Architectural Support for Verification and Validation of High-Confidence Software Systems

Oleg Sokolsky
Department of Computer and Information Science
University of Pennsylvania

Model-based design holds high promise for software systems that require high confidence in their correctness and quality of service. Modeling allows designers to precisely capture their evolving understanding of the system. Analysis of models helps to catch errors in system behavior before the system is built. Finally, generative techniques — generation of code, test suites, and configuration data — can help with tedious and error-prone tasks, ensuring that verified properties are preserved.

At the same time, if any benefit is to be derived from model-based development, it is imperative to ensure that the model itself if adequate and consistent with the designer's expectations. The process of modeling can be as tedious and error-prone as coding itself. There are two main sources of errors in a behavioral model. On the one hand, just like software itself, large models are constructed from separate components or modules, usually mirroring the physical structure of the system. Incompatibilities between interfaces of model components affect large models just as adversely as they affect large software systems. The other, more serious problem is that different components can have different levels of abstraction, reflecting incompatible assumptions made by the designers during the modeling process.

To help designers ensure consistency of their behavioral models, we need a higher-level modeling layer that concentrates on the architecture of the system being designed. Such a layer should capture interfaces between model components - and, ultimately, components in the system itself. The interfaces described in an architectural model should capture all ways in which two

components can influence each other. Communication channels, clearly, is one way of interaction that is explicit in system's behavior. In embedded systems, components can have other, implicit means of interaction such as limited shared resources.

The benefit of taking time to explicitly spell out means of interaction between model components makes model designers less likely to accidentally omit important implicit interactions from the model. More importantly, guiding model development by the architectural specification of the component interface will help designers to ensure that abstractions affecting component interactions are done consistently in all components.

The challenging problem, however, is identifying and capturing all possible sources of interactions and reflecting them in the component's interface. In complex embedded systems, such as aviation software, mutual influence transcends purely software interactions. Various physical sources of interference can adversely affect quality of service provided by a component and even its correctness. Capturing such interference in structural and architectural models, and using these high-level models in crafting behavioral models, will help us arrive at faithful models that ensure faithful verification results.