A Formal V&V Framework for UML Models Based on Model Transformation Techniques

Soon-Kyeong Kim and David Carrington

Information Technology and Electrical Engineering
The University of Queensland, St. Lucia, 4072, Australia
soon@itee.uq.edu.au
davec@itee.uq.edu.au

Abstract. *This paper proposes a formal V&V framework where multiple formalisms are integrated into an existing modeling environment for UML. In our framework, each formal specification technique is defined in term of a metamodel and integrated into the Eclipse Modeling Framework. The integration of informal models and various formalisms is achieved using model transformation techniques. Using metamodel-based model transformation, we provide precise and explicit integration. Integrating multiple formalisms allows the developer to choose formalisms that are appropriate for each analysis task.*

1. Introduction

Formal methods have been used in conjunction with informal or semiformal modeling techniques in software development [1, 6, 9, 17]. In such integrated approaches, formal techniques provide an effective means to check the validity of models, thus providing increased quality for both models and implementation. Despite its potential, application of the integrated approach to large scale systems has been limited. Beside those well-known barriers for formal techniques such as difficulties in dealing with mathematical notation [17, 21], some barriers can be overcome, for example:

- Inappropriate modeling techniques for analysis tasks
- Imprecise and implicit integration (e.g. informally defined transformation rules)
- Lack of tool support for the integration (e.g. manual transformations)
- Using a separate environment for the integration (e.g. stand-alone translation tools).

Among other factors, the selection of appropriate formal techniques is a key aspect for the success of the integrated approach in terms of appropriateness, semantic comparability, and availability of tools and techniques such as type checker, model checker and theorem prover. Unfortunately no single formal analysis technique can address all these aspects because different formal techniques and their supporting tools have different strengths in different areas. For UML [19], Object-Z [3] can be considered as an appropriate formal modeling technique because it is a mature object-oriented formal specification technique and provides well-defined modeling constructs and structuring facilities (e.g. classes and inheritance). Also the integration between UML and Object-Z does not require any semantic shift since both use the same object-oriented paradigm. Nevertheless, Object-Z lacks automatic analysis tools such as a model checker. Consequently, Object-Z is appropriate as a formal modeling tool for UML, but it is not well-suited as an effective means for tool supported formal analysis of UML models. In contrast, the Symbolic Analysis Laboratory (SAL) formalism [16] provides a rich set of analysis tools (e.g. type-checker, deadlock-checker and model checkers), but it is not yet clear how to capture the object-oriented concepts in UML models using the SAL input language. Nevertheless, the tool environment provided by SAL can provide an effective formal V&V environment for UML models when we have well-defined transformation rules between UML and the SAL input language.

For these reasons, it seems necessary to develop approaches and tools that integrate more than one formalism for the V&V of informal models. In this position paper, we propose a formal V&V framework that integrates multiple formalisms into an existing modeling environment for UML (see Fig. 1).

In our framework, each formal specification technique is defined in term of a metamodel integrated into the Eclipse Modeling Framework (EMF) [5]. The integration of UML models and various formalisms is achieved using model transformation techniques in the same modeling environment. The significance of our approach is summarized below:

- An informal UML model is readily transformed to any formalism integrated into the development environment and developers can choose that formalism(s) that are appropriate for each analysis task – *addressing the inappropriate modeling techniques integration problem*.
- The Eclipse-based metamodeling technique provides a rigorous semantic foundation for such integration - *addressing the precision problem in the integration*.
- Using model transformation, we provide precise, explicit and automatic transformation - *addressing the implicit transformation problem and lack of tool support*.
- The integration is incorporated into an existing modeling environment facilitating the application of the integration approach in practice - *addressing the integration environment problem*.

