Language Semantics

Towards a Common Underlying Domain Theory

Ioannis L. Kotsiopoulos
Zenon S.A., Athens, Greece, ikotsiopoulos@zenon.gr

Abstract: The paper is a response to the call for a common underlying domain theory to address the mismatch between syntax and semantics of enterprise modelling languages. To this end, we propose categorical morphisms of object interactions as a strong candidate theory on which all modelling constructs can potentially be mapped. Implications of such a framework are fundamental properties of models such as genericity of modelling, modelling environments and model correctness. Semantics of particular modelling languages and architectures can be obtained as specialisations of the general theory. Basic features of CIMOSA are derived as an example.

1 INTRODUCTION

The mismatch between syntax and semantics of languages for enterprise modelling has been identified by ICEIMT'97 as a serious shortcoming in the course of enterprise integration. This is the fundamental reason why models, even syntactically compatible ones, can neither be exchanged between tools, nor can they be federated leading to "agility" nor "level 4 and 5 integration in the enterprise", (Hollocks, et al, 1997). To this end, Working Group 2 at Workshop 4, (Petit, et al, 1997), identified the need for both a Unified Enterprise Modelling Language (UEML) and precise semantics for its constructs. To achieve the latter, the working group called for a single definition of the constructs in terms of a *common underlying domain theory*. Although theories such as situation calculus, state transition diagrams, temporal logic, process algebras etc (Petit, et al, 1997) have already been employed, fragmentation and lack of co-ordination of research effort has so far prevented an overall evaluation of their adequacy and suitability. This renders the do-

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35621-1_43

K. Kosanke et al. (eds.), *Enterprise Inter- and Intra-Organizational Integration*

main open to competition from candidate theories, one of which is the subject of this paper.

We argue the case that category theory morphisms of object interactions is a strong candidate for an underlying domain theory within which:

- All generic enterprise modelling concepts and constructs can potentially be mapped so as to provide definition, unification and resolution mechanisms for semantics of modelling languages,
- Fundamental properties such as genericity of modelling, architecture completeness and model correctness can be addressed using the objects of the theory.

The reason for such confidence stems from the nature of category theory itself. It is essentially a meta-mathematical theory, designed to deal with mathematical objects at the most general and abstract level allowable by the axiom-driven deductive process of mathematics and logic. If we accept that this is the highest form of human reasoning known to us, models themselves are nothing but maps of selected aspects of the real world to mechanisms of this reasoning. Loosely speaking, the "modelling power" of a semantically rigorous modelling language appears analogous to the generality allowed by its underlying domain theory.

2 OBJECTS

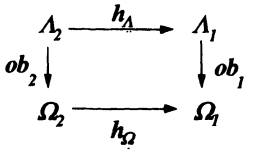
We introduce objects themselves as categorical morphisms called "Observed Processes" and we extend this characterisation to interactions among them, (Ehrich, et al, 1990). The generality of category theory can ensure that neither "useful real objects" nor "useful interactions" are excluded from such a formalism.

Following (Ehrich, et al, 1990), an object is seen as a mapping device between events and values of certain types called *attribute-value pairs* or *observations*. Both attribute-values and events share a general property: they are *atoms of behaviour*, that is, they occur at some single point in time and they "express" themselves through a designated *alphabet*. A collection of such atoms with the same single time occurrence is appropriately called a *behaviour snapshot*.

Dynamic behaviour of an object in this setting results from the attachment of behaviour snapshots to points in time. Formally this is defined as a map $\lambda : t \rightarrow S, t \in TIME$, with *TIME* some time domain be it discrete or continuous. λ is called a *trajectory* over S, the set of behaviour snapshots, while a set of trajectories is called a *behaviour*.

An *object* is a mapping $ob : \Lambda \rightarrow \Omega$ between different behaviours, which, formally, can be given the structure of a morphism in the (complete) cate-

gory of behaviours, (Ehrich, et al, 1990). Both Λ and Ω are snapshots on the time axis, where Λ is termed *process* part and Ω *observation* part.



Remark. This notion of an object as a behaviour morphism allows complete freedom of choice for the process and observation parts. Moreover, an object triggers its own observations according to the process behaviour (the principle of encapsulation).

