Composition by Anonymous Third Parties

Farhad Arbab

Center for Mathematics and Computer Science (CWI), Amsterdam and
Leiden Institute for Advanced Computer Science, Leiden University
The Netherlands

Composition of algorithms has dominated software composition since the inception of programming. The ubiquitous subroutine call acts as the primary composition operator in virtually all programming models and paradigms, appearing in various guises such as function call, method invocation, remote procedure call, etc. The inadequacies of the tight coupling imposed by such composition mechanisms and the need for more flexible alternatives have become clearer along the evolution through object-oriented to component-based, and now, service oriented computing.

Interaction arises out of how a composition allows the active entities in a composed system to play against one another. Communication primitives used in classical models of concurrency to allow interaction among processes in a composed system share the targeted message passing nature of function calls: in order to interact, they generally require a process to directly address foreign entities, such as other processes or channels, that belong to the environment of the process. Interaction constitutes the most interesting and the most difficult aspect of concurrent systems. We have studied protocols for, and various aspects of, interaction in concurrency theory. Curiously, however, no model of concurrency has hitherto considered interaction as a first-class concept! This makes dealing with interaction protocols more difficult than necessary, by erecting a level of indirection that acts as an obstacle between the concrete structures constructed and manipulated in a model, on the one hand, and interaction as the subject of discourse, on the other.

Recognizing the need to go beyond the success of available tools sometimes seems more difficult than accepting to abandon what does not work. Our concurrency and software composition models have served us well-enough to bring us up to a new plateau of software complexity and composition requirements beyond their own effectiveness. In this sense, they have become the victims of their own success. Dynamic composition of behavior by orchestrating the interactions among independent distributed components or services has recently gained prominence. We now need new models for software composition to tackle this requirement.

In this presentation, I describe our on-going work on a compositional model for construction of complex concurrent systems out of simpler parts, using interaction as the only first-class concept. This leads to a simple, yet surprisingly expressive, connector language, together with effective models and tools for composition of complex systems of distributed components and services.

M. Lumpe and W. Vanderperren (Eds.): SC 2007, LNCS 4829, p. 1, 2007.