Budapest University of Technology and Economics
Department of Control Engineering and Information Technology

Model Driven Construction of Interoperable WS-* Applications with Predictable Response Times

Ph.D. Thesis Booklet

Balázs Simon

Supervisor:

Károly Kondorosi, Ph.D.
honorary professor

Budapest, 2015

Contents

Abstract

Web services provide platform independent distributed communication by building all the related standards to XML. The interface description language (WSDL), the communication protocol (SOAP), and the middleware extensions of SOAP, called WS-* protocols, are all based on XML. Composite web services can be created by using business processes described in BPEL, which is also based on XML. The major benefit of using XML is platform independence. However, the software vendors implementing these standards provide different configuration methods for the WS-* protocols, and matching these configurations with each other can be tedious. In addition, XML interface descriptions are not user friendly and it is hard to provide versioning for them. Moreover, XML messages have a large communication overhead compared to binary protocols, such as RMI and CORBA. The process engines developed by the software vendors are capable of executing BPEL processes, however, for long running processes it may be required to be able to migrate running instances of these processes between process engines to avoid vendor lock-in. These are the most common challenges in web service development.

The dissertation gives a solution to all of these challenges. The contributions of the dissertation to the state of the art of web services can be grouped into three main parts. The first part covers the development productivity of web services. In this part a platform independent domain specific modeling framework for web services is introduced. The framework is capable of generating interoperable source codes and configuration files for the most important SOA products, and it increases the development productivity and portability of web service applications. The second part gives a solution to the business process instance migration task. Although this solution requires some restrictions on the process description, the solution works with all process engines. The third part provides a performance model for predicting the communication overhead of web service calls still at design time. This is very useful if the services are required to meet strict QoS properties.

1

Preliminaries and Objectives of the Research

Service-Oriented Architecture (SOA) [OAS06] defines a set of principles for connecting distributed software components. SOA is an architectural style, which can be implemented using various technologies. Earlier examples are RMI, CORBA and DCOM. Today, web services are the main implementation technology of SOA. Web services realize distributed communication in a platform independent way by building all the related standards on XML. The communication protocol (SOAP), the interface description language (WSDL) and the middleware aspects, called WS-* standards (WS-Addressing, WS-ReliableMessaging, WS-Security, WS-SecureConversation, etc.), are all represented in XML. These middleware aspects can also be configured in a platform independent way through XML chunks called WS-Policy assertions which are usually included in the WSDL. Web services can be combined into processes using the Business Process Execution Language (BPEL) standard. BPEL processes are executable, their implementation is described in XML, and they are also web services themselves.

Web services, being an industry standard and using XML instead of a binary representation, can solve the problem of interoperability of different programming languages, execution environments and hardware architectures. The standards related to web services deal only with interface descriptors and protocol messages, and they do not define how these concepts should be mapped to programming languages and execution environments. This may also be the reason why most software vendors adopted the technology and why it gained such popularity. The major adopters are Microsoft, Oracle and IBM, and open source communities such as the RedHat and the Apache community. They provide application servers, frameworks and development tools which are intended to make the developers' life easier.

On one hand, interoperability based solely on XML is successful, since everyone can choose how to implement the standards; all they need to care about is to send the appropriate messages through the wire. On the other hand, XML introduces a number of challenges for the developers of the services. In the dissertation I identify the most common challenges that hinder the development of web services and I provide a solution that covers these challenges. I show that my solution increases the productivity of web service development more than the other already existing solutions. The identified challenges come from the literature, the practical experience of developers across the world and from my own personal experience by being involved in industrial software projects.

1.1 The most important challenges in web service development

Using XML for interface description and for communication can solve the problem of interoperability of different programming languages and execution environments. However, the use of XML comes at a price. There are a number of challenges that developers have to face because of XML, and these challenges have performance and development productivity implications regarding web services.

The fundamental software development process activities are requirements analysis, specification, design, implementation, testing and maintenance [Som11]. The various software development life cycle (SDLC) models differ only in the order and magnitude of these steps. SOA applications and web services are also developed using one of the SDLC models with additional focus on reuse and composition. The use of XML introduces productivity challenges in the specification, implementation and maintenance activities.

The WSDL interface description is a complex XML with some redundant parts, and thus, it is not suitable for being written by humans. The software vendors came up with a number of solutions for this problem. The most common solution is to generate WSDL based on the implementation of the service. This bottom-up style of development is good for marketing purposes and maybe for prototyping. However, in real-life projects this style is undesirable, since any minor change in the implementation may result in a change in the WSDL interface itself. One could argue that the initial WSDL could be generated this way, and then it can be used as a starting point for the interface definition. But maintaining this definition can be as hard as writing it from scratch. Another way of creating a WSDL is through graphical designers provided by the tools. The problem with these graphical designers is that they do not increase the abstraction level enough and they maintain the redundancy of the WSDL description. In addition, they do not provide graphical editing features for the WS-Policy assertions, so the WS-* protocols cannot be easily configured. The tools provided by the software vendors do not support WSDL creation in a convenient way, which makes it hard to follow the recommended top-down (WSDL-first) development in real-life projects [FM14].

The WS-Policy assertions used for configuring the WS-* protocols make the problem of handling WSDL files even worse. The WS-Policy assertions can be large and complex XML structures and it can be hard to craft them by hand. To make the developers' task easier, most SOA products provide policy repositories (e.g. Oracle JDeveloper, IBM WebSphere) containing complete assertions that can be used for configuration. There are even products which offer GUI designers (e.g. Netbeans) or transform WS-Policy assertions into a more convenient configuration representation of their own (e.g. Apache CXF, Microsoft WCF). Although these configuration approaches may ease the task of the developers, they can also introduce interoperability problems between different products, since the representations of the WS-Policy assertions vary between products, and matching these representations requires detailed knowledge of both products.

Most issues arise with the configuration of security protocols, especially security tokens and security token services, since they require a lot of parameters, such as certificates, encryption and digital signature algorithms. The XACML (eXtensible Access Control Markup Language) standard is designed for defining policies and rules for authorization decisions. It provides Attribute-Based Access Control (ABAC) in a platform independent way. However, since it is also based on XML, it is not convenient to write XACML policies by hand. Another problem with XACML is that it is not yet widespread. For example, Microsoft WCF does not yet support it and most open-source implementations rely on the old Sun XACML library.

Another challenge arises from the different implementations of the standards. The WS-* standards have many versions and they are so complex and ambiguous, that the various software vendors may implement only parts of them or may implement them differently, resulting in interoperability problems between the tools developed by the software vendors. This is not desirable when the whole point of web services is that they should provide interoperability. This problem was soon realized and the organization called Web Services Interoperability (WS-I) was founded. The goal of this organization is to define a set of best practices and constraints on the WS-* standards so that they become unambiguous, and thus the interoperability between the WS-I compliant tools could be achieved easier. Although WS-I provides test cases to check these constraints, the organization itself does not certify the WS-I compliance of the vendors' tools. In addition, most constraints can only be checked at runtime by actually calling the running services, and this does not guarantee that the vendors' WS-* implementations are WS-I compliant in all cases. And even if the tools are or at least claim to be WS-I compliant, it does not ensure interoperability. Configuring the tools to use the WS-I compliant versions of the protocols and matching these configurations of the different tools with each other can still be a challenge.

