Web Services Computing: Beyond Component-Based Architectures

Dr. Dionisis X. Adamopoulos
Dept. of Technology Education & Digital Systems
University of Piraeus, Greece

Dr. Constantine A. Papandreou
Hellenic Telecom. Organisation (OTE) and
University of Piraeus, Greece

Abstract

Web services are emerging technologies that can be considered as the result of the continuous improvement of Internet services due to the tremendous increase in demand that is being placed on them. They are rapidly evolving and are expected to change the paradigms of both software development and use, by promoting software reusability over the Internet, by facilitating the wrapping of underlying computing models with XML, and by providing diverse and sophisticated functionality fast and flexibly in the form of composite service offerings. In this paper, the different facets of Web services are identified and a flexible approach to engineering complex Web services is adopted in the form of a proposed framework for the development of Web services. After the examination of its main constituent parts, it is argued that its full potential and that of Web service engineering in general, is realized through the gradual formation of a rich service grid offering value-added supporting functionality and therefore the main desirable properties of such a service grid are highlighted. Finally, the paper outlines a validation approach for the proposed framework and assembles important pointers for future work and concluding remarks.

1. Introduction

Web-enabled service-oriented computing is becoming the prominent paradigm for distributed computing and e-commerce, creating significant opportunities for a variety of providers to develop value-added services by specifying new Web services or by combining already existing ones. These services are self-contained, Web influenced programming entities capable not only of performing business activities on their own, but also possessing the ability to engage other Web services in order to complete higher-order business transactions. Therefore, Web services can be considered as a special category of telematic services (new telecommunications services), that although they have several unique characteristics, they remain geographically distributed entities (actually an encapsulation of a number of cooperating entities distributed over a geographical environment) providing a number of people (users, subscribers) a predefined, carefully selected, set of capabilities / facilities regarding the integrated coverage of a (possibly) wide range of information and communication needs, utilising the resources of (existing and future) telecommunication networks [1].

This paper attempts to determine the boundaries of Web service engineering and, considering the needs of companies that deploy Web services to support increasingly sophisticated business processes, proposes a framework for the development of Web services and examines its main constituent parts, addressing important issues for the creation and provision of a new generation of functionally rich, adaptable, web-centric, composite applications. Furthermore, it introduces the concept of service grids as the necessary infrastructure that will enable Web services to transform the Web from a collection of information into a distributed computational entity, and identifies open matters and current technical challenges for the Web services community in association with the proposed framework.

2. Web service engineering

A Web service is programmable application logic accessible using standard Internet protocols, fulfilling a specific task or a set of tasks and representing a discrete unit of business or system functionality, that can be combined with other Web services to maintain business transactions or workflows [3][7]. By exploiting Web services an organization is able to provide ("expose") any business function to any other entity, such as another business function, an organization, a particular community, as well as end users. A Web service can be provided by the organization directly, through trading networks or specific publishing hubs and other intermediaries on the Web.

Web services are self-contained, self-describing, modular software entities that can be published, located and invoked across the Web. Each discrete Web service can be deployed on and accessed from any node on the Internet, because once a Web service is deployed, other applications (and other Web services) can discover and invoke the deployed service. Therefore, multiple Web services can be combined or assembled to form new service configurations and deliver more valuable and sophisticated functionality supporting diversified business

objectives. In this way, Web services offer a degree of flexibility and granularity not previously possible or economically viable, as they facilitate on-the-fly software creation through the quick assembly of loosely coupled, reusable software components [4]. Furthermore, they can interact with each other ("be orchestrated") in an infinite variety of manners and through multiple iterations in order to deliver a particular task within any context [2].

As a new domain or scientific discipline at the boundaries of software engineering and telecommunications, Web service engineering addresses the technologies and engineering processes required to define, design, implement, test, verify, validate, deploy, combine, maintain, and manage Web services that meet user needs in the current or future networks. Its main objective is to ensure the introduction of new and enhanced Web services and their management, in a fast and efficient manner. It relies heavily on open distributed object-oriented processing and Internet technology, and ambitiously promises to significantly facilitate the offering of a wide variety of highly sophisticated and personalised services over the widest possible coverage area.

