

Web Services-Based Collaborative and Cooperative Computing

W. T. Tsai, Z. Cao, Y. Chen, R. Paul*,

*Department of Computer Science and Engineering, Arizona State University
Tempe, AZ 85287-8809, USA*

**Department of Defense, Washington DC*

Abstract

This paper presents an integrated development process for Web services. The key differences with the traditional software development are that this new process involves collaboration and cooperation among all parties involved: developers, brokers, and clients. The process defines a new way of developing trustworthy Web services based on the existing internet infrastructure. The paper serves as a roadmap to research on Web services specification, discovery, ontology, composition, re-composition, testing, reliability assessing, ranking, and collaboration and cooperation among all parties involved in Web services research, development, and application.

Keywords: Web services, service composition, testing, verification, reliability modeling.

1. Introduction

Web Service is an emerging technology that is changing the way computer software is designed and used. Many industrial standards have been defined in the past a few years to facilitate and to regulate the development of Web services. However, there are still a number of barricades preventing Web services from being widely applied or being used as the platform for the trustworthy and high-assurance systems. Brent Sleeper identified five missing pieces of Web service technology: reliability, security, orchestration, legacy support, and semantics [2]. Among these five issues, reliability is the least addressed and probably most difficult one, for the following reasons:

- Web services are based on unreliable and open Internet infrastructure, yet they are expected to be trustworthy.
- Web services have a loosely coupled architecture, yet they are expected to collaborate closely and seamlessly.

- Web services can be invoked by unknown parties with unpredictable requests, and thus Web services must be robust.
- Web services involve runtime discovery, dynamic binding with multi-parties including middleware and other Web services, and runtime composition using existing Web services. Thus, Web services must support dynamic and runtime behaviors.
- Web services must support dynamic configuration and reconfiguration to facilitate fault-tolerant computing in a relatively unreliable internet environment
- Web services must support dynamic composition and re-composition to cope with the changing environment and changing requirements.
- Web services involve concurrent threads and object sharing. It is difficult to test concurrent processes due to their non-deterministic behaviors..

We propose an integrated collaborative and cooperative Web service development process to achieve highly dependable or trustworthy computing. The process is implemented in a framework consisting of three major modules dealing with the construction, publishing, and testing of Web services. Section 2 gives an overview of this development framework. Sections 3, 4, and 5 elaborate the three modules, respectively. Finally, section 5 summarizes and concludes the paper.

2. Overview of the development process

As shown in Figure 1, the Web service Cooperative and Collaborative Computing (WSC3) framework consists of three modules: cooperative Web service construction; publishing; and testing, assessment, and ranking (WebStrar). The framework can be used by service requestors, service providers, as well as researchers experimenting Web services.

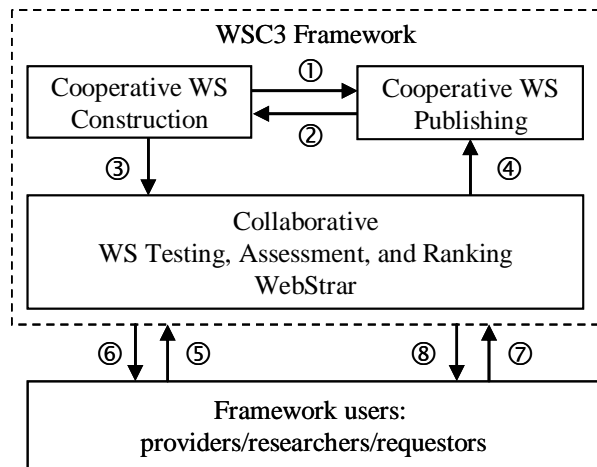


Figure 1. WSC3 model overview

As examples, the arrows and numbers in the figure outline scenarios of the cooperation between the components and are explained as follow.

1. The Web service construction module, in a process of dynamically constructing a composite Web service based on existing Web services, requests information from the Web service publishing module.
2. The publishing module provides required information, including the specification, interface of using the Web services.
3. After a composite Web service is constructed, the construction module submits the Web service to the WebStar module for rigorous testing.
4. If the Web service passes the test, it will be registered to the publishing module and a new Web service is available for online accessing.
5. A Web service provider or a researcher submits its Web service for publication or testing. The Web services will be tested by the WebStar module rigorously based on the test scripts submitted by the provider as well as the test scripts generated by WebStar. Sharing the test scripts represents the collaboration between the framework and Web service providers and researchers.
6. The framework publishes the Web service and informs the Web service provider if the submitted Web service passes the test.
7. A Web service requestor requests a service. The requestor can request testing before using a Web service. It can use the test scripts provided by the framework or submit his/her own test scripts. Sharing the test scripts represents the collaboration between the framework and Web service requestors. Web service requestors can also accesses the

reliability data and ranking information of published Web service.