Interaction between objects can be expressed as operations (morphisms) on objects. An *object morphism*, symbolised by $ob_2 \rightarrow ob_1$ is a pair of behaviour morphisms $h_\Lambda : \Lambda_2 \rightarrow \Lambda_1$ and $h_\Omega : \Omega_2 \rightarrow \Omega_1$ on objects $ob_1 : \Lambda_1 \rightarrow \Omega_1$ and $ob_2 : \Lambda_2 \rightarrow \Omega_2$ so that the left diagram commutes.

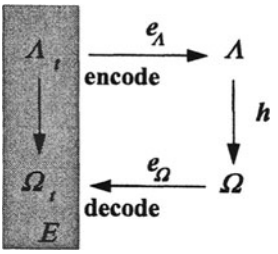
New objects within this category can also be constructed via morphisms. Of particular importance to us are the abstractions of objects produced by restricting either the observation part of an object (object view) or the process part (object trigger). Combining both operations and under some technical conditions (Kotsiopoulos, 1993) a *partial observation morphism* on ob_2 can be defined and used as the main abstraction mechanism. The outcome of this morphism is an object ob_1, which selectively restrict both the process and the observation part of the original object ob_2.

We are in a position now to give substance to the relationship of the "real world", i.e. physical objects, such as an industry processed product under intermediate or final treatment (e.g. a semi-completed form, a large paper roll etc.), to models. We consider a (finite) set of primitive objects, which correspond to all the functions of the physical objects of the enterprise. To those we apply partial observation morphisms, in order to construct a new set of processable objects, which we term the *enterprise objects*. Interaction between them is represented as another set of object morphisms. In this way, what is commonly called "the enterprise and its objects" is nothing but a set of objects and object morphisms partially observed from a physical space and time ensemble and performing some target function.

3 ENVIRONMENTS

Objects are usually seen as operating within a larger set or a larger object, frequently called an "environment". Here we also define a similar concept, adapted to the requirements of modelling architectures, that is amenable to hierarchical control structures similar to the organisation structures of an enterprise (business-process models).

The general idea of an environment is shown in the left diagram. The grey areas are linked with time scale preserving behaviour morphisms to an object (left) and to an object morphism (right). The morphisms e_Λ and e_Ω representing the links to the environment are called *encoding* and *decoding* morphisms accordingly. The encoding morphism "translates" the commands of the environment into commands, which the process part of the object "understands", while the decoding one does the reverse. Should there be other behaviour morphisms with corresponding domains and encoding and decoding morphisms, a common environment for all of them can be set by suitably enlarging the original.



The idea of an environment here depicts it both as a supplier and a recipient of selected subsets of behaviour, without reference to the internal workings of the embedded objects or morphisms (events and/or observations), a notion close to the common concept of a physical environment. In (Ehrich, et al, 1990) this is closer to the definition of an implementation of an object over another. In (Kotsiopoulos, 1993) this corresponds to the Petri Net identification of the CIMOSA constructions.

4 IMPLICATIONS OF THE CATEGORICAL FOUNDATION

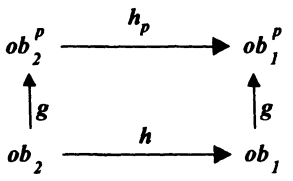
As mentioned in (Kotsiopoulos, et al, 2002) all enterprise modelling languages have intuitive semantics; they represent real world "objects", "processes", "actions". Although what we examine here is formal (mathematical) semantics (taking advantage of the reasoning power of an underlying domain theory), intuitive or "physical layer" semantics cannot be ignored. It is actually through them that such a language is judged by the user.

This puts additional requirements on our formal semantics framework: it must not only be embedded within a theory, but also be broad enough to allow mapping to real physical entities and their actions. The categorical framework we introduced views virtually all quantifiable or machine processable activity within the enterprise as an object and/or object interaction, that is morphisms of behaviour (objects) and morphisms of such morphisms (object morphisms). It is this ability provided by the categorical framework, which allows us to formalise the relationship between an enterprise model and reality.

