# THE VISION OF SEMANTICALLY ENABLED SERVICE ARCHITECTURES

Ljupcho Antovski
University Ss. Cyril and Methodius, PMF, Institute of Informatics
Skopje, Macedonia

## ABSTRACT

The Semantic differences are the main obstacle for automated web service based applications integration. Until there is no direct way for applications to understand each other, the effect of web services technology will be quite limited. This paper is an overview of the current state of the art mainly in the research of the European community in the field of the semantically enabled service architectures. It presents the possibilities, prospects but also the constraints.

## KEY WORDS

Web service, SOA, SESA, service automation, integration

## I. INTRODUCTION

Over the past few years, we face a growing interest in the potential of web services and the use of more automated and interpretable content on the WWW, involving the development of ontologies. At the same time, very little has been done to combine the full power of both of these approaches. The idea is to design information systems which adopt paradigms of Service Oriented Architectures (SOA). Although the idea of SOA is to provide more adaptive systems friendly to change in business requirements, existing SOA solutions are difficult to scale without a proper degree of automation. While today's service technologies around WSDL, SOAP, UDDI and BPEL certainly brought a new potential to SOA, they only provide partial solution to interoperability, mainly by means of unified technological environments.

## II. SOA OVERVIEW

The Service-oriented architecture [1] is believed to become the future e-government technology solution that promises the agility and flexibility the business users have been looking for by leveraging the integration process through composition of the services spanning multiple agencies [8]. SOA is an approach to loosely coupled, protocol independent, standards-based distributed computing where software resources available on the network are considered as Services. The services in SOA have minimum amount of interdependencies. The communication infrastructure used within an SOA should be designed to be independent of the underlying protocol layer. It should offer coarse-grained business services, as opposed to fine-grained software-oriented function calls. It implements service granularity to provide effective composition, encapsulation and management of services. The key concepts SOA is based on are: Loose Coupling, Coarse Granularity and Asynchrony. Loosely coupled, coarse grained, asynchronous SOAs provide a layer of abstraction that hides the complexity of the underlying technical implementation details from the user who takes advantages of the Services the SOA exposes.

- Coarse Granularity – The traditional approach to getting information in and out of an application is via an application programming interface, or API. APIs are typically fine grained, which means that each method call is a detail-oriented, technical construct for use by programmers. For two systems to communicate a complex business task via an API, they typically must exchange many of these fine-grained messages. Web Services are at their most powerful when they are used to exchange coarse grained information between systems. Coarse granularity clearly depends on loose coupling, because the Web Service consumer does not care how the Web Service puts together the information it needs [7].

- Loose Coupling - traditional distributed computing architectures is that they are tightly coupled, Making changes to one tightly coupled system often affects the whole architecture, requiring expensive and difficult reworking. SOA based on Web Services is loosely coupled. Each Web Service describes how other systems, known as Web Service consumers, can connect to it and exchange information with it. A developer can make changes to a Web Service without breaking the Service-oriented architecture.

- Asynchrony - synchronous communications consist of round-trip messages in which the sender waits for a reply. With an asynchronous message, the sender can submit a request, and then go about its work. If a reply does come, then the original sender can pick it up when it wants. Email works asynchronously, for example. Web Services based SOAs enable sending and receiving both synchronous and asynchronous messages.

In order to visualize SOA, one can use the SOA generic model, represented on Figure 1 [6].

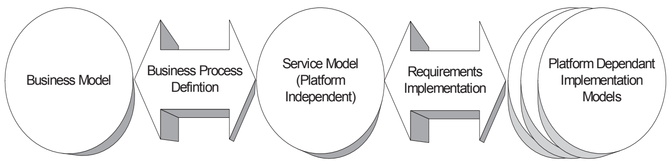
Business Model
Business Process Defintion
Service Model (Platform Independent)
Requirements Implementation
Platform Dependant Implementation Models

Figure 1: SOA Generic Model

The basic of the SOA Meta-model is the Service Model. The Service Model represents the Services an agency has in production. The Business Model represents the users and their requirements. The Implementation Models represent the technology underlying the Services. The Service Model becomes the point of contact between the business and technology. The most important feature of the SOA Meta-model is in fact this description between the business and technology domains, coupled with the explicit modelling of the two-way interaction between. This balance enables the business to drive the technology in an environment of transformation [10] [11].