Finally, it has to be stressed that Web services represent the convergence between Service-Oriented Architectures (SOAs) and the Web. SOAs (as the one proposed by the Telecommunications Information Networking Architecture-Consortium, TINA-C) have evolved over the last 10 years to support high performance, scalability, reliability and availability [1]. To achieve these properties, applications are designed as services, that can be accessed through a programmable interface and run on a cluster of centralized application servers. In the past, clients accessed these services using a tightly coupled, distributed object protocol, such as Microsoft's DCOM, OMG's CORBA or Sun's Java RMI. While these protocols are very effective for building a specific application, they limit the flexibility of the system. Furthermore, each of the protocols is constrained by dependencies on vendor implementations, platforms, languages or data encoding schemes that severely limit interoperability and none of them operates effectively over the Web [2]. Web services inherit all the best features of the SOAs and all the best aspects of component-based development in general and combine them with the Web. Like components, Web services represent functionality that can be easily reused without knowing how the service is implemented. However, the Web supports universal communication using loosely coupled connections and Web protocols are completely vendor-, platform-, and language- independent.

3. A framework for the development of web services.

Because of the inherent complexity of Web technologies and the recent diversification of the telecommunications environment, Web service engineering activities should satisfy a number of requirements, in order to maximise their usefulness, fulfil the emerging increased expectations regarding their value and impact, and lead eventually to a Web populated by a variety of service objects. The most important requirements that Web service engineering activities should support during their desired evolution process are the following [1][3][4][5]:

- The efficient and effective development of Web services by guiding successfully service developers during the entire Web service creation process.
- The successful application in a Web services context (through the appropriate adaptation when necessary) of carefully selected concepts, models, techniques, design patterns and practices that are developed, tested and (extensively) applied during the creation of conventional telecommunications services.
- The reduction of complexity and the increase of efficiency during the design and implementation of Web services by hiding from the service developers commonly encountered implementation details.
- The efficient automation of the Web service creation process, without semantic loss, with the use of appropriate, carefully designed and tested, customisable, and user-friendly software tools.
- The development of a rich variety of Web services with enhanced content, which can efficiently support a wide range of communication, information, business, education, entertainment and cooperation needs.
- The representation, processing, management, and transmission (possibly in an integrated manner) of all the basic information types.
- The adoption of precise service semantics, because in an open telecommunications market, it is important for reasons of service interoperability and for maximising customer satisfaction to specify Web services in a clear and unambiguous way by using concepts that their semantic content can be accurately defined.
- Reusability at different abstraction levels (e.g. reusable service requirements, service specifications, service components), with the intention to promote rapid Web service design and deployment.
- The use of a variety of document types and their population with appropriate values so that they are semantically coherent and are interpreted correctly by the service requesters and providers.
- The management of Web services in a flexible manner.
- The interoperability of Web services in a multi-provider (open) telecommunications environment (with multiple domains of management and ownership of services) by facilitating and promoting Web service composition.
- Openness to all types of potential end-users of a Web service considering all the possibly interested people (e.g. mobile users, residential users, etc.).
- Openness to change of Web service software and hardware (computer and network infrastructure), because as technology advances, or as prices change, or as purchasing policies and needs dictate, different hardware

should be able to be used without requiring new investment in the accompanying software, and vice versa.
- The accommodation of legacy telecommunications services and systems as they represent significant investments that should be protected.
- Security in each message exchange between a service requester and a service provider, which should be private and unmodified, as well as non-reputable.
- The accommodation of relevant standards (if necessary).

Current Web service technology scores rather low compared to the above mentioned requirements. Therefore, in an attempt to revitalize Web service engineering and enable it for the crucial role that is anticipated to have in the new emerging telecommunications environment, the Web service engineering framework of Figure 1 is proposed with the objective to provide a rich conceptual model for the development and the description of Web services bringing this technology to its full potential.