Suppose we have a very competent observer, who can record all the states of all entities in an enterprise at any single moment in time. Provided

physical objects of the enterprise have been identified, his observation will be just a set of cause and effect behaviour snapshots, representing the causality of enterprise processes. Using the framework of section 2, these can be readily seen as enterprise objects and object morphisms. Actually, the parallel composition of those into a single object morphism, representing the entire enterprise is also possible.

Let us think that a certain enterprise has been selected and that enterprise objects and morphisms between them have correctly been identified and abstracted from real objects and their interactions. To examine issues of integration and monitor overall performance, a more structured expression of the enterprise machinery is desired and therefore a "model" is employed. But what is a model and is there a way of separating good models from bad ones? The representation of objects as observed processes and of interactions as morphisms can be employed to characterise a "correct" model of an enterprise, with respect to the required abstraction.



Definition. Let $h: ob_2 \rightarrow ob_1$ be a morphism between enterprise objects and $g: ob_2 \rightarrow ob_2^p$ be a partial observation morphism on ob_2. Then h_p belongs to a (correct) *model* of the enterprise if for every morphism h between enterprise objects the attached diagram exists and is commutative.

The images of the morphism g are the *objects of the model*, while a composition of those (and their morphisms) is the enterprise model itself.

Indeed, following (Ehrich, et al, 1990), the category *OB* of objects and their morphisms is cocomplete. Parallel composition of objects can therefore be semantically mapped as a colimit in *OB*. It follows that the entire enterprise, as well as its model, can be thought of as an interacting set of two composite objects (representing an input and an output object accordingly) connected via an object morphism.

An aggregation based on more permanent characteristics i.e. a type of those objects together with a corresponding type of morphisms is what we commonly call a *reference architecture* (for enterprise modelling).

Note that the constructs of a single reference architecture cannot always ensure model correctness; indeed, this is the very reason for which different architectures exist: to ensure model correctness, i.e. that each object morphism of the model, such as h_p, satisfies the previous definition. This is particularly important for model-enacted or regulated operation (Kotsiopoulos, 1999), where a correct model implies that:

- Not every action which takes place in reality is modelled
- No action inconsistent with the model does take place.

A related property of an architecture, *completeness*, concerns the suitability of a certain architecture for a purpose or a particular class of enterprises.

This has interesting implications on practical issues such as model re-use and software tool requirements for true support of the modeller. A mathematical logic representation of the criteria involved is possible through model theory and situation theory of meaning (Bernus, et al, 1996). Associating completeness criteria with our correctness principle in a uniform semantic and representational context (objects, morphisms, model theory and situation semantics) remains an interesting open question.

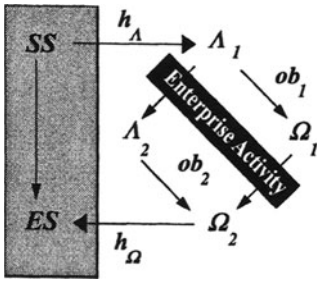
5 EXAMPLE. MAPPING OF ENV 12204 AND CIMOSA SEMANTICS

Both ENV 12204, (CEN, 1995), as well as CIMOSA, (1994), models are based on objects and their properties, themselves being abstractions of enterprise objects. Whether the abstraction is "good" or not is for the model correctness test to say. The formalism is built around object processing blocks, called "(Business/Domain) Processes" or "(Enterprise) Activities" accepting objects as "Function Inputs" and producing objects as "Function Outputs".

We shall realise this by posing additional structures to enterprise objects, thus abstracting them to ENV12204 (CIMOSA) objects. The same will be done with their corresponding morphisms and environments. In our view, *an ENV12204 (CIMOSA) compliant model is a set of category morphisms on ENV12204 (CIMOSA) objects, embedded in an ENV12204 (CIMOSA) environment.*

The method is of importance in its own right as it may be used for the construction or the extension of modelling architectures. We apply a bottom-up approach by starting from the smallest functional unit in ENV12204 the Enterprise Activity.

