Formalisation of enterprise modelling standards using UML and the B method.

Hervé Panetto, Jean-François Pétin, Dominique Méry

Proceedings of the 8th International Conference on Concurrent Enterprising, ICE2002, 17th-19th June, 2002, Rome, Italy, pp. 93-101, ISBN : 0 85358 113 4

Formalisation of enterprise modelling standards using UML and the B method

Hervé Panetto[1], Jean-François Pétin[1], Dominique Méry[2]

[1] *CRAN – CNRS UMR 7039, University Henri Poincaré Nancy I, BP 239,*

F-54506 Vandoeubre-les-Nancy, {Herve.Panetto, Jean-Francois.Petin}@cran.uhp-nancy.fr

[2] *LORIA – CNRS UMR 7503, University Henri Poincaré Nancy I, BP 239,*

F-54506 Vandoeubre-les-Nancy, Dominique.Mery@loria.fr

Abstract

This paper deals with the verification of the existing enterprise modelling standards. Our approach is based on the UML meta-modelling of enterprise standards in order to establish enterprise constructs and to use the formal B method to cover verification issues. Two points are discussed : the checking of the global consistency of the standard itself, and the verification of the instantiation of constructs to design particular enterprise models. This work is illustrated using the ENV12204/N177 particular enterprise constructs standard.

Keywords

Enterprise construct, ENV12204, UML, OCL, B Method, verification

1 Introduction

Most major Enterprise Modelling and Integration projects (e.g. ESPRIT/CIMOSA, ICAM/IDEF, IPK/IEM, ESPRIT/CCE-CNMA, LUT/CIM-BIOSYS, PERA, GRAI/GIM, GERAM) have demonstrated the necessity of developing enterprise models to support analysis, design and management of business processes that are executed in companies. Representing the reality of an extended enterprise through the construction of enterprise models requires to capture the whole needed and produced information, processes and its behaviours, organisation constraints with the goal of providing an efficient operation support [Jochem, 2002].

The consistency between the various representations involved in enterprise modelling is partially reached by providing unified notations such as the UML [UML, 1997] or integrated reference architectures such as CIMOSA [Kosanke, Vernadat, Zelm, 1999], GERAM, GRAI/GIM. These notations are able to deal with syntactic interactions between the different modelled points of view, but they suffer from a lack of mathematical foundations to check its semantics interactions. For example, class, state-transition and collaboration diagrams are standardised in UML notation, even if the specification modelled in some diagrams can be not compliant with other ones. First way consists in providing these unified notations and frameworks with consistent semantics and verification mechanisms [Vernadat, 1998]. Moreover, this approach allows one quantitative evaluation of enterprise models with regards to expected properties such as processes performance, safety, capability, etc.

A complementary way is to consider that any enterprise models result from an instantiation of generic constructs that are supposed to be correct with respect to basic knowledge about enterprise modelling. This approach relies on the meta-modelling of the syntax and the semantics of the “objects” involved in enterprise models. The modeller is then assisted by a methodical approach (cf. Figure 1) that promotes the use of validated components libraries (constructs) and their association rules to design enterprise models.

