Integrating Object-Oriented Modeling Techniques with Formal Specification Techniques

German-Argentinean Scientific-technological Cooperation

Claudia Pons, Gabriel Baum and Miguel Felder

Lifia, Universidad Nacional de La Plata
Calle 50 esq.115, 1er.Piso,
(1900) La Plata, Buenos Aires, Argentina

Ralf-D Kutsche and Susanne Busse

Technical University of Berlin, FB Informatik
Einsteinufer 17
D-10587 Berlin, Germany

1. Introduction

The increasing complexity of software systems makes their development complicated and error prone. A widely used and generally accepted technique in software engineering is the combination of different models (or views) for the description of software systems. The primary benefit of this approach is to model only related aspects (like structure or behavior). Using different models clarifies different important aspects of the system, but it has to be taken into consideration that these models are not independent and they are semantically overlapping.

The models constitute the fundamental base of information upon which the problem domain experts, the analysts and the software developers interact. Thus, it is of a fundamental importance that it clearly and accurately expresses the essence of the problem. On the other hand, the model construction activity is a critical part in the development process. Since models are the result of a complex and creative activity, they tend to contain errors, omissions and inconsistencies. Model verification is very important, since errors in this stage have an expensive impact on the following stages of the software development process.

Models are constructed using a modeling language (which may vary from natural language to diagrams and even to mathematical formulas).

The success of software development methods, such as Object Oriented Analysis (Coad and Yourdon, 1991), Object Oriented System Analysis (Shlaer and Mellor, 1988), Object Modeling Technique (Rumbaugh et al., 1991), Booch's design method (Booch, 1994) and the *Rational Unified Process* (Jacobson et al., 1999) are mainly based on their use of intuitively appealing modeling constructs and rich structuring mechanisms, which are easy to understand, apply and transmit to the customers. However, the lack of precise semantics for the modeling notations used by these methods can lead to several problems:

- Misunderstanding of models: a reader's interpretation could be different to the model creator's interpretation, even if recommend textual explanations (e.g. data dictionary) are given by the methodology.

- There are several separate models (e.g. static, dynamic, functional models) that are difficult to integrate and maintain consistent;

- Models cannot be validated (neither informally nor formally) leading to insecure systems where, for example behavior under certain conditions may be unpredictable.

- Since the meaning of some language constructions are not accurately defined, the people involved in the project often waste time discussing the different possible interpretations that can be allocated to the language.

On the other hand, formal languages for modeling, such as Z (Spivey, 1992), VDM (Jones, 1990), F-Logic (Kifer and Lausen, 1990), DS-Logic (Wieringa and Broersen, 1998) has a well-defined syntax and semantics. However, its use in industry is not frequent. This is due to the complexity of its mathematical formalisms, which are difficult to understand and communicate. In most cases, experts on system domain who decide to use a formal notation, center their effort upon the managing of formalism instead of focusing on the model itself. This leads to the creation of formal models that do not properly reflect the real system.

As a consequence, it has been proposed to combine the advantages of both approaches, intuitive graphical notations on the one hand and mathematically precise formalisms on the other hand, in development tools. The basic idea for this combination is to use mathematical notation in a transparent way, hiding it as much as possible under the hood of graphical notations. This approach has advantages over a purely graphical specification development as well as over a purely mathematical development because it introduces precision of specification into a software development practice while still ensuring acceptance and usability by current developers.

2. Project description

We address the problem of gaining acceptance for the use of an unfamiliar formalism by providing sound semantics to a well-known graphical method. The main components of the proposal are rules to associate syntactic structures of the modeling language with elements within a formally defined semantic domain.

The main advantage of the semantic proposal regarding the others, resides in that graphic language is turned into a formal language hence, thus the specifications expressed in a graphic language can be formally analyzed to early find out contradictions and ambiguities in the software development process. One of the keys to the success of this proposal resides in hiding the mathematical notation, as much as possible, behind the graphic notation. For example, it should be possible to use formal semantics to develop CASE tools. Only language developers should use formalism to build the CASE tools and justify their correction, while application software developers could handle graphic models with no need to know the underlying mathematical formalism.

The Unified Modeling Language UML (1999) is a standard graphic language for modeling and specifying object-oriented systems. The language consists of a set of constructs common to most object-oriented languages. From the standardization of the UML active discussions have risen about the semantics accuracy of its constructions. While the Object Management Group OMG was responsible for the standardizing of the UML as notation, the semantics of the UML is still a research issue.

There are an important number of theoretical works - see, for instance, (Muller and Bezivin, 1998) and (France and Rumpe 1999) - that deal with different part of UML, formally defining its syntax and semantics. However, there is still a long way to run regarding this matter. It is particularly hard to compare the results of the respective articles, and it is even harder to combine such results with the aim of obtaining a semantic standard for UML. This difficulty arises because of the different works that use diverse formal methods (or languages), or cover a notation subset, or assume a particular system subclass to be specified. However, an important amount of the proposals can be classified in two groups: formalizations based on the model and formalizations based on the metamodel. We explain this classification in the following section.

