

Correct by Construction Components Or: Would Nasreddin Use Components?

Asuman Suenbuel
Kestrel Institute,
3260 Hillview Avenue, Palo Alto, CA 94304
asu@kestrel.edu

Current technology is shifting bit by bit from familiar interactive computers to non-stationary physical devices with computing, communication and sensing capabilities. Reacting to environmental dynamic, self-assembly, self-configuration, self-repair and other forms of adaptation becomes a mandatory task. Component-based software development is a promising approach to meet the demands of these kinds of settings. However, current development methods used in practice either restrict the notion of components to executable objects – following a more or less traditional software development process -- or supporting specific component technologies like Java Beans or COM/DCOM.

With our focus on the elements connecting components, we will find further techniques such as RPC, script languages, middleware and ORB like approaches, event channels, adapting or wrapping of components, also known as glue-code. The major drawback of using general-purpose programming languages for the implementation of glue-code are the strict type systems of these languages. This makes it hard to “glue” together components based on different data models. In general, the above-mentioned approaches have a strong emphasis on solving technical component interaction problems. This means, that the same abstraction principles are not applied to the connectors as they do for the components themselves.

Instead of having to deal with code level components and their unpredictable side-effects and interaction problems, a better design approach is one, which starts writing a high-level description of the system and transforms it into executable code using refinement steps where the same abstraction mechanisms are applied to both components and connectors.

The semantics of this software design approach as well as its basic entities: services, components, connectors, views, well-formedness rules, refinement rules, consistency rules for specifications and constraints, should be represented in a formal way, so that it is possible to reason about every development step, thus facilitates building systems that are correct by construction. Further, a concept of generic views needs to

be introduced in order to cover different aspects of a concrete component model instantiation in a flexible manner. This means that the model should abstract from concrete notations for specification and constraints. A possible way to represent these predicates are so called “external functions” which can be used as hooks to abstractly formulate consistency rules of the model semantics. For concrete realizations of the component model, these predicates are then instantiated with concrete external functionality given as theorem provers.

This talk describes the component-based, architectural design of software systems, or *Service Layer Model (SLM)* that perhaps offers a better approach to component based software development than the traditional approaches and all the drawbacks mentioned above. The development process defined in the SLM can roughly be described as follows: starting from a system specification, an initial design model is developed containing components that should be combined in the new system. This model is then step by step transformed into a description of the system that can directly be used to derive an implementation. This process is structured into different phases, which are supported by a model-checker that is used to evaluate correct specifications.

References

- [1] M. Anlauff. Xasm -- *An Extensible, Component-Based Abstract State Machines Language*. Proceedings of the ASM 2000 Workshop, LNCS 1912
- [2] I. Crnkovic, H. Schmidt, J. Stafford, and K. Wallnau. *Component Certification and System Prediction*. 4th ICSE Workshop on Component-Based Software Engineering, 2003
- [3] A. Suenbuel, *Architectural Design of Evolutionary Software Systems in Continuous Software Engineering*, DAV, September 2001, ISBN 3-935316-84-4
- [4] B.J. Krämer, R.H. Reussner, and H.-W. Schmidt: *Predicting Properties of Component Based Software Architectures through Parameterized Contracts* in: M. Wirsing, S. Balsamo (Eds.) *Radical Innovations of Software and Systems Engineering in the Future*, 2002