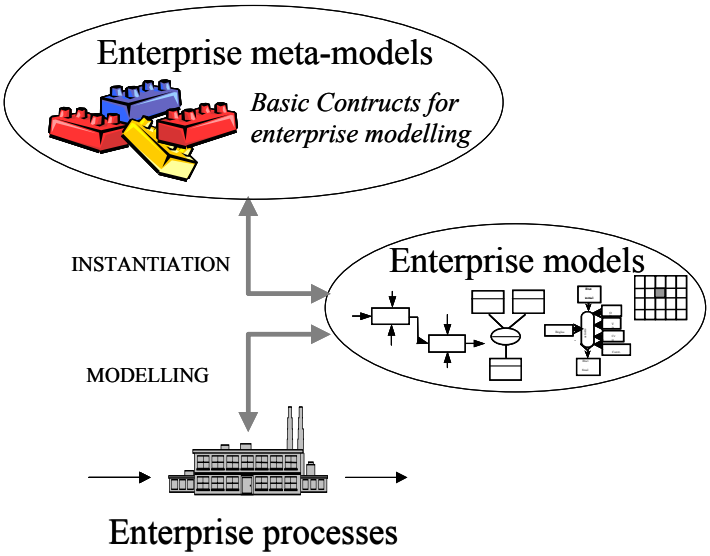


Figure 1 : Enterprise modelling approach

For the syntactic definitions of constructs, UML proposes an extensibility mechanism to formalise meta-classes and their associations (UML Profiles and OCL constraints), and models such as CIMOSA, GERAM, GRAI/GIM, IEM have contributed to standardisation and to unification efforts to harmonise concepts and terminology (ENV 12204 [ENV 12204, 1995] and its reworked version [N177, 2002], UEML (Unified Enterprise Modelling Language) [Panetto, 2002] [Chen, Vallespir, Doumeingts, 2002]). These efforts could contribute to the definition of an "Enterprise backbone" (cf. Figure 2) in the same way as the EAI (Enterprise Application Integration) specification [EAI, 2002], that helps in integrating enterprise models and tools such as ARIS ToolSet, Bonapart, MOOGO, GRAI tool, etc.

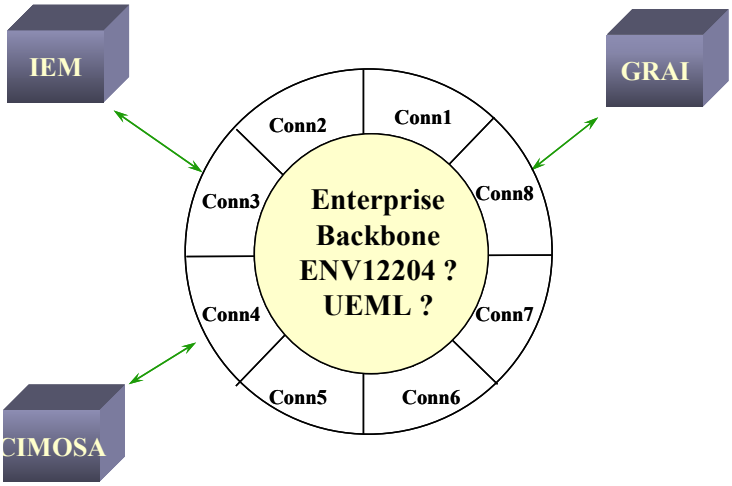


Figure 2 : UEML backbone

However, these models and tools need to complete their syntax with a formalisation of its semantics, in order to improve interoperability. Our approach then is based on the UML meta-modelling of the existing ENV 12204 rework in order to define consistent semantics for enterprise models. Formal verification issues are proposed to be supported by joining to UML semi-formal meta-models a B formal description [Abrial, 1996] that provides underlying proof mechanisms. Therefore, it becomes a necessity to define a unified language for universal use by business users as well as within the enterprise modelling community and which would address these problems. This work aims then to the development of a Unified Enterprise Modelling Language (UEML), by analogy with the development of the UML devoted to conceptual systems modelling.

2 ENV 12204/N177 standard for enterprise modelling

The ENV 12204 standard defines a set of constructs together with its relationships and its attributes, using textual templates and an UML class diagram graphical representation (cf. Figure 3). The standard also defines behaviour rules which are a specialisation of the relation construct that describes the sequencing relationships of constituent activities. There are five types, namely serial, junction, loop, conditional and exception. These constructs define an interpreted language specifying business processes behaviours.

