# On the Management of Compositions of Web Services

**Vladimir Tosic, Bernard Pagurek, Babak Esfandiari, Kruti Patel**
Network Management and Artificial Intelligence Lab
Department of Systems and Computer Engineering, Carleton University, Ottawa, Ontario, Canada
{vladimir, bernie, babak, kpatel}@sce.carleton.ca

**ABSTRACT**
In this position paper, we present our work on Web services with multiple classes of service, called service offerings, and on management and dynamic (i.e., runtime) adaptation of their compositions. We explain the motivation for Web services with multiple service offerings, present some management and dynamic adaptation algorithms based on the manipulation of provided service offerings, and discuss some of the issues with the corresponding management infrastructure, called DAMSC (Dynamically Adaptable and Manageable Service Components), that we are developing. We also briefly present our work on WSOL (Web Service Offerings Language) – an extension of WSDL (Web Services Description Language) that enables specification of various types of constraints—including functional, non-functional (QoS – Quality of Service), authorization policies, etc.—and specification of Web services with multiple service offerings. At the end, we summarize some of the challenges for future research in the area of management and dynamic adaptation of compositions of Web services.

**Keywords**
Management of compositions of Web services, management of Web services, dynamic adaptation, Web services, classes of service, service offerings, WSOL.

## 1 INTRODUCTION

Our research group has extensive experience in the area of management of computer and communication networks, distributed systems, and services. In the past, our research focussed on challenges particular to the domain of network management and exploration of application of advanced technologies for network management. We have extensive experience with network management standards like SNMP (Simple Network Management Protocol) and CMIP (Common Management Information Protocol), as well as the CIM (Common Information Model) standard for distributed system management. We investigated the use of mobile agent technologies and artificial intelligence techniques to enhance how a network is controlled and managed. In addition, we studied the issues how to extend SNMP network management solutions to allow monitoring and control of mobile agents visiting the network and how to integrate mobile agents and SNMP network management solutions. We have also explored some other management technologies, e.g., policy-based management and web-based management. In recent years, our focus has shifted more towards service-level management and to researching some software management issues in the context of the broader management of distributed systems, networks, applications, and services. Consequently, some of our recent research projects investigate issues that related to the management of Web services and their compositions.

The project on dynamic service composition [7] explored three different techniques for dynamic (i.e., runtime) creation of composite services from service components. The three techniques are: 1) creation of a composite service interface; 2) creation of a new composite service based on the pipe-and-filter architecture; and 3) creation of a new composite service by extracting and weaving code of composed service components. We designed and implemented a general-purpose dynamic service composition architecture called the Infrastructure for Composability At Runtime of Internet Services (ICARIS). Jini, JavaBeans, and XML were used for the prototype implementation of the ICARIS architecture. To illustrate a justifiable use of dynamic service composition techniques, this implementation of ICARIS was successfully used for dynamic construction and deployment of security associations between a client and a server in a network. As Web services are a special case of our concept of service components, the experiences and insights from this project are useful for researching dynamic composition of Web services.

The project on software hot-swapping [4] is ivestigating the issue of dynamic evolution of software without disrupting its operation. As noted in [9], a number of researchers tried to address this problem taking very different approaches. Our approach is based on the concept of swappable modules (S-modules) and corresponding non-swappable proxies (S-proxies). We have developed our own hot-swapping infrastructure and successfully applied it to a modular SNMPv3 (SNMP, version 3) network management system implemented in Java. The experiences from this project are relevant for dynamic evolution of Web services without disrupting operation of their compositions.

To enable a composition of Web services to be more flexible and adaptable to various changes and disturbances that can occur, we are also exploring Web services with multiple classes of service and management algorithms based on the manipulation of provided classes of service. We are also developing a corresponding management infrastructure and a specification language for Web services with multiple classes of service. Pay-per-use e- and m-business (i.e., elec-

tronic and mobile business) Web services are one of the main motivating examples for this project that will be described in more detail in this paper.

Several of our other ongoing research projects are also related to the management of Web services and their compositions. For example, one project studies management of Internet telephony service compositions based on the AT&T's DFC (Distributed Feature Composition) component-based architecture for telecommunication services. Also, some members of our research group are working on security management of peer-to-peer (P2P) architectures.

After the description of our work on Web services with multiple classes of service and dynamic management of their compositions in the next two sections, we will summarize some issues related to the management of compositions of Web services in the last section.

