Discussion Summary: Characteristics of Web Services and Their Impact on Testing, Analysis and Verification

Shriram Krishnamurthi
Brown University
sk@cs.brown.edu

Tevfik Bultan
University of California, Santa Barbara
bultan@cs.ucsb.edu

ABSTRACT

This is a summary of the comments from a discussion session at the end of the Workshop on Testing, Analysis and Verification of Web Services (TAV-WEB) 2004. The comments were made by the workshop participants, and were compiled and edited by the workshop organizers.

TAV-WEB 2004 was held in conjunction with ISSTA 2004 in Boston, Massachusetts on July 11, 2004. The abstracts of the papers presented in TAV-WEB 2004 appear in the September issue of SEN.

DISCUSSION SUMMARY

The scalability issues in Web services may be different from those of traditional programs. Scale isn't so much about lines of code as it is about crossing enterprise boundaries. Some dimensions along with Web services must confront scale include:

- the complexity of the workflow
- the volume of data
- the number of nodes
- the complexity of operations
- the differences in usage patterns

Web services rely on the standardized exchange of data, though standards like WSDL may be more important than lower-level syntactic notations such as XML. Indeed, WSDL files sometimes offer two interfaces to a given service: one, a "standard" interface, that uses an XML notation, and a "customized" interface that uses a domain-specific and more concise, standardized or otherwise appropriate data exchange format.

Security is an important problem given the cross-enterprise exchange of potentially sensitive data. Some research is attempting to build security into the messages, but this doesn't yet appear to be mature. Meanwhile, many people use SSL for communication; while a useful start, this is clearly not a long-term (or end-to-end) solution.

Web services are also standardizing rich interface notations. For instance, choreography languages specify lists of operations with partial orderings between them. The heterogeneity and distribution of Web services forces the need for such richer interface languages.

Web services seem to inherently be asynchronous. This helps them better adapt to problems of availability and distribution. This in turn makes them truly different from traditional RPC mechanisms and more like a distributed dataflow notation.

Distribution is an important part of Web services, and this significantly impacts any form of analysis—do you have access to the internals, even the specifications? These problems are especially manifest when systems are no longer within a single enterprise.

While many people expect Web services to eventually span several enterprises, many corporations are already trying to use Web services in-house even as cross-enterprise standards are being established. Developers must therefore handle "legacy" applications such as SAP by creating appropriate wrappers—not only service fresh applications written under the Web services rubric.

The current focus in Web services seems more on specifying what processes do as opposed to what they are *supposed* to do. That is, there is a lack of of good languages for defining properties for validation, analogous to those available for, say, hardware.

There is also some lack of definition in the validation question. For instance, we only have access to a WSDL specification, what meaningful result can we obtain from verification? While WSDL isn't rich enough to do much verification, other standards—such as BPEL, which can contain an abstract workflow that behaves as a contract—contains more information. These standards are evolving out of practical needs but also helping in validation.

On the other hand, Web services are multi-stakeholder, since they necessarily cross institutional and role boundaries. It is therefore essentially impossible to find consistent global requirements. Furthermore, individual services will not be aggressive about specifying their behavior, so the interfaces they publish will necessarily be rather sparse. Indeed, there may even be incentive for parties to lie.

Web services create problems of availability and dynamicity, especially in the context of automated discovery and composition of (possibly unavailable) services. Indeed, availability (or the lack thereof) can genuinely change the behavior of a system. Nevertheless, many Web service applications lack a directory or fail to use UDDI: the Web services standards simply serve as a protocol between nodes in a loosely-coupled distributed application.

Some of the problems of scale imposed by Web services are new. Traditional programming languages simply don't scale to enterprise-wide or cross-enterprise-scale systems—you need workflow orchestration to scale to that level. Web services are becoming the infrastructure for this, and the challenges are to restate and address both traditional and new questions atop this platform.

While Web services are still new, the set of applications built atop them is clearly going to grow in complexity and will explore new territories. For instance, they may represent new platforms for program- and even computation-migration, not only data-transfer for remote processing. Therefore, we should be thinking in terms of a broader vision of what Web services can accomplish, not be limited by the current set of applications.

The interplay between testing, analysis and verification of Web services, and the work on the Semantic Web, is still a little unclear. Clearly, problems of discovery involve questions of stating what it is we want to discover, and discovery is a necessary component of an overall Web service solution even if people are currently not yet using UDDI in interesting ways. The point of discovery may be an interface between work on the Semantic Web and work on the mechanisms of Web services, though eventually there should be interplay between the two.

We have general agreement that we need more and richer examples of concrete Web services; in particular, industry needs to better feed academic research with such examples. In the meanwhile, we can find some interesting examples at places such as the WS-I's Web site.

Finally, Web services appear to be a fascinating branch of computer science that bring together expertise in numerous subdisciplines: databases, operating systems, distribution, software engineering, programming languages, compilers, etc. This makes it harder to understand the whole, but also makes the area more rich and interesting.