Systematically Designing Component Frameworks (Abstract)

Dr. Wolfgang Weck
Independent Software Architect, Zürich
www.wolfgang-weck.ch

A component framework has been defined as *a set of interfaces and rules of interaction that govern how components 'plugged into' the framework may interact. Typical component frameworks also provide an implementation that partially enforces these rules of interaction. The implementation of the component framework and those of the participating components remain separate.*[1]

In this tutorial we will learn a few lessons from looking at a small but easy to understand (toy) example. (i) Plug-and-play composition relays on standardized contracts. (ii) These contracts often are symmetric and will often consist of several interfaces (in the sense of programming languages) defining various roles. (iii) Defects in plug-and-play components may cause hard to resolve problems to third-party composers, who then experience component hell. (iv) Contract design can help to avoid these problems, and (v) this may require some small implementation to become part of the contract itself.

Reflecting these lessons and the illustrating example, we will derive a systematic way to design component frameworks.

Dr. Wolfgang Weck takes over various tasks in software architecture design, interface management, and architecture reviews. He is author of the chapter about the role of software architects in the German handbook of software architecture (dpunkt-Verlag, 2006). He lectures at Universities of Applied Sciences in Switzerland and Germany. Since he received a doctoral degree from the Polytechnical University Zürich (ETH), he has been both working with, and researching in depth concepts and application of component software and the related architectural approach.

[1] C. Szyperski: *Component Software – Beyond Object-Oriented Programming*, Second Edition. Addison-Wesley, 2002.