Although the web service standards do not provide a mapping to programming languages, the software vendors do. These mappings are called web service APIs. In the .NET world the current API is the Windows Communication Foundation (WCF). In the Java world it is the Java API for XML-Based Web Services (JAX-WS). The two APIs are basically equivalent, they define similar metadata (attributes in .NET, annotations in Java) to provide a mapping between interfaces in the language and the WSDL, and between values in the language and SOAP messages. WCF does not use WS-Policy assertions directly, but it provides a mapping between WS-Policy assertions and its own configuration format. The JAX-WS specification provides a common API for JavaEE application servers. However, the specification does not deal with WS-Policy assertions, which means that each Java software vendor has its own way of handling these assertions. This can cause portability issues between the different application servers if WS-* protocols are enabled.

The goal of the JavaEE specifications is to ensure the portability of Java enterprise applications between different JavaEE servers. Although there can be vendor-specific settings, most of the source codes and configurations should be portable between servers. As it was already mentioned, the configurations of the WS-* protocols cannot be ported easily, but this problem can be solved by learning the new tool's configuration methods. When an application is ported between application servers, its database can be reused or the database itself can also be migrated to another database server. However, there are vendor-specific databases, which cannot be migrated. Even the migration of these databases between different versions of the same application server from the same vendor can be difficult. For example, the states of BPEL process instances are stored in such vendor-specific databases, therefore, migrating BPEL process instances is also an important challenge to solve.

Service interface versioning [EKW$^+$09] is a challenging part of the maintenance activity of an application. The top-down development of web services requires the versioning of WSDL and XSD files, however, these files are hard to maintain and update. Following and merging them in version control systems can also be a complex task because of the XML syntax. Graphical WSDL designers can hide the inconveniences of the XML format, however, these designers provide no support for versioning and merging of WSDL and XSD files, either.

Another issue of web services is that XML messages burden the communication with a significant serialization and deserialization overhead compared to binary protocols. The serialization overhead can even be comparable with the execution time of the service's application logic itself. This performance loss is one of the main criticisms against web services. When the Quality of Service (QoS) has to meet the expectations set by a Service Level Agreement (SLA), it is essential to know the performance implications of XML serialization at design time, even before the services are actually implemented.

The challenges explained above are the main obstacles that hinder the development of web services. These challenges are summarized in the following list:

- there is no user friendly way of creating, maintaining and versioning WSDL files for top-down development
- achieving interoperability between different tools is hard, because they have different configuration methods for the WS-* protocols, and they cover different parts of the standards
- JAX-WS does not deal with WS-* protocols, therefore, portability of web services with WS-* protocols between different JavaEE servers is unsolved
- XACML is not a convenient API for attribute-based access control
- the states of BPEL process instances are hard to migrate
- the XML serialization overhead of SOAP messages is significant, but it is not predictable at design time

If these tasks are solved, the productivity of the web service development process can be greatly increased especially in the design, implementation and maintenance phases. In the dissertation I provide a solution for all of these tasks, and I show that my solution has a better productivity than the other solutions currently available.

1.2 Related works

Researchers have been trying to give solutions to the challenges listed in the previous section. However, none of the existing approaches provide a solution, which is user friendly, productive and platform independent at the same time.

One of the most obvious ways to model web services is through UML. There are approaches without stereotypes [GSSO04] and with stereotypes [CMV04, DNsmGW08, EES10], however, none of these solutions is capable of modeling WS-* protocol configurations.

H. Wada et. al. [WSO08, WSO06] have a very detailed UML profile which can describe a lot of middleware aspects of SOA systems, however, more advanced security protocols such as WS-SecureConversation, WS-Trust and WS-Federation are not supported. They do not provide a mapping for models to configurations for the SOA products of important vendors like Microsoft, IBM or Oracle.

Most solutions [OMG09, JB08, PWH08, CBZ$^+$04, QN10] completely lack the middleware aspects of WS-* protocols.

Another way of describing services is using semantic web technologies [W3C12b, W3C12a, ESS12, KPKH05, VAG05, SSVS04]. The major goal of Semantic Web Services (SWS) is to create intelligent software agents to provide automated, interoperable and meaningful coordination of web services [Klu08]. Hence, SWS were not designed for modeling, and they require their own execution environment, so they cannot be used with the most important SOA products of the industry.

The problem with existing approaches is that they do not cover the most important WS-* standards, they do not support the most important SOA products of the industry and they do not use the appropriate abstraction level to describe web services.

There are existing solutions for business process instance migration, too. Zaplata et. al. [ZHKL10] provide a general solution, which does not require any constraints or modifications on the process, the migration data is calculated automatically from the process description and process instance execution. However, their approach requires extensions to the process engines which may not always be possible with commercial process engines.

Other approaches [CAB08, KG99, DVRS03, PSSvdA07, RRKD05, RMR10] focus on schema evolution of business processes. These, however, either require the modification of the process engine or they need their own execution environment. The problem with these intrusive solutions is that they cannot be used with SOA products of industry vendors. Some of them are even incomplete solutions, since they not cover timing activities and parallel branches, and the non-migratable instances are not handled at all.

The performance implications of web services have also been examined by previous works. Most of these approaches [NSP$^+$10, REB11, JRB$^+$06, JRH00, SSFG04, vEZ08] compare web services to earlier technologies (RMI, DCOM and CORBA). Their conclusion is what expected: web services have a large performance overhead compared to binary protocols, especially if security is enabled. But none of these approaches provide a performance prediction mechanism.

Performance prediction for composite services has a rich literature [AdL08, ESBS07, AGM08, KKK08, MM07]. These papers all focus on service composition, however, at their bottom, the response time for each service has to be provided, and this piece is missing from all of them.

G. Imre et. al. [IKLC10] developed a cost model for XML serialization in Java and .NET. XML serialization is the heart of SOAP message serialization, therefore, it is an important contribution in this area, although, they only examined three primitive types (string, int and double). A more extensive examination of the primitive types and the WS-* protocols is still needed.

I have not found any related work that gives an extensive examination of the response time overhead of web services, and no previous work provides performance prediction based merely on the service interface.

1.3 Research methodology

Most of the outlined challenges originate from real-life projects, so most of them stem from practical needs. I soon realized the development productivity challenges when I was involved in the project for the Hungarian e-Government Infrastructure, where solving the interoperability between SOA products of different software vendors was essential. This determined the main path of my research.

At first, I examined the existing solutions. However, none of the previous approaches covered the required WS-* protocols, none of them had the appropriate abstraction level for platform independent modeling and none of them provided a mapping to the most important SOA products of the industry. Hence, I started a new direction of research by creating a domain specific modeling language designed especially for web services, which had the appropriate abstraction level and could provide the productivity that was required in real-life projects.

I also examined existing solutions for business process instance migration, but none of them were sound, platform independent and non-intrusive at the same time. The portability of business process instances between SOA products of the industry is essential in an e-government infrastructure, and only a platform independent non-intrusive solution is acceptable here, since proprietary process engines are not freely customizable. Therefore, I started a new and previously unconsidered approach, which is based on recording and replaying messages to business processes. This approach was promising, however, the exact constraints for migratable processes had to be developed, and the soundness of the approach had to be proved, too.