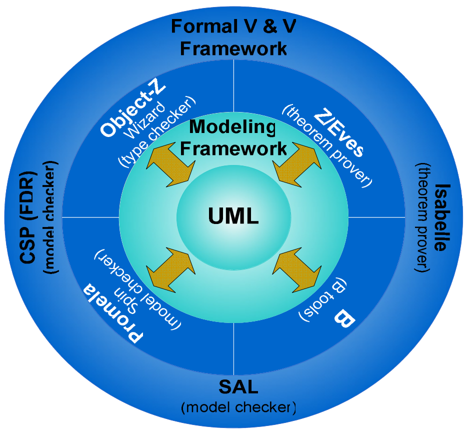

Fig. 1 Formal V&V environment for UML

The structure of the rest of this paper is as follows. Section 2 presents motivations and the potential of our approach. Section 3 describes the integration environment. Section 4 demonstrates an integration example using UML and Object-Z. Finally, Section 5 concludes.

2. Motivation and the potential of our approach

2.1 Why UML?

UML has become the dominant object-oriented modeling language in both industry and academia. Despite its popularity, the quality of UML as a precise formal modeling language has been challenged by numerous researchers [7, 15]. A lack of precision in the modeling language definition, and lack of support for rigorous analysis of UML models and verification activities are the well-identified drawbacks in UML. Our integrated approach aims to tackle these limitations. There is a significant amount of research that considers mappings from UML to other (mostly formal) modeling techniques to validate UML models (e.g. using B [14], CSP [6], SPIN [15, 17], PVS [21], Z/Eves [1] and our work with Object-Z [9, 10, 13]). We can integrate these existing approaches into our framework with a minimum of effort - facilitating reuse of existing work. UML has been proposed as a core modeling language in the OMG's Model Driven Architecture (MDA) [18], thus our UML-centered approach provides a solution for V&V related to MDA.

2.2 An extended formal V&V environment for informal models

In our framework, UML models can be potentially integrated to any formalism. For example, adopting existing work integrating Object-Z with other languages (e.g. the integration work with Object-Z and CSP to use a model-checker FDR [8] and the integration work with Object-Z and Higher Order Logic (HOL) to use the theorem prover Isabelle [20]), UML models can be potentially translated to any of these languages indirectly via Object-Z. Consequently, the tools developed for these languages could be used to check the UML models. In our integrated framework it is crucial that transformation rules defined at each local level (e.g. transformation between two languages) are correct, consistent and complete within the scope of the integration. Unless these properties hold, any analysis performed on the transformed models would not be reliable. Our metamodel-based transformation approach addresses these issues associated with the transformation.

2.3 The integrated V&V framework and MDA

Our integrated approach can deliver benefits to model driven development approaches such as MDA. To get the full potential of the MDA, the MDA transformation infrastructure should include the ability to use modelling notations that are the most appropriate to capture different aspects of a system, and should have a capability of transforming between models in these different notations. Also there must exist efficient ways to check models for properties such as consistency and correctness. Currently UML and MOF are proposed as the central modelling languages by the OMG in the MDA. However, using only UML limits the provision of these capabilities that are required for the MDA. Our integrated approach combining formal and informal modeling techniques can contribute to this area. For example, it provides the convenience to choose appropriate modeling techniques to capture different aspects and to integrate the techniques.

3. Integration environment

Developing a special tool for each integration is expensive. We tackle this issue by using an existing integration environment (EMF) and model transformation techniques. In our approach, integrating a new formal technique requires a metamodel of the language and a set of transformation rules to map that language to other languages. Then the actual transformations are achieved using transformation tools. Figure 2 shows the overall tool architecture used in our work. We use the Eclipse Platform [4] as an integration environment. Two plug-ins used in our integration are EMF [5] and the DSTC's transformation engine Tefkat [2]. EMF is used to define and implement metamodels of the languages in the integration. Transformation rules are defined by using the DSTC's model transformation language [2] and the actual transformations are achieved using the DSTC's Tefkat transformation engine (i.e. once the OMG finalizes a standard transformation language, our rules will be expressed in the standard language).