## 2 WEB SERVICES WITH MULTIPLE CLASSES OF SERVICE AND THEIR SPECIFICATION

A Web service is a unit of business, application, or system functionality that can be accessed over a network by using XML messaging. It can provide not only software functionality and data, but also access to some hardware resources like memory, printing, network bandwidth, etc. The motivation for the W3C's (World Wide Web Consortium) Web services framework [3] is to develop a standard platform for distributed application-to-application (A2A) and business-to-business (B2B) integration. Consequently, although in principle Web services can be used for providing services to end users, the true power of the W3C's Web services framework is leveraged through compositions of Web services. The composed Web services can be distributed over the network, running on different platforms, implemented in different programming languages, and provided by different vendors. The composition provides an added value, either to end users or for further A2A integration, when a composition of Web services can itself become a higher-level Web service. Particularly important is the possibility to compose Web services dynamically (i.e., during runtime), in a manner possibly unanticipated during design-time and deployment-time. Dynamic composition of Web services promises to increase agility and flexibility of A2A and B2B integrations, while minimally interrupting running applications.

Hereafter, by a consumer of a Web service *A* we assume another Web service that is composed with *A* and collaborates with it, not an end user (human) using *A*.

A Web service can serve many different consumers, possibly at the same time. To improve flexibility and adaptability of compositions of Web services, it can be useful for a Web service to offer several different classes of service. Providing differentiated services and multiple classes of service are well-known concepts in telecommunication service engineering and management [1], [6]. We have extended these concepts to address issues relevant for Web services and their compositions.

We define a service offering as one class of service of one Web service. Service offerings of one Web service relate to the same functionality, but differ in constraints, like authorization rights and quality of service (QoS) constraints, as well as cost. For example, service offerings can differ in usage privileges, service priorities, response times guaranteed to consumers, verbosity of response information, etc. Service offerings can also differ in the utilization of the underlying hardware and software resources.

The issues of QoS and balancing of limited underlying resources are particularly motivating for having multiple classes of service for Web services. If the underlying resources were unlimited, all consumers would always get the highest possible QoS. Unfortunately, this is not the case, so it is suitable to provide different QoS to different classes of Web service's consumers. Web services provided by third parties on a pay-per-use basis are very illustrative in this respect. Their providers want to achieve maximal monetary gain with the optimal utilization of resources. On the other hand, consumers want to receive service and QoS they need and are willing to pay for, while minimizing the price/performance ratio. Providing different classes of service for a Web service increases the chance of succeeding in the market because of the flexibility to accommodate several classes of consumer and provide to a consumer an appropriate level of QoS. It also supports different capabilities and rights of consumers of the Web service.

The advantage of having a relatively limited number of classes of service over other types of service customization is limited complexity of required management. Our approach does not exclude applying in addition other methods for customization of service and QoS (e.g., parameter-based), but in the latter case management can be more complex. Similarly, we find personalization techniques aimed at human users to be too complex for customization of Web services in compositions of e- and m-business Web services. As we are particularly addressing compositions of Web services, we want to limit the complexity in order to assure that solutions are scalable to large compositions of Web services. Note also that, as the level of common infrastructure for Web services has to be as minimal as feasible, we believe that too complex management solutions do not fit into the W3C's Web services framework.

To specify Web services with multiple service offerings we are developing an extension of WSDL (Web Services Description Language) called WSOL (Web Service Offerings Language). In WSOL, we place a particular emphasis on the comprehensive formal specification of different types of constraints. WSOL enables specification of functional constraints (pre- and post-conditions, and invariants), non-functional constraints (QoS guaranteed to consumers and QoS required from supplier Web services), authorization policies, cost, and other relevant information and constraints. Known relationships with and dependencies on other Web services and/or infrastructure (including hardware) can also be specified in WSOL. Note that authoriza-

tion policies specified in WSOL describe what subset of a Web service's functionality a service offering provides and under what conditions. Due to security reasons, authorization policies describing conditions under which particular classes of consumer may use a service offering are specified outside the WSOL description of a Web service. While a service offering contains specification of different types of constraints, these specifications are separated into multiple distinct layers to achieve greater flexibility and reusability of specifications. We are still studying some issues related to this separation and integration of different types of constraints and different dimensions of QoS.

