Bob Sproull, Sun Microsystems

We build complex systems by connecting components together and hoping the collection works. We attach one chip to another with wires; we connect one computer to another with an Ethernet; we connect one software system to another by binding APIs. In every case, the composition is designed to be "correct." But also in every case, we have no way to formally (or even informally!!) check that it is indeed correct.

Today, we have rudimentary tools. Type systems in programming languages, together with run-time binding checks, may insure some measure of correctness in the composition of two programs. We know how to extend type systems to the network (RMI; I'll even be generous and grant you XML/SOAP), but the practice is not widespread. But types alone tell you very little.

A next step is to decide whether the dynamic use of an interface is correct, i.e., sequences of interface events. Study of asynchronous and distributed systems has formalized the notion of a correct composition, but the analysis techniques work for only very small compositions (small state spaces, usually verified by model checkers). Remember path expressions?

We can't verify in advance correctness of composition and we rarely do adequate run-time checking of correctness. How many modules take a truly defensive approach to checking inputs, i.e., what the "environment" does to the module? Where should "defensive perimeters" lie in a system? If they're everywhere, the system will be too slow; if they're nowhere, it will be too fragile. When a new component or new version is introduced into a system, dynamically checking correct operation at its interfaces is advisable: trust results from experience.

Which brings us to yet another aspect of composition: performance. In addition to being able to check correctness, we need to be able to predict (at least estimate) performance of a composition. Even first-order predictions would be extremely valuable.

What should our expectations be for such a "composition challenge?" Surely not to be able to *prove* correctness of large systems. But let's enrich our way of engineering interfaces beyond today's type systems so that a great deal more static checking and analysis of compositions is possible. Other dynamic checking is also possible, e.g., checking when a network client contacts a network service whether that pair of protocol versions (or implementations) has successfully communicated before. One could imagine a systematic way of describing and cataloging protocol versions (or API versions) and implementations, and building an on-line compatibility database that can be checked for composition. When a server detects an error, it might use knowledge of the client's implementation to provide better diagnostics (e.g., this error is commonly the result of a mis-configured client).

Better interface engineering can also more rigorously define and implement "compatibility" (e.g., "backward compatibility"). There may be additional engineering techniques that reduce composition problems.

While the formal proving of correctness may be beyond our reach, the challenge is to devise good engineering approaches to composition. Rather than "correct by composition," we could aspire to "confidence in composition."

BIO

Robert F. Sproull, Vice President and Fellow at Sun Microsystems, founded and led the Massachusetts branch of Sun Microsystems Laboratories for over ten years. Now he serves in a research role. Since undergraduate days, he has been building hardware and software for computer graphics: clipping hardware, an early device-independent graphics package, page description languages, laser printing software, and window systems. He has also been involved in VLSI design, especially of asynchronous circuits and systems.

Before joining Sun in 1990, he was a principal with Sutherland, Sproull & Associates, an associate professor at Carnegie Mellon University, and a member of the Xerox Palo Alto Research Center. He is a coauthor with William Newman of the early text, Principles of Interactive Computer Graphics. He is an author of the recently-published book Logical Effort, which deals with designing fast CMOS circuits. He is a member of the National Academy of Engineering and has served on the US Air Force Scientific Advisory Board.