Citace:
BUCHALCEVOVÁ, Alena. Where in the curriculum is the right place for teaching agile methods? Prague 20.08.2008 – 22.08.2008. In: Proceedings 6th ACIS International Conference on Software Engineering Research, Management & Applications (SERA 2008). Prague : Copyright, 2008, s. 205–209. ISBN 978-0-7695-3302-5.

Get ready for SOA

Alena Buchalcevová, Roman Hauptvogl

University of Economics Prague, Czech Republic, Department of Information Technology
buchalc@vse.cz
Cleverlance Enterprise Solutions a.s, Czech Republic
romi74371@gmail.com

Abstrakt
Service-oriented architecture (SOA) is nowadays a hot topic in an enterprise development. Building SOA is a very complex problem that needs qualified people. There are only few courses at universities and therefore the main pressure nowadays is on SW development and SW vendors companies as well as training companies to train skilled developers. In this paper we aim to present how to prepare Web services courses.

Keywords
Service-oriented architecture, SOA, Web services, course

1 Introduction

Service-oriented architecture (SOA) is nowadays a hot topic in an enterprise development. Many IT professionals see the potential of SOA in effective business integration that delivers flexibility by allowing business resources (both inside and outside an organization) to work together to support a company's business strategy. These resources can include data, applications, processes, and people. One way to begin to achieve integration throughout a business is by providing the ability for the business systems to work together. In other words, there is a need to integrate the software and applications. To do this, we need technologies that can interoperate easily, standards that specify consistent styles of interaction and communication and especially understand that business integration is not just a technical problem but a business problem. To truly solve it, the IT development and business management sides of the organization need to work hand in hand. For many organizations that may require a change in the way they think and work.

SOA is an architectural style for building distributed systems that delivers application functionality as services to end-user applications or other services. A service is an application function packaged as a reusable component that can be used in a business process. Service exposes a well defined interface and has no dependencies on the state of other services. SOA is usually realized through standards-based web services. A web service is a service that communicates with clients through a set of standard protocols and technologies. Web services standards are implemented in platforms and products from all the major software vendors, making it possible for clients and services to communicate in a consistent way across a wide spectrum of platforms and operating environments.

Another SOA potential, especially a web services based, is seen in dramatically shorter application development process and more adaptable applications. Not only is SOA a hot topic, but it's clearly the wave of the future. Gartner reports that "By 2008, SOA will be a prevailing software engineering practice, ending the 40-year domination of monolithic software architecture" and that "Through 2008, SOA and web services will be implemented together in more than 75 percent of new SOA or web services projects."

Building SOA is a very complex problem that needs qualified people. There are only few courses at universities and therefore the main pressure nowadays is on SW development and SW vendors companies as well as training companies to train skilled developers. In this paper we aim to present how to prepare Web services courses.

2 Web Services Courses

Although web services are designed to be language and platform neutral, the Java programming language is ideal for developing web services and applications that use web services. The Java applications portability and interoperability mesh well with the objective of web service interoperability. Therefore, in this paper, we concentrate on Web services development in Java courses.

The Web Services courses are usually intended for developers who are already familiar with J2EE application development and have some basic knowledge about Web service technologies. In these courses students learn the technologies involved in developing, deploying and securing Web services in-detail. Generally, the courses are organized in three to five days. The length of the course depends as well on the platform on which it is presented. The general concepts and technologies courses are shorter than courses designed for a specific platform (for example BEA , WebSphere). Platform specific courses focus apart from common concepts and technologies as well on specific features of given platform, concrete wizards and differences. The Web services courses are design to give you enough knowledge to be able to:

- Develop, test, debug and deploy Web services,
- Describe the Service Oriented Architecture and supporting technologies, such as SOAP, WSDL and UDDI,
- Describe the motivations for synchronous and asynchronous Web services,
- Implement and invoke synchronous and asynchronous Web services,
- Develop and use custom data types,
- Create handlers and handler chains for pre- and post-processing of messages,
- Manipulate SOAP attachment,.
- Secure the Web services.

