A Programming Model for Service Oriented Applications

Francisco Curbera

IBM Research, USA
curbera@us.ibm.com

Service oriented computing (SOC) and service oriented architectures introduce a model for distributed software components. Full inter-component interoperability, based on Web services standards, is a core assumption of the SOC model. SOC, as a platform independent approach to software development and management, is not limited to a particular distributed computing stack (Web services), since the benefits of a distributed component model extend to legacy protocols and platforms as well. Web services has successfully stressed the notion that implementation characteristics should be decoupled from interoperability concerns, and has focused on defining an XML based interoperability stack. SOC is directly concerned with the implementation and management of service oriented applications and stresses the ability to incorporate multiple runtimes and programming models into an architecture of distributed software components.

The Service Component Architecture (SCA) is the first realization of SOC as an explicit component model. Just as and Web Services provide the common abstraction of interoperability concerns, SCA provides a common abstraction of implementation concerns. SCA introduces a common notion of service components, service types and service implementations as well as an assembly model for service oriented applications. SCA's goal is to be able to accommodate multiple implementation platforms into a single set of component oriented abstractions. J2EE, BPEL4WS, COBOL, SQL or XML components are only part of the possible implementation artifacts that SCA intends to support. Portability of component assemblies and implementations is an important concern of SCA. SCA is already backed by a Java open source initiative in Apache.

An initiative so ambitious necessarily raises many open issues. Foremost among them is the formalization of an SCA runtime model sufficiently complete to ensure portability of implementations, but at the same time generic enough that it can be supported by multiple platforms and programming models. Once an SCA runtime model is defined, the question arises of whether a "native SCA" platform would be able to provide better support for the execution and deployment of SOC applications. Other significant issues include the possibility of formalizing the component and assembly models beyond their current state, and the support for non-functional requirements and capabilities in the definition and assembly of components.

This talk will review the motivation and major elements of the SCA model, and will discuss the main open issues surrounding the SCA effort.

L. Baresi and R. Heckel (Eds.): FASE 2006, LNCS 3922, p. 1, 2006.