Using Service Orientation to Drive Business Processes

Arsalan Minhas, Prof. Dr. Friedrich H. Vogt
Telematics Department, Hamburg University of Technology, Germany
{arsalan.minhas, f.vogt}@tuhh.de

Abstract

This contribution proposes an approach that is necessary to model cross enterprise business processes using Service Oriented Modeling and its realization by leveraging standard Web Services protocols. It stresses the need for validation of produced models at every step of development which is essential to reduce the mistakes at early stages and raise the level of confidence. It also assures that all important aspects are considered at the right level of abstraction whereas programmatic support allows reducing project completion time

Key words: *Service Oriented Architecture, Web Services, cross enterprise business processes, validation*

1. Introduction

As a result of mass customization and due to the complexity of products in the industries, the size and complexity of the supply chain network are increasing rapidly. It is therefore crucial to assure the successful collaboration of the overall supply chain and to improve the cross enterprise communication processes. More efficient and faster communication would contribute to a better quality of the final products and reduced time-to-market due to shortened delivery times and efficient handling of exceptions.

Service Oriented Architecture (SOA) based on Web Services technology provides the flexibility and loose technology coupling needed to support vast supplier networks. The flexibility is needed because globally distributed supplier network partners can change over time and loose technology coupling is required since various companies use plethora of applications running on disparate platforms for managing their work.

We propose the sequence consisting of six steps:

1. Business/ Cooperation processes modeling
    a. Requirements capturing
    b. Graphical modeling
2. Models validation
3. Business processes to Web Services mapping
    a. Cooperation parameters identification
    b. Web Services protocols selection
4. Web Services protocols validation
5. Protocols implementation
6. System validation.

The intermediate validation steps are needed to assure the correctness of the developed solution and to reduce the probability of mistakes. Another important aspect to stress is the need for programmatic support at each level of Service Oriented System Architecture creation i.e. from requirements capturing to system validation to reduce human error and hence shortened project completion time.

The proposed sequence is set in to practice by a research initiative at the Telematics Department of Hamburg University of Technology that aims at strengthening the Small and Medium Enterprises (SMEs) involved in the Aviation industry. The goal of the initiative is to offer a solution that would provide safe and correct collaboration base fostering innovation process of Airbus and supporting cooperating partners and suppliers in fulfilling their contractual agreements.

The rest of the paper is organized as follows. Section 2 outlines the Service Oriented methodology while section 3 delineates how Web Services are an appropriate choice to build Service Oriented Architectures. Our proposed approach is presented in detail in section 4.

Finally conclusions are drawn based upon experiences gained by applying this approach to model business processes between Airbus and one of its suppliers, Innovint Aircraft Interior GmbH in section 5.

2. What is Service Orientation?

Service Orientation (SO) is NOT a technology or a product but a paradigm or an approach to building Systems using Services which adhere to four principles of Service Orientation [1]:

1. Boundaries are *Explicit*
2. Services are *Autonomous*
3. Services share *Schema* and *Contract,* not *Type*
4. Service compatibility is determined based on *Policy*

A System is a set of deployed Services cooperating in a given task. System is built to change and adapt to new services after deployment. Service can be considered as an entity that is interacted with via message exchanges. Unlike Systems, they are built to last and their availability and stability are of critical importance. Services are fractals which mean that a service can be composed of sub-services; sub-services can be composed of further sub-services and so on.

SO provides us an opportunity to rethink how we can design and architect tomorrow's systems by minimizing hard interdependencies, enhancing independence and easing delivery of composite business applications. It enables a high level of interoperability and makes explicit some of the core assumptions that compromise today's systems particularly around boundaries and locality.

The four principles of SO offer various advantages for businesses. It promotes technology reuse resulting in cost savings, modeling business capabilities, serving the business, formalizing inter-departmental or inter-organizational relationships and expressing through service interaction and facilitation through outsourcing and focusing on core competencies.

3. How Web Services fit in?

The SOA is *usually* built from separate Web Services. Web Service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a format that machines can process (specifically WSDL [2]). Other systems can discover a Web Service (specifically via UDDI [3]) and interact with it in a manner prescribed by its description using SOAP [4] messages, typically conveyed using HTTP with XML serialization in conjunction with other Web-related standards (W3C) [5]. Service is the important concept. Web Services are the set of protocols by which Services can be published, discovered and used in a technology neutral, standard form. So Web services provide us with certain architectural characteristics and benefits specifically platform independence, loose coupling, self description, and discovery - and they can enable a formal separation between the provider and consumer because of the formality of the interface.

4. Our Approach

The creation of efficiently working and flexible systems in today's business environment where the supply chain partners use several applications installed on different platforms is just one of the challenges. Many business applications use proprietary protocols, do not integrate with other applications, and require their own user credentials to sign in. More and more organizations are looking for ways to integrate their business processes so they can streamline operations and share information more effectively. It is also impossible to push organizations for a new solution and abandon investments in existing systems because of the costs, reliability, correctness, security and other factors tied with new implementations. It is therefore necessary to find ways to make existing processes more efficient, while making the best use of the systems implemented today. To realize this objective the current inflexible applications should be integrated with the help of additional services to provide needed flexibility, transparency and scalability.

