Will Reliability Kill the Web Service Composition?

Department of Computer Science
Rutgers University
Piscataway, NJ 08554, USA

Abstract:

Web service composition is a distributed model to construct new web service on top of existing primitive or other composite web services. The advantage of web service composition has already been demonstrated and highlighted for its capability to compose autonomous services to achieve new functionality. However, an important question is whether this distributed model will be realized or not, due to the reliability problem of composite web services.

We take the position that the trend of web service composition will not be terminated because of the reliability problem. We defend the position by arguing that the reliability of both constituent web services and web service composition itself would be greatly improved due to the emergence of research providing cost-effective solutions. More interestingly, we show that web service composition could even achieve much higher reliability by leveraging the redundancy intrinsic to the Web.

As the conclusion, we believe that the high reliability can be achieved by the support of existing solutions and ongoing research efforts. And the reliability problem will forward rather than reverse the growth of composite web services. More and more autonomous web service would be composed to provide different services to address the wide range of demands.

1. Introduction

Increasing numbers of interesting services are moving online and the Web is fast transforming from a collection of static pages to a provider of numerous useful services. Though those autonomous web services are provided and operated independently, some of them are often related to each other in different ways and semantics. Thus, there is an inherent need for composing those web services provided by independent providers to achieve the new functionality that meets the different needs of end-users.

On the other hand, with the standard protocols like SOAP [1], WSDL [2], and UDDI [3], Web services are turning the Web into a huge programmatic backbone for various applications. Composing new web service on top of existing primitive or composite web services are becoming an efficient way to provide new web services.

Web service composition can take place among big commercial organizations, or across cooperative academic institutes, or even in P2P networks in which ordinary user could deploy and publish their web services from their own machines.

In the past few years, though the advantage of web service composition has already been demonstrated and highlighted for its capability to compose autonomous services to achieve new functionality [6], it is still an open question whether this distributed development model for web services will become reality or not, due to the challenges posed by the reliability of composite web services.

Will reliability kill the web service composition?

In this paper, we take the position that the trend of web service composition will not be terminated because of the reliability problem. We defense our position by arguing that the reliability of both constituent web services and web service composition itself would be greatly improved due to the emergence of research providing cost-effective solutions. More interestingly, we show that web service composition could even achieve much higher reliability by leveraging the redundancy intrinsic to the Web.

The rest of the paper is organized as follow. In Section 2, we introduce the web services and composition on top of them. Section 3 states reliability problem in the composite web services, and articulates our position in favor of the web service composition. Section 4 discusses the cost-effective solutions to achieve higher reliability for web service composition. Furthermore, Section 5 shows that web service composition can also improve the reliability by leveraging the redundancy in web services. Section 6 summarizes our conclusions.

2. Web Services Composition

A web service is a software application identified by a URI, whose interfaces and bindings are capable of being defined, described, and discovered as XML artifacts. It supports direct interactions with other software agents using XML based messages exchanged via Internet-based protocols [7].

Web service composition is a distributed development model to construct new web service on top of existing web services. We name resulted web service as composite service, and constituent web service for ones to be composed. A composite service could also be the constituent service for another composite service

Composition may take place within an organization. For example, in a big company, each department may have a set of web services providing its basic functionalities, such as sales, accounting, and warehouse management. Those web services may then be integrated into much more complex business process by web service composition.

More interestingly, the individual web services can be composed across organization boundaries. Take a travel agency for example, it could provide a composite travel service which is composed from the external web services that provide flight booking, car renting, room booking, etc.

3. Reliability of Composite Web Services

As pointed out in [10], failures are inevitable in the modern Internet-Connected environments due to a fundamental mismatch between traditional high-reliability approaches(fault-tolerant hardware, careful software testing, vendor-supplied technicians)and the realities of current heterogeneous, distributed server environments that provide web services.

Assuming the failure of any individual web service will cause the failure of the composite service, the overall reliability of composite service is the product of the reliability of constituent web services. Even all the other web services are reliable, one unreliable web service could decrease the overall reliability to a very low level. The upper bound of overall reliability is often determined by the weakest constituent web services.

Also, since operations of a composite service often span multiple individual web services, web service composition itself also brings unreliability with the presence of the failures. The heterogeneous, distributed, dynamic nature of the web service composition also gives rises to many more unpredictable failures, compared with centralized web service.

Consequently, composing the web service on top of web services is often viewed as the task destined for unreliability. With such great unreliability with the web service composition, a fundamental question is raised: will reliability kill the future of the web service composition?

Our answer is no. The confidence comes from two arguments. Firstly, we believe that the reliability of both constituent web services and web service composition would be improved greatly due to the emergence of research providing cost-effective solutions. More interestingly, we believe that web service composition can even help to achieve much higher reliability by leveraging the redundancy intrinsic to the Web.

4. Higher Reliability to Web Service Composition

Our discussion on achieving higher reliability for composite web services starts with revisiting the reliability for the individual web service. We then examine the reliability issues in the web service composition itself.

4.1 Reliability of individual web service

As we mentioned in Section 3, the very different computing environment and fast evolution of the web services make traditional fault-tolerant solutions not effective.