The process of service-oriented modelling and architecture consists of three general steps: identification, specification and realization of services, components and flows [9]. The process of identification consists of a combination of top-down, bottom-up, and middle-out techniques of domain decomposition, existing asset analysis, and goal-service modelling. In the top-down view, a blueprint of business use cases provides the specification for business services.

It is important to start service classification into a service hierarchy, reflecting the composite nature of services. Classification helps determine composition and layering. Also, it helps minimize the service proliferation syndrome in which an increasing number of small-grained services get deployed with very little governance, resulting in major issues. The service realization recognizes that the software that realizes a given service must be selected or custom built. Other options that are available include integration, transformation, subscription and outsourcing of parts of the functionality using Web services.

## III. Semantic Issues with SOA

The essential challenges of computing that are not addressed by SOA are search and integration. Besides that SOA provide a basis for addressing these challenges, SOA significantly and fundamentally depends on solutions to fill the semantic gap to achieve its potential. SOA provides the potential of a global registry in which to search for services anywhere in the network. This is referred to as service discovery.

SOA provides the potential of invoking remote services to achieve the combined results of those services. This requires that the services interoperate or integrate with respect to their respective data, protocol, and process syntax and semantics. These actions are called service composition and integration. Service automation does not address the challenges of automating discovery, composition, or integration.

In the SOA world service requirements include functional and non-functional requirements. Functional requirements of a service require that a match be made based on a meaningful match of the required service functions with the capabilities of the functions that will provide the offering service. The semantics of the functions provided by the service must match those defined in the requirement. It is the same situation with the non-functional requirements. For service discovery and matching to be achieved dynamically syntactic and semantic aspects of service requirements must be considered.

The service discovery is a process of matching the user needs for service with the capabilities of the service-candidates and discovery of unique service or complex of unique services. In SOA the service requests contain functional and non-functional descriptions. More specifically the semantic requests must match with the semantic possibilities of the server. The same counts for the non-functional descriptions. In order to have dynamic or so called automatic discovery and selection of services, one must consider the syntactic but more the semantic characteristics of the service. We argue that SOA will not work in near future when billions of services emerge if t automatic way of service discovery will not exist. The selection of services is a more complex task, bearing in mind the orchestration, and in some cases composition of complex services.

Without optimization, the computational cost of the algorithm for service discovery is linearly dependant O(n) to the number n of available services. The number of service in SOA can be very large. If the search space is not organized, the efficiency of finding service can be extremely low. All the published algorithms for semantic discovery are dealing with automation of the process and definitions of the goal oriented architecture. Very few algorithms deal with the optimization and improvement of efficiency in an environment with large number of services.

The service discovery and composition is the factor indented as bottleneck point from the aspect of performance in SESA. When we talk about web services, one must bear in mind that today there are 91.323.472 domains on the Net [4]. From the perspective of performance, there is an identified need for optimized algorithm for service directory search.
The use of semantic organization, like graphs [2] is an efficient approach in reducing the search space. In this dissertation we propose a new model to improve the efficiency with shorter search time and use of fewer resources.

## IV. Current WSMO Solutions

The Web Service Modelling Ontology (WSMO) consists of several components presented in the following text. WSMO [12] provides a conceptual model for adding semantics to SOA. Its main elements are goal definitions of user, service definitions of providers, and Ontologies and mediators as declarative and procedural means to facilitate Interoperability at the level of data, protocols, and processes.

The Web Service Modelling Language (WSML) [3] is a set of languages providing formal semantics for WSMO models. Its four major dialects form a lattice based on rule languages and descriptions logics as well as on their minimal and maximal intersection.