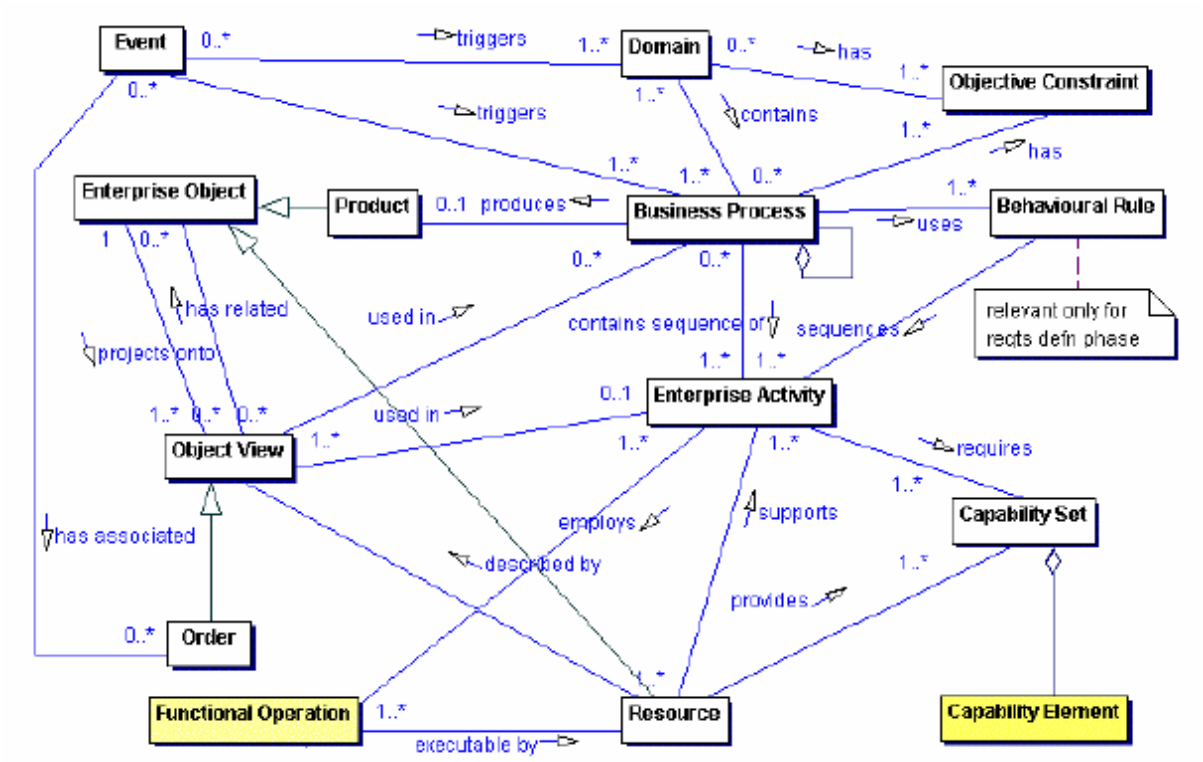


Figure 3 : Part of the current ENV 12204 revision constructs model [N177, 2002]

3 UML modelling of ENV 12204/N177

3.1 A common language for enterprise modelling.

The Unified Modeling Language (UML), an OMG standard, is a widely adopted and used modelling language. The UML emerged from the unification of various object-oriented methods that occurred in the 1990s and is defined by 9 languages. In this work, we use the *Class Diagram* which defines objects with its attributes, its operations and its relationships and the *State-transition Diagram* which describes the dynamic behaviour of operations. Moreover UML standard specifies the Object Constraint Language that supports the description of *constraints* to be applied on object-oriented models. OCL is a formal constraint language based on 1st order predicate logic. It formalises constraints which can be a restriction on a static relationship between one or more values of some objects attributes, or a dynamic guard that defines the pre and/or post-condition to be satisfied by an operation.

In order to extend its meta-model, UML provides an expendability mechanism through the definition of so called “Profiles”. A profile contains one or more related extensions of standard UML semantics. These are normally intended to customize UML for a particular domain or purpose. They can also contain data types that are used by tag definitions for informally declaring the types of the values that can be associated with tag definitions.

Indeed, these extension mechanisms are a means for refining the standard semantics of UML and do not support arbitrary semantic extension. They allow the modeller to add new modelling elements to UML for use in creating UML models for process-specific domains such as enterprise models. Moreover, as the UML specification relies on the use of well-formedness rules to express constraints on model elements, this profile uses the same approach. The constraints applicable to the profile are added to the ones of the stereotyped base model elements, which cannot be changed. Constraints attached to a stereotype must be observed by all model elements branded by that stereotype. If the rules are specified formally in a profile (for example, by using OCL for the expression of constraints), then a modelling tool may be able to interpret the rules and aids the modeller in enforcing them when applying the profile. UML is currently used to define common semantics to the existing various frameworks [Panetto, Mayer, Lhoste, 2000] within the scope of the UEML (Unified Enterprise Modelling Language) international IFAC-IFIP Task Force.

3.2 UML formalisation of ENV 12204/N177 constructs

Each construct is modelled by an UML class associated with OCL constraints that describes the constraints to be verified by a particular application.

The following example (cf. Figure 4) formalises that an “Enterprise Object” (EO) is defined by some attributes (identifier, name, description, a set of properties which can either be of type “String” or another EO). It can be decomposed by other ones (which are part of it). Moreover, subtypes of an EO (“is-a” relationships) may exist. The OCL invariant formalises well-formedness rules verifying that each EO could not be part of itself and that if an EO has a property which is another EO, then this last one could not be part of the former one.