One of the biggest criticisms against web services is that they have a large communication overhead, and in business environments there can be strict QoS requirements for the response time. In order to meet these requirements it is important to predict this response time even before the services are actually implemented so that the interface of the service can be designed with the appropriate granularity. I searched for existing approaches for predicting this response time overhead, however, no previous solution covered all the primitive types and the WS-* protocols had not been considered before at all.

Since the challenges originate from practical needs, my research can be classified as applied research. I also applied my theoretical results back into real-life projects. My domain specific modeling framework for web services proved to be very productive especially in the second phase of the e-Government project and in a project with the Hungarian Railways, so the original goals of the research have been achieved. In addition, the new theoretical results opened other research directions, too.

1.4 Contributions

The dissertation gives a detailed description of the following contributions to the area of web service development:

- I created a domain specific language, which provides an intuitive platform independent description for systems of distributed web services configured with the most important WS-* standards
- I created a generator for producing interoperable configurations for the most important SOA products of the software industry, providing greatly increased productivity compared to the previous solutions
- I developed a method based on recording and replaying SOAP messages to provide migration of BPEL process instances between different process execution engines
- I developed a performance model for predicting the response time overhead of web service calls based on the interface description of the service still at design time

2

New Scientific Results

After briefly summarizing the general architecture of the web service protocol stacks implementing the WS-* standards, this chapter gives a high level overview of the contributions of the dissertation to the state of the art of web service development. At first, the outline of the proposed domain specific metamodel for web services is shown, which can be used for modeling distributed services in a platform independent way. The metamodel is capable of describing simple and complex types, interfaces of services and middleware aspects including addressing, reliability, transactions and security, as well as contracts (pre- and post-conditions) for services and also claims-based authorization. This chapter also outlines the proposed algorithm behind the business process instance migration. The last part of this chapter gives a high level overview of the proposed performance prediction method for web services.

2.1 Web service stack implementation architecture

The most widely implemented WS-* protocols are WS-Addressing, WS-ReliableMessaging, WS-Security, WS-SecureConversation and WS-AtomicTransaction. The Microsoft .NET Framework, the GlassFish server, the Oracle WebLogic server, the IBM WebSphere server and the Apache CXF framework support most of them. The difference between these frameworks is in the configuration of these protocols. There are many other WS-* protocols. However, the most widely used SOA products do not implement them; therefore, they are also omitted from the dissertation.

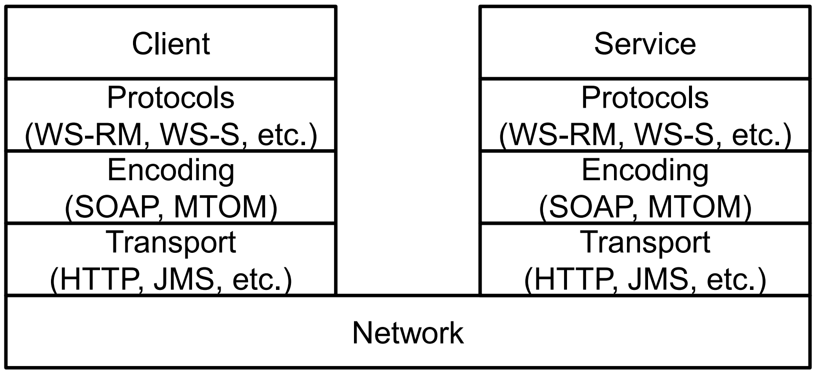


Figure 2.1: Typical architecture of the web service stack implementations

Figure 2.1 shows the general architecture of the web service stack implementations. The Windows Communication Foundation (WCF) [Mic14] stack (part of the Microsoft .NET framework) and the Metro [Ora14] stack (the reference implementation of JAX-WS) follow this structure, and other framework implementations can also be modeled this way.

The **network** is at the bottom, it is used for sending bytes from one end to the other. The network protocol for web services is usually HTTP, but it can be replaced with other protocols.

The **transport** layer is responsible for handling the network protocol. On the service side it waits for client connections, on the client side it connects to services. It also transfers bytes between the two participants.

The **encoding** layer translates between bytes and a framework specific message object representation, i.e. it is responsible for serialization and deserialization (e.g. into SOAP, with or without MTOM). The transport and encoding layers are always mandatory.

The **protocol** layers are optional, and they implement the various WS-* standards (e.g. WS-ReliableMessaging, WS-Security, etc.). The protocol layers usually produce bootstrap messages or insert additional headers into service invocation messages. For example, the WS-ReliableMessaging protocol includes bootstrap messages for initiating and terminating the reliable session. WS-ReliableMessaging also extends the messages with additional SOAP headers containing acknowledgement information about the messages already received by the parties. WS-SecureConversation also has bootstrap messages for establishing the security context and the session key. WS-Security and WS-SecureConversation have additional security headers in the messages for timestamps, digital signatures and encrypted keys.

2.2 Metamodel for web services

Since almost every web service framework uses a different kind of configuration format it is hard to match the options of the different frameworks. This challenge opens a new direction of research for creating a user friendly domain specific language (DSL) for the WS-* standard family in order to be able to model the services in a platform independent way. From this model the platform dependent configurations of the various frameworks can be automatically generated. In the context of the dissertation, a platform is a SOA product with a web service framework. The domain specific language introduced in the dissertation is called SOAL (Service-Oriented Architecture Language).

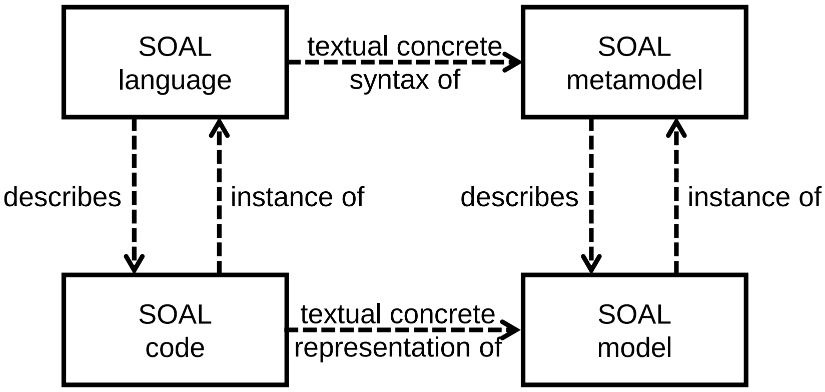


Figure 2.2: SOAL metamodel and language

SOAL is a technical DSL, since it targets developers, who need to construct web services between different web service frameworks. Most developers prefer textual concrete syntax instead of graphical concrete syntax [ZS09] [Voe08], therefore, SOAL has a textual concrete syntax described by the SOAL programming language. The abstract syntax is described by the SOAL metamodel. A SOAL program code is an instance of the SOAL programming language, and it is compiled to a SOAL model, which is captured by the SOAL metamodel (see figure 2.2).