Consider an object morphism called an *Enterprise Activity (EA)* embedded in an environment called the *Business Environment*. Enterprise Activities act on objects in an input/output fashion and are deployed by the Business Environment, realised by the ENV and CIMOSA as a hierarchical structure performing sequencing operations on them. Although not currently present in the ENV the embedding contains:



- Encoding morphisms "triggered" by start events in *SS*, the set of all *starting events* for the launch of all EA morphisms
- Decoding morphisms valued in *ES*, the set of all attribute-value pairs for the *ending statuses* of the EAs launched by a model

In this way, the Business Environment has a finite alphabet equal to $SS \cup ES$. ENV and CIMOSA map this to a simple discrete event calculus called Procedural Rules, (CIMOSA, 1994), which, in turn, can be mapped to Hoare's Communicating Sequential Processes, (Hoare, 1985) and is termed *Business Process*. Hierarchies can also be built in similar ways. As only events are present at this level, each Business Process is indeed a composite object morphism and its input and output objects are ENV12204 (CIMOSA) compliant objects (schematic diagram on the left).

A further level of decomposition is done by CIMOSA by decomposing Enterprise Activities into *Functional Operations*. Our framework allows this mapping too. We define an object ob_e and call it an *elementary object* such that: $ob_e : e \rightarrow \Omega$, where e is a singleton in the event space and Ω is a singleton in the space of observations consisting of n-tuple vectors with typed components. ob_e acts as an object analysis mechanism mapping objects into named entities with attributes. Indeed, an elementary object is uniquely identified by the name of a single event and a set of attribute-value pairs. Moreover, the attachment of the single event on the time axis is independent of the particular point of attachment.

By the same token, a morphism between elementary objects, called an *elementary morphism*, can be defined and mapped to a computable (in the sense of ENV 12204) structure similar to a callable function of a programming language, or, in CIMOSA terms, a *Functional Operation*. We refer to (Kotsiopoulos, 1995) for full technical details. An (CIMOSA) *Activity Environment* can also be constructed, such that the encoding morphism is a partial observation with identity in its process part and the decoding morphism is an inclusion. We endow this environment with all the control structure of the CIMOSA Activity Behaviour pseudo-code as described in the Technical Baseline (CIMOSA, 1994).

Remark 1. All elementary objects release their events and attribute values to the Activity environment. This is in accordance to the CIMOSA definition of a Functional Operation as a grain of the model which is either fully executed (i.e. its elementary objects released to the environment) or not at all.

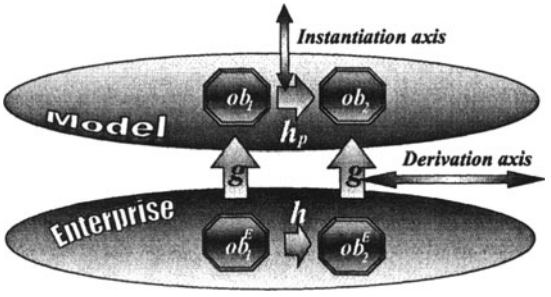
Remark 2. As far as the (CIMOSA) Business Environment is concerned an Enterprise Activity is seen only as a starting event and an ending status, occurring at single moments in time. It follows that for the purposes of this environment, an Enterprise Activity is semantically mapped to an elementary morphism.

Remark 3. CIMOSA considers *"object views"* instead of *"objects"*. Since either of them is constructed through a partial observation morphism on some enterprise object, we chose not to rename here.

Remark 4. Both notions of an environment and an elementary morphism can accommodate in a natural way the call of ICEIMT'97 for level 4 and 5 integration, (Hollocks, et al, 1997). New concepts such as that of an agent, identified by the working group of the 1st workshop of ICEIMT'02 as a potential constituent of the next generation of integrated systems, (Goranson, et al, 2002), could be incorporated as a special property of an environment. The same can be said about the ability of Enterprise Activities in CIMOSA for direct communication between them by exchanging events or messages. In this case, either the Business Environment is considered able to "mediate" for this communication, meaning that the Enterprise Activity is not seen as an elementary morphism any more, or it totally ignores this exchange. It is all a matter of modelling architecture but not of language semantics: the fundamental semantics framework is already there! Similar "environmental" constructions may give models the (now missing, (Goranson, et al, 1997) ability to represent their own state and causality (introspection). Finally, as also observed in (Goranson, et al, 2002), the insight into an object's internal mechanism demanded by level 4/5 integration can be served by suitable object decompositions (e.g. elementary objects and morphisms in the case of CIMOSA Functional Operations).

