



SCA: A Model for Building Applications and Systems with SOA

By Michael Caughey, TopCoder, Inc.
June 2008

Many TopCoder clients are, or soon will be, implementing an SOA-based development model. Service Component Architecture (SCA) is receiving growing attention as the use of diverse, heterogeneous components occurs increasingly throughout the modern enterprise.

Service Component Architect is one of two main projects ongoing at Open Service Oriented Architecture (www.osoa.org). The project was originally created by a group of vendors including IBM, BEA, Oracle, Cape Clear, Red Hat, IONA and many more and is now managed by OSOA. SCA's goal is to provide a model for developing and deploying components agnostic to technology, which allows for portability of code and designer and developer resources. SCA is broken up into Components, Composites and Domains. It further encapsulates and externalizes how components are connected within a domain.

The most basic element of the SCA is that a component is defined, managed and consumed independent of its specific implementation technology. There are already a number of technologies with APIs for SCA such as Java, C++, BPEL and Spring. The technologies with which the components are implemented is abstracted from the consumer through the SCA API. Components are wired to other dependant components using annotations and Inversion of Control (IoC) through dependency injection. Simply stated this removes the need for a developer to worry about where the component is located - in process, out of process, or even on the same machine - and how the implementation will be constructed. The developer simply uses an Interface for the component and the SCA container insures that there will be an implementation available at run time.

One or more components are then grouped together in a composite or a working set of components generally grouped by logical functionality. The process of building a composite is known as an assembly in SCA. Composites are defined using the Service Component Definition Language, or SCDL (pronounced "skiddle"). It is in the composite definition file, which uses SCDL that the components are wired together. A composite set of components may or may not be co-located on the same computer. The definition file contains a set of binding definitions, which define how services communicate. Bindings may be through Web Services, JMS, EJB or SCA native. Separating how the components communicate allows the developer to focus on the business logic of the component, and allows deployment modification without code modification.

One or more composites are grouped by domain, which is implicitly defined by the vendor implementing the SCA Container. A composite cannot span a domain. Domains are the boundary implied by deploying to a specific vendor implementation. Cross-domain communication is not addressed by the SCA specification.

Component policies are also externalized, allowing an administrator to ensure that policies are set correctly. This includes common tasks like authentication and authorization as well as encryption. This approach allows the administrator to make modifications to the deployment of the components without code changes and it further removes cross-cutting development concerns from the component developer's scope.



SCA continues to build on the methodologies of the past like CORBA, EJB and Web Services and makes it possible to build new applications where the best technology for the job can be employed, yet legacy components still work together with new components. SCA removes the burden from the developer, who can then focus on a particular implementation technology and the business logic necessary. By externalizing configuration, bindings and policies of components, applications are easier to deploy and maintain. By allowing developers to focus on business logic, applications can be assembled and brought to market faster.

Resources:

OSOA Home page

<http://www.osoa.org/display/Main/Home>

SCA Specifications

<http://www.osoa.org/display/Main/Service+Component+Architecture+Specifications>