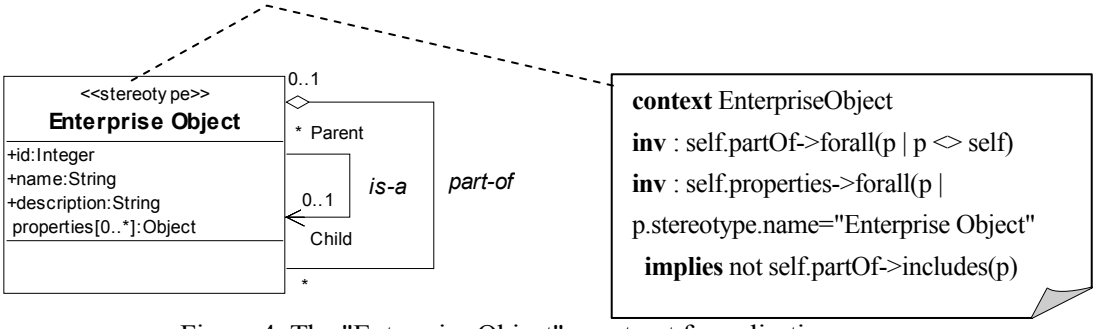


Figure 4: The "Enterprise Object" construct formalisation

4 Checking ENV12204/N177 with B Method

4.1 The B method

Introduced by (Abrial, 1996), the B Method is a formal method for the specification, the design and the implementation of software applications which supports properties proofs and refinement mechanisms (Cancell, Mery, Weinzoepflen, 2001). The B language is based on the first order logic and the set theory. The first one includes classical operators of a propositional calculus ($\neg$ P, P $\vee$ Q, P $\wedge$ Q, P $\Rightarrow$ Q, P $\Leftrightarrow$ Q) and quantifiers ($\forall$ X . p, $\exists$ X . p). The second one includes set-theoretical operators (S $\cup$ T, S $\cap$ T, S $\subset$ T, x $\in$ S, #S, …), functions and relations defined as subsets of Cartesian product (A $\leftrightarrow$ B $\cong \mathbb{P}$ A $\times$ B), and generalised substitutions ([S], [x:= f(y)]).

The B Method provides the Abstract Machine Notation (A.M.N.). Data, functions and relationships issued from set theory aim are described using SETS, VARIABLES and PROPERTIES clauses. Processing part is described using OPERATIONS clause that is based on generalised substitutions that allow modifying an element or a set. An operation can

be pre-conditioned[1] (or guarded) by a predicate *Pre* (i.e. the result of the operation is established only if the predicate *pre* is satisfied : [Pre | S] I <=> Pre & [S] I) . INVARIANT properties, described as a predicate, can be proved[2] as being satisfied (or maintained) through the execution of an operation.

B proof mechanisms are supported by a theorem prover. The invariant to be proved give rise to proofs obligations which means the underlying hypotheses needed for the proof. If these hypotheses are included in the known theories, the invariant is proved and considered as a new theorem; if not, user operation is requested to help the prover by suggesting proving strategy or by correcting the initial B specification.

The refinement mechanism of the B method provides support for an incremental specification of models, by proving that invariant properties of a given abstract model are preserved by a more concrete model adding specification details. The features (inclusion, inheritance, …) of the B method allows a modular specification using visible or shared variables between several machine and called operations.

4.2 From UML to B

The set theory basis and the object oriented features make easy an automatic translation from object diagram, such as OMT [Facon, Laleau, Nguyen, 1998] or UML class diagram [Meyer, 2001] into the B language.

4.2.1 Class diagram

UML classes are translated into B machines where a class is declared as a set (that will contain its instances), the attributes are defined as relationships (in the set theory meaning) between its values domains and the class set. UML and B operations are equivalent concepts. (cf. Figure 5). Note that the relationship between attributes and class is defined as an invariant that must be always preserved whatever the operation modifies.

