

Model Integration and Composition Problems: An Important Aspect of a Science of Design

José Meseguer, University of Illinois at Urbana-Champaign
Carolyn Talcott, SRI International

September 30, 2003

A good system design should specify what components the system is built of, and how they are combined. A good design methodology should be compositional, resulting in techniques that facilitate integration of systems from reusable components. It is also very important that a design, and not just its implementation, can be used to predict and analyze system behavior.

We believe that systematic use of mathematical models of different system aspects is a crucial part of a science of design of software intensive systems. Use of such models is routine in many engineering disciplines and is one of the key reasons for their success.

Design of a complex software system will involve modeling a variety of system aspects from different perspectives. For example, functionality, real-time behavior, concurrency and synchronization, fault tolerance, probabilistic behavior, security properties, resource requirements, and so on. Many of these aspects cut across multiple system components.

A non-trivial problem for a science of design is *the model integration problem*. That is, how to structure a system so that the models of different aspects/perspectives can be semantically integrated and composed to support the prediction and analysis of overall system behavior, and to ensure that system requirements are satisfied. This will greatly facilitate system integration itself, and the reusability of system components.

A related problem is the problem of *composing heterogeneous models*. That is to say, for each aspect models can have a compositional structure, but this structure and the composition operations may be different for different aspects. These

different models must be integrated to understand the implications of the overall system design.

To support such a design methodology we need languages and tools to formally specify and analyze the different, possibly heterogeneous, models and their composition and integration. In addition, languages and tools are needed to specify and analyze system designs, including structure and models. System design specifications that are *executable* are important to fill the gap between requirements and implementations.

Background

The authors and their collaborators have developed the Maude formal executable specification language based on rewriting logic, as well as a number of supporting reasoning and analysis tools. The Maude system has been used extensively by the authors and many others to model a variety of systems including network protocols, real-time systems, programming languages, and biological systems. Maude supports a rich algebra of module composition operations, which have been used in a number of the system modeling applications. It has also been used as a meta-logical framework to design and implement reasoning tools for different logics and mappings between such logics.