The SOAL framework is built around the SOAL metamodel and language. The framework consists of a compiler and a generator, see figure 2.3. The compiler compiles a SOAL code into a SOAL model. The generator produces standard XSD and WSDL files, and also source files, configuration files and projects for the various web service frameworks. The currently supported frameworks are Windows Communication Foundation for Microsoft .NET with Visual Studio and IIS, Metro with GlassFish server and Netbeans, Apache CXF with JBoss Application Server and Eclipse, Apache CXF for Tomcat Server and Eclipse, IBM WebSphere with Rational Application Developer, and Oracle WebLogic with JDeveloper. The primary goal of the SOAL framework

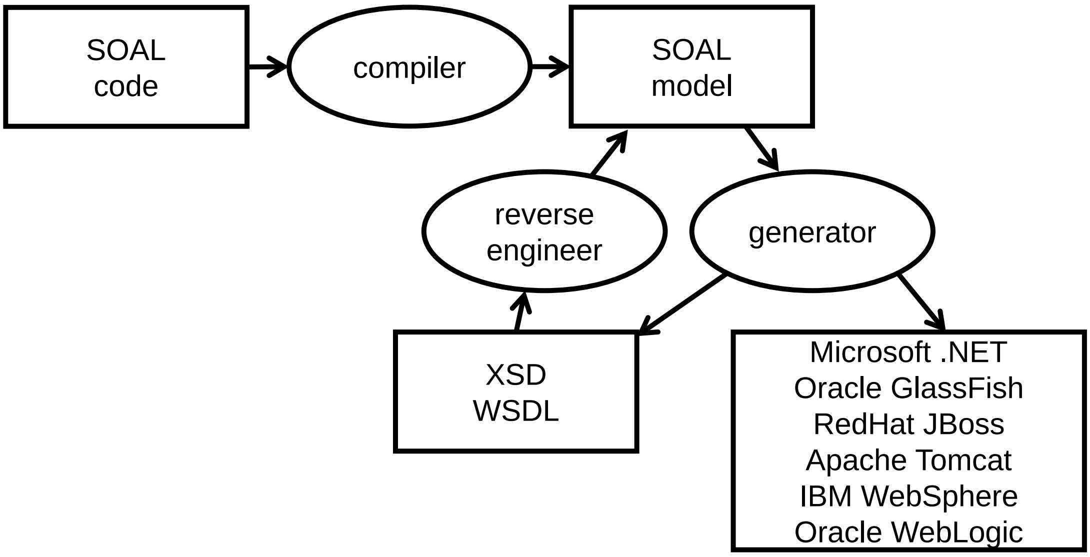


Figure 2.3: Architecture of the SOAL framework

is to increase the productivity of top-down development of web services, therefore, the XSD and WSDL files are generated from the SOAL model.

The framework also supports reverse engineering of existing XSD and WSDL files into SOAL models. This increases productivity for already existing services with given WSDLs, since these services can be easily migrated between different web service frameworks, primarily between different JAX-WS implementations of different vendors.

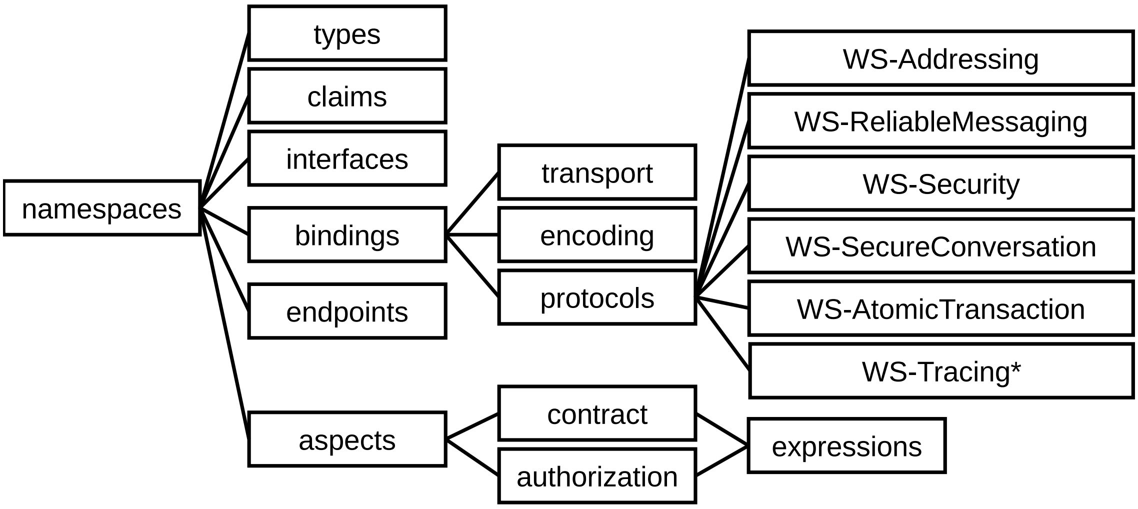


Figure 2.4: Architecture of the SOAL metamodel

Figure 2.4 shows the architecture of the SOAL metamodel. Most of the WSDL parts are mapped to the SOAL metamodel. The only exception is the message part which is a redundant element in the WSDL description; therefore, it is omitted from the SOAL metamodel. All the other parts (types, portType, bindings and endpoints) are included in SOAL and there is an additional declaration for modeling claims to support claims-based identity. Bindings define the transport, the encoding and the WS-* protocols to be used. The supported protocols are WS-Addressing, WS-ReliableMessaging, WS-Security, WS-SecureConversation, WS-AtomicTransaction. WS-Tracing is not a standard protocol. I created it to be able to trace web service calls, which is needed for my proposal for the business process instance migration. There are two aspects that can be defined in SOAL: a contract aspect for pre- and post-conditions, and an authorization aspect for access rights used in claims-based identity solutions. Both of them use expressions to describe the required conditions.

The new scientific results in web service modeling can be phrased as:

Thesis I. *I defined a new domain specific metamodel called SOAL for describing distributed systems of web services with WS-* protocols in a platform independent way. I defined a new programming language which serves as a textual concrete syntax for the metamodel. The SOAL metamodel and SOAL programming language provide a simpler and more user friendly description of web services at a more appropriate abstraction level and in a more productive way than the previous approaches. I specified a transformation which maps SOAL models to interoperable program codes and configuration files for the most important web service stack implementations in the industry. I verified the correctness of the transformation with interoperability tests. I implemented the metamodel, the language compiler and the transformation in .NET.*

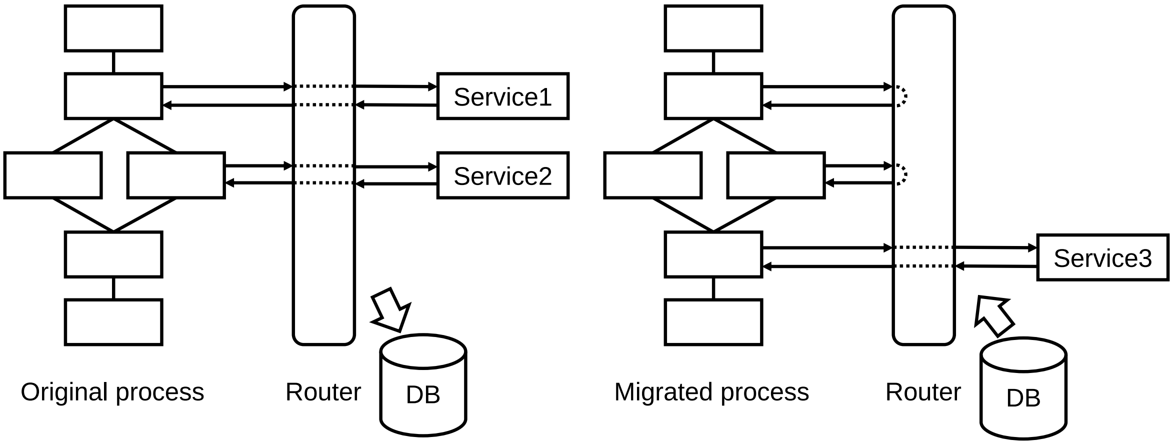
Publications related to this thesis: [1] [2] [4] [5] [6] [7] [8] [9] [10] [11] [15]

