SCA: A Model for Data and Service Integration on the Web?

Anish Karmarkar, Oracle

Anish.Karmarkar@oracle.com

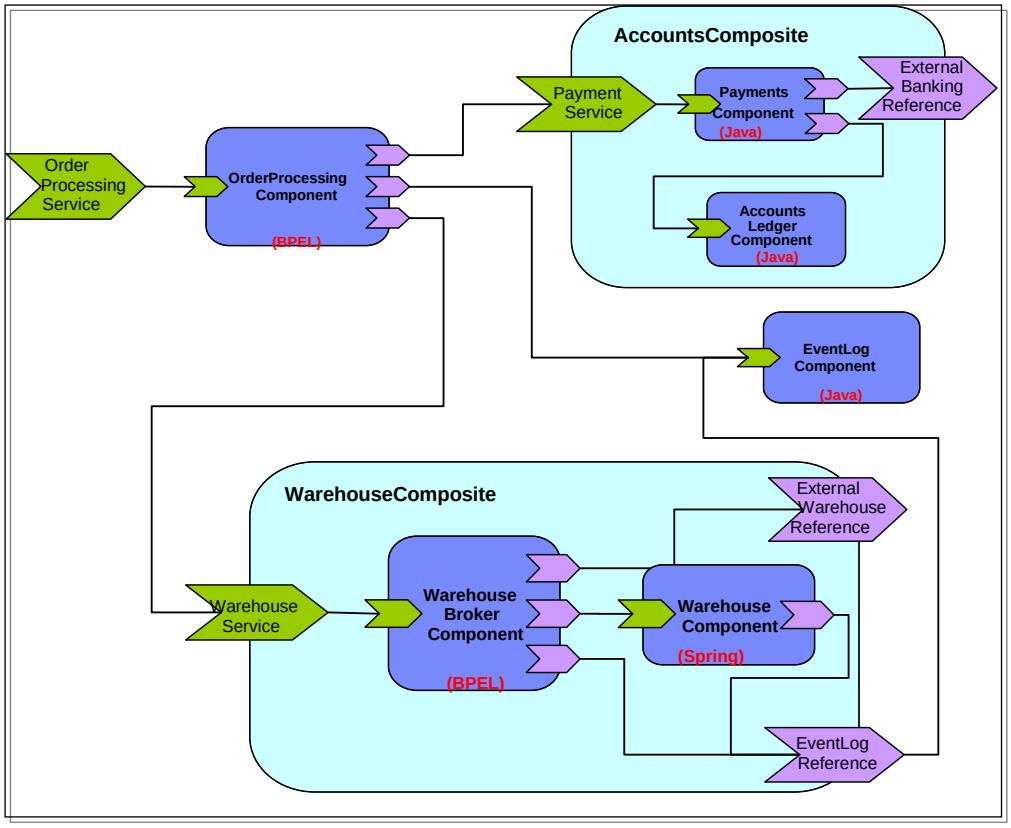
September 16, 2011

1. Introduction and motivation

This position paper contains ideas and solutions for solving some of the data and service integration problems encountered on the web. Service Component Architecture (SCA) [1] is a suite of specifications developed in OASIS [2], which are close to being finalized and becoming a standard. SCA has been used to solve problems faced in the enterprise, especially as it relates to composite applications, service/application integration, mediation, amongst other things. Specifically it allows components that use different implementation technologies, languages, transports, and protocols to be reused to build a composite application that provides added value. It allows for using existing services as well as deploying new services. But the problems that SCA tries to solve are not unique to the enterprise. This short paper investigates three data and service integration problems in the context of the Web and emerging web-based Cloud services, and proposes ways in which SCA may be able to address them.

1.1 Brief Introduction to SCA

Service Component Architecture (SCA) is a set of specifications [1], which define a model for building applications and systems using a Service-Oriented Architecture. In SCA, services are provided by *service components*, which may in turn use services provided by other components. Multiple service components can be configured and assembled into groupings called *composites*, to provide specific business capabilities, which model connections between components through *wires*. SCA supports the creation and reuse of service components using a wide variety of implementation technologies, including general purpose programming languages such as Java™, C++; orchestration languages such as WS-BPEL; components frameworks such as Java Enterprise Edition and the Spring Framework. Integration of both new and existing components is a key feature of SCA assemblies, including the ability to assemble existing components that are not SCA-aware. An example SCA composite that uses multiple technologies is depicted below.



2. Scenarios

The scenarios below do not try to exhaustively cover all the data and service integration cases that occur on the web. These scenarios represent a class of problems that can be potentially solved by SCA.