2.1 Structure of the Web services course

The structure of the course, no matter on which platform it is presented, is usually the same. The challenge is how to present complex concept of Service Oriented Architecture and Web services in a time of three to five days. The course usually begins with the introduction to distributed computing. Distributed computing enables processes and application in one location to access functionality (operation and data) available in another location, which is also one of the principles of Service Oriented Architecture. There is need to show the benefits of distributed computing and technologies which tried to address issues related to distributed computing. Next subject covers SOA and Web services definition and fundamentals of these concepts and technologies (see 2.2). The core of the course presents developing Web services according to two key J2EE specifications: JAX-RPC and Web services for J2EE. (see 2.2.2) Programming models for developing web services (J2EE Web Services Programming Model, Packaging and Deployment Model, Message Handlers and SOAP Programming Model) are there introduced in this part. The remaining subjects involve advanced features of Web services development such as: JMS support, Web services security and Web services interoperability.

2.2 SOA and Web Services Fundamentals

Implementing a SOA can involve developing applications that use services, making applications available as services or both. Some key aspects of the SOA are:

- to leverage open standards to represent software assets as services,
- to provide a standard way of representing and interacting with software assets,
- to allow individual software assets to become building blocks that can be reused in developing other applications,
- to shift focus to application assembly rather than implementation details,
- can be used internally to create new applications out of existing components,

- can be used externally to integrate with applications outside of the enterprise.

The key goal of a SOA is to achieve loose coupling. This means that services maintain a relationship that minimizes dependencies and only requires that they retain an awareness of each other. Another key principle, to achieve services reusability, is done by dividing logic into services with the intention of promoting reuse. The SOA vision of interaction between clients and loosely-coupled services demands widespread interoperability. In other words clients and services communicate and understand each other no matter what platform they run on. This objective can be met only if clients and services have a standard way of intercommunication. More about SOA and Web Service Fundamentals can be found in [1].

SOA uses the *find-bind-execute* paradigm as shows Figure 1. According to this paradigm, service providers register their service in a public registry. This registry is used by consumers to find services that match certain criteria. If the registry has such a service, it provides the consumer with a contract and an endpoint address for that service.

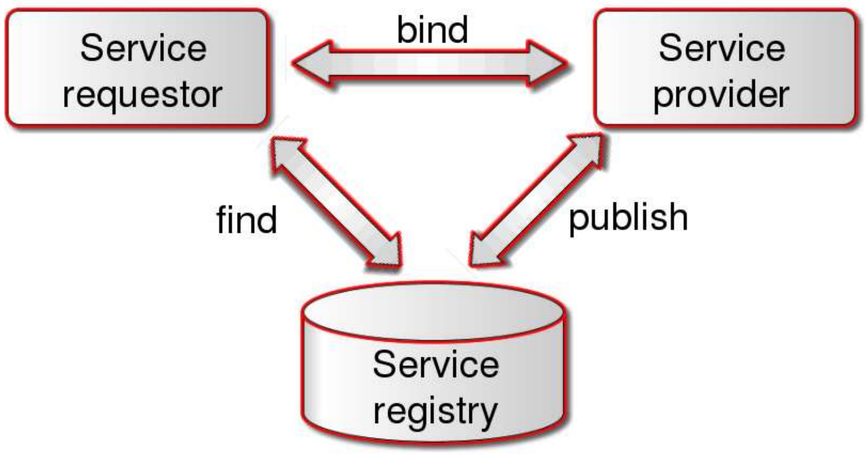

Figure 1: SOA components and operation

According to W3C Web services Architecture Group [4] we can define Web Service as “A software system identified by a URI, defined, described and discovered by XML artifacts, which interacts with other software systems using XML messages through Internet-based protocols”.

Web services comprise a maturing set of protocols and technologies that are widely accepted and used, and that are platform, system, and language independent. In addition, these protocols and technologies work across firewalls, making it easier for business partners to share vital services. Promising to make things even more consistent is the WS-I basic profile, introduced by the Web Services Interoperability Organization (an organization chartered to promote web services interoperability). The WS-I basic profile identifies a core set of web services technologies that when implemented in different platforms and systems, helps ensure that services on these different platforms and systems, and written in different languages, can communicate with each other. The WS-I basic profile has widespread backing in the computer industry, virtually guaranteeing interoperability of services that conform to the profile.