8. The framework process and responses to the Web service requestor.

In the following three sections, we will elaborate the three modules in the WSC3 framework, respectively.

3. Cooperative web service construction

Figure 2 elaborates the cooperative Web service construction module in Figure 1. This module has six components.

The cooperative Web service Specification component provides guidelines and tools for the users to write Web service specifications in a specification language, e.g., in OWL-S. The component will then use WebStar to perform C&C (Consistency and Completeness) check on the specification.

Once the specification passes the check, the code generation component can automatically generate the executable code. The Web service generated in this way is atomic, because its implementation detail is not accessible for the users. All Web services submitted by Web service providers are atomic too.

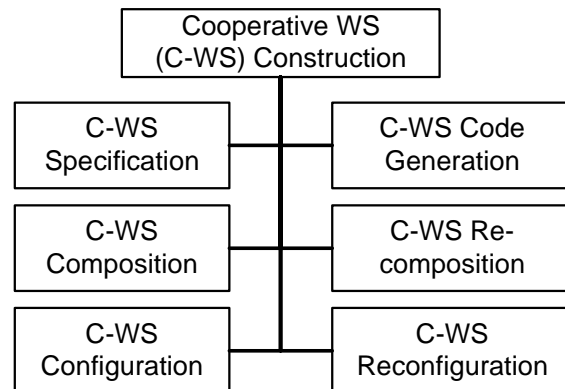


Figure 2. Cooperative Web service Construction

The cooperative Web service composition component provides automated high-level Web service composition based on existing atomic Web services and their specifications.

The re-composition component can reconstruct a composite Web service if the requirement and specification are changed.

Composition and re-composition components construct Web services based on the functional requirement while the configuration and reconfiguration components deal with the management of redundant resources and the reliability of Web

services. The configuration component adds redundant structure into composite Web services to meet the reliability requirements and reconfiguration maintains the redundancy after the environment is changed, for example, if some Web services become faulty or unavailable.

4. Cooperative web service publishing and ontology

This section elaborates the cooperative Web service publishing module in Figure 1. Current Web service publishing is based on the Universal Description, Discovery, and Integration (UDDI) technique. The UDDI discovery part is based on simple term/text matching, which does not have the intelligent to find synonyms and semantically related terms. For example, if the phrase "red wine" is searched, the terms like Cabernet Sauvignon and Merlot should be found too.

OWL-S and Protégé are recent projects that support the ontology description. Ontology defines the basic terms and relations comprising the vocabulary of a topic area as well as the rules for combining terms and relations to define extensions to the vocabulary [3]. It represents knowledge about a domain and describes specific situations in the domain. The dynamic composition and re-composition need to discover Web services at runtime over the internet, add them to the ontology domain, and then compose a Web service at runtime.

Current ontology methods do not include the verification process. In our cooperative Web service Publishing module, we integrated the collaborative verification and validation (CV&V) process into the ontology by including the necessary test scripts in the ontology domain. When a Web service is chosen for composition, re-composition, configuration, or reconfiguration, the stored test scripts will be immediately applied to test the Web service. This integrated ontology with CV&V is called Cooperative Ontology. Table 1 compares and contrasts the cooperative ontology and traditional ontology.

The ontology-based architecture plays a key role in runtime Web service composition. Runtime verification can choose different level of test scripts to verify the found services. To further explain the idea, a simple ATM example is used here. Assume the ATM offers login, balance-checking, withdrawal, deposit, and logout services.

The service tree of the cooperative ontology representing the ATM composite Web service is given in [5]. Each node has the services interface definition,

services constraints, service scenarios, and user-specific requirement by using test scripts. For instance, if we want to choose the login service that supports a specific character set "&*^%" in the user name, the user-defined test scripts can be:

Execute Register("abc&^%123", "123456");*

If this test script failed, the found services does not conform the requirement and will be rejected. As discussed before, there are multiple levels of test scripts. This test script includes the interface test scripts. In the service tree specification, the internal relations among test scripts are also important to support dynamic service composition and re-composition.

Table 1. Cooperative vs. traditional ontology

	Traditional Ontology	Cooperative Ontology
Test Scripts	Do not include test scripts.	Include test scripts and execute these test scripts at runtime.
Non-functional Property	Use certain terms to present the value of these non-functional properties, such as performance and security.	Use test scripts to present the non-functional properties. Translate the non-functional properties to the measurable features.
Behavior constraints	Allow to specify the constraints, but can not be executed at runtime.	Execute the constraints at runtime to check if the assigned services match the constraints.
Interface	Do not include the interface information in the description.	Use specific test scripts to present the interface constraints.