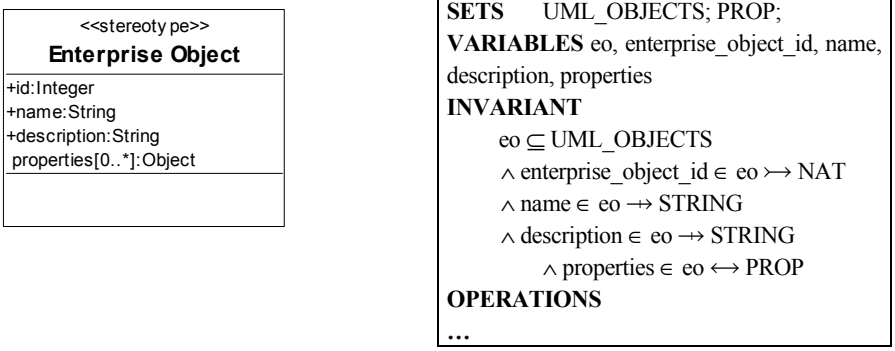


Figure 5 : B formalisation of a UML class

In the same way, relationship between two UML classes give rise to a B machine where the composition mechanism *USES* enables access to the variables and invariant of the two associated classes and where invariant characterise the relationship between the two classes (cf. Figure 6). Its multiplicities (referential integrity) are given by the nature of relationship simple relation (↔), bijective (⤖), injective (↣), surjective (↠), combined with partial function (⇸) and total function (→), and with declaration of domain (dom) and range (ran) if needed. For example, the "properties" relationship (eo ↔ PROP) means that the attribute "properties" of the class "Enterprise Object" is multi-valuated.

[1] preconditioned substitution [Pre | S] I ⇔ Pre & [S] I

[2] operation S maintains the invariant I ∧ termination(S) ⇒ [S] I

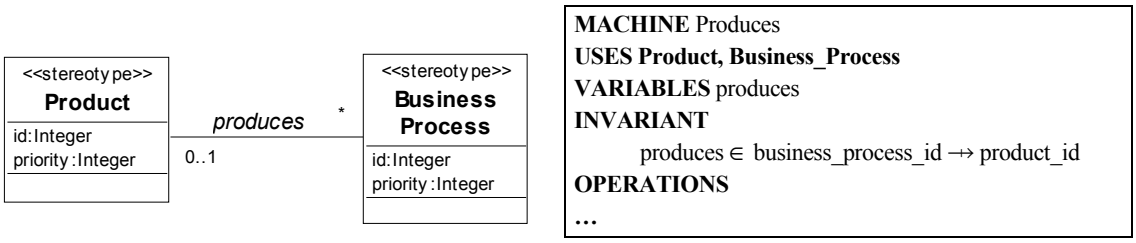


Figure 6 : B formalisation of UML classes association

4.2.2 OCL constraints

At least, OCL constraints are described by a logic predicate included in a B *invariant* in order to describe specific constraints to be applied to the relationship between attributes values. Using the previous example (cf. Figure 6), a constraint specifies that if a "Business Process" (BP) produces a specific "Product" (P), then, this product priority has to be equal to the BP one. Figure 7 shows the OCL specification of that constraint and its B formalisation.

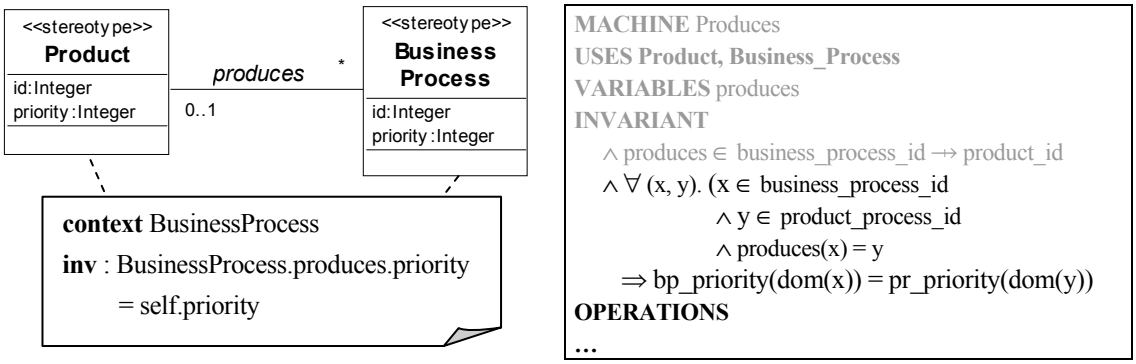


Figure 7 : B formalisation of an OCL constraint

This translation from UML to B can then be used to formalised ENV12204 UML representation in order to check the global consistency of the standard and to verify the conformance of instantiation rules with regards to the referential integrity and OCL constraints defined in the ENV12204/N177 standard.