2.1 Monitoring and Management of Cloud Applications

This scenario consists of an application or instances of multiple applications that are deployed across multiple cloud service providers. There is a need to monitor various components that are deployed in this multi-cloud environment and manage them as needed. This requires aggregation of various status data, a dashboard if you will, which is presented to the administrator. The administrator may from time to time take action based on the status or this may be automated based on predefined policies. For example, a change in average response time may result in the administrator provisioning additional compute-power, data-store, or bandwidth. This may be a decision based on pricing and knowledge of various bottlenecks within a particular application.

2.2 Exposing a Multifaceted Service(s)

A company wants to open up its proprietary data on local businesses and their ratings and make it available on the Web. It wants to take advantage of 'the wisdom of crowds' to enhance its data and keep it current. This requires them to expose the data over the Web using RESTful APIs. It has to also cater to implementations that want to use Web services and has prior commitments to make the same data available over legacy proprietary interfaces and protocols. This means that the same data needs to

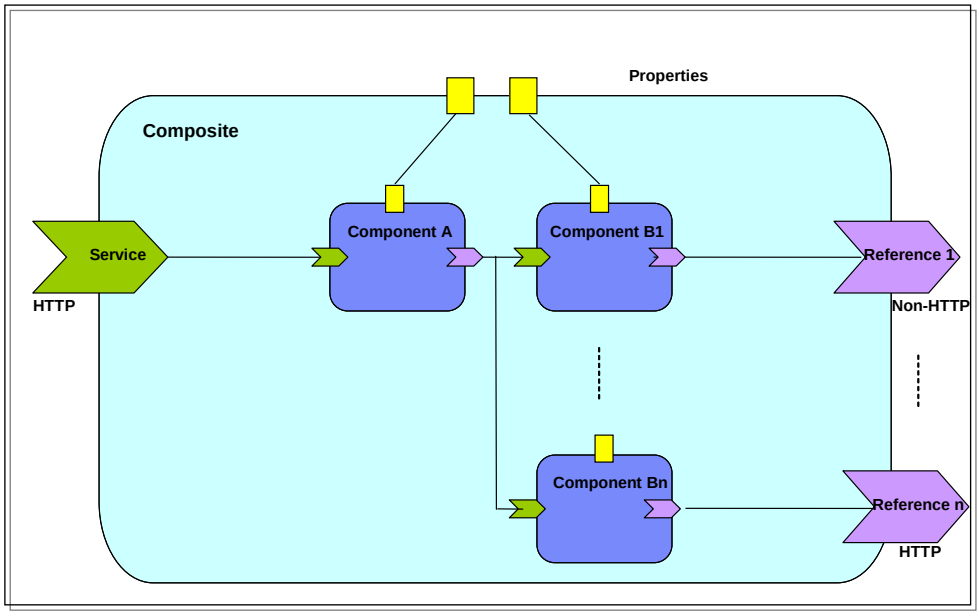
be available via multiple access points, each providing support for different format/protocol. It is also necessary to provide consistency across various access points without disadvantaging a particular format or protocol.

2.3 Fault-tolerant and load-balancing services

With the advent of SIP-based VOIP technology, several VOIP providers have sprung up that are based on different business and pricing models. Unfortunately, these providers are notoriously unreliable with respect to their availability and capacity. Furthermore they have different pricing models depending on usage, quality-level etc. A softphone application is to be written to take advantage of these varying business/pricing models and combine it with regular 'landlines' to provide a cheaper service with higher availability. Therefore, the application cannot rely on a single provider and has to be written to work with multiple-providers some of who may be periodically unavailable or may not have the needed capacity. The application therefore must balance the call load across multiple providers based on capacity, pricing and usage and must work around unavailable providers.

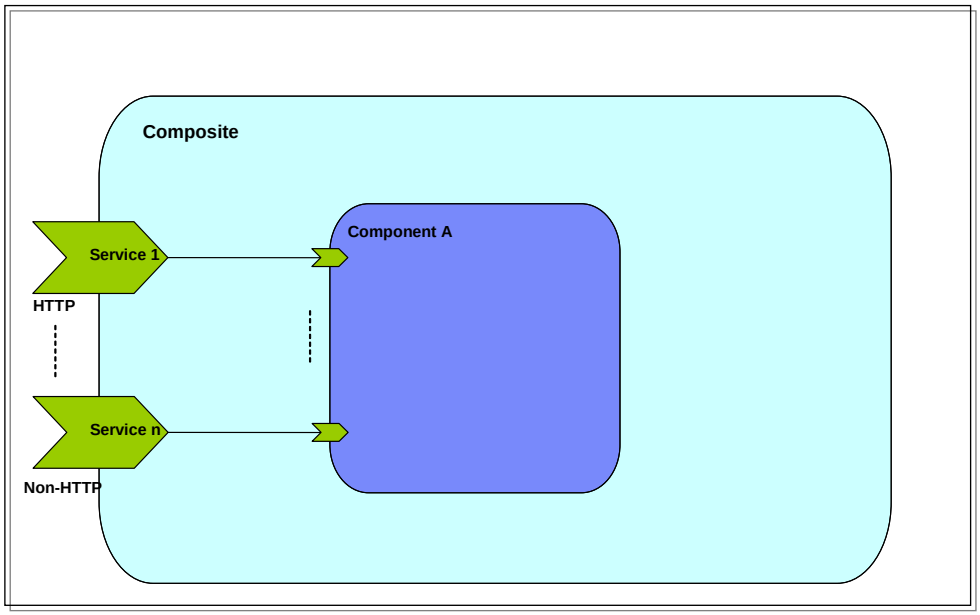
3. How does SCA address these scenarios?