As the number of Web services in the market that offer similar functionality increases, the offered QoS and price/performance ratio, as well as adaptability, will become the main competitive advantages. The comprehensive specification of Web services and service offerings in WSOL supports choosing appropriate Web services and service offerings. In addition, it can be beneficial to minimize unexpected feature interactions between Web services in their compositions.

The ideas from [2] and some ideas from [5] influenced the work on WSOL. Although [8] also described an XML-based specification of various constraints, their specification is not WSDL compatible, does not address QoS constraints in depth, and does not address specification of multiple classes of service.

## 3 DYNAMIC MANAGEMENT OF COMPOSITIONS OF WEB SERVICES WITH MULTIPLE SERVICE OFFERINGS

While dynamic composition of Web services promises to increase agility and flexibility of A2A and B2B integrations, we see it only as a part of the agile, flexible, and adaptable e- and m-business solutions. To further increase flexibility and adaptability of e- and m-business solutions compositions of Web services have to be managed.

One of the goals of our research is to achieve management and dynamic adaptation of compositions of Web services without breaking an existing relationship between a Web service and its consumer. This goal differentiates our work from the past work on adaptable software, like the architecture-based approaches based on finding alternative components and rebinding [9]. To achieve this goal we are exploring management and dynamic adaptation mechanisms that are based on the manipulation of service offerings. Further, we are developing an appropriate management infrastructure, called DAMSC (Dynamically Adaptable and Manageable Service Components). Our dynamic adaptation mechanisms include switching between service offerings, deactivation/reactivation of existing service offerings, and creation of new appropriate service offerings.

We believe that our mechanisms are beneficial for both Web service providers and their consumers, especially in the case of pay-per-use relationships in e- and m-business. Pay-per-use Web service providers do not want to lose existing consumers when changes occur. If consumers have to find another Web service to accommodate the change, they might chose one from a competing vendor. On the other hand, in many cases the change has to be accommodated very quickly. In some cases, finding and choosing an appropriate alternative Web service can turn out to be too slow and its success cannot always be guaranteed. In addition, for some consumers it may be inconvenient to look for another Web service every time the circumstances of operation change. Such a situation may occur in e- and m-business systems when choosing an alternative Web service would require establishment of new trust relationships.

Dynamic switching between service offerings is the basic method for dynamic adaptation in our work. Switching can be initiated by a consumer or by the Web service. This mechanism enables consumers to dynamically adapt the service they receive without the need to find another Web service. It also enables Web services to gracefully degrade or upgrade their service and QoS in case of changes.

Deactivation and reactivation of service offerings is used by a Web service in cases when changes in operational circumstances affect what service offerings it can provide to consumers. Some service offerings provided by a Web service cannot be used in all circumstances. For example, it can be sometimes impossible to achieve high QoS or too dangerous to offer low security service offerings. When a change of circumstances occurs (e.g., QoS provided by used service components can rise and fall), a Web service can dynamically and automatically deactivate service offerings that cannot be supported in the new circumstances. The issue is what to do with consumers using the deactivated service offering. We are developing support for handling such cases. In our solution, the service component automatically switches the affected consumers to another service offering and then notifies them about the change. The crucial question in this automatic switching is how to relate service offerings in order to decide on which one to switch. We are exploring several possible alternatives for representing these relationships. Note that if there is no appropriate replacement service offering, an alternative service component has to be sought. When an affected consumer receives a notification of automatic switching of service offerings, it can decide what to do next in the specific situation. Some examples of consumer decisions are accepting the automatic change of service offerings, explicit switching to another service offering that it estimates to be more appropriate, and discarding the affected Web service invocation. Note that as consumers are other Web services these decisions are done by some programming logic and not by human intervention. The deactivated service offering might be reactivated automatically at a later time after another change of circumstances and, eventually, the consumers can be automatically switched back to their original service offering and notified about the change. This helps to achieve as much as possible of the originally intended level of service and QoS. However, the consumer

also has a possibility to notify the Web service that it is not interested in automatic restoration of the original service offering. The described simple algorithm for automatic switching of service offerings helps in almost instantaneous autonomous adaptation to disturbances with minimal loss of service and QoS.