Simple Object Access Protocol (SOAP) is an XML-based protocol for exchanging information in a distributed environment. SOAP provides a common message format for exchanging data between clients and services. Complete description can be found in W3C Note [5].

Web Services Definition Language (WSDL) is an XML based language that provides a model for describing Web services. Version 1.1 has not been endorsed by the World Wide Web Consortium (W3C) [6]. Version 2.0, for which several drafts have been released, is expected to become a W3C recommendation. A WSDL document serves several purposes. It provides a model for implementing the actual Web service. It describes how a service requester connects to a service and what messages to send and receive from the service. It defines a XML schema for describing a web service and its operations, bindings and location of specific implementation. To uncover the description for a Web service, a client needs to find the service's WSDL document. A programmer uses the interface information in the WSDL document to construct the appropriate calls to the service.

Universal Description, Discovery and Integration (UDDI) is an open industry initiative, sponsored by OASIS, enabling businesses to publish service listings and discover each other and define how the services or software applications interact over the Internet. It defines the structure of the Web service "metadata" database, as well as an API for accessing the database. This API allows clients to discover, bind and invoke Web services at run time. Structure of the UDDI registry and its relationship with WSDL is shown on the Figure 2.

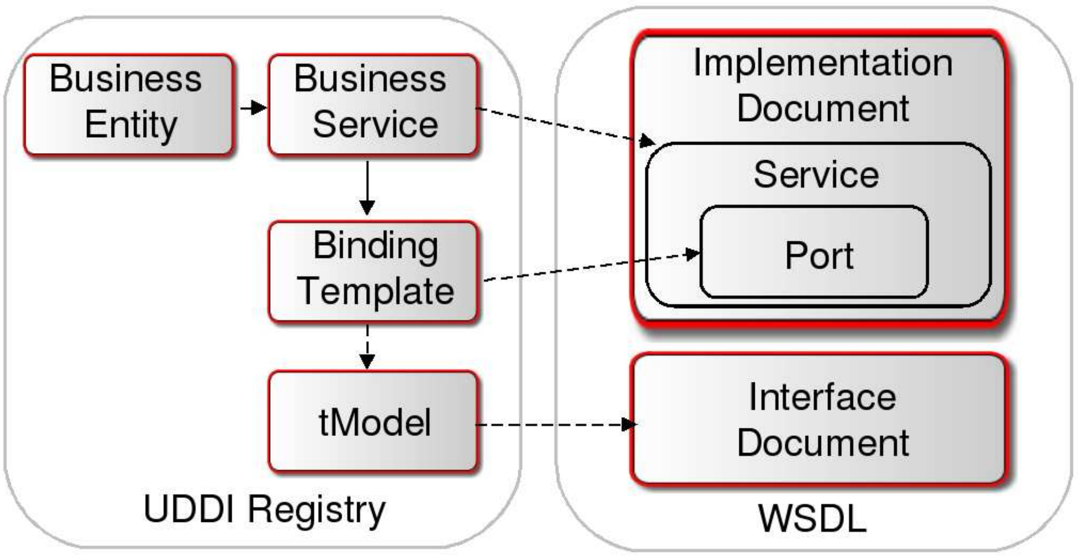

Figure 2: Relationship between UDDI and WSDL

In the UDDI Registry, business entities are service providers that offer one or more business services (Web services). A business service holds one or more binding templates, providing access information to the actual Web service. The access information is a URI to a WSDL port element. Binding templates also reference one or more tModels, metadata about type specification and service category. The tModel in turn holds a reference to a WSDL namespace.