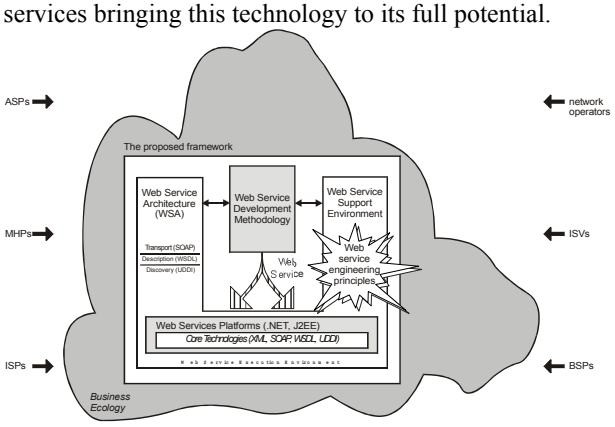

Figure 1. The proposed Web service engineering framework

As can be seen from Figure 1 the proposed framework is placed inside a composite organisational context (a "*business ecology*"), in order to signify that Web service engineering activities are normally performed by a variety of entities / business formations. Although in practice many of the companies operating in this sector / area blend various functions into a composite offering and adopt many different roles, the major players in this new always-on Web services landscape are Application Service Providers (ASPs), Managed Hosting Providers (MHPs), Internet Service Providers (ISPs), network operators, Independent Software Vendors (ISVs) and Business Service Providers (BSPs) [6]. Therefore, the proposed framework is influenced by their business objectives, their general telecommunications and IT strategic orientation, their knowledge, their problem solving attitude and their experience.

The main constituent parts of the proposed Web service engineering framework, which are depicted in Figure 1, are:

- *A Web service development methodology:* It is a methodology that guides service developers during the entire process of Web service creation.
- *A Web service support environment:* It is an environment aiming to facilitate, both the development of Web services (in cooperation with the Web service development methodology) and their execution under real conditions. It consists mainly of:
  - *Web service engineering principles:* These are concepts, guidelines, design patterns, practices and (in general) mental constructs that are applicable to Web service engineering activities.
  - *A Web service architecture:* It contains in a structured manner all necessary details for the information and computational modelling of Web services.
  - *A Web service execution environment:* It encompasses the necessary computing and network infrastructure and the appropriate ancillary software (e.g. operating systems, database management systems, etc.), which is needed for and during the execution of a Web service. Its most important part is the Web platform, which abstracts over all the other parts and reduces greatly the effort needed for the implementation of a Web service. Furthermore, the Web platform is accompanied by a collection of software tools (together with a reuse infrastructure) that are used according to the Web service development methodology with the aim to assist the service developer(s) when applying the methodology.

A Web services environment conforms to the conceptual roles and operations that characterize every SOA. The three basic roles are the service provider, the service consumer and the service broker (see Figure 2). A service provider offers the service and publishes the contract that describes its interface. It then registers the service with a service broker. A service consumer queries the service broker according to its specific needs and finds a compatible service. Then, the service broker informs the service consumer on where to find the service and its service contract. Finally, the service consumer uses the contract to bind the client to the service. In order for the three conceptual roles to accomplish the related conceptual operations, a SOA system must supply / specify three core functional architecture components; namely transport, description, and discovery [7].

Table 1. A comparison of SOA middleware and Web Services

	DCOM	**CORBA**	**Java RMI**	**Web Services**
Invocation Mechanism	DCE RPC	CORBA RMI	Java RMI	JAX-RPC, .NET, etc.
Data Format	NDR	CDR	Serialised Java	XML
Wire Protocol	PDU	GIOP	Stream	SOAP
Transfer Protocol	RPC CO	IIOP	JRMP	HTTP, SMTP, etc.
Interface Description	MIDL / DCE IDL	CORBA IDL	Java Interface	WSDL
Discovery Mechanism	CDS	COS naming	Java Registry	UDDI

As can be seen in Table 1, the distributed object platforms (middleware) that form the basis of SOAs

(DCOM, CORBA or Java RMI) define their own vertical set of formats and protocols to implement the core SOA functions. This approach ensures consistency among applications that share the same middleware, but prevents interoperability with applications that use different middleware. It also requires that every service producer and service consumer that engages in a conversation must have the appropriate middleware installed and loaded on its computing infrastructure.

On the other hand Web services (Internet middleware), unlike traditional SOA systems, do not require an entirely new set of protocols. The most basic Web services protocol is XML (an industry standard), which is used as the message data format and is also used as the foundation of all other Web services protocols. Web services use XML to describe their interfaces and to encode their messages. However, XML by itself does not ensure effortless communication. The applications need standard formats and protocols that allow them to properly interpret the XML. Hence, the following XML-based technologies have emerged as the de facto standards for Web services:

- *Simple Object Access Protocol (SOAP)* establishes a common format for addressing messages and defines a standard communications protocol for Web services.
- *Web Services Description Language (WSDL)* defines a standard mechanism to describe a Web service.
- *Universal Description, Discovery and Integration (UDDI)* provide a standard and uniform mechanism to register and discover Web services.