Formalizing modeling languages. Classification of approaches

A number of approaches for giving semantics to modeling languages (specially the UML) can be classified in two different groups: model-based and *meta-model-based* approaches. This classification is inspired from the four levels in the architecture of modeling notations (UML, 1999). The main difference between these approaches is the focus of the formalization. Formalizations in the first group concentrate their attention on the model level, while formalizations in the second group focus on the metamodel level:

- In the *model-based* approaches (Moreira and Clark, 1994; France et al., 1997; Waldoke et al., 1998; Wieringa and Broersen, 1998; Lano and Biccaregui, 1998; Kim and Carrington(a), 1999) the individuals in the semantic domain are the business objects, for example accounts and clients of a bank. (i.e. the formalization focuses on the particular system that is being described).
- In the *meta-model-based* approaches (Evans et al., 1999; Breu et al., 1997; UML, 1999; Evans et al., 1998; Kim and Carrington(b), 1999) the objective is to give a precise description of core concepts of the graphical modeling notation and provide rules for analyzing their properties. The individuals appearing in the semantic domain are modeling elements, such as classes, attributes, operations, associations, generalizations, etc. (i.e. the formalization is focused on the language itself instead of on any particular system described by the language).

Our approach

We have defined the M&D-theory, a proposal for giving formal semantics to the UML. The basic idea behind this formalization is the definition of a semantics domain integrating both the model level and the data level. In this way, both static aspects and dynamic aspects of either the model or the modeled system, can be described within a first order formal framework.

3. Work and time schedule

The project includes the following research topics:

a) Developing a conceptual framework for the integration (duration: 4 months):

The UML consists of several languages (e.g. class diagrams or statecharts). It is necessary to develop a conceptual framework for the integration of these languages. Based on this framework, the formalization of each individual language as well as the integration with other languages can be achieved.

b) Formalization of the structural part of the UML (duration: 4 months):

The UML provides graphical notation for class diagrams. Using the envisaged formalization it is possible to define whether a class diagram fulfills general consistency constraints, e.g. absence of circular inheritance between classes. Besides such static constraints it is necessary to define the impact of the structural model on the behavioral parts.

c) Extending the UML by formal specification techniques (duration: 4 months)*:*

The underlying semantical model is extended by a suitable predicate logic. The logic is integrated in the structural model. This enables us, for example, to express class invariants based on the name space constituted by the structural model. In this way, the integrated models can also serve as basis for type-checking facilities.

d) Introducing further semantical objects (duration: 6 months)*:*

It is useful to introduce further implicit semantical objects. These objects do not have representations on the modeling language level but they are necessary in order to relate system states at different moments in time. The semantics of operations can be defined as a relation on these system states.

e) Integration of graphical dynamic description techniques (object-oriented Statecharts, message sequence charts) (duration: 6 months)*:*

Graphical dynamic description techniques are highly important for expressing system behavior in a user-friendly manner. These techniques should be integrated with the structural model. The UML uses the concepts of object-oriented statecharts as recently defined by Harel [Harel and Gery, 1997]. The meaning of message sequence charts as used in the UML is up to now only sketched. Furthermore, use cases has been shown in practical context although the corresponding language is less developed in the UML up to now.

4. Conclusion

Due to the missing formal foundations of the Unified Modeling Language UML the syntax and the semantics of a number of UML constructs are not precisely defined. The aim of this project is to produce a rigorous object-oriented analysis technique that combines the UML with a formal object oriented specification language, ensuring that the integrated technique is accessible to main stream software engineers.

The principal benefits of the proposed formalization can be summarized as follows: the different views on a system are integrated in a single formal model. This allows us to define rules of compatibility between the separate views, on syntactical and semantic level. Using formal manipulation, it is possible to deduce further knowledge from the specification. The faults of specifications expressed using a user-friendly notation can be revealed using analysis and verification techniques based on the formal kernel model.

References

Alencar, A. and Goguen, J., OOZE: an object-oriented Z environment, ECOOP'91 Proc., Lecture Notes in Computer Science vol.512, Springer-Verlag, (1991).

Booch, G., Object Oriented Analysis and Design with Applications, Second Edition, Addison-Wesley Publishing Company, Inc, (1994).

Breu,R., Hinkel,U., Hofmann,C., Klein,C., Paech,B., Rumpe,B. and Thurner,V., Towards a formalization of the unified modeling language. ECOOP'97 procs., Lecture Notes in Computer Science vol.1241, Springer, (1997).

Coad,P. and Yourdon,E., Object Oriented Analysis, Yourdon Press, Englewood Cliffs,NJ, (1991).

Cook,S. and Daniels,J., Let's get formal, Journal of Object-Oriented Programming(JOOP), July-August, (1994).

Duke,R., King,P., Rose,G. y. Smith,G., The Object-Z specification language, T.Korson, V.Vaishnavi and B.Meyer, editors, Technology of Object-Oriented Languages and Systems:TOOLS 5. Prentice Hall, (1991).

Evans,A., France,R., Lano,K. and Rumpe, B., Developing the UML as a formal modeling notation, UML'98 Beyond the notation, Muller and Bezivin editors, Lecture Notes in Computer Science 1618, Springer-Verlag, (1998).