2.3 Developing Web Services in Java

As we have meant earlier the Java programming language is ideal for developing web services and applications using them. Java technologies are designed for use with XML, and conform to web services standards such as SOAP, WSDL, and UDDI. Java Technology and Web Services are organized into these subcategories (more about these subcategories can be found on [12]):

- Java API for XML-Based RPC (JAX-RPC)
- Java API for XML Web Services (JAX-WS)
- Java API for XML Registries (JAXR)
- Java API for XML Processing (JAXP)
- Java Architecture for XML Binding (JAXB)
- SOAP with Attachments API for Java (SAAJ)
- XML Web Services Security (XWSS)
- XML Digital Signatures (XMLDSig)
- Java API for Web Services Addressing (JAX-WSA)

Developing Web services in Java involves two approaches, developing Web services according to JAX-RPC and Web services for J2EE. There are two basic specifications for these two approaches:

- JAX-RPC (JSR-101) [11]

  - Defines a standard programming model for Java Web service clients and endpoints.
  - Ensures that conforming Web service run-time environments provide the same level of functionality.
- Web Services for J2EE (JSR-109) [10]
  - Defines a structure and a packaging standard for Web services implemented as a J2EE application.

Java API for XML-Based RPC (JAX-RPC) is a Java API for accessing services through XML (SOAP-based) RPC calls. The API incorporates XML-based RPC functionality according to the SOAP 1.1 specification. JAX-RPC allows a Java-based client to call web service methods in a distributed environment, for example, where the client and the web service are on different systems. Although JAX-RPC is a Java API, it doesn't limit the client and the web service to both be deployed on a Java platform. A Java-based client can use JAX-RPC to make SOAP-based RPC calls to web service methods on a non-Java platform. A client on a non-Java platform can access methods in a JAX-RPC enabled web service on a Java platform.

JAX-RPC is designed to hide the complexity of SOAP. When there is JAX-RPC used to make an RPC call, there is no need to explicitly code a SOAP message. Instead there is the call coded in the Java programming language, using the Java API. JAX-RPC converts the RPC call to a SOAP message and then transports the SOAP message to the server. JAX-RPC on the server converts the SOAP message and then calls the web service. Then the sequence is reversed. The web service returns the response. JAX-RPC on the server converts the response to a SOAP message, which is then transported back to the client. JAX-RPC on the client converts the SOAP message and then returns the response to the application. This process is shown on

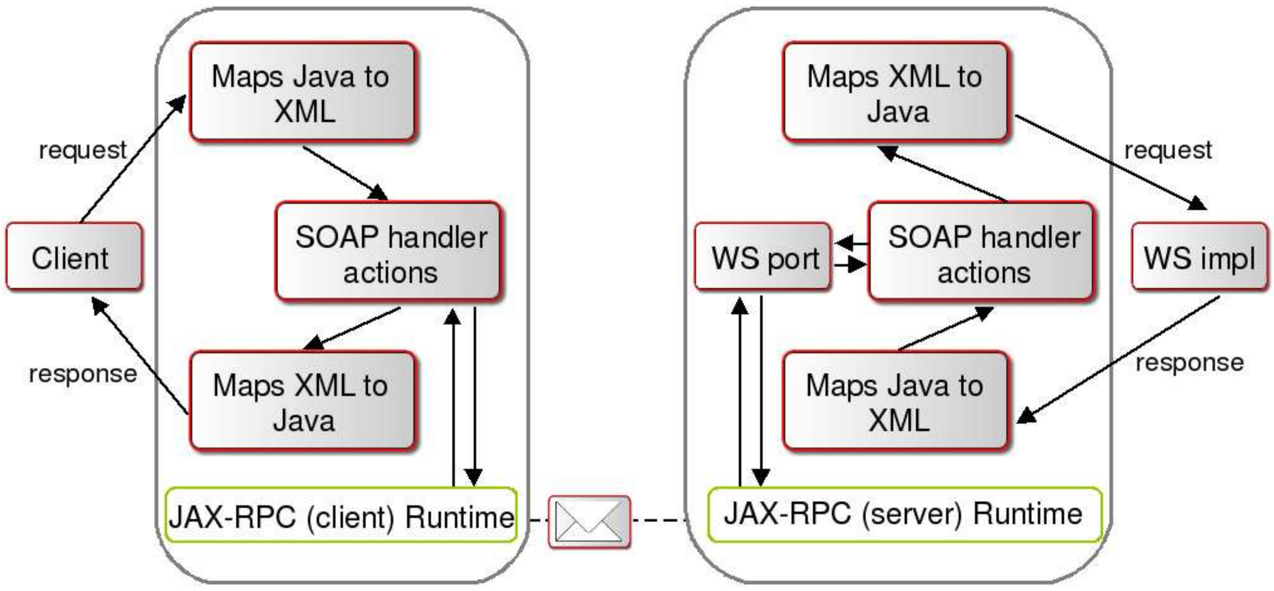

Figure 3: JAX-RPC
Figure 3.