These core Web service technologies define the transport, description and discovery mechanisms respectively in the way depicted in Table 1, and have a close relationship with a strong semantic underpinning. As most Web service configurations suggest, the three core functional architecture components (transport, description, and discovery) are implemented using SOAP, WSDL, and UDDI, respectively, forming the Web Services Architecture (WSA) that can be seen in Figure 2. A UDDI registry has the role of a service broker. The register and find operations are implemented using the UDDI Inquiry and UDDI Publish APIs. A WSDL document describes the service contract and is used to bind the client to the service. All transport functions are performed using SOAP.

The Web Services Architecture (WSA) provides the necessary means to create Web services for the coverage of an infinite variety of needs and to dynamically combine them to satisfy more specialized business requirements at any point in time, by knitting together micro-services (individual process components) into a broader application entity offering enriched functionality. However, such Web service creation activities can be extremely risky and difficult as Web services can be relatively simple, like the delivery of a currency converter or stock quotes to a cell phone, but also very complex, like a payment processing service where millions of euros are being transferred in individual transactions from one account to another.

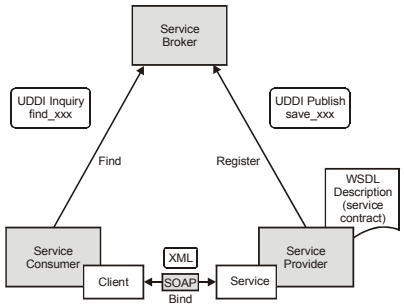

Figure 2. Conceptual roles and operations in a Web Services Architecture (WSA)

Furthermore, all Web services are currently composed in a rather ad hoc and opportunistic manner by simply combining their operations and input and output messages. If the requirements change or need to be adjusted, then the composition will have to be respecified and recreated by possible interlinking additional or modified service interfaces. This approach leads to a proliferation of badly specified service operations and results in unmanageable and cluttered solutions. In this case, the needs of service developers that want to reuse the design and implementation of existing Web services only by extension or restriction, without developing them from scratch, cannot be satisfied.

An equally important problematic situation arises also from the fact that unlike a traditional telecommunications enterprise network, many different providers share the multi-layered network and software infrastructure of Web services. While a particular group of dominant network operators (telcos) provide backbone and last-mile wireline network services, other providers contribute server management, data storage, edge cacheing, security, VPN and wireless services, and several other ancillary services. Therefore, creating an integrated, end-to-end application delivery infrastructure incorporated into a Web service requires close cooperation between all the interconnected, autonomous participants (providers and enterprises).

It is evident that as Web services become more sophisticated and more global in reach and capacity, it becomes increasingly important to provide additional assistance to service developers in order to ensure the effective encounter of the above mentioned problems and the efficient support of commercial-grade application functionality by Web services in an incremental manner, with little risk and at low cost. Recognising these needs, the Web service development methodology of Figure 3 is proposed. This methodology "covers" in a systematic and structured manner the entire Web service creation process through a requirements capture and analysis phase, a Web service analysis phase, a Web service design phase, a Web service implementation phase, and a Web service validation and testing phase. It recognises the inefficiency of current general-purpose software engineering methodologies to address successfully Web service engineering matters and proposes a novel Web service creation process based on fundamental object-oriented analysis and

design concepts and on important results of service creation research regarding the development of telematic services upon distributed object platforms utilising SOAs. The novel character of the proposed methodology is reinforced by the adoption of an incremental and iterative use case driven approach, by the consideration of the special needs imposed by the Web Services Architecture, by the careful incorporation of the Unified Modelling Language (UML) notation and the XML technology throughout the service creation process, by the exploitation of specially constructed design patterns, and by the promotion of reusability and dynamic Web service compositions.