We are also working on infrastructure support and appropriate algorithms for dynamic creation of new service offerings for existing Web services. As all circumstances of runtime operation (e.g., some issues related to QoS) and needs of all consumers cannot be predicted in advance, this mechanism is needed as an addition to the concept of Web services with multiple service offerings to enable further flexibility, customizability, and adaptability. Note that creation of new service offerings is not creation of new functionality, but creation of new sets of constraints (primarily QoS constraints and authorization policies) for the existing functionality. However, it can be used as support for dynamic evolution of Web services. When a Web service is dynamically updated to improve performance and/or add new functionality, new service offerings can describe changes for consumers, e.g., new levels of QoS and authorization policies for the new functionality. When they are notified about the new service offerings, consumers may choose to switch to them or the service component can do the switching automatically when some old service offerings are no longer supported. Dynamic creation of new service offerings is both a powerful and a dangerous feature that cannot be performed arbitrarily due to various possible conflicts. Therefore, we suggest that service components have strict control over the service offerings that they offer, performing creation of new service offerings only in certain circumstances and after rigorous conflict checks. Examples of possible circumstances include the cases when: 1) the implementation has dynamically changed (e.g., in the case of dynamic versioning/evolution); 2) some other service components used dynamically have updated their service offerings (e.g., offer better QoS); 3) an important (e.g., "premium") client has requested creation of a new service offering.

Although the dynamic adaptation mechanisms that we are developing have limited power compared to finding alternative Web services, they enable faster and simpler adaptation and enhance robustness of the relationship between a Web service and its consumer. These mechanisms do not require human intervention and can be performed almost instantaneously. They enable also Web services to retain existing consumers and do not require establishment of new trust relationships between e- and m-business Web services. We believe that these mechanisms are suitable for situations when the required adaptation is relatively limited and acceptable for the consumer. Examples of such adaptations are a small temporary degradation of service and a temporary switch to a more expensive class of service in order to sustain the overall level of service. These mechanisms could be used for accommodating changes, especially relatively frequent changes, which cannot be accommodated on lower system levels.

The idea of sessions in the DAMSC infrastructure is similar to the idea of persistent connections between Web services and their consumers mentioned in [3]. A consumer may use a Web service either invoking individual operations or by opening sessions. With sessions, the authentication and authorization of a consumer to use the Web service and its service offerings, as well as other appropriate checks (e.g., leasing, current availability of resources, etc.), occur only once, at session opening. Sessions also enable keeping the information about the state of interaction, including what service offering is used. A Web service may (but need not) allow multiple parallel sessions with the same consumer and the consumer may use different service offerings for these sessions. The consumer may use only one service offering in one session with the Web service at a time, but it can dynamically switch service offerings without session termination or losing the session state. Some of the checks performed at session opening need not be repeated when switching occurs. However, checks related to the switched service offering (e.g., the usage rights, current availability of resources, etc.) must be performed.

We have researched several alternatives for service offering enforcement, monitoring, and control and developed the concept of a session object. A session object is automatically generated at the Web service side on session opening. It stores the session state and information and code for service offering enforcement, monitoring, and control. The consumer gets an encrypted reference of the session object and provides this encrypted reference in every communication within this session. For individual operation invocations outside sessions, a lightweight session object might be generated if needed for management purposes, but its reference is not passed to the consumer.

## 4 ISSUES AND CHALLENGES

As already noted, we firmly believe that appropriate management of compositions of Web services is crucial for increasing flexibility and adaptability of e- and m-business Web service-based solutions. In the long run, it might also be very significant for the success of the Web service platform for complex A2A and B2B integrations. We want to emphasize the need for Web service management solutions that are independent from vendors of Web services and vendors of underlying infrastructure, thus preventing lock-ins. Web service industrial initiatives and standardization committees do not currently address management issues.

Note that the W3C's Web services framework adopts the "start simple, gradually grow more complex (as needed)" approach and leveraging existing already widely used standards. Therefore, it can be very useful to explore applying and/or extending proven network, distributed system, software/application, and service management solutions for the management of Web services and their compositions. Let us now note several related issues.

As Web services can encapsulate not only software but also hardware functionality, to successfully manage Web services, solutions from different management domains (software/application, network, device), levels (resource, desktop, system, enterprise, service), and areas (configuration, security, performance, fault, and accounting management) have to be integrated. Further, we need appropriate mappings between technically-oriented solutions for management of Web services and their compositions and business-oriented service-level management that end users are ultimately interested in. In order to better support such mappings, solutions for management of Web services must address the price/performance ratio, uninterrupted service availability, and other issues (both technical and non-technical) relevant to end users.