2.3 Business process instance migration

Many business processes are long running processes, since they usually require human activities. Such processes may run for days or even weeks in e-Governmental environments. When a new version of the underlying process execution engine is published, it is usually desirable to migrate the running process instances to the new version; however, the engine's software vendor may not provide any support to do this. An even greater risk of using a specific process engine is vendor lock-in. There are no converters that could migrate process instance states between different vendors' engines; hence, moving process instances to another engine is a complex task.

The direct migration of the product specific inner state representation of BPEL processes is a complicated and unmaintainable task, since all the desired process engines would have to be examined and even all their different versions would have to be taken into account.

Therefore, my proposition is that the process state should be restored from the outside in a non-intrusive way. The process engine is handled as a black box, but the processes have to be prepared with a special design pattern by including identifiers about the sender and the receiver in the messages exchanged by the processes, and also non-deterministic execution must be handled with care. If the proposed patterns are followed, the instances of a process can be migrated as follows. At first, the calls made by the original process and the responses to those calls sent by the outer world are recorded by a message router (see figure 2.5a). The router normally does not interfere with the communication.



(a) Original process instance

(b) Migrated process instance

Figure 2.5: Running of the original and the migrated process instance

When the instance migration of the running processes is required, the processes are deployed to the new process engine, and a new process instance is started for each of the original process instances. The new process instances are not communicating with the outside world until they have reached the same state as of their original process instances. Instead, the calls made by them are captured, and the original replies are replayed to them by the router (see figure 2.5b). This way, the new processes behave exactly as the original ones and can easily reach the same state. This method works between different versions of the same process engine and even between products of different vendors. It is also a non-intrusive solution, since the modification of the process engines is not required.

The only requirement for this solution to work is that the processes should be carefully engineered. In the dissertation I propose a design pattern for such processes and I also propose a framework that supports recording and replaying SOAP messages to support instance migration of business processes.

The new scientific results in BPEL process instance migration can be phrased as:

Thesis II. *I defined the architecture of a new BPEL process instance migration framework which is based on recording and replaying SOAP messages. Unlike previous approaches, the framework is non-intrusive and it is process engine independent. I defined the concept of migration compatibility for the correct migration of BPEL processes. I defined a new protocol called WS-Tracing for correlating SOAP messages with BPEL process instances. I defined a novel routing algorithm to perform the process instance migration within the proposed framework. I showed that if the process descriptions fulfill the constraints of migration compatibility, then the routing algorithm performs correct migration. I implemented the migration framework in .NET and verified the correctness of the implementation with real-life examples.*

Publications related to this thesis: [12] [13] [14]

2.4 Performance prediction for web services

The major performance overhead of web services results from the use of XML for serializing messages. The WS-* protocols extend simple SOAP messages with additional headers, bootstrap messages, and even XML encryption and digital signatures can burden the communication further. In the dissertation I propose a performance model for web services in order to be able to predict the response time overhead of web service calls for services with arbitrary interfaces. The coefficients of the performance model have to be calculated in advance so that the performance model can be used for prediction.

I also propose a method to calculate these coefficients, see figure 2.6. This method requires the execution of some measurements between the various frameworks by using a predefined set of services and clients. These are called reference services and reference clients. The interface of the reference services defines operations, where the input and output parameters are arrays of structures containing fields of various types. These types are the most common primitive types used in programming languages. The reference services differ only in the enabled WS-* protocols. All the reference services and all their respective clients have to be implemented in every web service framework. Then all the clients have to call all their respective services, even between different frameworks, and the response times have to be measured. The measurements have to be performed with different array lengths and with different number of calls.

My measurements show that different environments have the same characteristics: the response time is in linear correlation with the array length and also with the num-

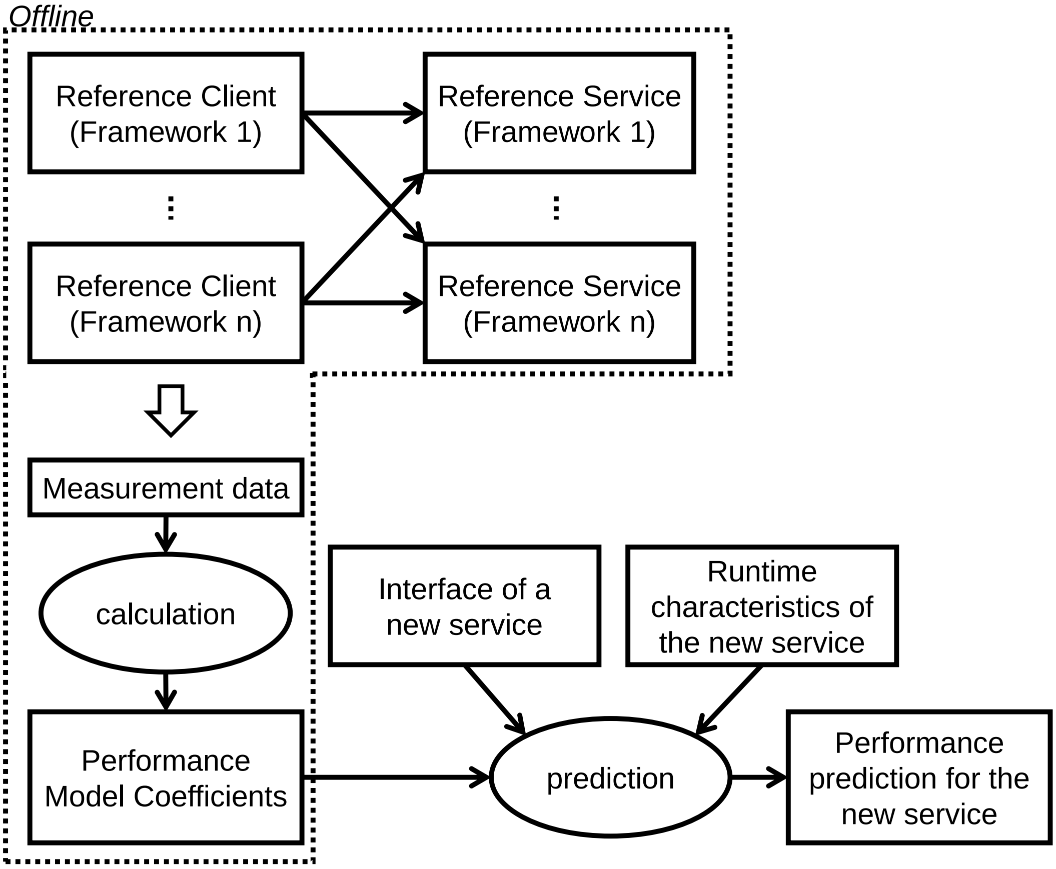

Offline
Reference Client (Framework 1)
Reference Service (Framework 1)
Reference Client (Framework n)
Reference Service (Framework n)
Measurement data
calculation
Performance Model Coefficients
Interface of a new service
Runtime characteristics of the new service
prediction
Performance prediction for the new service

Figure 2.6: Performance prediction process overview

ber of calls. Hence, a common performance model can be developed for the different environments. If this performance model is used, the environments only differ in the coefficients of the model. These coefficients can be calculated from the measurements performed between the reference clients and reference services.

