Integrating Informal and Formal Approaches to Requirements Modeling and Analysis*

Betty H. C. Cheng and Laura A. Campbell
Software Engineering and Network Systems Laboratory
Department of Computer Science and Engineering
Michigan State University
East Lansing, Michigan 48824
{chengb,campb222}@cse.msu.edu

Abstract

The Unified Modeling Language (UML) comprises several different notations for object-oriented modeling with no formal semantics attached to the individual diagrams. We have developed a generic framework for formalizing a subset of UML diagrams in terms of various formal languages, with a focus on embedded systems. We have formalized UML in terms of Promela, thus enabling analysis of the UML diagrams by the SPIN model checker and simulator. We have also developed a number of visualizations to assist in the interpretation of the analysis results. This paper presents a case study of the UML design and automated analysis of an industrial automotive embedded system using our formalization techniques, supporting tools, and existing analysis tools.

1. Project Overview

The software for embedded systems is in general more difficult to design and debug because embedded software usually involves time-dependent sections in difficult to instrument situations. For example, embedded systems rarely have a keyboard or display, and correctly handling real-time events is often critical to the system's success. Nonetheless, embedded systems usually must achieve a higher level of robustness and reliability because they control real-world physical processes or devices upon which we depend, frequently, in a critical way. Errors in embedded systems can also have disastrous *economic* effects when large numbers of devices require code retrofits after being deployed. Consequently, methods for developing and modeling embedded systems and rigorously verifying behavior *before* committing to code, are increasingly important. Currently, much of the embedded systems industry use *ad hoc* approaches for developing embedded systems [1]. Frequently, there are few, if any, intermediate steps between high-level, prose descriptions of requirements and code written in the target implementation language, such as C.

The *Unified Modeling Language* (UML) [2] is an object-oriented notation broad enough in scope to represent a variety of domains, including embedded systems. In addition, there appears to be interest from the embedded systems community in exploring how object-oriented modeling, specifically UML, can be used for embedded systems. At present, however, UML is only a collection of notations, with no formal semantics attached to the individual diagrams, nor is there a formally defined semantics for the integration of the diagrams. In the absence of formally defined semantics, it is not possible to apply rigorous automated analysis such as model checking [3] or simulation, or to execute a UML model in order to test its behavior, short of writing code and performing exhaustive testing.

In order to address this problem, we have developed a general framework [4] for deriving formal language specifications from a subset of the UML diagrams; this framework enables us to attach a specific semantics to the UML diagrams from a range of possible semantics. Furthermore, this framework facilitates the construction of a consistent set of rules for transforming UML models into specifications in the formal language. (A description of the formalization framework and details regarding the development of the formalization rules may be found elsewhere [4].) The resulting specifications derived from UML diagrams enable either execution through simulation or analysis through model checking, using existing tools.

*This work is supported in part by NSF grants EIA-0000433, CDA-9700732, CDA-9617310, CCR-9633391, CCR-9901017, DARPA grant No. F30602-96-1-0298, managed by Air Force's Rome Laboratories, Department of the Navy, Office of Naval Research under Grant No. N00014-01-1-0744, Eaton Corporation, and a Motorola doctoral fellowship.

Proceedings of the Fifth International Symposium on Requirements Engineering (RE'01)
1090-705X/01 $10.00 © 2001 IEEE

One overarching goal of this research, motivated by technology transfer objectives, is to enable developers to continue to use widely accepted development techniques, both in terms of modeling languages as well as target specification languages. The framework that we have developed enables UML diagrams to be formalized in terms of a variety of formal languages. While UML offers several different notations, our preliminary investigations indicate that for modeling requirements and high-level design, the class and state diagrams are sufficient for modeling embedded systems. In addition, we have developed a suite of tools encapsulated by MINERVA and Hydra that respectively support the graphical construction of UML models and, using our formalization framework, automatically generates the corresponding formal specifications in terms of a selected target language.

In order to leverage existing specification languages and tools, we formalized UML in terms of Promela [3], the specification language for the model checker SPIN [3]. The specifications generated for the UML diagrams by the tool suite enable the model to be either directly executed in random simulation or analyzed with model checking, using existing tools (SPIN). The tool suite also supports the display of preliminary error checking results, simulation traces, and counterexample traces (paths of execution that lead to errors in the design) within the UML diagrams.

