For Web services to grow as predicted, researchers must not wait to address the challenges of software trustworthiness.

Jia Zhang

Trustworthy Web Services: Actions for Now

Over the past several decades, software has become an integral part of all government, military, and business systems, growing more complex and functional as its use expands to critical systems. Software's pervasiveness in these systems, including those driving and supporting the national infrastructure, homeland security, healthcare, and e-commerce, demands a certain level of confidence that the software will function as intended. In 1990, David Parnas and colleagues coined the term *software trustworthiness* to represent this confidence ("Evaluation of Safety-Critical Software," *Comm. ACM*, June 1990, pp. 636-648), and many researchers and practitioners have since explored aspects of this term—reliability, security, safety, survivability, interoperability, availability, and fault tolerance.

Only in the past few years, however, have software *vendors* begun to recognize that software trustworthiness is of paramount importance. The "Trustworthy Computing" sidebar explains why. Accelerating the issue of software trustworthiness is the growing popularity of Web services—programmable Web applications that are universally accessible through standard Internet protocols ("What Are Web Services?" Christopher Ferris and Joel Farrell, *Comm. ACM,* June 2003, p. 31). A variety of Web services appear daily, and an increasing number of software producers announce products that are Web-service-enabled. The Web services paradigm has gained enough momentum in academia and industry that the view of the Internet as data repository is changing. The "Changing the Face of the Internet" sidebar tells how this transformation is occurring.

But not everyone is willing to take the Web services plunge. Although Web services are a boon to e-commerce, they come at the high price of shaky security. Current methods and technology simply cannot ensure trustworthiness in loosely coupled Web services whose integration must be seamless. At present, interest remains high enough in implementing Web services that the lack of trustworthiness is not a bottleneck to industry's adoption of this Internet model. But why wait to find a solution? What better time to explore ways of measuring, testing, and enhancing the trustworthiness of systems that are Web services-oriented than when system implementers are still hammering out details? If research on trustworthiness lags implementation too much, developers will have to retrofit techniques to achieve trustworthiness. They will have fewer options and most likely end up with a substandard solution.

It is possible to build in trustworthiness by creating a layer atop the current Web services framework. At present, the framework stops with WS-Security, a standard that IBM and Microsoft jointly proposed to enhance the quality of protection for Web services (http://www-128.ibm.com/developerworks/webservices/library/ws-secroad/index.html). The framework needs a new trustworthiness layer that defines criteria for determining that a Web service is indeed trustworthy and that measures, enhances, and guarantees trustworthiness.

The first step in creating such a layer is to iden-

Inside

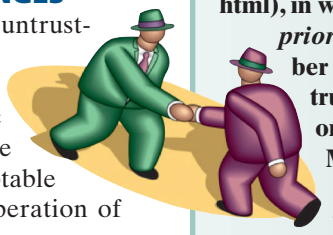
Trustworthy Computing
Changing the Face of the Internet
Resources

 Published by the IEEE Computer Society 1520-9202/05/$20.00 © 2005 IEEE

tify what it must do to overcome trustworthiness obstacles such as unfulfilled requirements and poor interoperability, which can eventually derail industry's progress in adopting the Web services Internet model.

FOUR CRITICAL CHALLENGES

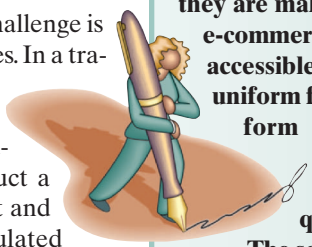
Web services can become untrustworthy for four reasons: unfulfilled requirements, malicious acts and code changes, erratic Internet behaviors or resource scarcity that result in unacceptable delays, and the poor interoperation of selected services.



Unfulfilled requirements

A Web service is an independent service component that aims to serve a variety of needs; it is not a single specific application. When the selected Web service component does not thoroughly fulfill its target need, the service becomes untrustworthy. Testing a service to determine that it fulfills all requirements is tricky because testing must verify the satisfaction of both functional and nonfunctional requirements. The nonfunctional requirements (reliability, availability, survivability, interoperability, and a host of other attributes) decide the service quality and require individual testing. Because the user can invoke a Web service only over the Internet, the challenge is how to test for all possible use scenarios. How can testing determine that the service operates effectively and efficiently for a specific application and environment?