5. Collaborative testing and evaluation

Web services composed using the process in Figure 1 will be tested, assessed and ranked. Figure 4 depicts the module that tests and assesses the reliability of Web services and assures the tools involved. The solid arrows indicate that a component can be decomposed into several sub components, while the dotted arrows indicate the data flow between components. WebStrar itself is a framework supporting the development of trustworthy Web services (<http://asusrl.eas.asu.edu/srlab/projects/webstrar/index.htm>). This section explains the individual techniques developed and to be developed in this framework. At the top level,

WebStrar consists of three components: CV&V, Web service reliability assessment, and Ranking.

The idea of CV&V is to involve all parties (Web service providers, brokers, and clients) in verifying and validating Web services, because the Web service provider does not have the full information how their Web services will be used by clients and brokers in user-composed composite Web services [5][6]. Before test script generation, the consistency and completeness of the Web service specification in OWL-S or WSDL will be checked through model checking or other methods, which may detect inconsistent conditions or in complete coverage of the requirements [8]. Once the specification passes the check, Boolean expression can be extracted and the expressions then can be used for test script generation. Different techniques can be applied here. We have applied Swiss Cheese [8] and BLAST [1] techniques in our experiments. The system generated test scripts, along with the test scripts provided by other parties will be verified for their correctness and ranked on the effectiveness in detecting faults in Web services.

Group testing is a key technique developed to test potentially large number of Web services available on Internet [6]. Web services with the same specification

can be tested in group and the results are compared by a discriminant voter, which can identify correct and faulty output based on the majority principle. The majority results are used as the oracles in future testing.

Reliability assessment of Web services is different from that of traditional software. Web service reliability models do not have the access to Web service source code. Web service reliability can only be assessed at runtime because Web services can be composed and modified (re-composed) at runtime. A group testing based dynamic reliability model has been developed to assess the Web service reliability [7]. Reliability is one of the criteria used to rank Web services. Other criteria include security, performance, real-time ability, etc.

WebStrar is an open platform and has a service-oriented architecture itself and thus allows researchers and Web service providers to submit their models and tools for evaluation and ranking. WebStrar supports ranking of test scripts in terms of fault detection capacity, test script generation algorithms in terms of generating effective test scripts, reliability models in terms of the accuracy of their assessment results, and ranking models themselves in terms of ranking accuracy.