4.3 B formalisation of ENV 12204/N177

Let us take the example of Enterprise Object given by Figure 4. Formalisation of the Enterprise Object class is given by Figure 5.

First step is to complete this formalisation by defining the B machine (cf. Figure 8) associated to the relationship part-of between EO class and itself and to integrate into its invariant a predicate that describes OCL constraints of Figure 4.

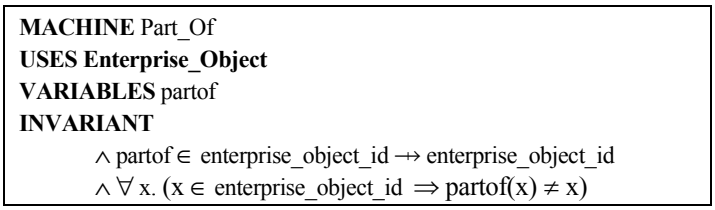
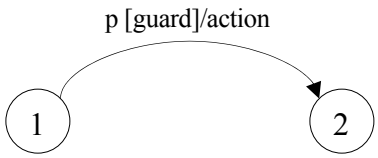


Figure 8 : B machine of the part-of relationship

In this step, this formalisation of ENV12204/N177 is efficient for checking the global correction of the UML model that is standardised. Indeed, the dynamics of the enterprise business processes is described by an informal "Behaviour rules" construct that is not demonstrated to be compliant with the information structure of ENV12204/N177.

Benefits of our formalisation consists in addition of dynamics features to the B formalisation using B operations. These last ones are able to specify basic operations such as object creation, deletion, and modification but also more complex sequential rules (cf. Figure 9)

related to the dynamics of enterprise processes. B proof mechanisms allow us to guarantee that the dynamics specification of objects maintains the static specification of constructs.



MACHINE FSA
SETS STATE; T; GUARDS */* set of states, triggering events and guards*/*
VARIABLES st1, st2, guard, p, s
INVARIANT st1 ⊆ STATE ∧ st2 ⊆ STATE ∧ guard ∈ GUARDS ∧ s ∈ STATE ∧ p ∈ T
OPERATION tr(s,p) =
PRE guard ∧ s ∈ st1 ∧ p ∈ T
THEN st1 := st1 – {s} || st2 := st2 ∪ {s} || action
END

Figure 9 : B formalisation of a state-transition diagram [Lano, 1996]

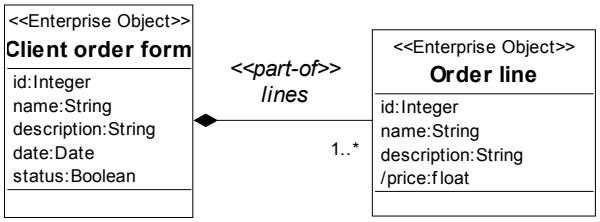
Second step is related to the instantiation of constructs for the design of a particular enterprise model. The B formalisation is then expected to ensure that any enterprise model instantiated from the ENV 12204/N177 is correct with regards to the modelling rules specified in this standard. Let us use again the example given in Figure 4. The instantiation of the meta-class "Enterprise Object" produces two classes named "Client order form" and "Order line". Its formalisation leads to two B machines ("ClientOrderForm" and "OrderLine") that refer to the construct machine ("Enterprise_Object") using the clause *EXTENDS* and a dot notation (cf. Figure 10).

MACHINE ClientOrderForm
EXTENDS ClientOrderForm.Enterprise_Object;
VARIABLES date, status;
INVARIANT
date ⊆ ClientOrderForm.properties
∧ status ⊆ ClientOrderForm.properties
∧ date ∈ ClientOrderForm.eo → DATE
∧ status ∈ ClientOrderForm.eo → BOOL

MACHINE OrderLine
EXTENDS OrderLine.Enterprise_Object;
VARIABLES price;
INVARIANT
price ⊆ OrderLine.properties
∧ price ∈ OrderLine.eo → NAT

Figure 10 : B Instantiations of "Enterprise_Object"

The instantiation of the "part-of" relationship produces the "lines" relationship between the two previous classes (cf. Figure 11). The B formalisation follows the same principle (cf. Figure 11) applied on the "Part_Of" machine described in Figure 8.