Another aspect of this testing challenge is the dynamic nature of Web services. In a traditional software system, testers know all the components and their relationships before the software executes and so can conduct a thorough test of each component and component interactions in a simulated environment. Theoretically, they can determine the "trustworthiness" of each component and component interaction before the system starts. Web services, on the other hand, dynamically locate and assemble components across the Internet—an extremely large and complex environment. When the Web-services-oriented system requires a service component, it searches a public registry, such as universal description, discovery, and integration



Trustworthy Computing

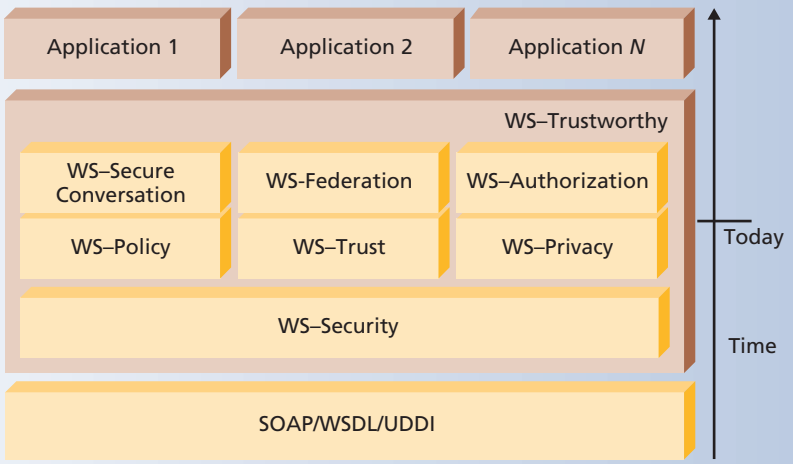
In January 2002, Microsoft's Bill Gates sent an e-mail to all employees to announce the company's Trustworthy Computing initiative (http://www.wired.com/news/business/0,1367,49826,00.html), in which he stated, "*Trustworthy Computing is the highest priority for all the work we are doing*." An increasing number of vendors are following suit, recognizing that software trustworthiness determines a software product's success or failure in the market ("Quality-Evaluation Models and Measurements," Jeff Tian, *IEEE Software*, May 2004, pp. 84-91). Vendors are right to be concerned, because the emphasis on trustworthy computing has changed not only the way developers create and deliver software, but also the way society views it.

Despite increased interest, trustworthy computing is still in its infancy, and many research issues remain open. The scope and challenge of trustworthy computing were prominent topics at the May 2004 Second National Software Summit. Indeed, NSS2 and other forums are urging a US R&D agenda on software trustworthiness. Although these two terms—trustworthy computing and software trustworthiness—appear different, in essence they are interchangeable: Certainly there is more to computing than software, but software is the ultimate deliverable and therefore has the most potential to influence the user's confidence level in the entire computing process.

Changing the Face of the Internet

Web services are transforming the Internet in three ways. First, they are making it easier to conduct business-to-business (B2B) e-commerce, since each organization makes its software services accessible through standard interfaces. Second, they provide a uniform framework to increase cross-language and cross-platform interoperability for distributed computing and resource sharing over the Internet. Third, they represent a cost-effective way to engineer software by quickly developing and deploying Web applications. The software dynamically integrates these applications with other independently published Web services components to conduct new business transactions. These three capabilities are powerfully shaping Internet computing for the future. Indeed, Gartner Group, a leading industry analyst firm, predicted that by 2008 more than 60 percent of businesses will adopt Web services and transform into new types of enterprises ("Composite Applications Head Toward the Mainstream," 16 Oct. 2003; http://www.gartner.com).

Figure 1. Current Web services framework with an additional trustworthiness layer.



At the framework's base is the combination of Simple Object Access Protocol (SOAP), an XML-based protocol that lets applications communicate over the Internet, Web Services Definition Language, which lets services recognize each other, and universal description, discovery, and integration (UDDI), a public registry that enables providers to publish their services on the Internet and discover each other. The next layer, WS-Security, consists of six models that focus on establishing secure services. The WS-Trustworthy layer will ensure the services' trustworthiness—availability, reliability, and so on—which goes beyond security.

(UDDI), where Web services providers publish their services. The system then chooses the optimal Web service that fulfills its requirements, binds to the service's Web site, and invokes the service. In this dynamic-invocation model, users might not even know which Web services they will use, much less how trustworthy those services are.

