Introduction from the session chair Component Models and Technologies

Merijn de Jonge
Department of Mathematics and Computer Science,
Eindhoven University of Technology, P.O. Box 513,
5600 MB Eindhoven, The Netherlands
M.de.Jonge@tue.nl

Component-based software engineering (CBSE) is concerned with building software systems from prefabricated components. But what exactly are components?

Components exist in different flavors. For instance, a component can be a binary building block according to the definition of Szyperski [6]. Their internals cannot be inspected or modified (black-box reuse) and they are directly usable. Components can also be in source form [4, 5]. Such components have to be compiled first, before they can be used. Furthermore, their internals can be inspected and they can be modified (white-box reuse), although the latter is usually forbidden. Components can also be collections of models describing different aspects of a component. Examples are, the documentation model which contains documentation for a component, execution model which contains the reusable asset, and performance model which describes performance aspects of the component.

To build a software system from a set of components, these components need to be assembled (i.e., composed). Since there are many different forms of components, there are also many different forms of composition. Consequently, if we talk about CBSE, we have to precisely specify what we mean my component *and* what kind of composition we have in mind. This definition of component and composition constitutes a *component model*.

When developing a software system according to a particular component model, techniques are used that prepare components for composition and techniques that perform the composition. For instance, COM [3] uses global unique identifiers to identify components, ToolBus [2] uses an event handling mechanism for dispatching operation calls, Koala uses C function call binding to link function calls to function bodies. A *component technology* is the set of techniques that enable composition in a particular component model. Observe that there may exist multiple component technologies for a single component model. The set of tools, libraries etc. that implement the component techniques for a component model forms a *component architecture*. For instance, the architecture for Koala includes the Koala compiler, a generic build environment, the Koala software development environment, and the Koala modeler.

There exist several moments of composition during the life time of a software system. Each composition moment may require a different component model. For instance, at development-time, composition typically involves composing source files, Makefiles etc. Files play the role of components, and typical component models are the Koala and the source tree composition models. At deployment-time, composition involves obtaining all the parts of a software application and installing it on a target machine. Software distributions play the role of components and RPM [1] is a typical component model. Composition at run-time usually implies loading requested components in a running system. Components are in binair form, COM is a typical component model.

An additinal challenge of CBSE is to combine all the different component models that are involved during the life time of a software system in a uniform way.

References

[1] E. C. Bailey. *Maximum RPM.* Red Hat Software, Inc., 1997.

[2] J. A. Bergstra and P. Klint. The ToolBus coordination architecture. In P. Ciancarini and C. Hankin, editors, *Coordination Languages and Models*, volume 1061 of *Lecture Notes in Computer Science*, pages 75–88. Springer-Verlag, 1996.

[3] D. Box. *Essential COM.* Addison-Wesley, 1998.

[4] M. de Jonge. Source tree composition. In C. Gacek, editor, *Proceedings: Seventh International Conference on Software Reuse*, volume 2319 of *Lecture Notes in Computer Science*, pages 17–32. Springer-Verlag, Apr. 2002.

[5] R. van Ommering, F. van der Linden, J. Kramer, and J. Magee. The Koala component model for consumer electronics software. *IEEE Computer*, 33(3):78–85, Mar. 2000.

[6] C. Szyperski. *Component Software: Beyond Object-Oriented Programming.* Addison-Wesley, 1999.