MACHINE Lines
EXTENDS Lines.Part_Of;
USES ClientOrderForm, OrderLine;
INVARIANT
Lines.partof ∈ OrderLine.enterprise_object_id
⇸ ClientOrderForm.enterprise_object_id

Figure 11 : B formalisation of an instantiated model

The major benefit of this formalised instantiation is that the correctness of the particular enterprise model can be proved with regards to the OCL constraint that is specified in the generic construct. Indeed, the verification of the invariant defined in the "Lines" machine implies the underlying verification of the invariant (including the OCL constraint) that is defined in the "extended" generic machine "Part_Of". However, an intensive combination of B structuring mechanisms (USES, EXTENDS, …) with the refinement mechanism leads to numerous proof obligations which are not easily computed by the B foundations supported by the Atelier B tool[3]. This aspect could be a limit for the formalisation of the ENV

3 Atelier B is a ClearSy product

12204/N177 using of the B Method and justifies further work in this area, namely on the refinement.

5 Conclusion

Our approach combines UML widely used diagrams with a more formal formalism in order to provide semantics to the ENV 12204/N177 enterprise modelling standard. Two major benefits have been illustrated : the verification of the standard itself when adding dynamics to enterprise constructs, and a safe instantiation of the standard to any particular companies that respects the modelling rules of the constructs. On-going work focuses on applying this kind of formalisation (joint use of UML and B) on a real case study in order to investigate the limits of our approach and to use the B refinement. Another important aspect for a wide dissemination of these techniques within the enterprise world consists in hiding, as far as possible, the use of the B formalism in order to provide the various actors, that are not familiar with formal language, with a proved UML enterprise representation.

References

Abrial J.R. : The B Book: Assigning Programs to Meanings. Cambridge Univ. Press, 1996

ENV 12204 : CEN European Pre-Standard, Advanced Manufacturing Technology, Systems Architecture, Constructs for Enterprise Modelling, TC 310/WG1, 1995

EAI : UML Profile and Interchange Models for Enterprise Application Integration (EAI) Specification, Object Management Group standard, 2002, Web page : http://www.uml.org

Cancell D.; Mery D. Weinzoepflen A. : Modélisation et analyse de la documentation technique d'un système, MSR'2001, Modélisation des Systèmes Réactifs, pp. 481-496, ISBN : 2-7462-0329-4, Toulouse, France, October 17th-19th, Hermes Sciences, 2001

Chen D.; Vallespir B.; Doumeingts G. : Developing an Unified Enterprise Modelling Language (UEML) – Requirements and Roadmap, 3rd IFIP working conference on infrastructures for virtual enterprises (PRO-VE'02), Protugal, May 1st-3rd, 2002

Facon P., Laleau R., Nguyen H.P. : The Invoicing System Problem : From OMT Diagrams to B Specifications, in International Workshop on Comparing Systems Specification Techniques "What questions are prompted by ones particular method of specification ?", Nantes, France, 1998.

Jochem R. : Common Representation through UEML - Requirements and Approach, Proceedings of ICEIMT international conference, Valencia, Spain, April 24th-26th, 2002

Kosanke K.; Vernadat F.; Zelm M. : CIMOSA: enterprise engineering and integration, Computers in Industry, Volume 40, Issues 2-3, Pages 83-97, November 1999

Lano K. : The B language and method, a guide to practical formal development, Springer Verlag, ISBN 3-540-76033-4, 1996.

Meyer E.: Développement formels par objets : utilisation conjointe de B et UML, thèse de l'Université de Nancy 2, 23 mars 2001

N177 : ENV 12204 CEN European Pre-Standard revision, Advanced Manufacturing Technology, Systems Architecture, Language Constructs for Enterprise Modelling, TC 310/WG1, 2002, restricted

Panetto H.; Mayer F.; Lhoste P. : Unified Modeling Language for meta-modelling : towards constructs definitions, Proceedings of ASI'2000 Conference, September 18-20, 2000, Bordeaux, France, ISBN 960-530-050-8.

Panetto H. : UML semantics representation of enterprise modelling constructs, Invited conference, Proceedings of ICEIMT international conference, Valencia, Spain, April 24th-26th, 2002

UML : Unified Modeling Language, Object Management Group standard, 1997, Web page : http://www.uml.org

Vernadat F. : The CIMOSA languages, Handbook of Information Systems. Bernus P., Mertins K. and Schmidt G. Ed., Springer Verlag, pp 243-263, 1998.