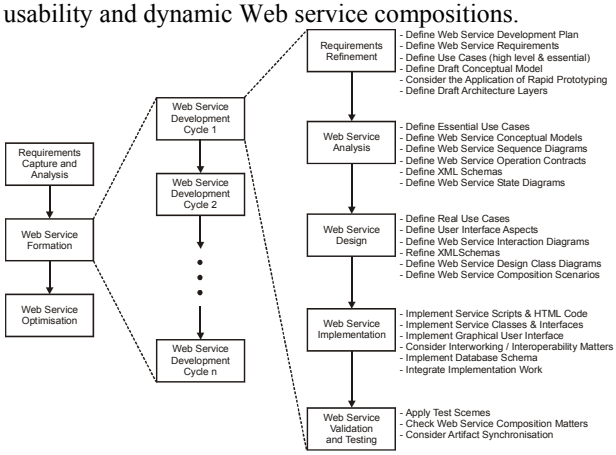

Figure 3. Outline of the proposed Web service development methodology

Unlike other SOA systems, Internet middleware does not define a specific invocation mechanism. It simply defines the communication protocols (XML, SOAP, etc). The specifics of how Web services interact with SOAP and WSDL have been left as an exercise to the service developer's community. Since the WSA is based on standard XML, Web services can be implemented by using the pervasive XML processing techniques that are supported by a variety of software tools, together with ad hoc invocation implementation patterns. However, efficiency can be greatly improved by using specialized Web services platforms, which provide a ready-made foundation for building and deploying Web services, based on a set of carefully selected invocation mechanisms. The advantage of using a Web services platform is that developers don't need to be concerned with constructing or interpreting SOAP messages. They simply write the service code (application logic) and rely on the Internet middleware to do the rest.

The two most prominent Web services platforms currently are Microsoft's .NET and Sun's J2EE. More specifically, Microsoft has defined a set of standard programming interfaces and class libraries for the Visual Studio .NET languages within the .NET framework, and the Microsoft SOAP Toolkit provides support for COM-based applications written in Visual Basic and Visual C++. On the other hand, the Java Community Process' (JCP) has recently defined a set of standard programming interfaces for Java Web services, as part of the J2EE specification: JWSDL (the Java API for WSDL), JAX-RPC (the Java API for XML based RPC), JAXM (the Java API for XML Messaging), SAAJ (the SOAP with Attachments API) and JAXR (the Java API for XML Registries). Although J2EE is an open standard, there are several competing J2EE development environments (Jboss+Tomcat, BEA WebLogic, IBM WebSphere, Sybase EAServer). It is evident that due to the increased capabilities of these platforms and their continual improvement the selection process is a challenging task [8].

4. The importance of service grids

A distributed Web services infrastructure will be required before Web services technology can be broadly deployed to support mission critical applications within and across enterprises. The difference between traditional Web content and Web services originates from the addition of process – the sequence of events that need to happen in order to produce a result. The fact that the users / participants of a Web service are distributed across the Web and need to complete certain processes adds important new operating requirements, such as consistency, authenticity, timeliness, integrity, and persistence [6].

The distributed Web services infrastructure must support these requirements. Service grids constitute a key component of this infrastructure, especially as its scope expands beyond the boundaries of the enterprise to encompass a broad range of business partners. Service grids provide a set of enabling utilities and ancillary services to support more robust connections between providers and users of Web services. This enabling functionality offered by service grids is distinct from application functionality that is directly useful to end-users. It focuses on supporting the application logic with functionality like security, routing of messages across Web services or data transformation so that one Web service can access data from another Web service. It can be considered as the equivalent of the supporting functionality provided by object-oriented middleware in SOAs, with the difference that in this case it is delivered as a set of managed services, rather than installed in the computing infrastructure communicating at either end of the connection [5].

The full value of the proposed framework for the development of Web services is realized when it is used in combination with a carefully created service grid in an integrated manner. This will also require the enhancement of the proposed methodology in order to take into account the availability of the managed services offered by the service grid, especially during the specification of non-functional requirements.

Although service grids are still at a very early stage of development, there is no doubt that the adoption of Web services technology will be significantly affected by the pace and scope of service grid deployments. Nevertheless, significant initiatives are already taking place leading to

the emergence of early generations of service grids and specialized service grid utilities.

5. Conclusions and future work

Web service engineering provide a sound basis for developing and deploying interoperable Web services, allowing the gradual transformation of the Internet to a global common information networking platform where organizations and individuals communicate with each other to carry out various commercial activities and to provide value-added functionality. With the emergence of Web services the Internet has ceased to be solely a content transmission network. It has become a computing execution network, processing commercial transactions and business applications.