However, we also found that more and more research efforts are made in achieving high reliability in the web services.

There are two main groups among those researches. One group tries to extend the traditional solutions to address the new challenges in the web services. The guiding principle for this group is to prevent or avoid the occurrence of the failures, i.e. to reduce the Mean Time to Fail (MTTF). Those solutions often rely on very expensive and specialized systems.

Instead of putting huge amount of resource and efforts to avoid all possible failures, the other group focuses on how to achieve fast repair or recovery once failures occur. This group tries to reduce the Mean Time to Recover (MTTR) on the commodity systems composed of off-the-shelf hardware and software components.

[11] proposed the recursive micro-reboot techniques to reduce the mean time to recovery (MTTR). The key idea of micro-reboot is only to reboot the faulty component instead of the whole server. Such a fine-grained reboot technique constrains the negative effect of recovery to a very small scope, without affecting other non-relevant components.

[12] proposed a novel recovery mechanism that gives human system operators the power of system-wide undo. As we mentioned in Section 3, human errors are one of the major source of the unreliability. In particular, human operators could easily damage a system by misconfiguration, incorrect updates, etc. System-wide undo allows operators to roll back erroneous changes to a service's state without losing end-user data or updates, to make retroactive repairs in the historical timeline of the service system, thereby to quickly recover from catastrophic state corruption, operator error, failed upgrades, and external attacks, even when the root cause of the catastrophe is unknown.

With more and more cost-effective technical solutions becoming available, we believe that individual web services will become more and more reliable. In this process, the nature of pursuit of the profit will be a driving force for the service providers to adopt those techniques. The basic motivation to provide web service is to attract all potential consumers and make profit from them. Once they expect the profit gained by improving the reliability would bypass the cost to take the measure, they will not hesitate to use those techniques.

Naturally, with the higher reliability achieved by constituent web services individually, higher composite reliability could be obtained.

4.2 Reliability within the web service composition

As we mentioned in the Section 3, leaving aside the unreliability of individual web services, the unreliability introduced by the web service composition itself is the other major concern of the overall reliability of composed web services.

The root cause of this unreliability is the fact that the operations of a composite service often span multiple individual web services. To make composition reliable, it often

requires atomicity of processing of a set of Web Services requests or the consistency of data transformations applied to set of individual Web Services, in the presence of the failures.

We found out that there are no standards to keep the consistency of failed transaction across multiple constituent services. WSDL and SOAP specifications did not address the reliability of web service composition. Even for proposed composition languages like WSFL and XLANG there is no reliability specification.

Fortunately, many researches are working on this problem way and several feasible solutions have been already proposed. We believe these solutions could make web service composition itself much more reliable.

[4] outlined ideas for a model of web transactions. The model is based on the OMG/J2EE Activity Service specification, an advanced object transaction service. Overall, they aimed at exploring a practical solution to web transactions that is compliant to web services standards. In their proposed model, the Activity Service is extended to support the communication models available to web services. This will enable flexible "web transactions" that span multiple web services and the objects that implement those services.

[5] showed how explicit transactional attitude descriptions could be used to automate the reliable composition of applications into larger web transactions, while maintaining autonomy of the individual applications. Provider transactional attitudes (PTAs) are a mechanism for Web service providers to explicitly describe their specific transactional behavior. By making transactional semantics explicit, PTAs can be used in automating the composition of individual transactional Web services into larger transactional patterns, while maintaining the autonomy of the individual services.

5. Higher Reliability via Web Service Composition

While the previous section discuss how to make web service composition reliable, this section will show that web service composition also holds the key to achieve higher reliability before needed reliability for all constituent web services are delivered by service providers individually. We found out that redundancy intrinsic to the web service could greatly improve the reliability even the reliability of individual web service is still relatively low. Web service composition is the right model to leverage this redundancy. From this perspective, web service composition is actually one way to achieve higher reliability.

5.1 Redundancy in the web services

Looking at reliable systems built on top of relatively unreliable subsystems, we often find out that the redundancy is a key element for the overall reliability. For example, Google File System (GFS)[9] is built from inexpensive commodity components that often fail. Disks, memories, machines are not trusted and their crashes are even deemed as norm

instead of exception. However, the overall reliability is achieved by the use of thousands of machines. For the millions end-users, the reliability of Google search service is amazingly high.

As another example, for applications in P2P networks, like the multimedia files download, there is no guarantee of the reliability of any participating peers: machines could be online or offline any time. However, the overall reliability is achieved by the great redundancy of the equivalent functionality. Most of end users of P2P applications are pretty satisfied with the reliability they provide.

Redundancy could also improve the overall reliability of composite web service a lot. Here, by redundancy we mean that the same kind of functionality is available in the web service provided by different service providers. For example, suppose a travel agency provides a composite web service composed of 3 types of web services from other company: flight booking service, room reservation service, and car renting service. Because lots of redundant web service existed for each type of constituent web services, the travel agency could still provide the same functionality even when some of them are not available due to failures or other reasons.

