# A DEVELOPMENT INFRASTRUCTURE FOR WEB SERVICES

Dionisis X. Adamopoulos
*Department of Technology Education & Digital Systems, University of Piraeus, Greece*

Keywords: Web services, Web service engineering, Internet middleware.

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 services can be considered as a special category of telematic services, that although they have several unique characteristics, they remain geographically distributed entities providing a number of people a predefined, carefully selected, set of capabilities / facilities, utilising the resources of telecommunication networks (Adamopoulos, 2003). 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.

## 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 (Curbera, 2004)(WSAWG, 2005). 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.

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


X. Adamopoulos D. (2008).
A DEVELOPMENT INFRASTRUCTURE FOR WEB SERVICES.
In *Proceedings of the Fourth International Conference on Web Information Systems and Technologies*, pages 256-259
DOI: 10.5220/0001522302560259

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 (Adamopoulos, 2003). 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 (Chung, 2003). 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. Current Web service technology scores rather low compared to these 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, a Web service engineering framework 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.

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) (Wainewright, 2006). 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 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. 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 (WSAWG, 2005).

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). 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.

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. 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, a Web service development methodology 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.

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.

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. 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. It is evident that due to the increased capabilities of these platforms and their continual improvement the selection process is a challenging task (Williams, 2003).

## 4 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 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.

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. 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

Adamopoulos, D.X., Pavlou, G., Papandreou, C.A., 2003. Advanced Service Creation Using Distributed Object Technology. In *IEEE Communications Magazine*, Vol. 40, No. 3, pp. 146-154.

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

Curbera, F., Khalaf, R., Mukhi, N., Tai, S., Weerawarana, S., 2004. The Next Step in Web Services. In *Communications of the ACM*, Vol. 46, No. 10, pp. 29-33.

Fensel, D., Bussler, C., 2005. The Web Service Modeling Framework WSMF. In *Electronic Commerce: Research and Applications*, Vol. 1, pp. 113-137.

Wainewright, P., 2006. *Web Services Infrastructure: The Global Utility for Real-Time Business*, White Paper.

WSAWG, 2005. *Web Services Architecture Requirements*, W3C Working Draft.

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