Malicious acts and code changes

Because Web service invocation is remote, there is always the chance that those hosting Web service components will act maliciously or erroneously and cause the service—even a fully tested one—to malfunction. Service providers might also change the service's underlying code, which could compromise its previous trustworthiness. The larger challenge then—one that goes beyond single malicious or errant acts—is how to ensure that a tested and trusted Web service maintains its trustworthiness.

Erratic Internet behaviors or resource scarcity

Because users can invoke Web services only over the Internet, unpredictable network traffic can significantly delay Web service delivery. If a Web service request has critical time or synchronization requirements, such as one for streaming video, sluggish delivery could render the service useless. Meanwhile, since a Web service is accessible to users worldwide, during holidays and other peak use times, an explosion of incoming requests could block the Web service host and make the Web service unavailable. Finally, the service's delivery medium—the Internet—could also come under attack, which could compromise the service as well.

Poor interoperability within the composed environment

Selected Web service components can act incorrectly in the composed environment. Testing must therefore include interoperability with other system components in the context of a specific environment. Again, remote access is at the heart of the challenge: Investigating how the entire system will react with every possible output from the Web service is a daunting problem.

CURRENT WEB SERVICES FRAMEWORK

At present, the Web services community is preoccupied with low-level mechanisms for implementing Web services—how to publish and compose a service, how to define the overall architecture of a Web-services-oriented system, how to facilitate the transportation of Web services, and so on.

Figure 1 shows the current Web services framework, including the WS-Security standard, a family of protocols that enhances messaging to solve three basic quality-of-protection problems for Web services: user authentication and authorization, message integrity, and message encryption. As the figure shows, WS-Security offers six models (from bottom to top) that help establish secure and interoperable Web services:

- *WS-Policy* provides a syntax-wired model to specify endpoint policies for Web services.
- *WS-Trust* defines methods to request and issue security tokens for establishing trust relationships.
- *WS-Privacy* describes a model for expressing privacy claims inside WS-Policy descriptions and for associating privacy claims with messages.
- *WS-Federation* defines mechanisms for identity, account, attribute, authentication, and authorization federation across trust realms.
- *WS-Secure Conversation* defines a security context that

is based on security tokens for secure communication.
- *WS-Authorization* defines how Web services manage authorization data and policies.

Unfortunately, WS-Security and related techniques and languages address only the *security* issue of Web services-centered computing. Trustworthiness is a holistic property that encompasses attributes beyond security, such as reliability, safety, survivability, interoperability, availability, fault tolerance, and performance ("Principled Assuredly Trustworthy Composable Architectures," Peter G. Neumann, http://www.csl.sri.com/users/neumann/chats4.pdf).

A NEW LAYER

As the top of Figure 1 shows, I propose adding a layer—WS-Trustworthy—containing the WS-Security layer but including much more. This layer would address the four critical challenges of Web services and provide solutions for three key issues: how to define trustworthy Web services and the criteria used to do so, how to measure and test Web services trustworthiness, and how to enhance and guarantee trustworthy Web services.

Criteria for trustworthiness

What is software trustworthiness in the domain of Web services? Generic software trustworthiness is a combination of software attributes: reliability (Re), security (Se), safety (Sa), maintainability (Ma), survivability (Su), availability (Av), testability (Te), interoperability (In), performance (Pe), fault tolerance (Ft), and so on. Thus, a Web service's trustworthiness would be a function of a specific set:

```
Trustworthiness = f(aRe, bSe, cSa,
dMa, eSu, fAv, gTe, hIn, iPe, jFt)
```

where $a, b, c, d, e, f, g, h, i$, and j are quantitative or qualitative measures.

The addition of measures captures the idea that each attribute can contribute differently to a service's trustworthiness in a specific context or scenario. The software testing community has provided precise definitions for these attributes, but in the Web services domain, these definitions might not be valid and so deserve reexamination. For example, what is "security" in the domain of Web services?

Measuring and testing trustworthiness

If software trustworthiness is a collection of measurable and testable attributes, then overall trustworthiness of Web services is measurable and testable. If, for example, a Web service scores high in every attribute (of course some will naturally conflict, such as Te and Ft), that service's trustworthiness is likely to be high as well. Thus, a feasible strategy is to investigate each attribute independently in the domain of Web services before exploring attributes together. Testing models and methodologies exist, but again, because of properties inherent in Web services—remote invocation and dynamic discovery and invocation—effective and efficient testing and measurement might require new techniques.