6 MODEL CORRECTNESS AS AN INVARIANT PROPERTY

Model correctness as introduced here is made possible by the generality of the object morphism characterisation. It also has an interesting invariance property: correctness should be maintained at two axes of the CIMOSA cube, derivation (from requirements to design-implementation) and genericity (from generic to partial-particular). For derivation: all models at different levels should be correct. For genericity: the partial and generic levels should be able to provide classes of morphisms between objects which, if instantiated (particular level), should maintain correctness (attached diagram).



7 EPILOGUE

We argued the case that the cocomplete category of objects and their morphisms can be used as a semantic framework for enterprise modelling languages. The very general setting obtained in this way can be mapped to physical (real world) objects to allow for the physical layer semantics of those languages. At the same time, it can accommodate the formal features of architectures and associated languages such as CIMOSA or their suggested extensions (to accommodate agents for example). New properties of models, such as model correctness, can also be defined.

The author believes that far more can be achieved by the application of the full power of category theory to modelling. To this end, the present paper serves only as an introduction and a call for further work in the future.

8 REFERENCES

Bernus, P. Nemes, L. Morris, R. (1996), *"The meaning of an Enterprise Model"*, in Bernus, Nemes, (Eds.), Modelling and Methodologies for Enterprise Integration, Chapman & Hall.

CEN (1995), ENV 12204, *"Constructs for Enterprise Modelling"* TC 310/WG1.

CIMOSA Association, (1996), *"CIMOSA, Technical Baseline"*, private publication.

Ehrich, H.D. Goguen, J.A. Sernadas, A. (1990), *"A Categorial Theory of Objects as Observed Processes"*, in Foundations of Object-Oriented Languages, REX School/Workshop Proceedings.

Goranson, H.T. (1997), *"ICEIMT in Perspective – 92 to 97"*, in Enterprise Engineering and Integration, Proceedings of ICEIMT'97, Springer-Verlag.

Goranson, H.T. (Ed.) Huhns. M, Nell, J.G. Panetto, H. Tormo Carbó, G. Wunram, M. (2002), *"A Merged Future for Knowledge Management and Enterprise Modeling"*, this conf.

Hoare, C.A.R., (1985), *"Communicating Sequential Processes"*, Prentice-Hall.

Hollocks, B.W. (Ed), Goranson H.T., Shorter D.N., Vernadat F.B. (1997), *"Assessing Enterprise Integration for Competitive Advantage – Workshop 2, Working Group 1"*, in Enterprise Engineering and Integration, Proceedings of ICEIMT'97, Springer-Verlag.

Kotsiopoulos, I.L. (1993), *"Theoretical aspects of CIMOSA modelling"*, CIM Europe Conference, Amsterdam, in Kooij C., MacConaill, P.A., Bastos J. (Eds.),"Realising CIM's industrial potential", IOS PRESS.

Kotsiopoulos, I.L. (1996), *"Objects and Environments in Dynamic CIMOSA Models"*, in Bernus, P. Nemes, L. (Eds.), *"Modelling and Methodologies for Enterprise Integration"*, IFIP TC5 Working Conference on Modelling and Methodologies for Enterprise Integration, IFIP/IFAC Task Force, Heron Island, Australia, Chapman & Hall.

Kotsiopoulos, I.L. (1999), *"Railway Operating Procedures: Regulating a Safety-Critical Enterprise"*, Computers in Industry, 40.

Kotsiopoulos, I.L. (Ed.), Engel, T. Jaekel, F-W. Kosanke, K. Mendez, J-C. Ortiz Bas, A. Petit, Raynaud, P. (2002), *"Steps in Enterprise Modelling – A Roadmap"*, this conference.

Petit M (Ed.), Goossenaerts, J. Gruninger, M. Nell, J.G. Vernadat, F.B. (1997), *"Formal Semantics of Enterprise Models – Workshop 4, Working Group 2"*, in Enterprise Engineering and Integration, Proceedings of ICEIMT'97, Springer-Verlag.