Rigorous Object-Oriented Modeling: Integrating Formal and Informal Notations

R. B. France\textsuperscript{1*}, J.-M. Bruel\textsuperscript{2}, M. M. Larrondo-Petrie\textsuperscript{1}, E. Grant\textsuperscript{1}

\textsuperscript{1} Department of Computer Science & Engineering
Florida Atlantic University
Boca Raton, FL-33431, USA.
\textsuperscript{2} Laboratoire IRIT/SIERA, bat. 1R1
118, rte de Narbonne
31062 Toulouse Cedex, France
Contact: robert@cse.fau.edu

Abstract. The high-quality modeling experiences embedded in the more mature graphical OO methods (OOMs) makes their application to complex systems attractive, but the lack of firm semantic bases for the modeling notations can significantly hamper the development of such systems. One approach to making OOMs more precise and amenable to rigorous analysis is to integrate them with suitable formal modeling techniques. In this paper we describe a technique for integrating an OOM, the Fusion method, and a formal specification notation, Z.

1 Introduction

The more mature informal object-oriented (OO) modeling techniques (e.g., Fusion \cite{6}) provide good support for developing concise, highly-structured, models of behavior from a variety of perspectives. These techniques are based on some of the best modeling experiences available, and consist of rich sets of structuring and abstraction mechanisms. A deterrent to the use of OO methods (OOMs) for the development of complex systems is their lack of support for rigorous analysis. This is a result of the loosely-defined semantics for the modeling notations.

An approach to making informal OO models more precise and amenable to rigorous analysis is to integrate them with suitable formal notations. Several studies in this area have been published (e.g., see \cite{1, 9}). In our work, “integration” means providing a bridge from the informal OO modeling concepts to the formal notation. Our work differs from others in that our focus is on producing formalizations that can directly support verification and validation activities, and for which mechanical support is possible. For this reason, we have used only formal notations for which there exist sound sets of analysis tools. The Z notation \cite{3, 14} we use is supported by typecheckers (e.g., ZTC \cite{12}), animators (e.g., ZANS \cite{11}), and theorem proving environments (e.g., Z/EVES \cite{7}). We assume

\* This work was partially funded by NSF grant CCR-9410396.
that the reader is familiar with the Z notation. For a detailed description of the Z notation see [14].

Our past work focused on the formalization of the analysis models of the Fusion method [6] and it resulted in a set of rules for transforming Fusion Object Models to Z specifications (e.g., see [1]). We built a prototype tool, FuZE [4], that automatically generates Z specifications from Fusion Object 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 [8]). 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 Object Models (OMs) to Z specifications that improves upon our previous technique. The new technique supports the automated transformation of a wider range of OMs.

We have also extended our transformation technique to the design models of Fusion. We present some of our current work on formalizing Fusion design models in this paper. In section 2 we give an overview of the Fusion analysis and design modeling techniques. In section 3 we outline our technique for formalizing Fusion's analysis models, and in section 4 we present the results of our recent work on formalizing Fusion's design models. We conclude in section 5 with a summary of our work and an outline of ongoing work.

2 The Fusion Modeling Techniques

Fusion is an object-oriented software development methodology that combines and extends existing techniques, e.g., Rumbaugh's Object Modeling Technique (OMT) [13], Booch's technique [2], Wirfs-Brock's Class Responsibility Collaborator [15] (CRC) technique, and Jacobson's Objectory [10]. Fusion claims to take the best ideas from these methods and incorporate them into a single coherent method that covers analysis, design, and implementation.

2.1 Fusion Analysis Models

In Fusion's analysis phase the required behavior of the system is described by the following models.

Object Model. An Object Model (OM) defines the static structure of the information manipulated by the application in terms of classes and the relationships among them. An example of an OM for a petrol dispensing system is given in Fig. 1 (taken from [6]). In an OM a class is represented by a box that consists of a partition containing the name of the class, and another containing a list of attributes for the class. Type information for attributes may or may not be shown in an OM. Relationships are depicted as diamond adorned lines between