Some researchers have started to explore this novel field. Some work is focused on deciding testing criteria. Daniel Menascé, for example, proposes counting the total execution time and cost of the whole composite Web service when facing multiple Web service components ("Composing Web Services: A QoS View," *IEEE Internet Computing*, Nov. 2004, pp. 88-90). Jorge Cardoso and colleagues use simulation to validate Web services composi-

Resources

Conferences and forums

- **IEEE International Conference on Web Services; http://conferences.computer.org/icws.**
- **IEEE International Conference on Services Computing; http://conferences.computer.org/scc.**
- **IEEE Technical Steering Committee for Services Computing; http://tab.computer.org/tcsc.**
- **Second National Software Summit; http://www.cnsoftware.org/nss2.**
- **Center for National Software Studies; http://www.cnsoftware.org.**

Journals

- ***International Journal of Web Services Research*; http://www.ideagroup.com/journals/details.asp?id=4138.**
- ***International Journal of Business Process Integration and Management*; https://www.inderscience.com/browse/index.php?journalID=115.**
- ***International Journal of Grid and Utility Computing*; https://www.inderscience.com/browse/index.php?journalID=108.**

Additional reading

- **"Trustworthy Computing," Craig Mundie and colleagues; http://www.microsoft.com/mscorp/innovation/twc/twc_whitepaper.asp.**
- **"Understanding Service-Oriented Software," Nicolas Gold and colleagues, *IEEE Software*, Mar.-Apr. 2004, pp. 71-77.**
- **"Are Web Services Finally Ready to Deliver?" Neal Leavitt, *Computer*, Nov. 2004, pp. 14-18.**
- **"Web Services: Been There, Done That?" Steffen Staab and colleagues, *IEEE Intelligent Systems*, Jan.-Feb. 2003, pp. 72-85.**

tion using a mathematical quality of service (QoS) model that emphasizes timeliness, cost of service, and reliability ("Modeling Quality of Service for Workflows and Web Service Processes," *J. Web Semantics*, Apr. 2004, pp. 281-308). Other research aims at test-case generation. Jeff Offutt and Wuzhi Xu propose to adopt data perturbation technique to generate test cases of testing message communications between pairs of Web services ("Generating Test Cases for Web Services Using Data Perturbation," *ACM SIGSOFT Software Eng. Notes*, Sept. 2004, pp. 1-10). I am also researching this area with the aim of developing testing tools that use mobile agents to select reliable Web service components cost-effectively ("An Approach to Facilitate Reliability Testing of Web Services Components," *Proc. IEEE Int'l Symp. Software Reliability Eng.* (ISSRE 04), IEEE CS Press, 2004, pp. 210-218).

Enhancing and guaranteeing trustworthiness

The WS-Trustworthy layer should provide a set of principles and practices that underlie trustworthy Web services. It should also help software engineers improve the trustworthiness of their Web services processes by providing an evolutionary path—as opposed to ad hoc techniques—to a mature and disciplined process. To produce trustworthy Web services, software engineers need direction through notations, guidelines, and methodologies that cover the entire Web service life cycle—publication, discovery, composition, invocation, and upgrade. Supportive toolkits, integrated development environments, and successful case studies fall into this category.

THINKING AHEAD

When Web services become mainstream, which could be soon, trustworthiness will become the bottleneck to their extensive adoption. A set of trustworthiness criteria and guidelines will provide an open and standard infrastructure for ensuring trustworthiness in this domain. Researchers must then set to work devising a technical strategy and roadmap, coupled with a standards-based architecture that is comprehensive yet flexible enough to meet the Web services trustworthiness needs of real business. There is serious work ahead, but the results will be far more rewarding if researchers take the first step now: Standardize on a precise and comprehensive definition of Web services trustworthiness. The rest of the tasks will follow logically from that. ■

Jia Zhang *is an assistant professor in the computer science department at Northern Illinois University and a guest researcher for the National Institute of Standards and Technology. Contact her at jiazhang@cs.niu.edu.*

We thank the anonymous reviewers for their insightful comments.

Join the IEEE Computer Society online at

computer.org/join/

Complete the online application and

- Take Web-based training courses in technical areas for free
- Receive substantial discounts for our software development professional certification program
- Get immediate online access to *Computer*
- Subscribe to our new publication, *IEEE Security & Privacy,* or any of our 22 periodicals at discounted rates
- Attend leading conferences at member prices
- Sign up for a free e-mail alias—*you*@computer.org

THE WORLD'S COMPUTER SOCIETY