2.3.1 Server Programming Model

A server view of a Web service is defined by a port or a port component. Port is created and managed by the container, which mediates access to the service implementation. Port consists of:

- Service Endpoint Interface (SEI) – declares Web service methods that can be invoked by clients.
- Service implementation bean – java class providing business logic for the service and provides implementation for methods declared in SEI.
- WSDL – provides a canonical description of the Web service.
- Security role references.

Service implementation is either a stateless session EJB or a JAX-RPC service endpoint deployed in a Web container. It implements methods of SEI which are described by WSDL. Regardless of the service implementation bean type, the class must follow these rules:

- provide default public constructor,
- implement all methods signatures in the SEI,
- be a stateless object,
- define methods that are public, must not be final or abstract,
- finalize method is not defined.

There are two different implementation approaches. First is called 'Bottom-up' and generates SEI and WSDL from the existing service implementation. Second is called 'Top-down' and generates SEI and service implementation from existing WSDL file

2.3.2 Client Programming Model

Clients can invoke Web service using:

- Static Stub – stub represents the client-side proxy to the remote Web service implementation,
- Dynamic Proxies – the JAX-RPC client environment creates dynamic proxies at run time,
- Dynamic Invocation Interface (DII) – *javax.xml.rpc.Call* interface allows clients to invoke Web service operation at run time.

There are two different types of clients. 'Managed' clients run in a J2EE container. 'Unmanaged' clients use the JAX-RPC run-time to invoke Web services. Such clients need to work directly with the generated Service interface to access stub, proxy or DII.

2.3.3 Message Handlers

Message handler is a mechanism for intercepting and acting upon SOAP requests and responses. It means that it allows SOAP message pre-processing and post-processing. They are available on both server and client side and are defined in JAX-RPC specification. A SOAP message handler gets access to the SOAP message that represents either an request or response. A typical use of a SOAP message handler is to process the SOAP header blocks as part of the processing of an request or response. SOAP headers are used to carry contextual data for a request – data that is only pertinent to the request and not necessarily to the business interface. A few typical examples of the usage of handlers are:

- Encryption and decryption handler,
- Logging and auditing handler,
- Caching handler.

SOAP message handlers are tied to web service endpoints (either on client or server) and are used to provide additional SOAP message processing facility as an extension to these components.

2.4 XML and Web Services Security

XML and Web Services Security (XWS Security) provides message-level security for applications that use JAX-RPC to access web services. In message-level security, security information is contained in the SOAP message header. One of the primary uses of message-level security is to secure a SOAP message from unauthorized access at intermediate nodes along the message path. Recall that a SOAP message can pass through a set of intermediate nodes as it travels from a client to a service, and that each node can independently process part or all of the message before forwarding it. Using message-level security, things like encrypt a SOAP message can be done and permit decryption can be available only by the target web service. For example, this could be used to protect credit card information from exposure until it's received by the target service In addition, XWS security gives the flexibility of securing different parts of a SOAP message and in different ways. Specifically, entire service, one or more service ports, or one or more service operations can be secured. For instance, some parts of the message and not other parts can be encrypted. Some parts of the message can be signed with a digital signature. In addition to including encryption information, the SOAP message header might include other security-related items such as an X.509 certificate or a security token. The SOAP header can also point to a repository of security information for the message. Issues that are addressed in a security framework:

- Authentication – the identity of the party should be validated based on some credentials,
- Authorization – the party should have the permissions to access the requested resource,
- Integrity – there should be a guarantee that information is not modified in transit,
- Confidentiality – only authorized actors or security token owners should be able to view the data,
- Non-repudiation – the sender and receiver are able to provide legal proof to a third-party that the sender sent the information and receiver received the information,
- Auditing – all of the transactions are recorded,
- Denial of service – An attack on a computer system causing loss of service.

More information about Web Services Security can be found on [13].