The Web Service Execution Environment (WSMX) [5] is an example implementation of an SESA that is compliant with the semantic specifications of WSMO. It supports semantically enabled shifting functions such as dynamic discovery, selection, and complex creation. WSMX also implements semantically enabled control and connection functions such as service invocation and interoperation. WSMX is an execution environment for the dynamic discovery, selection, mediation, invocation and Interoperation of the Semantic Web Services in a reference implementation for WSMO. The development process for WSMX includes defining its conceptual model, defining the execution semantics for the environment, describing architecture and software design and building real life implementation.

WSMO provides three main categories of semantic descriptions. First, it provides means to describe Web services, and then it provides means to describe user goals referring to the problem solving aspect of our architecture. Last it provides means to ensure interoperability between the various semantic descriptions of heterogeneous environments with ontologies and mediators. Goals provide means to characterize user requests in terms of functional and non-functional requirements. Web service descriptions build up on this by an interface definition that defines access of, as well as means to complex services composed from several services. More concretely, a Web service has a capability that is a functional description of a Web service, than interfaces that specify how the service behaves in order to achieve its functionality. A service interface consists of a choreography that describes the interface for the client service interaction required for service consumption, and an orchestration that describes how the functionality of a Web service is achieved. Ontologies provide an important means to achieve interoperability between goals and services. Mediators provide additional procedural elements to specify further mappings that cannot directly be captured using ontologies.

## V. Possibilities as Conclusion

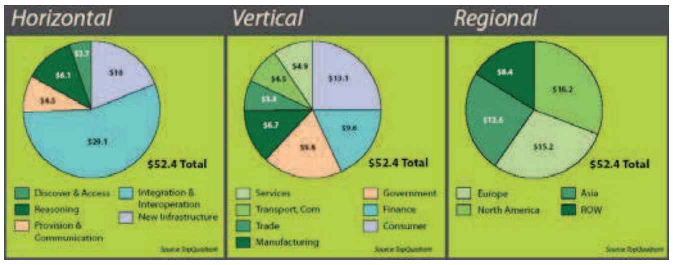

Figure 2: Estimates for 2010, source: TopQuadrant

This paper presented some possible ways of implementing SESA. The success will come in numbers of profit when the business is at stake. Figure 2 shows the potential of the market of SESA. According to the source, TopQuadrant, this market is estimated to be worth $52,4 billion in 2010. It shows the full potential of this technology.

## References

[1] Antovski, Lj. And Gusev, M., M-GOV –The Evolution Method, Proceedings of the Second European Conference on Mobile Government , University Sussex, Brighton UK, September 2006
[2] Antovski, Lj. Gusev M., Improving service matching in M-government with soft technologies, SWEB Workshop, 18th IEEE PIMRC 2007, Athens , Greece, 07 September, 2007
[3] Bruijn, J, et Al, Owl DL vs. OWL FLight: Conceptual modelling and reasoning for the semantic web, Proceedings of the 14th International World Wide Web Conference, 2005
[4] DomainTools (2007), Internet Domains Statistics, http://www.domaintools.com/internet-statistics/, accessed: 01.06.2007
[5] Haller, A., E. Cimpian, A. Mocan, E. Oren, and C. Bussler, WSMX A Semantic Service Oriented Architecture. Proceedings of the International Conference on Web Service (ICWS 2005), 2005
[6] Jason Bloomberg, The SOA Implementation Framework, http://www.zapthink.com, April 2004
[7] Jeff Hanson, Coarse-grained Interfaces Enable Service Composition in SOA, JavaOne, August 2003
[8] Kishore Channabasavaiah and Kerrie Holly, IBM White Paper, Migrating to a Service-Oriented Architecture, April 2004
[9] KIT(2005), National Strategy for Information Society Development, http://www.kit.gov.mk , April 2005
[10] Kushchu, I. and Kuscu, H (2003), “From E-Government to M-Government: Facing the Inevitable”, in the proceedings of European Conference on E-Government (ECEG 2003), Trinity College, Dublin
[11] Meta Group White Paper, Practical Approaches to Service-Oriented Architecture, November 2003
[12] Roman, D. et Al , Web Service Modelling Ontology, Applied Ontology, pp. 77-106, 2005