The response time overhead of a service with an arbitrary interface can be predicted based on the model if the model's coefficients and the runtime characteristics of the service are known (e.g. the number of calls, the lengths of the arrays).

Figure 2.7 shows how the performance model was constructed, how it was evaluated and how it can be used for prediction. At first, I made measurements on the reference hardware using the reference services and reference clients. By observing the results of the reference measurements I constructed a performance model which can be used to make predictions for the response time overhead of web service calls. I evaluated the performance model on the reference hardware by making predictions for some new and previously unmeasured services and then comparing the actual response times with the predictions. For other kinds of hardware and for other services the prediction can be made as follows. At first, the reference measurements have to be

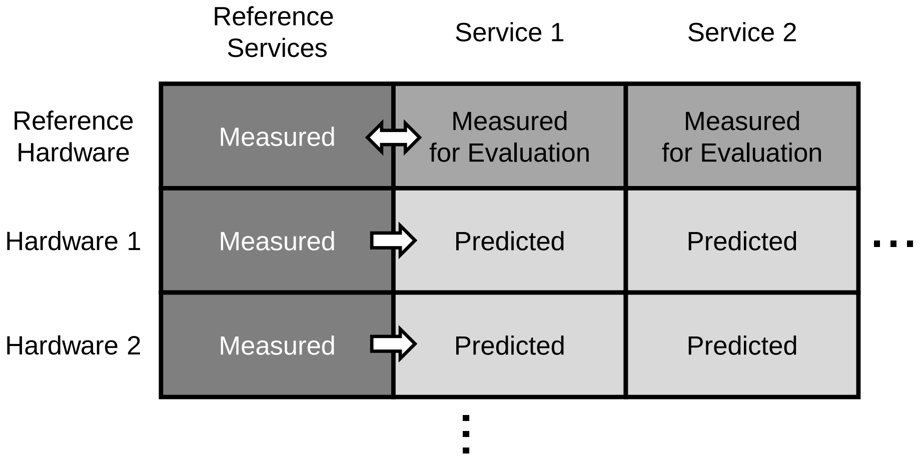


Figure 2.7: Performance prediction for different kinds of hardware and for different services

made on the new hardware using the reference services and reference clients. Then, the coefficients of the performance model specific to this new hardware have to be calculated from these measurements. Finally, the response time overhead of arbitrary services can be predicted on the new hardware, if these coefficients are known. I also evaluated this methodology on a hardware different from the reference hardware.

The new scientific results in web service performance prediction can be phrased as:

Thesis III. *I developed a new performance model for predicting the response time overhead of web service calls based on the interface description of the services. The prediction requires the knowledge of the coefficients of the performance model for the execution environment. I defined a set of reference measurements to calculate these coefficients. I calculated the coefficients for a selected hardware performing the reference measurements with two different web service stack frameworks. I verified the correctness of the performance model with independent measurements on a different hardware and on previously unmeasured web services. The performance model is also in correlation with the theoretical considerations of the WS-* protocols. Unlike previous approaches, the proposed performance model incorporates the most common primitive and complex types and also the most common WS-* protocols, and it can also be used in the design phase, when the services are not yet implemented.*

Publications related to this thesis: [3] [16]

3

Application of the Scientific Results

The scientific results discussed in the dissertation were also used in real-life projects. The SOAL modeling framework proved to be the most useful, it was utilized in three different projects.

The Hungarian governments are determined to migrate public administration processes to their electronic versions, therefore, the project for the Hungarian e-Government Infrastructure has been established. The Hungarian government agencies have different operating systems, different development frameworks from different software vendors, and they want to preserve their autonomy. The integration of these agencies could be made easier by developing a framework that supports the creation of interoperable components for different SOA products. The SOAL framework was applied in a pilot project for the Hungarian e-Government Infrastructure, where a real-life public administration process (foundation of a private entrepreneurship) had to be implemented using web services and BPEL. The pilot system included the simulation of four government agencies. There were seven web services, a BPEL process and a web site in the pilot system. At first, the implementation of the process and the services was done manually, and it took about one month to complete. The detailed description of the pilot system and the results were published in [9]. After the SOAL framework was ready, we reimplemented all the services by specifying them in SOAL at first, and then we generated WSDL files, program codes and configuration files for the various SOA products we had to use. In this second round we also included the test system of the real Hungarian Electronic Governmental Portal in the pilot system. This second approach took only three days, which shows, that the SOAL framework can greatly increase productivity in the development of distributed SOA systems.

In the second phase of this pilot project we also tested the prototype of the proposed business process migration framework on two real public administration processes: the foundation of a private entrepreneurship, and the request for financial parenting support. The process instances were migrated successfully between ActiveVOS, Oracle BPEL Process Manager and IBM WebSphere Process Server.

The SOAL framework was also useful in a project with the Hungarian Police Department. In this project we had to implement an administration system for storing the data of people who are banned from sport events or sport facilities because of their ill behavior. The organizers of the sport events have to check each person on entry whether they are allowed to enter the event or the facility. These requests are made online through a web service. The interface and the skeleton of the implementation of this service along with the appropriate configuration files were generated by the SOAL framework.

The third and most important application of the SOAL framework was a project with the Hungarian Railways. We wrote the software of a new ticket vending machine. The machine communicates with an online ticketing system and with a central maintenance system. The interfaces of both systems were designed in SOAL, and the skeleton of the implementation and the configuration files for .NET and Java were generated from the SOAL description. In the design phase the web service interface describing the protocol of the online ticketing system was changing weekly for about three months. Using the SOAL framework these changes could be made easily and very quickly, and also the different versions of the service interfaces could be maintained easily. Also the contracts defining pre- and post-conditions proved to be useful for checking the results coming from the ticketing system which was also still under development. The SOAL framework proved to be very productive and spared us a lot of work in this project.

The proposed performance model for predicting the response time overhead of web service calls is a recent result, and thus it has not yet been used in industry projects.

I also used the SOAL framework to create WSDL files and program codes for demonstration purposes for my lectures. I could easily create laboratory exercises for web services and BPEL processes with the help of the framework, too. Hence, the SOAL framework proved to be useful also in education.

Publications

Journal papers (3)

[1] B. Simon, B. Goldschmidt, and K. Kondorosi, "A metamodel for the web services standards," *Journal of Grid Computing*, vol. 11, no. 4, pp. 735–752, 2013.

