

UML based Verification of Software *

S. Van Langenhove and B. De Leeuw

Department of Pure Mathematics and Computer Algebra
Ghent University, Belgium
{Sara.VanLangenhove, Benjamin.DeLeeuw}@UGent.be

The Unified Modeling Language (UML) is a graphical language for modeling a multitude of systems. This discussion focuses on the design of software systems in UML. We discern two distinct phases in software factoring. In the *design phase* we unveil system properties and inconsistencies by analyzing requirements, and decomposing the system in several communicating components, using standard (and well known to programmers) object oriented reasoning. Inconsistencies and design errors are subsequently removed *before* entering our next phase, the *code generation*. Most contemporary programmers and engineers do not discern such a design phase. We are convinced, as are many researchers, that it is tedious practice to recover from faulty reasoning *during* or *after* the coding phase. Moreover, removing such design errors requires a lot of money and time spent on coding and testing of software. This situation is particularly unacceptable for security, safety or commercially critical systems. Alas, modeling with UML neither guarantees a flawless design nor has the formal backbone required to do so. Therefore, it is important to conceive and to subsequently use a UML based verification method to identify and remove errors, inconsistencies, and misconceptions during the design phase.

We present a translation template in the Cadence SMV modeling tool¹ for UML state charts. This template allows us to convert arbitrary state charts to the modal logic of SMV, covering all behavioral model elements as defined in the UML specification², in particular, hierarchy, sequentialism, parallelism, non-determinism, priorities, and run-to-completion step semantics. The SMV model checker is subsequently used to verify the behavioral design. Additionally we are able to verify requirements between communicating objects of different classes using this template. Communicating objects have their behavior specified in separate state charts. The communication is based on signal and call event exchange, also known as asynchronous and synchronous communication respectively, and their queuing.

SMV has some technical limitations that force us to transform the original state charts to semantic equivalents. More specifically, we are forced to insert additional states and transitions to cope with these limitations. We introduce corresponding transformations on the temporal requirements of the system, as specified by the designer, using temporal logics.

During our work we utilized a formal characterisation of the logic used in the construct of SMV. We applied, in particular, the mathematical background of classical and intuitionistic Kripke structures and their application to state charts and SMV models. Also, since we are working with connected classes of objects, we filter important additional information out of the static structure of the design. The larger part of this static information is specified in UML object model diagrams. We use typed reasoning and attribute domain information to inject type specific operations and constraints³ in the SMV model checker. We are currently analyzing the formal basis to reason about class based representations: description logic. We are planning to use this description logic, not only to infer type information, but also to correct and refactor the static (class based) design of the system.

The template we introduce here makes way for the automatic translation of UML state charts. In particular, we will use the standardised translation of UML in XML to fill out the parameters of our template. With such an automatic translation available, UML would not only be a standard for systems development, but also portal to formal verification. Hence UML would become a more powerful and interesting tool, particularly for engineers and programmers.

*The research activities are funded by Ghent University (BOF/GOA project)

¹Cadence SMV was developed by K. McMillan and is available at <http://www-cad.eecs.berkeley.edu/~kenmcmil/smv>

²Available at www.omg.org

³The Object Constraint Language (OCL) is the de facto standard in UML to specify constraints