Evans,A., France,R., Lano,K. and Rumpe,B., Towards a core metamodelling semantics of UML, Behavioral specifications of businesses and systems, H,Kilov editor, , Kluwer Academic Publishers, (1999).

France,R., Bruel,J. and Larrondo-Petrie. An integrated object-oriented and formal modeling environment, Journal of Object Oriented Programming (JOOP), 10(7), (1997).

France, R. and Rumpe, B. editors, Proceedings of the UML'99 conference, Beyond the Standar, Colorado, USA, Lecture Notes in Computer Science 1723, Springer-Verlag (1999).

Goldsack,S. and Kent,S., Formal Methods and Object Technology", Chapter 3: LOTOS in the Object-oriented analysis process. Editors S.J. Goldsack, S.J.H. Kent. Serie FACIT, Springer-Verlag, (1996).

Harel David and Gery, E.. Executable Object Modeling with Statecharts. IEEE Computer, 30(7):31{42, (1997).

Jacobson, I, Booch, G, Rumbaugh, J. The Unified Software Development Process, Addison Wesley. ISBN 0-201-57169-2 (1999)

Jungclaus,R., Saake,G., Hartmann,T., Sernadas,C., TROLL- a language for o-o specifications of information systems. ACM Transactions on IS, vol.14 no.2. (1996).

Jones,C., Systematic software construction using VDM. Prentice Hall, (1990).

Kifer,M. and Lausen,G., F-Logic: a higher order language for reasoning about objects, inheritance and scheme. Proceedings of the ACM SIGMOD symposium on principles of database systems, SIGMOD RECORD. Vol.18, No.6, (1990).

Kim, S. and Carrington,D., Formalizing the UML Class Diagrams using Object-Z, proceedings UML'99 Conference. Lecture Notes in Computer Sciencie 1723, (a) second part of the paper, (b) first part of the paper ((1999).

Lano,K., Z++, An object-oriented extension to Z. In John Nicholls, editor, Z user workshop, Oxford 1990. Workshops in Computing, Springer Verlag, (1991).

Lano,K., and Biccaregui,J., Formalizing the UML in Structured Temporal Theories, Second ECOOP Workshop on Precise Behavioral Semantics, TUM-I9813, Technische Universitat Munchen, (1998).

Meseguer,J., Winkler,T., Parallel Programming in Maude. Proceedings of Research Directions in High Level Parallel Programming Languages. France, (1991).

Moreira,A. and Clark,R., Combining Object_Oriented Analysis and Formal Description Techniques, In 8th European Conference on Object Oriented Programming, Procs. Lecture Notes in Computer Science821, Springer, (1994).

Muller. P. and Bezivin. J. editors, Proceedings of the UML'98 conference, Beyond the notation, Mulhouse. France. Lecture Notes in Computer Science 1618, Springer-Verlag (1998).

Pastor,O. and Ramos,I., Oasis 2.2 : A Class-Definition Language to Model Information System Using an Object-Oriented Approach". SPUPV-95.788, Universitat P. de Valencia. (1996).

Pons,C., Baum,G., Felder,M., Integrating object-oriented model with object-oriented meta-model into a single formalism, Second ECOOP Workshop on Precise Behavioral Semantics, European Conference on Object-oriented Programming, Brussels, Belgium, LNCS, (1998).

Pons,C., Baum,G., Felder,M., Foundations of Object-oriented modeling notations in a dynamic logic framework, Fundamentals of Information Systems, Chapter 1, T.Polle,T.Ripke,K.Schewe Editors, Kluwer Academic Publisher, (1999).

Pon, C. , Baum,G. , Felder,M. and Kutsche, R., Formalizing Evolution of UML Models, OOPSLA99 Workshop on Behavioral Semantics, Denver, (1999).

Reggio,G. and Larosa,M., A graphic notation for formal specification of dynamic systems, proceedings of FME'97. Lecture Notes in Computer Science 1313, Springer.(1997).

Rumbaugh,J., Blaha,M., Premerlani,W., Object Oriented Modeling and Design, Prentice Hall, (1991).

Shlaer,S. and Mellor,J., Object Oriented Systems Analysis: Modeling the World in Data, Yourdon Press Computing Series, Yourdon Press, Englewood Cliffs, NJ, (1988).

Spivey,M., The Z notation: a reference manual. Prentice Hall, Englewood Cliffs, NJ, Second edition, (1992).

UML 1.3, Object Management Group, The Unified Modeling Language (UML) Specification – Version 1.3. en http://www.omg.or, (1999).

Waldoke, S., Pons,C., Paz Mezzano,C. and Felder,M., A Formal Approach to Practical Object Oriented Analysis and Design, Procs of Argentinean Symposium on Object Orientation, Buenos Aires, (1998).

Weber,M. "Combining Statecharts and Z for the Design of Safety-Critical Control Systems", Proceedings of Third International Symposium of FME'96. Oxford (1996).

Wieringa,R. and Broersen,J., Minimal Transition System Semantics for L Class and Behavior Diagrams. In Wsh. on Precise Semantics for Software Modeling Tech., T.U. Munchen, Report TUM-I9803, (1998).