[2] B. Simon, B. Goldschmidt, and K. Kondorosi, "A platform independent access control metamodel for web services," *Periodica Polytechnica Electrical Engineering and Computer Science*, vol. 58, no. 3, pp. 93–108, 2014.

[3] B. Simon, B. Goldschmidt, and K. Kondorosi, "A performance model for the web service protocol stacks (accepted paper)," *Services Computing, IEEE Transactions on*, vol. PP, pp. –, 2014.

International conference papers (11)

[4] B. Simon, B. Goldschmidt, P. Budai, I. Hartung, K. Kondorosi, Z. Laszlo, and P. Risztics, "Generic contract descriptions for web services implementations," in *ICDS 2011, The Fifth International Conference on Digital Society*, pp. 51–56, 2011.

[5] B. Simon, B. Goldschmidt, P. Budai, I. Hartung, K. Kondorosi, Z. Laszlo, and P. Risztics, "A metamodel of the ws-policy standard family," in *ICDS 2011, The Fifth International Conference on Digital Society*, pp. 57–62, 2011.

[6] B. Simon, Z. László, B. Goldschmidt, K. Kondorosi, and P. Risztics, "Evaluation of ws-* standards based interoperability of soa products for the hungarian e-government infrastructure," in *Digital Society, 2010. ICDS'10. Fourth International Conference on*, pp. 118–123, IEEE, 2010.

Number of independent citations: 7

[7] B. Simon and B. Goldschmidt, "A human readable platform independent domain specific language for wsdl," in *Networked Digital Technologies*, pp. 529–536, Springer Berlin Heidelberg, 2010.

[8] B. Simon, B. Goldschmidt, and K. Kondorosi, "A human readable platform independent domain specific language for bpel," in *Networked Digital Technologies*, pp. 537–544, Springer Berlin Heidelberg, 2010.

Number of independent citations: 1

[9] B. Simon, Z. László, and B. Goldschmidt, "Soa interoperability, a case study," *Proceedings of the IADIS International Conference, Informatics*, pp. 131–138, 2008.

Number of independent citations: 1

[10] B. Simon and Z. László, "A framework for traversing models using loops," in *Proceedings of the IADIS International Conference on Applied Computing 2007*, pp. 17–20, 2007.

[11] Z. László and B. Simon, "Model-based code generation with tree construction," *Proceedings on the IADIS International Conference on Applied Computing*, pp. 498–502, 2006.

[12] B. Simon, B. Goldschmidt, and S. Imre, "Engineering instance-migratable bpel business processes," in *Proceedings of the 15th International Multiconference INFORMATION SOCIETY – IS 2012*, pp. 233–236, Narodna in Univerzitetna Knjiznica, Ljubljana, Slovenia, 2012.

[13] B. Simon, B. Goldschmidt, and Z. László, "Bpel movie framework: Replaying bpel processes from logs," *Proceedings of the 13th IASTED International Conference, Software Engineering and Applications*, pp. 242–249, 2009.

[14] V. Müller, B. Simon, Z. László, and B. Goldschmidt, "Business monitoring solution for an e-government framework based on soa architecture," *Proceedings of the IASTED International Conference*, vol. 669, no. 039, p. 234, 2009.

Number of independent citations: 3

Hungarian conference papers (2)

[15] B. Simon, Z. Laszlo, and B. Goldschmidt, "Soa mintarendszer kialakításának tapasztalatai," in *Networkshop 2008, Dunaújváros, Magyarország*, pp. 1–8, 2008.

[16] B. Simon, B. Goldschmidt, and K. Kondorosi, "Webszolgáltatások kommunikációs overhead-jének becslése," in *Networkshop 2012, Veszprém, Magyarország*, pp. 1–8, 2012.

References

[AdL08] Rubén González Antolín and Juan de Lara. A Model-Driven Approach to Service Performance Prediction and Analysis. In *Proceedings of 6th Workshop on System Testing and Validation*, STV08. Fraunhofer IRB, 2008.

[AGM08] Danilo Ardagna, Carlo Ghezzi, and Raffaela Mirandola. Model Driven QoS Analyses of Composed Web Services. In *Proceedings of the 1st European Conference on Towards a Service-Based Internet*, ServiceWave '08, pages 299–311, Berlin, Heidelberg, 2008. Springer-Verlag.

[CAB08] Mohamed Amine Chaâbane, Eric Andonoff, and Rafik Bouaziz. Dealing with business process evolution using versions. In *ICE-B*, pages 267–278, 2008.

[CBZ$^+$04] Fei Cao, Barrett R. Bryant, Wei Zhao, Carol C. Burt, Rajeev R. Raje, Andrew M. Olson, and Mikhail Auguston. A meta-modeling approach to web services. In *Proceedings of the IEEE International Conference on Web Services*, ICWS '04, pages 796–, Washington, DC, USA, 2004. IEEE Computer Society.

[CMV04] V. De Castro, E. Marcos, and B. Vela. Representing wsdl with extended uml, 2004.

[DNsmGW08] C. Dumez, A. Nait-sidi moh, J. Gaber, and M. Wack. Modeling and specification of web services composition using uml-s. In *Next Generation Web Services Practices, 2008. NWESP '08. 4th International Conference on*, 2008.

[DVRS03] Paulo Dias, Pedro Vieira, and António Rito-Silva. Dynamic evolution in workflow management systems. In *Database and Expert Systems Applications, 2003. Proceedings. 14th International Workshop on*, pages 254–260. IEEE, 2003.

[EES10] A. Elgammal and M. El-Sharkawi. Using uml to model web services for automatic composition. Open Access publications from Tilburg University urn:nbn:nl:ui:12-4561873, Tilburg University, 2010.

[EKW+09] Thomas Erl, Anish Karmarkar, Priscilla Walmsley, Hugo Haas, L Umit Yalcinalp, Kevin Liu, David Orchard, Andre Tost, and James Pasley. *Web service contract design and versioning for SOA*. Prentice Hall, 2009.

[ESBS07] Julian Eckert, Stefan Schulte, Rainer Berbner, and Ralf Steinmetz. An Approach for Capacity Planning of Web Service Workflows. In *13th Americas Conference on Information Systems*, AMCIS 2007, 2007.

[ESS12] ESSI WSMO working group. *Web Service Modeling Ontology (WSMO)*. `http://www.wsmo.org/`, accessed: 29.01.2012.

[FM14] Greg Flurry and Manish Modh. *Web services development patterns*. `http://www.ibm.com/developerworks/websphere/library/techarticles/0511_flurry/0511_flurry.html`, accessed: July 13, 2014.

[GSSO04] Roy Gronmo, David Skogan, Ida Solheim, and Jon Oldevik. Model-driven web services development. In *The 2004 IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE-04)*, 2004.

[IKLC10] G. Imre, M. Kaszó, T. Levendovszky, and H. Charaf. A Novel Cost Model of XML Serialization. *Electron. Notes Theor. Comput. Sci.*, 261:147–162, February 2010.

[JB08] Harshavardhan Jegadeesan and Sundar Balasubramaniam. An mof2-based services metamodel. *Journal of Object Technology*, 7(8):71–96, 2008.

[JRB+06] Matjaz B. Juric, Ivan Rozman, Bostjan Brumen, Matjaz Colnaric, and Marjan Hericko. Comparison of performance of Web services, WS-Security, RMI, and RMI-SSL. *J. Syst. Softw.*, 79(5):689–700, May 2006.

