a taxonomy of typical situations which involve the exchange of EMs we can analyse what pragmatic interpretation processes are performed by these agents. The analysis allows us to formulate a list of requirements for the implementation (and tool support) of enterprise engineering processes.

The requirements are expected to be valid for computer-aided collaborative workbenches, but also for any group-work in which there is a potential for problems with mutual understanding.

A full very detailed taxonomy is beyond the scope of this paper, but we discuss its top level. According to the goals of enterprise modelling there are:

- Enterprise design situations to create EMS and make them explicit;
- Enterprise design situations to take decisions, including EM validation;
- Learning situations using EMs to support mutual understanding of some domain of the enterprise by a set of stake-holders;
- Enterprise control situations to operate enterprise business processes.

In each of the above situations EMs may be about the product of the enterprise, the enterprise itself, the enterprise engineering process, or strategic enterprise management (Bernus-Nemes, 1994).

[10]The "model" is probably a theory only.

4.1 Situated Pragmatic Interpretation as Constraint Satisfaction

In the pragmatic interpretation process the interpretation of utterances assumes that there is a planned action which the interpreting party wants to take, and that this action needs its inputs (or controls) explicitly stated.

Since many elements of the utterance situation and described situation may be known when the situation is entered into, many symbols of forthcoming utterances are constrained. There often is little selection as to which symbols of the interpreting agent's Herbrand Universe may be considered as legitimate groundings of a symbol in the utterance.

The more the interpreting party can trust that the utterance situation is perceived by all parties in the conversation in the same way, the more constrained and efficient the interpretation process can be. For example, a common reference present in the utterance situation has such beneficial effect.

For efficient interpretation, the participants of the Enterprise Engineering process often start with assumptions about the grounding of some symbols in the utterances exchanged and the grounding of symbols in the described situation. These assumptions may not prove correct later, in which case assumptions (and false consequences) must be backtracked.

The pragmatic interpretation process may have to backtrack less frequently if the utterance situation has many externalised common references available in it. For example, if *immediate* reference to common representations (such as encyclopaedic information) is provided on the spot, then the interpreting party is given the chance to check the assumptions made (as to which grounding of symbols is correct) – *before starting the constraint satisfaction process.*

Therefore enterprise engineering tools need to allow low cost checking of assumptions about the proper grounding of symbols. Passive models (such as pictures, figures, videoclips) allow the grounding of symbols in the Herbrand pre-interpretation. Executable models (such as simulators) allow the grounding of relationship symbols as well, and therefore the grounding of the entire Herbrand model.

4.2 Situated Meaning of Enterprise Models

An EM 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 in as much 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.

5 LIMITATIONS AND POSSIBILITIES OF REUSE

5.1 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 of EM-s as perceived by the enterprise engineer[11].

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.*[12])
- 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.

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

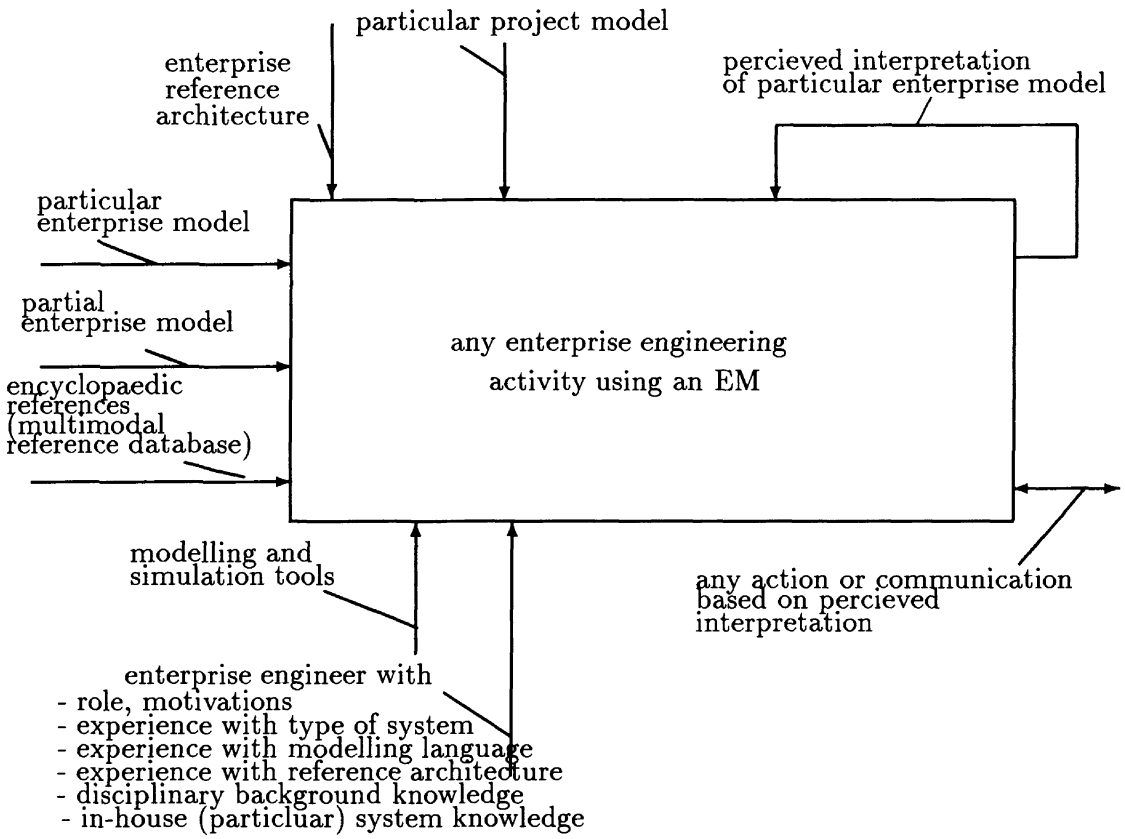

Figure 4 Factors effecting the pragmatic interpretation of enterprise models

- 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.

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

5.2 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 encyclopaedic 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 6) 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.

6 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 short-list 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 (Bernus, 1983);
- 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)[13]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..
- Ability to navigate in the EM via queries and links;
- Ability to discover implicit properties by simulation;

[13],

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 displaying some "inherent" form of intelligence.

7 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.

REFERENCES

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

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

Bernus, P. (1983), "Rigour and Permissiveness in the Design of CAD/CAM Systems," **in** *Integration of CAD/CAM*, D.Kochan (ed), North Holland, Amsterdam, 161-178

Bernus,P., Nemes,L., (1994) "A Framework to Define a Generic Enterprise Reference Architecture and Methodology," Proc. ICARV'94, Singapore, pp88-92

Bunge, M. (1979) "Ontology II: A World of Systems," (Treatise in Basic Philosophy Vol 4.), D.Reidel, Dodrecht.

Dorval,B., (1990) "Conversational organisation 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, 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 Heterogeneous Distributed Database Systems," Proc. AFIPS, **50**,*5* 487-499

T.J. Williams P *et al*, (1993) "Architectures for Integrating Manufacturing Activities and Enterprises," *Computers in Industry* Vol 24, No 2-3, Special Issue on CIM (1994) pp111-140

Winograd,T., Fores,F., (1986) 'Understanding computers and cognition : a new foundation for design,' Norwood, Ablex.