# Transforming UML Models to Formal Specifications

Jean-Michel Bruel
Laboratoire TASC
Universit de Pau et des Pays de l'Adour
64012 Pau Université cedex, France
Jean-Michel.Bruel@univ-pau.fr

Robert B. France
Computer Science Department
Colorado State University
Fort Collins, CO 80523, USA
france@cs.colostate.edu

**Abstract**

The Unified Modeling Language (UML) is considered a significant step in the development of object-oriented (OO) notations in that it is based on some of the best OO modeling experiences. In this paper we are considering more specifically its use in the context of high-quality modeling of complex systems. We believe that the lack of firm semantics for the modeling notations used makes, among others, the identification of requirements problems difficult. One approach to making UML more precise and amenable to rigorous analysis is to integrate it with suitable formal notations. In this paper we present the benefits from a similar approach that is the result of integrating Fusion modeling techniques and the Z formal specification notation. The result is a technique that produces highly-structured, graphical, rigorously-analyzable models that facilitates early detection of errors. We believe that UML could gain from this experience.

## 1 Research motivation

In the area of software specification we can distinguish two classes of techniques: the *informal, structured techniques* (ISTs) (e.g., OMT [7], Fusion [2]) which emphasize ease-of-use and understandability, often at the cost of rigor, and the *formal specification techniques* (FSTs) (e.g., Z [9]) which emphasize formality, often at the cost of ease-of-use and understandability.

The lack of precise semantics for popular ISTs can lead to the following problems [4]:

- Understanding of models can be more apparent than real.
- Developers can waste considerable time resolving disputes over usage and interpretation of notation.
- Rigorous semantic analysis is difficult.

The need for a rigorous basis for OO modeling suggests the use of traditional FSTs, where a FST consists of a notation that directly utilizes mathematical concepts, and provides mechanisms for formally analyzing models expressed in the notation. OO extensions of traditional FSTs attempt to provide a richer set of structuring mechanisms, but such extensions often result in semantics that are more complex.

The integration of ISTs and FSTs provides an avenue for transforming an imprecise model into a more precise model. We propose that rather than attempting to extend traditional FSTs with OO concepts, a more workable approach would be to provide a formal basis for the OO modeling notations and concepts that have been shown to be effective in practice. In other words, the approach we advocate is to make more precise the notations and concepts that are currently perceived as reflecting the best practical experiences in OO modeling. Formalizing such concepts and notation would deepen our understanding of OO modeling and result in OO techniques that are at least rigorous. In this position paper we give an overview of a first attempt in this direction. Our approach for requirements modeling and analysis, called *FuZed*, integrates an informal OO analysis technique, Fusion [2], with a formal specification notation, Z [9]. In section 2 we detail our

approach and its main benefits. In section 3 we discuss our ideas of making UML more rigorous according to our experience and we conclude in section 4.

# 2 FuZed: A Rigorous Object-Oriented Analysis Modeling Technique

Development of a requirements model using the FuZed technique involves developing Fusion analysis models and then transforming them to Z specifications so that they can be analyzed using Z analysis tools such as animators and theorem-checker/prover. Our experiences with applying FuZed indicates that defects in the requirements are uncovered not only during analysis of the Z specifications, but also during the transformation of the Fusion models to the Z specifications (the "formalization" activity).

## 2.1 Formalizing object-oriented analysis

In our work, "integration" means providing a bridge from the informal OO modeling notations to a more formal notation. The objective of integration is to produce modeling techniques that allow one to obtain precise, rigorously analyzable models from less formal, but more understandable OO models.

In developing FuZed we focused on producing formalizations that can directly support verification and validation activities, and for which computer-based support is available. Fusion is used because it incorporates some of the best OO modeling concepts from mature OO techniques in a coherent modeling framework. The use of Z reflects our desire to use a stable, mathematically precise and structured notation that can express object-oriented analysis (OOA) modeling concepts in a straightforward manner, and that is supported by computer-based analysis tools. Hall has shown that Z can be used to express OO modeling concepts [6], and Z is supported by typecheckers, animators, and theorem proving/checking environments.

In this process, the modeling phase involves the use of a mature OOA modeling technique. In the case of FuZed the Fusion OOA techniques are used in the modeling phase. Analysis of the models is accomplished by generating *validation models* from the informal OOA models. The validation models allow the developer to rigorously validate the behavior and structure captured by the informal OOA models. In the FuZed technique, validation models are expressed in the Z notation.

In our work, a validation model is a precise and analyzable representation of a system view captured by an OOA model. This allows developers to check each view for consistency. Furthermore, the use of a single formalism for the validation models (Z) allows developers to check consistency across views.