[JRH00] Matjaz B Juric, Ivan Rozman, and Marjan Hericko. Performance comparison of CORBA and RMI. *Information and Software Technology*, 42(13):915 – 933, 2000.

[KG99] Markus Kradolfer and Andreas Geppert. Dynamic workflow schema evolution based on workflow type versioning and workflow migration. In *Cooperative Information Systems, 1999. CoopIS'99. Proceedings. 1999 IFCIS International Conference on*, pages 104–114. IEEE, 1999.

[KKK08] Jong Myoung Ko, Chang Ouk Kim, and Ick-Hyun Kwon. Quality-of-service oriented web service composition algorithm and planning architecture. *Journal of Systems and Software*, 81(11):2079 – 2090, 2008.

[Klu08] M. Klusch. *CASCOM - Intelligent Service Coordination in the Semantic Web*, chapter 3. Birkhäuser Verlag, Springer, 2008.

[KPKH05] Vladimir Kolovski, Bijan Parsia, Yarden Katz, and James Hendler. Representing web service policies in owl-dl. In *In International Semantic Web Conference (ISWC*, pages 6–10, 2005.

[Mic14] Microsoft. *Windows Communication Foundation*. `http://msdn.microsoft.com/en-us/library/dd456779%28v=vs.110%29.aspx`, accessed: May 11, 2014.

[MM07] Moreno Marzolla and Raffaela Mirandola. Performance prediction of web service workflows. In *Proceedings of the Quality of software architectures 3rd international conference on Software architectures, components, and applications*, QoSA'07, pages 127–144, Berlin, Heidelberg, 2007. Springer-Verlag.

[NSP+10] Marc Novakouski, Soumya Simanta, Gunnar Peterson, Ed Morris, and Grace Lewis. *Performance Analysis of WS-Security Mechanisms in SOAP-Based Web Services*. http://www.sei.cmu.edu/reports/10tr023.pdf, November 2010.

[OAS06] OASIS. *Reference Model for Service Oriented Architecture 1.0*. `http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.pdf`, October 12, 2006.

[OMG09] OMG. *Service oriented architecture Modeling Language (SoaML)*. `http://www.omg.org/spec/SoaML/Current`, 2009.

[Ora14] Oracle. *Metro*. `http://metro.java.net/`, accessed: May 11, 2014.

[PSSvdA07] M. Pesic, M.H. Schonenberg, N. Sidorova, and W.M.P. van der Aalst. Constraint-based workflow models: Change made easy. In Robert Meersman and Zahir Tari, editors, *On the Move to Meaningful Internet Systems 2007: CoopIS, DOA, ODBASE, GADA, and IS*, volume 4803 of *Lecture Notes in Computer Science*, pages 77–94. Springer Berlin Heidelberg, 2007.

[PWH08] Baba Piprani, Chong Wang, and Keqing He. A metamodel for enabling a service oriented architecture. In *Proceedings of the OTM Confederated International Workshops and Posters on On the Move to Meaningful Internet Systems: 2008 Workshops: ADI, AWeSoMe, COMBEK, EI2N, IWSSA, MONET, OnToContent + QSI, ORM, PerSys, RDDS, SEMELS, and SWWS*, pages 668–677, Berlin, Heidelberg, 2008. Springer-Verlag.

[QN10] Xhevi Qafmolla and Viet Cuong Nguyen. Automation of web services development using model driven techniques. In *Computer and Automation Engineering (ICCAE), 2010 The 2nd International Conference on*, volume 3, pages 190 –194, feb. 2010.

[REB11] Douglas Rodrigues, Julio C. Estrella, and Kalinka R. L. J. C. Branco. Analysis of Security and Performance Aspects in Service-Oriented Architectures. *International Journal of Security and Its Applications*, 5(1):13–30, January 2011.

[RMR10] S. Rinderle-Ma and M. Reichert. Advanced migration strategies for adaptive process management systems. In *Commerce and Enterprise Computing (CEC), 2010 IEEE 12th Conference on*, pages 56–63, Nov 2010.

[RRKD05] MU Reichert, SB Rinderle, Ulrich Kreher, and Peter Dadam. Adaptive process management with adept2 (tool demo). 2005.

[Som11] Ian Sommerville. *Software Engineering*. International Computer Science Series. Pearson, 2011.

[SSFG04] Satoshi Shirasuna, Aleksander Slominski, Liang Fang, and Dennis Gannon. Performance Comparison of Security Mechanisms for Grid Services. In *Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing*, GRID '04, pages 360–364, Washington, DC, USA, 2004. IEEE Computer Society.

[SSVS04] Natenapa Sriharee, Twittie Senivongse, Kunal Verma, and Amit Sheth. On using ws-policy, ontology, and rule reasoning to discover web services. In Finn Arve Aagesen, Chutiporn Anutariya, and Vilas Wuwongse, editors, *Intelligence in Communication Systems*, volume 3283 of *Lecture Notes in Computer Science*, pages 246–255. Springer Berlin / Heidelberg, 2004. 10.1007/978-3-540-30179-0_22.

[VAG05] Kunal Verma, Rama Akkiraju, and Richard Goodwin. R.: Semantic matching of web service policies. In *Proceedings of the Second Workshop on SDWP, 2005*, pages 79–90, 2005.

[vEZ08] Robert A. van Engelen and Wei Zhang. An Overview and Evaluation of Web Services Security Performance Optimizations. In *Proceedings of the 2008 IEEE International Conference on Web Services*, ICWS '08, pages 137–144, Washington, DC, USA, 2008. IEEE Computer Society.

[Voe08] Markus Voelter. A family of languages for architecture description. In *in 8th OOPSLA Workshop on Domain-Specific Modeling (DSM'08)*, 2008.

[W3C12a] W3C. *OWL-S: Semantic Markup for Web Services*. `http://www.w3.org/Submission/OWL-S/`, accessed: 29.01.2012.

[W3C12b] W3C. *Semantic Annotations for WSDL and XML Schema (SAWSDL)*. http://www.w3.org/TR/sawsdl/, accessed: 29.01.2012.

[WSO06] Hiroshi Wada, Junichi Suzuki, and Katsuya Oba. A model-driven development framework for non-functional aspects in service oriented grids. In *Proceedings of the International Conference on Autonomic and Autonomous Systems*, ICAS '06, pages 30–, Washington, DC, USA, 2006. IEEE Computer Society.

[WSO08] Hiroshi Wada, Junichi Suzuki, and Katsuya Oba. A model-driven development framework for non-functional aspects in service oriented architecture. *International Journal of Web Services Research (IJWSR)*, 5(4):1–31, 2008.

[ZHKL10] Sonja Zaplata, Kristof Hamann, Kristian Kottke, and Winfried Lamersdorf. Flexible execution of distributed business processes based on process instance migration. *Journal of Systems Integration*, 1(3):3–16, 2010.

[ZS09] Uwe Zdun and Mark Strembeck. Reusable architectural decisions for dsl design. In *Proceedings of the 14th Annual European Conference on Pattern Languages of Programming (EuroPLoP)*, Aachen, Germany, 2009.