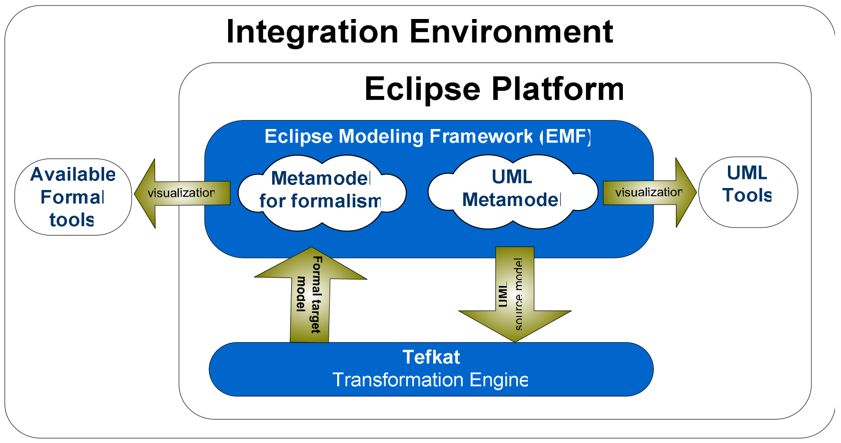

Fig. 2 Integration environment

4. Transformation example from UML to Object-Z

We demonstrate an integration example using UML and Object-Z. In our approach, different types of UML models such as the Class model, the State machine and Interaction diagrams are transformed into Object-Z models. Then they are integrated into a single Object-Z model to represent the overall system. The model integration is achieved using the instantiation and inheritance mechanisms in

Object-Z. The Object-Z models developed in this way provide a precise semantic basis from which various levels of checking activities to models can take place. For example, the individual Object-Z models provide a precise basis to check intra-model consistency, while the integrated Object-Z model provides a precise basis to check inter-model consistency. The Object-Z models also provide a precise semantic basis to map the UML models to other languages – providing a precise semantic domain for UML.

Due to the page limitations, we show one example rule to transform a UML class to an Object-Z class (Fig. 3 shows partial metamodels of UML and Object-Z) [1].

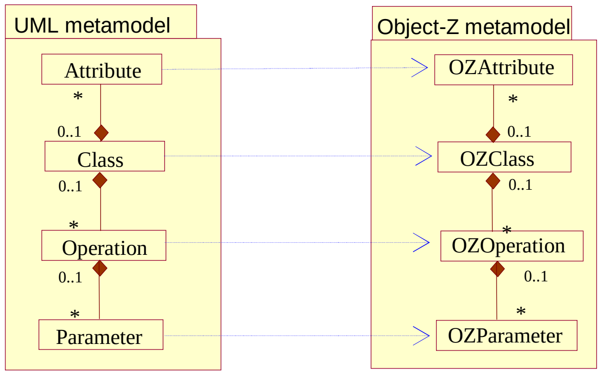

Fig. 3 UML and Object-Z metamodels

The following rule UMLClass2OZClass in the DSTC's language maps a UML class to an Object-Z class. Line 13 declares the rule name and variables to be used in the rule. Line 15 introduces a WHERE...LINKS statement to find the correct Object-Z model into which to place the created Object-Z class. Lines 15 and 16 effectively find the UML model that contains the source UML class (umlm.ownedMember). Line 17 creates the target Object-Z class, while Line 18 introduces a SET statement that sets the attributes and references of created target elements. In this case, the Object-Z class name is set to the same name as the UML class, and the Object-Z class is added to the Object-Z model. Line 19 preserves the tracking relationship by storing the corresponding UML class and Object-Z class as these will be used in other rules.

```
13   RULE      UMLClass2OZClass(umlm, ozs, umlc, ozc)
14   FORALL  Class umlc
15   WHERE   UMLModelOZSpec LINKS umlmodel = umlm, ozspec = ozs
16   AND        umlm.ownedMember = umlc
17   MAKE      OZClass ozc
18   SET         ozc.name = umlc.name, ozc.owner = ozs
19   LINKING  UMLClassOZClass WITH umlmodel = umlm, ozspec= ozs, umlclass =
umlc, oz-class = ozc;
```

[1] we use the UML2.ecore file supplied by the UML2 project [4] for the UML metamodel, and implement the Object-Z metamodel in [11] using EMF.