FuZed differs from other works on integrated techniques in its focus on precisely defining a compositional semantics for Fusion models. In the FuZed approach, mappings for the basic modeling constructs are precisely defined and composed to obtain the meaning of more complex structures. While it is possible to express these mappings and their compositions in a formal declarative functional notation, we chose to express the transformation rules in an algorithmic manner. The algorithmic style makes the operational aspects of the transformation clearer and provides a path for development of computer-based tools. We implemented an early version of the transformation rules in a tool called FuZE (see section 2.3).

## 2.2 From Fusion Analysis Models to Z Specifications

Rather than detail the rules and guidelines we developed for transforming Fusion's object models to Z specifications (see [4] for more details) we outline the major ideas behind our approach.

A transformation is referred to as a *formalization* of the informal model. Invariably, the informal model does not have all the information needed to generate a formal expression of desired properties, or the information is expressed in language that is open to multiple interpretations. Human

intervention is needed to provide additional information and to provide formal expressions of informal statements. The intervention forces the analyst to examine the informal models more closely and helps him/her formulate appropriate questions about its content. Essentially, the intervention provides the analyst with opportunities for uncovering problems with the models. For this reason, we do not view the inability to completely automate the formalization process as a weakness, rather it is a desired feature of our requirements analysis process.

In our technique, a class diagram (CD) is a characterization of a family of structures consisting only of linked instances of depicted classes. These instance (object) structures are often referred to as *instance (object) models* in the literature. We refer to the instance structures characterized by a CD as *configurations*. Intuitively, a configuration is a valid, stable state of the system modeled by the CD. A system is in a valid stable state when there are no operations executing and the state components satisfy a specified invariant. Denotationally, the semantic domain for a CD is a collection of sets of configurations. The meaning of a CD is a set of configurations, that is, instance structures that conform to the constraints expressed in the CD. The Z specification generated from a CD is a characterization of the valid, stable system states.

In FuZed, CD constructs are transformed to characterizations of configuration components. These construct characterizations are composed to form characterizations of CD configurations. The meaning of a CD (or CD construct) is the set of all configurations (or configuration elements) that satisfy the characterizations produced by the FuZed technique. In general, FuZed takes a property-oriented approach to formalizing Fusion models.

Denotationally, a Fusion Operation Schema (OS) is a collection of configuration pairs, where each pair denotes the before- and after-effect of the operation on the system state. The Z operation schema corresponding to a Fusion OS is a precise characterization of the set of configuration pairs defining the desired effects.

Z operation schemas can be composed to form specifications of more complex behaviors. In FuZed, the Fusion Life-Cycle model is used to constrain how Z specifications of system operations can be composed, that is, it is used to determine the complex behaviors that can be built up from specified operations.

For CDs, the generation of Z specifications can be completely automated in the cases where there are no informal annotations. When there are informal annotations, they must be re-expressed formally by the human analyst during the formalization process. The transformation of CDs to Z specifications is based on a set of transformation rules that map basic CD constructs to Z constructs.

Fusion OSs are expressed largely in natural language, thus more human effort is required to transform them to Z specifications. We present guidelines, rather than rules, for transforming Fusion OSs to Z specifications. A subset of these guidelines can be automated. The result of automation is a tool that automatically produces skeletons of Z specifications from OSs. Human effort is required to complete the generated specification shells.

## 2.3 FuZE: Fusion/Z Environment

We built a prototype tool, FuZE [1], that uses an early version of one set of transformation rules to automatically generate Z specifications from Fusion class models. Z analysis tools (ZTC and ZANS) can be called from within the tool to analyze the generated Z specifications. The tool has been applied by graduate students at Florida Atlantic University on non-trivial projects and case studies (e.g., see [3]). In general, our experiences indicate that formalization and analysis of informal models can uncover problems with the informal models, and lead to a deeper understanding of the problem. Our applications of the integrated methods on case studies also uncovered limitations of our previous rules. In this paper we present a technique for transforming Fusion CDs (OMs) to Z specifications that improves upon our previous technique. The new technique supports the automated transformation of a wider range of OMs.

# 3 Making UML more precise

In the OO modeling community the Unified Modeling Language (UML) [5] is attracting much industrial attention. It is difficult to dispute that the UML reflects some of the best modeling experiences and that it incorporates notations that have proven to be useful in practice. Yet, the UML does not go far enough in addressing problems related to the lack of precision. The architects of the UML have stated that precision of syntax and semantics is a major goal. Currently the UML semantics are expressed in terms of a semantic meta-model that is supported by natural language descriptions. The meta-models do capture semantic relationships among UML modeling constructs, but they do little in the way of answering questions related to the interpretation of non-trivial UML structures. It does not help that the semantic meta-model is expressed in the notation that one is trying to interpret. The natural language text supporting the models often mimic the meta-models (i.e., it describes semantic relationships) and provides very little additional insights into the semantics of UML structures. More seriously, it is not clear how the UML semantics as described in UML documents supports rigorous analysis and refinement of UML models.

