Chapter 16
Unifying Services and Resources:
A Unified Architectural Style for Integrations

José C. Delgado
Instituto Superior Técnico, Universidade de Lisboa, Portugal

ABSTRACT

Current integration solutions are still based on technologies developed for the original Web problem, which is browsing remote hypermedia documents with text as the main media type. Text-based data descriptions such as XML and JSON and stateless and connectionless protocols such as HTTP are still the norm to achieve distributed integration. However, the Web today is much more dynamic, in that resources are no longer passive hypermedia documents but are active and implement services. SOA and REST are the most used architectural styles to implement distributed integration, and each exhibits advantages and disadvantages. This chapter illustrates that they are dual architectural styles—one oriented towards behavior and the other towards state—and contends that it is possible to combine them to maximize the advantages and to minimize the disadvantages. A new architectural style, designated Structural Services, is proposed and described. Unlike REST, resources are able to offer a variable set of operations, and unlike SOA, services are allowed to have structure. To minimize resource coupling, this style uses structural interoperability based on the concepts of structural compliance and conformance, instead of schema sharing (as in SOA) or standardized and previously agreed upon media types (as in REST). To delineate how this style can be implemented, a new distributed programming language is presented.

DOI: 10.4018/978-1-4666-6178-3.ch016

Copyright © 2014, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.

INTRODUCTION

The main integration technologies available today are based on the *SOA* (Erl, 2005) and *REST* (Webber, Parastatidis, & Robinson, 2010) architectural styles, with SOAP Web services (WS) and HTTP RESTful services as the main workhorses. They constitute two different approaches to solve the same problem: application integration. SOA has the goal of a low semantic gap, since it models real world entities by resources with services that express their capabilities. This is good in modeling terms, but entails a coupling between the provider and the consumer that hampers dynamic changeability and adaptability. If the provider's interface changes, the consumer's interface also needs to change in accordance. There is no apparent structure, since service composition is hidden behind each service interface.

One of the main goals of REST is to reduce the coupling between provider and consumer, both to increase scalability and adaptability. Real world entities are modeled in a data-oriented manner by resources, all with the same syntactical interface (same set of operations). Semantics are restricted to a set of data types (or schemas), either standardized or previously agreed upon between the interacting entities. The variability of the characteristics of entities is modeled by visible structure (resources composed of other resources) and the semantics of the agreed data types.

Unfortunately, the decoupling goal of REST is somewhat elusive. Messages cannot be understood simply by exploring their data structure, touching links blindly. Semantics and behavior need to be considered as well, and this is determined by the type of resources used. For example, if the provider decides to change its specifications, the code at the consumer will most likely be unable to cope with that.

What happens in practice is that REST on HTTP is simpler to use than SOA style SOAP Web services and many applications are simple enough to adopt a data-oriented interface i.e., REST style. This means that, although REST represents a modeling shift from real world entities (lowering the modeling level and increasing the semantic gap), it is still simpler to use than a full-blown SOA environment. Unless, of course, the application is sufficiently complex to make the semantic gap visible and relevant enough. This is why REST is preferable in simpler applications and SOA is a better match for complex, enterprise-level applications.

Nevertheless, REST gets one aspect right: in a distributed context, interoperability has to be based on structural composition of previously known entities. Forcing these to have one single set of operations, however, does not increase adaptability in the general sense; it only leads applications to adopt a data-oriented style, which is not adequate for all classes of applications.

The following questions then arise:

- Why do we have to choose between the service-oriented (SOA) and data-oriented (REST) styles, instead of combining both and using the best approach for each part of the application?
- How can we increase adaptability without enforcing some particular application style?

This chapter revisits the integration problem with an open mind, without being restricted a priori by existing technologies. The only assumption is that there are entities that need to interact, by using messages. Then, an integration model is defined and its various characteristics compared with those of current technologies, showing how this model can solve some of their limitations.

The main goal of this chapter is to propose and describe a new architectural style, designated *Structural Services*, which combines the best char-

32 more pages are available in the full version of this document, which may be purchased using the "Add to Cart" button on the publisher's webpage:
www.igi-global.com/chapter/unifying-services-and-resources/115437

Related Content

Business Process Modeling with Services: Reverse Engineering Databases
Youcef Baghdadiand Naoufel Kraiem (2014). *Uncovering Essential Software Artifacts through Business Process Archeology (pp. 177-200).*
www.irma-international.org/chapter/business-process-modeling-with-services/96620

Automated Synthesis and Ranking of Secure BPMN Orchestrators
Vincenzo Ciancia, Jose Martin, Fabio Martinelli, Ilaria Matteucci, Marinella Petrocchiand Ernesto Pimentel (2014). *International Journal of Secure Software Engineering (pp. 44-64).*
www.irma-international.org/article/automated-synthesis-and-ranking-of-secure-bpmn-orchestrators/113726

Design Patterns and Design Principles for Internal Domain-Specific Languages
Sebastian Günther (2013). *Formal and Practical Aspects of Domain-Specific Languages: Recent Developments (pp. 156-214).*
www.irma-international.org/chapter/design-patterns-design-principles-internal/71820

Finding Optimal Transport Route and Retail Outlet Location Using Mobile Phone Location Data
Giridhar Majiand Soumya Sen (2022). *International Journal of Software Innovation (pp. 1-20).*
www.irma-international.org/article/finding-optimal-transport-route-and-retail-outlet-location-using-mobile-phone-location-data/301226

Building a Self-Sustaining World: How AI and Self-Sustaining Systems Converge
Prithi Samuel, Reshmy A. K., Sudha Rajeshand Karthika R. A. (2024). *The Convergence of Self-Sustaining Systems With AI and IoT (pp. 85-103).*
www.irma-international.org/chapter/building-a-self-sustaining-world/345507