However, many of the standards required for Web services are not yet fully defined. The SOAP, WSDL and UDDI specifications that underpin current Web services technology form a de facto standard infrastructure with little endorsement by official standards organizations. For this reason, the existing specifications contain a number of ambiguities and inconsistencies, and address only basic Web services communications. Two standards groups are currently working on the definition of official Web services standards: The World Wide Web Consortium (W3C) and the Organisation for the Advancement of Structured Information Standards (OASIS). W3C focuses on core infrastructure specifications and OASIS focuses on higher-level functionality.

In general, Web services computing poses significant theoretical and engineering challenges as developers determine how to leverage emerging technologies to automate semantically rich application domains and to create software entities with an open interoperable character, based on cross-organisational, heterogeneous software components. The proposed framework for the development of Web services aims to address this movement towards Web-enabled service-oriented computing, where application logic is offered as a set of services both within and across enterprises. Regarding this framework, it is currently attempted to examine it and specify it in greater detail, focusing especially on the Web service development methodology, as it presents increased research interest and practical value. Furthermore, the validation and evaluation of the proposed methodology is considered by applying it to the development (from requirements elicitation up to actual implementation) of a complex representative Web service (eXtended e-Learning Interactive eXperience using Internet Services, eXeLIXIS). This Web service enables students to specify their learning objectives and learning preferences, attend specially designed e-classes that satisfy their needs, participate in training multiplayer games that enhance their understanding and test their knowledge, getting feedback in order to amend or enrich their learning objectives. Two alternative implementations of this Web service are being constructed (using Microsoft's .NET and Sun's J2EE) and special emphasis is placed on Web service composition matters, as the training material can originate by a variety of providers, and on the incorporation of session tracking, as the maintenance of state information for the students by the Web service improves the overall performance, simplifies program development and provides for a more intuitive user interface. This validation attempt aims to provide tangible evidence about the correctness, the efficiency and the true practical value of the proposed methodology. All these efforts treat the proposed framework as a conceptual "umbrella" for Web service engineering activities and gradually transform it to a more precise and concrete construct.

Web services constitute undoubtfully a promising technology that will increasingly assist the integration of heterogeneous islands of application logic (objects on the Web) to homogeneous component-based solutions (a web of objects), especially when supported by robust service grids. However, developers should keep in mind that Web services are still a fast moving target and an immature technology. Nonetheless, Web services technology provide the appropriate solution for the agility requirements that software engineering must cope with today and perhaps will encounter increasingly in the future. Existing object-oriented middleware such as COM+/.NET, CORBA, and EJB/RMI may be still necessary to implement sophisticated back-end services, but Web services claim a prominent role when these functionality islands must be connected to fully operational networked systems.

References

[1] D.X. Adamopoulos, G. Pavlou, and C.A. Papandreou, "Advanced Service Creation Using Distributed Object Technology", *IEEE Communications Magazine*, Vol. 40, No. 3, March 2002, pp. 146-154.

[2] J.-Y. Chung, K.-J. Lin, and R.G. Mathieu, "Web Services Computing: Advancing Software Interoperability", *IEEE Computer*, Vol. 36, No. 10, October 2003, pp. 35-37.

[3] F. Curbera, R. Khalaf, N. Mukhi, S. Tai, and S. Weerawarana, "The Next Step in Web Services", *Communications of the ACM*, Vol. 46, No. 10, October 2003, pp. 29-33.

[4] D. Fensel and C. Bussler, "The Web Service Modeling Framework WSMF", *Electronic Commerce: Research and Applications*, Vol. 1, 2002, pp. 113-137.

[5] J. Hagel and J.S. Brown, "Service Grids: The Missing Link in Web Services", *White Paper*, 2002.

[6] P. Wainewright, "Web Services Infrastructure: The Global Utility for Real-Time Business", *White Paper*, 2002.

[7] Web Services Architecture Working Group, "Web Services Architecture Requirements", *W3C Working Draft*, Aug. 19, 2002[http://www.w3.org/TR/2002/WD-wsa-reqs-20020819].

[8] J. Williams, "The Web Services Debate: J2EE vs. .NET", *Communications of the ACM*, Vol. 46, No. 6, June 2003, pp. 59-63.