On the other hand, we observed that redundancy is a fundamental feature of the Web as well as web services. Redundancy is the natural product of growth of the Web itself. For any kind of web service, if this is profitable or meaningful to the service providers, it will definitely enjoy a lot redundancy as numerous individual service providers are trying to take advantage of it. The temptation that anyone who has the access to the Web could be potential consumer for their services is extremely hard to resist for them.

Due to individual pursuits for the potential profit, the redundancy of web service often also enjoys much higher quality than the one we usually find in tight-coupled systems. For the latter, redundancy is often achieved by the simple accumulation of the same kind of instances.

On the contrary, the redundancy of web service often has much higher diversity. Web services delivered by different service providers are often developed individually. The diversity makes it much less likely that the same failure would hit all redundant web services. On the contrary, if two web services are implemented exactly the same, then the failure caused by the same software failure would not be eliminated by redundancy.

5.2 Discover the Redundancy

The Universal Description, Discovery and Integration (UDDI) specifications define a way to publish and discover information about Web services. With the UDDI, we could discover the redundant web services for the same type of functionality.

In addition to UDDI, Web Service Discovery Architecture (WSDA) is proposed by [8]. WSDA promotes an interoperable web service discovery layer by defining appropriate

services, interfaces, operations and protocol bindings, based on industry standards. It defines a small set of orthogonal multipurpose communication primitives for discovery. These primitives cover service identification, service description retrieval, data publication as well as minimal and powerful query support. The architecture is open and flexible because each primitive can be used, implemented, customized and extended in many ways.

5.3 Leveraging the redundancy

With the support of discovering redundant web services, web service composition is the right model to leverage the redundancy because its key capability is to build the composite web services over existing web services. When failures occur in any constituent web service, a composite web service can discover redundant web services, and replace the failed ones. The current research in self-management system like Autonomic Computing [13] can also be very helpful to make the design of the web service composition more resilient to the failures of web services.

6. Conclusion

After the investigation of the different aspects of the reliability of composite web services, we conclude that high reliability could be achieved by the support of existing solutions and ongoing research efforts. We also believe that web service composition could help to solve the reliability problem rather than destined to be unreliable.

Web service composition will be the popular mode to develop new web services. We predict that more and more autonomous web service will be composed to provide different services to address the wide range of demands.

7. Reference

[1]SOAP Version 1.2 Specification Assertions and Test Collection W3C Recommendation. June 2003 http://www.w3.org/TR/2003/REC-soap12-testcollection-20030624/

[2]Erik Christensen, Francisco Curbera, Greg Meredith,Sanjiva Weerawarana. Web Services Description Language (WSDL) 1.1, March 2001. http://www.w3.org/TR/2001/NOTE-wsdl-20010315

[3]UDDI Version 3 Specification. http://www.oasis-open.org/committees/uddi-spec/doc/tcspecs.htm#uddiv3

[4]Thomas Mikalsen, Isabelle Rouvellou, and Stefan Tai. Reliability of Composed Web Services--From Object Transactions to Web Transaction. In Workshop on Object-Oriented Web Services, OOPSLA 2001, October 15, 2001, Tampa, Florida.

[5]Thomas Mikalsen, Stefan Tai, Isabelle Rouvellou. Transactional Attitudes: Reliable Composition of Autonomous Web Services. In workshop on Dependable Middleware-based Systems (WDMS 2002), part of the International Conference on Dependable Systems and Networks (DSN 2002), Washington D.C., June 2002.

[6]B. Benatallah and F. Casati (Guest Editors). Special Issue on Web Services. Distributed and Parallel Databases, Kluwer Academic Publishers, 12(2-3), September 2002

[7]Donald F. Ferguson, Tony Storey, Brad Lovering, John Shewchuk. Secure, Reliable, Transacted Web Services: Architecture and Composition. September 2003, IBM Corporation

[8]Wolfgang Hoschek. The Web Service Discovery Architecture. In Proceedings of the 2002 ACM/IEEE conference on Supercomputing, 2002, Baltimore, Maryland.

[9]Sanjay Ghemawat, Howard Gobioff, Shun-Tak Leung. The Google file system. In Proceedings of the nineteenth ACM symposium on Operating systems principles, 2003, Bolton Landing, NY.

[10]Aaron B. Brown and David A. Patterson. Embracing Failure:A Case for Recovery-Oriented Computing. In High Performance Transacttion Systems Workshop, 2001.

[11]George Candea and Armando Fox. Recursive Restartability: Turning the Reboot Sledgehammer into a Scalpel. Proc. 8th Workshop on Hot Topics in Operating Systems (HotOS-VIII), Schloss Elmau, Germany, May 2001

[12]Brown, A. and D. A. Patterson. Undo for Operators: Building an Undoable E-mail Store. In Proceedings of the 2003 USENIX Annual Technical Conference, San Antonio, TX, June 2003 (Best Paper Award)

[13]The Vision of Autonomic Computing, Jeffrey Kephart, David M.Chess. http://www-3.ibm.com/autonomic/pdfs/AC_Vision_Computer_Jan_2003.pdf