3.1 Monitoring and Management of Cloud Applications



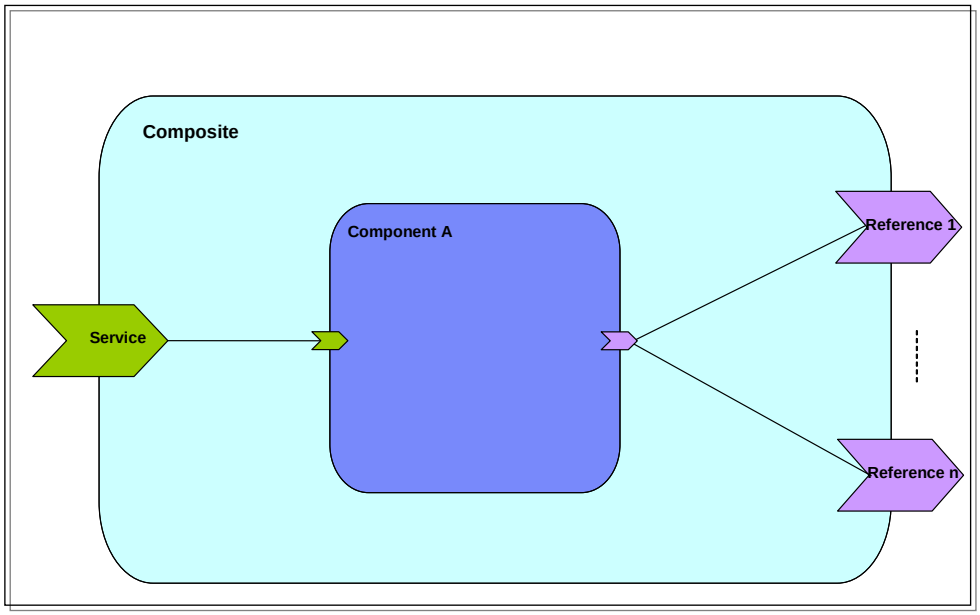
The figure above contains a composite that exposes a RESTful HTTP cloud dashboard service and contains component B1 through Bn, each of which connect to a different cloud provider over a variety of protocols. Component A provides the aggregation, control and presentation necessary for the dashboard service.

3.2 Exposing a Multifaceted Service(s)



The figure above consists of a composite with a single component that exposes access to the underlying data over multiple protocols/formats by exposing multiple services each with a different binding or format. This provides an ability for the consumer to choose the format/protocol that servers it the best by choosing the appropriate service.

3.3 Providing Fault-tolerant and load-balancing service



The figure above consists of a single composite that connects to multiple external SIP and PSTN providers. The component also offers a single service that provides the fault-tolerance/load-balancing needed to work around failures, trunk capacity, and policies.

4. What is missing?

Although the SCA model has built-in extensibility and support for heterogeneous implementation, interface, and binding technologies, the current proposed SCA standards use WSDL 1.1 interface as the convergence point for interoperability, requiring all interfaces to be mappable to WSDL 1.1. This, for example, does not work well with RESTful services and applications. Something similar to WADL [3], which is REST-friendly, would be more appropriate for this purpose. Furthermore, SCA does not include (although it does not prevent the creation of such an extension) a standard for an HTTP binding as well as various technologies that are popular on the web (PHP, Ruby on Rails, etc). SCA would have to be enhanced/extended to provide support for these technologies.

5. Summary

SCA model does seem like a good fit to solve the data/service integration problems listed in the scenarios above. This would require further experimentation and extension of the existing SCA standards to provide support for technologies that are popular on the web including support for WADL, HTTP-binding and similar REST-friendly technologies. SCA model is focused on allowing heterogeneous technologies with copious extension mechanisms that will enable such experimentation.

References

[1] http://www.oasis-opencsa.org/sca

[2] http://oasis-open.org/

[3] http://www.w3.org/Submission/wadl/