Architectural Styles for Distributed Interoperability

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

ABSTRACT

SOA and REST are the two main architectural styles used for distributed applications. Current SOA implementations use Web Services, which have a flexible service interface but lack resource structure. Current REST implementations use HTTP, with good structural support (through URIs) but fixed service interface, leading to a significant mismatch in modeling some application artifacts. In addition, current data description languages support distributed interoperability by schema sharing (XML) or predefined media types (JSON). This paper makes two main proposals: an architectural style, Structural Service, capable of combining resource structure with flexible services, and a distributed interoperability mechanism, based on structural compliance and conformance, which supports partial interoperability and reduces coupling between interacting services while ensuring that the interoperability requirements of the application are satisfied. Since current technologies do not readily support these proposals, the author developed a language and its execution platform that supports structural services and compliance and conformance.

Keywords: Architectural Style, Compliance, Conformance, Distributed, Interoperability, Lifecycle, Resource, Service, Structural, Web of Services

INTRODUCTION

The Web started out as a solution to the problem of browsing remote hypermedia documents (Berners-Lee, 1999). This was essentially an instance of the client-server paradigm geared for global human-to-human interaction, which in fact was the major factor that spurred all subsequent developments.

Up to then, computers could communicate, but this was basically limited to system-to-system or application-to-application data exchange, or human-to-human text messages in e-mail or newsgroups. This was the first time that a human could place a multimedia document in some server and any other human, with a suitable client (the browser), could easily access that document and others accessible from it, with just a few clicks. The world of information had left the realm of specialists and reached the general user.

In those days, text was king. The main inspiration came from the printed document, which essentially was text sprinkled with other artifacts such as pictures and tables. The choice of a text markup document description language (HTML), based on another one coming from the printing world (SGML), seems natural and more than justified (Toshniwal & Agrawal, 2004).

Today, the world is rather different. Text is just one among other formats, with binary information (e.g., pictures, video, voice) taking a great part of the data bandwidth. Humans are no longer satisfied with information browsing

DOI: 10.4018/irmj.2013100103
and want added value in the form of services. Service providers recognize this value and are moving away from face-to-face contact with customers in physical shops to virtual presence in sites that allow anyone, anywhere, to instantly access services or to buy products. 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.

The human-to-human Web of Documents has evolved into a global Web of Services (Tolk, 2006), in which the interacting parties are both humans and computer applications. Actually, contents are increasingly dynamic, generated on the fly according to some database, and the basic, manual hypermedia document creation (corresponding to the original human-to-human interaction) is decreasing its importance.

Services are replacing static documents and computer applications are filling in the web space once dominated by humans. Human-to-human interaction is now migrating to a higher level, the social network. However, computer applications were left using the technology once developed to describe data in a user-oriented way (hypermedia document browsing).

Granted, there was a major evolution milestone when HTML evolved into XML, with presentation separated from data description, language extensibility and metadata introduced to self-describe the information. However, XML is not really a general data description language, since it has a heavy document-style legacy (text and markup based, weak binary support, element and attribute discrimination, element sequence instead of a composition construct, and so on). Perfectly understandable in terms of evolutionary transition, but still a mismatch towards both humans (verbose and complex, aside trivial cases) and applications (limited regarding binary formats and inefficient due to parsing).

Web Services embodied another major evolution, the transition from the client-server to the service paradigm. However, Web Services were built on top of the text document technology (XML) and the actual work must be done by a separate mechanism (either a general programming language or an orchestration language such as BPEL).

Backwards compatibility in a new technology can be both a bonus (things done so far still work) and a curse (many new features become a compromise instead of what they should be). Wrapping a new technology around an older one is just one way to evolve.

This paper revisits this problem and considers the case of conceiving a Web of Services right from the start, using a service model rather than a document model as a seed, but that can still work with the current XML-based technologies. The main ideas can be summarized as follows:

- The basic web element is the (resource with a) service, not the document;
- The main paradigm is the service, not the process nor the client-server paradigms;
- One language is enough to contemplate several currently separated levels (data, service interface and service behavior, e.g., XML, WSDL and BPEL);
- This language is optimized both for applications and humans (with two formats, automatically synchronized);
- People interface the web not as mere clients but as fully-fledged resources with services, which means that they can both invoke and offer services (in other words, humans become first class citizens in the Web of Services). This can be achieved if their web access device is a browser-ver (a browser and a server working together) instead of a mere browser (Delgado, 2012c).

The main problems at stake here and that we are trying to solve are:

- Mismatch between the current architectural styles, such as SOA and REST, and the model of the Web of Services;
Implementing a Data Mining Solution for an Automobile Insurance Company: Reconciling Theoretical Benefits with Practical Considerations
www.igi-global.com/article/implementing-data-mining-solution-automobile/44533?camid=4v1a