2. Related Work

Other projects have explored adding formal semantics to various components of the UML. We focus our discussion on those projects that apply analysis techniques to the UML diagrams. Some of the projects have considered class or state [5] diagrams separately, but not in an integrated fashion. Others focus instead on architectural relationships [6] within a system. UMLAUT [7] provides a framework for refining UML models and can generate code (not Promela) to simulate a model. In earlier work [8, 9], we formalized the three models of the OMT (*Object Modeling Technique*) [10] approach, including the class [8] and state diagrams [9], but the project did not include a general formalization framework. The formalization was defined in terms of LOTOS [11], thus enabling simulation and property checks with existing LOTOS tools.

Due to space constraints, we do not include work related only to visualization of analysis tools. In short, none of the tools provides analysis of both the class and state diagrams in an integrated fashion. Finally, vUML [12] and v-Promela [13] are intended to be graphical front-ends to the SPIN model checker. The vUML tool translates UML models to Promela and displays counterexamples as UML sequence diagrams, while MINERVA animates the original UML state diagrams. Furthermore, vUML is tied specifically to Promela, whereas our approach is more general. On the other hand, although v-Promela translates UML-style collaboration and state diagrams to Promela, it does not appear to utilize the class diagram. Additionally, it allows many Promela-specific notations, especially in state transitions, while our framework was designed to be target language independent at the graphical modeling level.

References

[1] D. D. Gajski, F. Vahid, S. Narayan, and J. Gong, *Specification and Design of Embedded Systems*, ch. 1. P T R Prentice Hall, 1994.

[2] Rational Software Corporation, Santa Clara, CA 95051-0951, *UML Notation Guide*, 1.0 ed., January 1997.

[3] G. J. Holzmann, "The Model Checker SPIN," *IEEE Transactions on Software Engineering*, vol. 23, May 1997.

[4] W. E. McUmber and B. H. C. Cheng. A general framework for formalizing UML with formal languages. In *Proc. of IEEE Int. Conf. on Soft. Eng. (ICSE01)*, Toronto, Canada, May 2001.

[5] D. Latella, I. Majzik, and M. Massink. Towards a formal operational semantics of UML statechart diagrams. In *Proc. FMOODS99, IFIP TC6/WG6.1 3rd Int. Conf. on Formal Methods for Open Object-Based Distributed Systems, Florence, Italy, February 15-18, 1999*. Kluwer, 1999.

[6] P. Bose. Automated translation of UML models of architecture for verification and simulation using SPIN. In *Proc. of IEEE Int. Conf. on Automated Soft. Eng.*, Cocoa Beach, FL, October 1999.

[7] W. M. Ho, J.-M. Jezequel, A. L. Guennec, and F. Pennaneac'h. UMLAUT: an extendible UML transformation framework. In *Proc. of IEEE Int. Conf. on Automated Soft. Eng.*, Cocoa Beach, FL, October 1999.

[8] R. H. Bourdeau and B. H. C. Cheng. A formal semantics of object models. *IEEE Trans. on Soft. Eng.*, 21(10):799–821, October 1995.

[9] E. Y. Wang, H. A. Richter, and B. H. C. Cheng. Formalizing and integrating the dynamic model within OMT. In *Proc. of IEEE Int. Conf. on Soft. Eng. (ICSE97)*, Boston, MA, May 1997.

[10] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen. *Object-Oriented Modeling and Design*. Prentice Hall, Englewood Cliffs, NJ, 1991.

[11] K. J. Turner, editor. *Using Formal Description Techniques: An Introduction to Estelle, LOTOS and SDL*. Wiley, 1993.

[12] J. Lilius and I. P. Paltor. vUML: a tool for verifying UML models. In *Proc. of IEEE Int. Conf. on Automated Soft. Eng.*, Cocoa Beach, FL, October 1999.

[13] S. Leue and G. Holzmann. v-Promela: A visual, object-oriented language for SPIN. In *2nd IEEE Int. Symp. on Object-Oriented Real-Time Distributed Computing*. IEEE Comp. Soc. Press, May 1999.