The Precise UML (PUML) project is a collaborative effort that has as its objectives the development of a precise semantics for UML, and the development of mechanisms that support rigorous analysis and refinement of UML models.

The primary objective of our collaborative research is to develop a formal foundation for a core subset of the UML and to use the insights gained as a result of the formalization activity to fix any deficiencies uncovered as a result of formalizing the UML concepts and notations, and to develop techniques with mechanisms that facilitate the rigorous application of UML.

Our focus on developing rigorous techniques based on the UML, a notation that is widely perceived as reflecting the best industrial software development experiences, stems from our desire to see more widespread (direct or indirect) use of FSTs. The approach we propose uses traditional FSTs to better understand the concepts underlying OO modeling and refinement. The insights gained will be used to develop a more precise semantics for the UML, and to develop mechanisms that allow developers to rigorously analyze and refine their models without the need to directly manipulate mathematical reformulations of their OO models.

The specific goals of the proposed research are: to identify and formalize core UML modeling concepts and their compositions, to develop a formal basis for refinement of UML models, to develop an integrated set of rigorous UML modeling techniques, and to develop a computer-aided UML modeling environment that incorporates the rigorous modeling and refinement techniques that we develop.

We anticipate that the results produced by the proposed research will have at least the following impact: improved understanding of OO modeling concepts, help provide a firm foundation for standardization efforts, provide a firm base for the development of rigorous OO development methods, and enhance investment in OO technologies and encourage more widespread use of rigorous development techniques.

# 4 Conclusion

Although mature object-oriented analysis modeling techniques are widely used for software specification, their expressiveness and rich set of intuitive constructs are not used for the modeling of complex systems. This is mainly due to their lack of support for rigorous analysis, itself due to its loosely-defined semantic. Formal specification techniques can provide the precision and tools needed to support rigorous analysis of modeled properties [8]. But despite their strengths, the use of existing FSTs can be difficult because of the effort needed to use them. We believe that FSTs and mature, structured OO techniques can play complementary roles. We have given an overview of our approach of such an integration, called FuZed. Our experience suggests that an even better approach would be to move the UML notation to a more formal language by giving it a semantics and developing a calculus that utilizes the UML notation to support reasoning. This is the motivation behind the collaborative effort engaged in by the *Precise UML* group.

# References


[1] Jean-Michel Bruel, Robert B. France, Bharat Chintapally, and Gopal K. Raghavan. A Tool for Rigorous Analysis of Object Models. In *Proceedings of the* $20^{th}$ *International Conference on Technology of Object-Oriented Languages and Systems (TOOLS'96),* Santa Barbara, California, July 29–August 2 1996.

[2] Derek Coleman, Patrick Arnold, Stephanie Bodoff, Chris Dollin, Helena Gilchrist, Fiona Hayes, and Paul Jeremaes. *Object-Oriented Development: The Fusion Method.* Prentice Hall, Englewood Cliffs, NJ, Object-Oriented Series edition, 1994.

[3] Robert B. France and Jean-Michel Bruel. The Role of Integrated Specification Techniques in Complex System Modeling and Analysis. In Janusz Zalewski, editor, *Proceedings of the Workshop on Real-Time Systems Education (RTSE'96),* Daytona Beach, Florida, pages 111–119. IEEE Computer Society Press, 20 April 1996.

[4] Robert B. France, Jean-Michel Bruel, and Maria M. Larrondo-Petrie. An Integrated Object-Oriented and Formal Modeling Environment. *The Journal of Object-Oriented Programming (JOOP)*, 10(7), Nov/Dec 1997.

[5] The UML Group. Unified Modelling Language. Version 1.1, Rational Software Corporation, Santa Clara, CA-95051, USA, 1 September 1997.

[6] J. Anthony Hall. Using Z as a specification calculus for object-oriented systems. In D. Bjørner, C. A. R. Hoare, and H. Langmaack, editors, *VDM and Z – Formal Methods in Software Development*, volume 428 of *Lecture Notes in Computer Science*, pages 290–318. VDM-Europe, Springer-Verlag, New York, 1990.

[7] James Rumbaugh, Michael R. Blaha, William Premerlani, Frederik Eddy, and William Lorensen. *Object-Oriented Modeling and Design.* Prentice Hall, Englewood Cliffs, NJ, 1991.

[8] Hossein Saiedian. An Invitation to Formal Methods. *Computer*, 29(4):16–30, April 1996.

[9] J. Michael Spivey. *The Z Notation: A Reference Manual.* Prentice Hall, Englewood Cliffs, NJ, Second edition, 1992.