REQUIREMENTS FOR A NEXT-GENERATION WEB SERVICES ARCHITECTURE

Robert F. MacInnis, Ph.D.
June 2017

AetherWorks
501 Fifth Avenue
New York, NY 10017

E: info@activeaether.com

1 INTRODUCTION

This document defines a set of requirements that a next-generation architecture must meet. These requirements aim to encourage a separation of concerns between participants in the architecture, to advance the creation of mechanisms which can accommodate heterogeneous technologies, and to promote the development of architectures which are resilient to various types of failure. Although some of the existing architectures address some of the individual requirements, the satisfaction of this complete set of requirements is unique to this next-generation model.

2 ACCOMMODATING TECHNOLOGICAL HETEROGENEITY

There currently exist multiple, competing Web Services technologies for which a large number of provider agent and consumer agent applications have been developed, and it can be reasonably assumed that provider agent and consumer agent applications will continue to be developed using more than one set of technologies. If the goal of universal interoperation is to be realized, then technological heterogeneity must be taken into account in the design of any next-generation architecture. The first three requirements address the accommodation of heterogeneity in the technology used for developing, deploying, and hosting provider agent Web Service applications:

In order to promote the integration of existing closed-world systems and foster the development of new, open-world environments, it must be possible to include hosting resources with various configurations and application server technologies.

Requirement 1: The architecture must enable the incorporation and utilization of heterogeneous Web Service hosting resources.

Providing a model which enables the incorporation of existing Web Service implementations with minimal effort can encourage existing and future provider agent-publishers to participate in an open market for software services. This is argued to be of benefit to the Service Consumer as it increases access to services previously trapped in proprietary environments, which may in turn lower costs (through competition) and save on development time.

Requirement 2: It must be possible for existing Web Service provider agent application implementations to be utilized in the architecture without any modifications to their code.

Generic deployment mechanisms are argued to be of critical importance to any architecture which aims to support heterogeneous technologies. Different Web Service application hosting technologies, for example, require different sets of steps to be executed in order to deploy and expose a provider agent application as a Web Service. Identifying common traits amongst the deployment procedures of Web Service application hosting environments will allow generic mechanisms to be developed for Web Service deployment and undeployment, which in turn will enable the isolation (and possibly automation) of the responsibility of managing the provisioning of Web Service endpoints, independent of the specific technologies in use.

Requirement 3: The model must provide technology-neutral mechanisms for instigating the deployment and undeployment of Web Services.

3 SEPARATING ENTANGLED CONCERNS

The Service Provider role, as currently defined, involves high cognitive complexity and has high barriers to entry, discouraging participation by individuals due to the cost and complexity involved in fulfilling all the responsibilities of the role. Having identified the need for a separation of concerns and a more fine-grained approach to assigning actors' responsibilities, requirements 4 and 5 address hosting resources, while requirements 6 through 8 pertain to the development of provider agent applications (i.e. deployable Web Service implementations).

Requirement 4: The model must lower the barriers to participation in the provision of Web Services.

Implementing the business logic of a Web Service is very different from hosting a Web Service, requires a different skill-set, and a different set of software and hardware technologies. Separating the tasks of implementing and deploying a Web Service also protects the hosts by leaving the process of deployment within their control – and further enables the development of a marketplace for hosting resources.

Requirement 5: The tasks of developing and publishing a Web Service must be separate from the task of deploying and hosting a Web Service.

Separating the basic responsibilities of a 'host' (exposing interfaces for remote access to applications deployed within its domain) from activities surrounding 'hosting' (deploying and undeploying Web Service endpoints on host machines) allows hosts to be viewed as a consumable resource. Because statically provisioning resources can be wasteful, any new architecture should enable available hosts to be added to a pool of available hosting

capacity which can then be applied to suit the goals of the architecture as and when necessary. This separation further encourages a 'market-based' approach to pairing provider applications with suitable hosts.

Requirement 6: It must be possible to exploit the latent computational resources of hosts in a distributed computing environment.

Because Web Service standards do not specify the procedures for re-locating a migrated Web Service endpoint (or an alternative endpoint in the case of failure), ad-hoc methods must currently be developed & implemented, taking time and introducing complexity and the potential for failure into consumer agent applications.

Requirement 7: Mechanisms must be provided which disentangle distribution-related details from the business logic of consumer agent applications.

Monitoring and reporting of Web Service usage data is an important aspect of the Web Service lifecycle which enables managing entities to respond to failure and make decisions on the required level of resource provisioning. In the same way that implementing the business logic of a consumer agent application must be separated from handling distribution-related details, Web Service implementations must not be required to perform any environment-specific tasks which are unrelated to providing and fulfilling requests for its advertised operations.

Requirement 8: The architecture must provide means for the monitoring and management of Web Service provisioning which place no requirements on Web Service provider agent implementations to record or report usage data.

4 DESIGNING FOR FAILURE

For developers of consumer agent applications, programmatically detecting and recovering from failure is an onerous, complex, and time-consuming task, fraught with difficultly and opportunities for error. It is argued that there is an expectation of failure in distributed systems and, knowing this, that any next-generation system must be designed to gracefully accommodate failure within the architecture.

Requirement 9: The architecture must be resilient to the failure of its components and the Web Services which rely upon them.

Finally, it is important to take into account the intrinsic fallibility of hosts on the Internet. Designing for dynamically changing levels of hosting resources will enable systems to evolve to changing user demands and provide more reliable and highly available services.

Requirement 10: The model must gracefully handle the addition and removal of hosting resources.