5. Conclusion and future work

This paper has presented a framework for integrating multiple formalisms for analysing informal models. We use model transformation techniques to achieve the integration. Metamodel-based model transformation techniques allow us to define the integration unambiguously and explicitly. Using an existing development environment along with model transformation techniques, we achieve automatic integration. The framework provides a unified formal V&V environment for informal modelling techniques and an effective means to integrate different formalisms. Incorporating the integration into an existing modelling environment facilitates application of the integrated approach in software development. In addition, our approach incorporates an effective V&V mechanism for model driven approaches such as MDA. Currently, we are investigating integration between UML and SAL [12] to generate test sequences from UML models.

REFERENCES

[1] N. Amalio, S. Stepney and F. Polack, Formal Proof from UML Models, *ICFEM 2004*, LNCS 3308, pp. 418-433, Springer-Verlag, 2004.

[2] DSTC Transformation Language and Tefkat, http://www.dstc.edu.au/tefkat

[3] R. Duke and G. Rose, Formal Object-Oriented Specification Using Object-Z, Macmillan, 2000.

[4] Eclipse Foundation. http://www.eclipse.org/

[5] EMF, The eclipse modeling framework. http://download.eclipse.org/tools/emf/scripts/docs.php?doc=references/overview/EMF.html

[6] G. Engels, R. Heckel, and JM. Kuster, Rule-Based Specification of Behavioral Consistency Based on the UML Meta-model, Proc. UML’01, LNCS 2185, pp. 272–286, 2001.

[7] R. France, A. Evans, K. Lano, and B. Rumpe, The UML as a Formal Modeling Notation, Computer Standards and Interfaces, 19(7), pp. 325-334, 1998.

[8] G. Kassel and G. Smith. Model Checking Object-Z Classes: Some Experiments with FDR. *APSEC2001*, pp. 445-452. IEEE Computer Society, 2001.

[9] S-K. Kim and D. Carrington, Formalizing the UML class diagram using Object-Z, Proc 2nd IEEE conference on UML: UML'99, LNCS, No 1723, pp. 83 -98, 1999.

[10] S-K. Kim and D. Carrington, A Formal Mapping between UML Models and Object-Z Specifications, International conference on ZB2000, LNCS 1878, pp. 2-21, 2000.

[11] S-K. Kim, A Metamodel-based Approach to Integrate Object-Oriented Graphical and Formal Specification Techniques, PhD Thesis, ITEE, The University of Queensland, 2002.

[12] S-K. Kim, L. Wildman and R. Duke, A UML Approach to the Generation of Test Sequences for Java-based Concurrent Systems, Australian Software Engineering Conference 2005, pp. 100-109, IEEE Computer Society, 2005.

[13] S-K. Kim and D. Carrington, An MDA Approach towards Integrating Formal and Informal Modeling Languages, Formal Method 2005, LNCS 3582, pp. 448-464, 2005.

[14] K. Lano, D. Clark and K. Androutsopoulos, UML to B: Formal Verification of Object-Oriented Models, Proc. *IFM’04*, LNCS 2999, pp. 187 – 206, 2004.

[15] D. Latella, I. Majzik, and M. Massink. Automatic verification of a behavioural subset of UML statechart diagrams using the SPIN model-checker. Formal Aspects of Computing, 11:430-445, 1999.

[16] L. de Moura, S. Owre, H. Rueb, J. Rushby, N. Shankar, M. Sorea, and A. Tiwari, SAL2. Proc. Computer-Aided Verification (CAV2004), LNCS 3114, pp. 496-500. 2004

[17] W. McUmber and B. Cheng. A General Framework for Formalizing UML with Formal Languages, IEEE International Conference on Software Engineering, pp. 433–442, 2001.

[18] OMG, MDA Guide Version 1.0.1, 2003. http://www.omg.org/docs/omg/03-06-01.pdf

[19] OMG, UML 2.0 Superstructure Specification, OMG Document ptc/03-08-02. http://www.omg.org/docs/ptc/03-08-02.pdf, 2003.

[20] G. Smith, F. Kammüller and T. Santen. Encoding Object-Z in Isabelle/HOL. International Conference of Z and B Users, LNCS 2272, pp. 82-99. Springer-Verlag, 2002.

[21] I. Traore and D. B. Aredo, Enhancing Structured Review with Model-Based Verification, *IEEE Trans. Software Eng*., 30(11), pp. 736 -753, Nov., 2004.