3 Environment and exercises

For general courses it is worth to use Java Web Service Development Pack (Java WSDP) from Sun Microsystems. The Java WSDP is an all-in-one download containing key technologies to simplify building of Web services using the Java 2 Platform. It provides tools you need to quickly build, test, and deploy web services, clients that interoperate with other web services and clients running on Java-based or non-Java-based platforms. In addition, it enables businesses to expose their existing J2EE applications as web services. This pack presents both specifications for developing Web services: JAX-RPC and JAX-WS. It is also worth to present some examples on a light web server for example Apache Tomcat, to show students that they do not need to have application server to work with Web services.

To gain some practical experience, it is always useful to supplement theoretical knowledge with practical exercises. Due to shortness of time these exercises are not done from scratch. On the contrary they present only specific issues of Web services technology and Web services developing, debugging and monitoring. Exercises can be workshop based or individual study. For each exercise we need to specify requirements, recommended solution strategy, time for completion. Good idea that proves its worth is to prepare two types of exercises: core and extra. Core exercises reinforce core concepts. Students should complete all of these exercises. They usually represent the logical process associated with building one complete solution. Extra exercises explain concepts beyond core knowledge.

It is preferred to have one introductory exercise, where instructor explain pieces of Web service and demonstrate the process of building a simple Web service. The objective of this demonstration is for students to become familiar with:

- the structure of the exercises,
- the platform and environment where is the development of Web services presented,
- the pieces that constitute a Web service.

Another proven idea is to present some examples of on-line Web services. Very useful site for this purpose is http://www.xmethods.com. This site provides a hosting service for Web services under development. Developers can upload their Web services and let other developers test their solutions. Each Web service provides a description of the services using WSDL where developers explore methods that they can call remotely.

At the end one last but not least note about the exercises. Since the Web service wizards can perform the majority of the work, sometimes there is no coding required, there is need to remind students to avoid the temptation of rapidly clicking through wizard menus, without understanding the meaning of the wizard settings.

4 Conclusion

Service-oriented architecture (SOA) becomes step by step prevailing software engineering practice. Building SOA is a very complex problem that needs qualified people. This paper makes an effort to present some ideas and experiences associated with preparing Web services courses.

References

[1] Thomas, E.: Service-Oriented Architecture (SOA): Concepts, Technology, and Design, The Prentice Hall, 2006, ISBN 0-13-185858-0

[2] Eric Armstrong, Jennifer Ball, Stephanie Bodoff, Debbie Bode Carson, Ian Evans, Dale Green, Kim Haase, Eric Jendrock: The J2EE 1.4 Tutorial, December 2005, http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html

[3] The Java Web Services Tutorial, February 2006, http://java.sun.com/webservices/docs/2.0/tutorial/doc/

[4] W3C Working Group Note: Web Services Architecture Requirements, February 2004, http://www.w3.org/TR/wsa-reqs

[5] W3C: Simple Object Access Protocol (SOAP) 1.1, May 2000, http://www.w3.org/TR/soap/

[6] W3C: Web Services Description Language (WSDL) 1.1, March 2001, http://www.w3.org/TR/wsdl

[7] W3C: Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language, March 2006, http://www.w3.org/TR/wsdl20/

[8] W3C Recommendation: XML Schema Part 1: Structures Second Edition, October 2004, http://www.w3.org/TR/xmlschema-1/

[9] W3C Recommendation: XML Schema Part 2: Datatypes Second Edition, October 2004, http://www.w3.org/TR/xmlschema-2/

[10] Specification Lead Dhiru Pandey: Implementing Enterprise Web Services, May 2006, http://www.jcp.org/en/jsr/detail?id=109

[11] Specification Lead Roberto Chinnici: Java™ APIs for XML based RPC, October 2003, http://www.jcp.org/en/jsr/detail?id=101

[12] Java Technology and Web Services: http://java.sun.com/webservices/

[13] OASIS Standard: Web Services Security: SOAP Message Security V1.0, March 2004, http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf

[14] W3C Recommendation: Canonical XML Version 1.0, March 2001, http://www.w3.org/TR/xml-c14n

[16] XML Signature Working Group: http://www.w3.org/Signature/

[17] XML Encryption Working Group: http://www.w3.org/Encryption/2001/