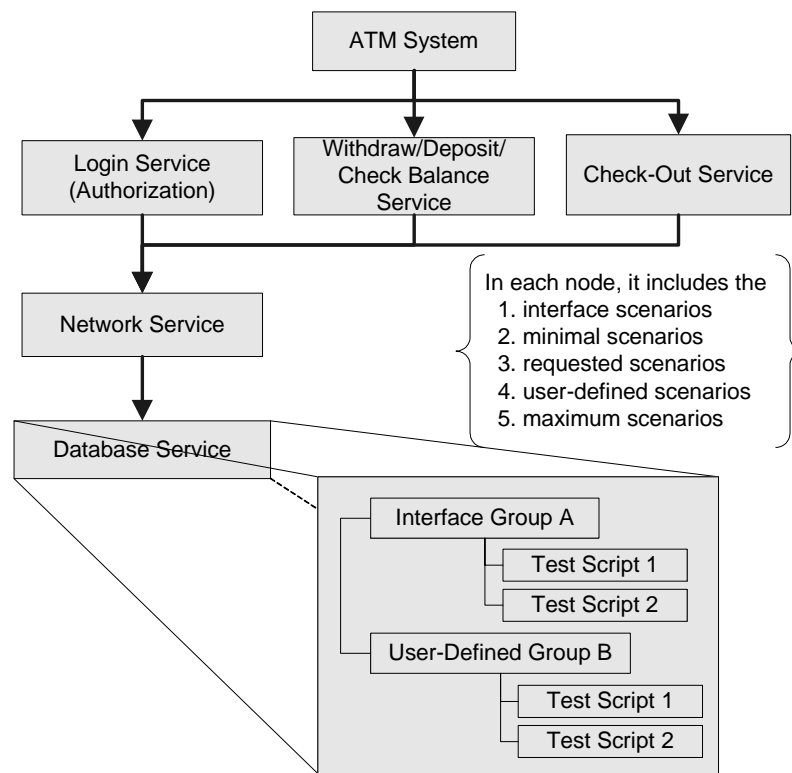


Figure 3. ATM service tree example

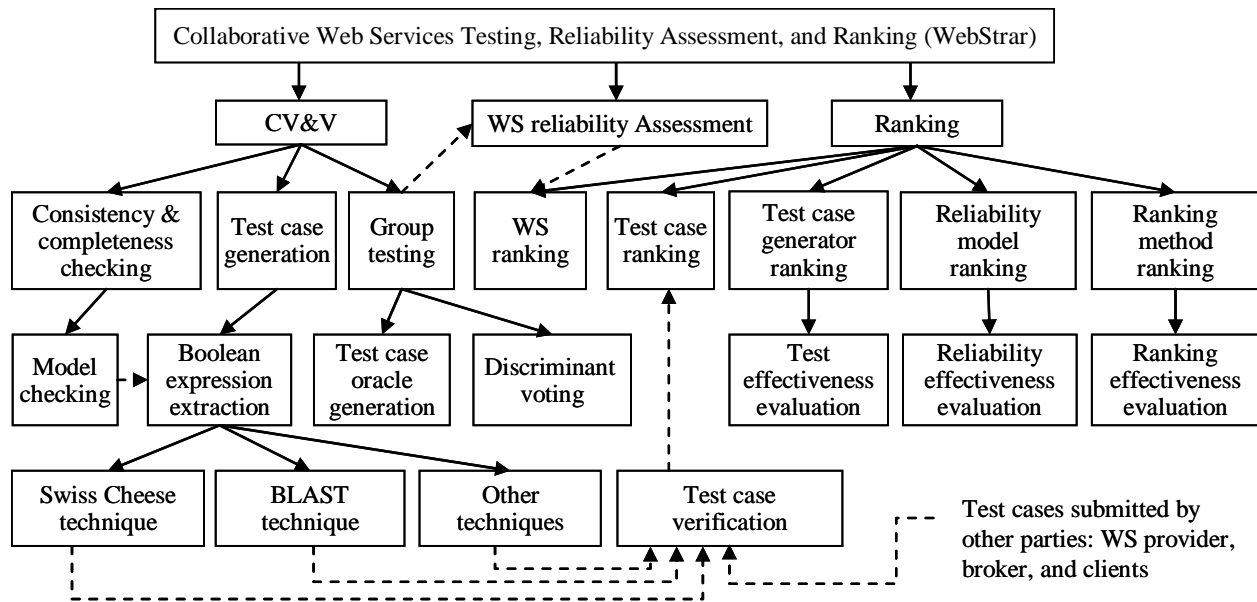


Figure 4. Collaborative testing, assessing, and ranking

6. Summary

It is more difficult to develop highly dependable Web services because of the unavailability of source code and the dynamic nature. Thus collaborative and cooperative computing concept is applied to improve the trustworthiness of Web services. This paper drew a big picture of the process and the individual techniques supporting this process. Several key techniques have been implemented but much research is still needed to achieve the ultimate goal of moving product-oriented software architecture to service-oriented architecture. We urge Web service researchers, providers, brokers, and clients to work together to achieve the goal.

References

- [1] D. Beyer, A. Chlipala, T. Henzinger, R. Jhala, and R. Majumdar, "Generating Tests from Counterexamples", Proceedings of the 26th International Conference on Software Engineering (ICSE'04), Scotland, UK, May 2004, pp. 326 – 335
- [2] B. Sleeper, "The five missing pieces of SOA", www.infoworld.com/article/04/09/10/37FEwebservmiddle_1.html, September 2004.
- [3] R. Neches, R. Fikes, T. Finin, T. Gruber, R. Patil, T. Senator, and W. R. Swartout, "Enabling Technology For Knowledge Sharing", AI Magazine, Volume 12, No. 3, Fall, 1991
- [4] R. Fikes, A. Farquhar, "Distributed Repositories of Highly Expressive Reusable Ontologies", IEEE Intelligent Systems, 14(2): 74-79, 1999
- [5] W. T. Tsai, R. Paul, Z. Cao, L. Yu, A. Saimi, and B. Xiao, "Verification of Web Services Using an Enhanced UDDI Server", Proc. of IEEE WORDS, 2003, pp. 131-138
- [6] W. T. Tsai, Y. Chen, R. Paul N. Liao, and H. Huang, "Cooperative and Group Testing in Verification of Dynamic Composite Web Services", in Workshop on Quality Assurance and Testing of Web-Based Applications, in conjunction with COMPSAC, September 2004, pp.170-173.
- [7] W. T. Tsai, D. Zhang, Y. Chen, H. Huang, R. Paul*, N. Liao, "A software reliability model for Web services", The 8th IASTED International Conference on Software Engineering and Applications, Cambridge, MA, September 2004, pp. 144-149
- [8] W. T. Tsai, X. Wei, Y. Chen, B. Xiao, R. Paul, and H. Huang, "Developing and Assuring Trustworthy Web Services", submitted to The 7th International Symposium on Autonomous Decentralized Systems (ISADS), April 2005