Chapter 12
Bridging the SOA and REST Architectural Styles

José C. Delgado
Technical University of Lisbon, Portugal

ABSTRACT

SOA and REST are the most used architectural styles for distributed interoperability solutions. Each is more adequate to a different class of applications and exhibits, advantages and disadvantages. This chapter performs a comparative study between them, after establishing a hierarchical classification of architectural styles and showing how to derive them by progressively introducing architectural constraints. An underlying model, which relates resources, services, and processes, is presented. It is shown that SOA and REST are dual architectural styles, one oriented towards behavior and the other towards state. This raises the question of whether it is possible to combine them to maximize the advantages and to minimize the disadvantages.

This chapter proposes a new architectural style, based on a combination of the best characteristics of SOA and REST, which the authors designate as Structural Services. Unlike REST, resources are able to offer a variable set of operations, and unlike SOA, services are allowed to have structure. This style uses structural interoperability, which includes structural compliance and conformance.

INTRODUCTION

Originally, the Web (Berners-Lee, 1999) was designed as a solution to the problem of browsing remote hypermedia documents, in which text was the main format. This justified the choice of a text markup document description language, HTML (Toshniwal & Agrawal, 2004). Today, the world is rather different. Text is just one among other formats, with binary information (pictures, video, voice) taking a great part of the data bandwidth. Users are no longer satisfied with information browsing and want more value, in the form of services. Organizations are also customers of other organizations, in a value-added chain, and began using the same infrastructure to setup their information systems and integrate them with their partners’ (enterprise integration), in which Web services replaced specific application-to-application protocols.

DOI: 10.4018/978-1-4666-2488-7.ch012
Fielding (2000, p. 107) contends that, since 1994, the REST style has been used to guide the design of the Web. In fact, most early Web applications were highly scalable, with many clients for each server, and a good match for the REST style. As Web technologies gained generalized acceptance and distributed interoperability became the usual scenario, enterprise class applications transitioned from classical object interoperability, based on RPC (Remote Procedure Call), such as Java-RMI or CORBA (Bolton, 2001), to standard, Web-based technologies. These applications require more flexibility than scalability, since they shifted the client emphasis from quantity (many people for one server) to business functionality. The concept of Web Service appeared as a solution to the interoperability problem, as an extension of the object concept in the distributed realm.

The Web of Documents has evolved into a global Web of Services (Tolk, 2006), in which the interacting parties are both humans and computer applications and content is increasingly dynamic, generated on the fly according to some database and business logic. The underlying logical architecture became less client-server and more peer based, in particular when enterprise applications are involved.

SOA (Earl, 2005), instantiated by Web Services, embodied a major evolution, the transition from the client-server to the service paradigm. Web services required a new protocol (SOAP) and a way to express the interfaces of services (WSDL). HTTP and HTML are technologies that were originally conceived for large-scale applications and human clients. XML, which maintained the text markup style of HTML, made computer based clients easier and, together with HTTP, became the cornerstone of Web Services technologies. This evolutionary transition is perfectly understandable in market and standardization terms, but still constitutes a mismatch towards both humans and applications.

HTTP is an application level protocol, synchronous and committed to the client-server paradigm. In particular, it does not support full duplex, long running sessions required by general services such as those found at the enterprise level. This has spurred workaround technologies such as AJAX (Holdener, 2008) and Comet (Crane & McCarthy, 2008). Web Sockets, now part of the HTML5 world (Lubbers, Albers, & Salim, 2010), removes this restriction, adds binary support and increases performance.

XML is verbose and complex, has limited support for binary formats, is inefficient in computer terms due to parsing and exhibits symmetric interoperability, based on both sender and receiver using the same schema, which constitutes a relevant coupling problem.

The notion of schema matching (Jeong, Lee, Cho, & Lee, 2008) is used only for service discovery, not for message exchange. This means that Web Services are more a legacy integration and interoperability mechanism than a true and native service oriented solution. Their universal service interoperability came at the price of complexity, with schemas and all the associated standards. Tools automate and simplify a lot, but they just hide the complexity and do not eliminate it. This has spurred a movement towards simpler, more manageable systems, in the form of a resource oriented architectural style, REST (Fielding, 2000), and of a simpler data format, JSON (Zyp, 2010), promoting what is known as RESTful applications (Laitkorpi, Selonen, & Systa, 2009; Li & Chou, 2010).

Whereas SOA emphasizes behavior and rich interfaces, the guiding aspects of REST are state, structure and uniform interfaces. All resources implement the same service, with the same set of operations, albeit with different implementations. These typically exhibit CRUD style (Create, Read, Update, and Delete) and are mapped onto HTTP verbs (GET, PUT, POST, and DELETE). Functionality that in SOA would be modeled as an operation is modeled in REST as a resource.

REST is defined at a lower level than SOA, with a higher semantic gap between the problem