In principle, a Web service in one composition or in one management domain could be viewed as analogous to a network node. However, there are important differences that make management of compositions of Web services more challenging. For example, Web services remain under full control of their vendors that are often third parties.

SNMP seems suitable for integration with Web services because it is simple and already widely adopted. Although a network management protocol it has also been used for some application management. It should not be too complex to make an appropriate Web services SNMP MIB (Management Information Base).

While WBEM (Web-Based Enterprise Management) management solutions, based on XML and CIM, are technically much more compatible with Web services than SNMP, they are not yet widely used and their rate of adoption seems relatively slow. The issues to be studied in this area include to what extent WBEM solutions can be applied and/or extended for Web services and what should be addressed by the corresponding CIM schema.

The usefulness of existing software (particularly component) configuration management solutions and the existing application management standards like ARM (Application Response Monitoring) should be also investigated. We have already stated some related challenges for software configuration management in [10]. An important configuration management issue currently not addressed by industrial initiatives for composing Web services is minimization and handling of unexpected feature interactions. Unexpected feature interactions cannot be always discovered and prevented before the composition, although using a comprehensive specification of Web services can help in their minimization. In addition, management solutions for detecting unexpected feature interactions after the composition and for recovering from such situations are needed.

Another issue is to what extent and at what level of granularity policy-based management can be used. The issue with policy-based management is centralization of policy storage and decision, which might not appropriate for third-party Web services distributed over the Internet.

**REFERENCES**


1. Aimoto, T., Miyake, S. Overview of DiffServ Technology: Its Mechanisms and Implementation. *IEICE Trans. Inf. & Syst.*, Vol. E83-D, No. 5 (May 2000), pp. 957-964.

2. Beugnard, A., Jezequel, J.-M., Plouzeau, N., Watkins, D. Making Components Contract Aware. *Computer*, Vol. 32, No. 7 (July 1999), pp. 38-45.

3. Curbera, F., Mukhi, N., Weerawarana, S. On the Emergence of a Web Services Component Model. In *Proc. of the WCOP 2001 workshop at ECOOP 2001* (Budapest, Hungary, June 2001). On-line at: http://www.research.microsoft.com/~cszypers/events/WCOP2001/Curbera.pdf

4. Feng, N., Ao, G., White, T., Pagurek, B. Dynamic Evolution of Network Management Software by Software Hot-Swapping. In *Proc. of the Seventh IFIP/IEEE International Symposium on Integrated Network Management - IM 2001* (Seattle, USA, May 14-18, 2001), IEEE Publications, pp. 63-76.

5. Hailpern, B., Ossher, H. Extending Objects to Support Multiple Interfaces and Access Control. *IEEE Transactions on Software Engineering*, Vol. 16, No. 11 (November 1990), pp. 1247-1257.

6. Kristiansen L. (ed.) Service Architecture, Version 5.0. TINA-C (Telecommunications Information Networking Architecture Consortium), June 16, 1997. On-line: http://www.tinac.com/specifications/documents/sa50-main.pdf

7. Mennie, D., Pagurek, B. A Runtime Composite Service Creation and Deployment and Its Applications in Internet Security, E-commerce, and Software Provisioning. To be printed in *Proc. of the 25th Annual International Computer Software and Applications Conference - COMPSAC 2001* (Chicago, USA, October 2001), IEEE Computer Society Press.

8. Mckee, P., Marshall, I. Behavioural Specification using XML. In *Proc. of the 7th IEEE Workshop on Future Trends of Distributed Computing Systems - FTDCS'99*, (Cape Town, South Africa, December 1999), IEEE Computer Society Press, pp. 53-59.

9. Oreizy, P., Medvidovic, N., Taylor, R. N. Architecture-Based Software Runtime Evolution. In *Proc. of the International Conference on Software Engineering 1998 - ICSE'98* (Kyoto, Japan, April 1998), ACM Press, pp. 177-186.

10. Tosic, V., Mennie, D., Pagurek, B. Software Configuration Management Related to Management of Distributed Systems and Services and Advanced Service Creation. In *Proc. of the SCM-10 workshop at ICSE 2001* (Toronto, Canada, May 2001). On-line at: http://www.ics.uci.edu/~andre/scm10/papers/tosic.pdf