Specifying the Abstract Layer of a Graphical Editor

Orieta Celiku
Turku Centre for Computer Science

August 28, 1999

We give a formal specification of an object-oriented software system using the Unified Modeling Language [3] and the Refinement Calculus techniques [4]. More specifically, we specify the abstract layer of a graphical editor for action systems, a language for parallel and distributed programming [1].

The Unified Modeling Language(UML) has emerged as a de-facto standard modeling language because of its intuitive and powerful graphical description techniques. Class diagrams, use case diagrams, and scenarios are effective means for expressing analysis and design. However, what UML's graphical notations mean is not always clearly defined; a lot of researchers argue for specifications more formal than what UML can express.

One of the recent approaches to formalizing UML uses the Refinement Calculus as a formal foundation for giving precise semantics to UML diagrams [2]. UML description entities are paired with refinement calculus constructs, for example, UML classes are described as the so-called attribute systems, the well-foundedness of which can be proved using the weakest precondition approach to program correctness. This method is appealing because it combines the power of graphical descriptions as modeling languages with the safe, stepwise development of abstract specifications into final programs that satisfy the initial requirements. However, since the method is quite new, it has not been extensively tested in practical tasks yet.

In this work we describe the experience of using UML and the Refinement Calculus to formally specify a small, yet non-trivial system. We model action systems as attribute trees and allow for editing operations. This specification is used for building a tool for utilizing the action systems formalism graphically [5], but initially nothing is said about how the constructs look like, hence the name abstract layer. UML proves to be very suitable for expressing what kind of objects our system has, their relationships, etc. Formal constructs help express system properties that cannot be expressed by UML entities. Furthermore, we can prove if procedures establish the required postconditions, and if the system remains consistent after procedure calls. To reduce the abstractness level, and to take the system closer to an implementation we perform several refinement steps. Translating the specification into a Java implementation comes smoothly.

References

[1] R. J. R. Back and R. Kurki-Suonio. Decentralization of process nets with centralized control. *Proc. ACM SIGACT-SIGOPS Symp. on Principles of Distributed Computing.* pp. 131-142, 1983.

[2] R. J. R. Back, L. Petre, I. P. Paltor. Formalizing UML Use Cases in the Refinement Calculator. TUCS Technical Report, Nr. 279, Åbo, 1999.

[3] G. Booch, J. Rumbaugh, and I. Jacobson. *The Unified Modeling Language User Guide.* Addison-Wesley, October 1998.

[4] R. J. R. Back and J. von Wright. *Refinement Calculus: A Systematic Introduction.* Graduate Texts in Computer Science, Springer-Verlag, 1998.

[5] O. Celiku. *Specifying a Graphical Editor for Action Systems.* Master's thesis, Åbo Akademi, 1999.