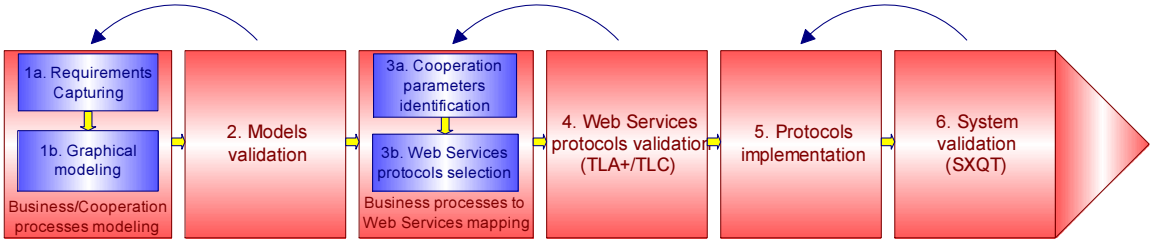

Figure 1: SO compliant system development steps

The architecture based on SO solves this dilemma of the enterprises. By providing Web Services interfaces, our solution is open for additional functionality and for integration into existing systems and processes. Using the standard Web Services protocols and adhering to the SO methodology provide an integrated environment for new and existing applications. We propose following steps to develop SO based system (Figure 1):

4.1. Business/Cooperation processes modeling:

With the goal to implement SO approach, development begins with the analysis of existing document, data and information flows as part of *requirements gathering phase* (step 1a, figure 1). The information concerning business processes is obtained by interviewing people involved in execution of different process steps and also by studying related documents. The outcomes of this research and observations are documented in form of Business Requirements Specifications (BRS) which serves as an essential ingredient to develop Software Requirements Specifications (SRS).

Graphical modeling (step 1b, figure 1) follows after the requirements gathering phase where captured requirements and observed interactions among different actors are depicted in a graphical view. To get the best results, an appropriate level of detail and the right perspective have to be chosen. Various perspectives can be used in order to reduce the information presented in one picture. Hence we need a right balance between the level of detail and the number of perspectives. Graphical models that are one of the most popular approaches today should ease communication between people coming from different domains like business managers and IT experts because all of them will have to make their own contributions in order to reach a complete and correct solution.

Different perspectives exist for model preparation like object-oriented, service-oriented or information flow-oriented. Consequently a number of different tools like ArgoUML [6] based on the Unified Modeling Language (UML) [7] or BizTalk [8] based on the Business Process Execution Language for Web Services (BPEL4WS) [9] exist in the market supporting various modeling perspectives. One of the best suited tools for requirements capturing and modeling of existing processes has to be chosen. The following criteria have been considered for successful modeling:

- *Several representation perspectives* (organizational-, informational-, activity-, role-, object-oriented) shall be provided by the tool in order to look at the process from different eyes and to cover all the aspects.
- *Several abstraction levels* (local/global view) shall be available in order to look at the process from different levels of detail and to provide possibility to increase this level of detail when decomposing the business processes.
- *Optimal level of detail* has to be kept in each of the views provided instead of overloading the person analyzing them with the information storm.
- *Code generation possibility* is desirable for decreasing development time and saving costs. It is acceptable that some editing might still be needed to make it workable.
- *Possibility to integrate with other tools* is considered to be important, because sometimes combined use of different tools provide better results when communicating the results to business owners. Possibility to reuse the results

developed with one tool like graphical model of data and information flows can make development process faster, if code can be generated according to the imported graphical model.

- *Representation of decision points* and parallel processing of activities is very important, because in the business environment different sets of actions can be taken according to the decisions made in one or the other point of the overall process or parallel processing can be used in order to speed up the business process.
- *Simulation possibilities* are required for validation of modeled processes and testing.
- *Fast learning curve* is desirable to save the development time and costs.
- *Expressive representation* of real-life processes is needed to communicate the results to process owners of the processes modeled. Process owners or business analysts should be able to understand the presented results and identify the mistakes in the models worked out so far.

Several tools supporting different modeling techniques and approaches have already been tested by us in [10] with the goal to identify the most suitable one to capture the requirements of business data and information processes and to communicate the results to the people actually involved in the process.

4.2. Models validation

The second step of our approach is *models validation* (step 2, figure 1). This can be done by demonstrating and discussing the produced models with the process owners because owners are in the best position to gauge them. The actor perspective makes it easier for the owners of different process steps to identify themselves within the information presented in the diagrams and allows getting precise feedback from them on the missing aspects or incorrect observations. The use of graphical cooperation models also allows verifying that all actual data and information flows are depicted. Usually it takes several cycles of models improvement to reach the final version of the business models.

4.3. Business processes to Web Services mapping

The third step after modeling and verification is to *derive the cooperation specific parameters* (step 3a, figure 1) like security, safety, messaging from the diagrams and feedback of the process step owners. It is also essential to identify the bottlenecks in the process and get early alerts for application specific exception handling.
Selection of appropriate WS-Protocols (step 3b, figure 1) satisfying the derived parameters and cooperation/business rules analysis follows this step. A large number of protocols using SOAP conventions for message exchange are now under development with the goal to achieve interoperable protocols for Coordination, Transactions, Reliable Messaging, Security and Business Processes in autonomous systems. The current available specifications for WS-* protocols can be found at [5].

4.4. Web Services protocols validation

The *validation of the co-existence of the selected protocols* (step 4, figure 1) is suggested to ensure that chosen protocols work as intended. Like other distributed systems, WS protocols can allow ambiguous behaviors which can make understanding and debugging hard and painful. WS standards have formal mechanisms to specify the format of XML data structures [11] and service interfaces [2]. However, they generally do not have methods of precisely specifying the complete behavior of a protocol [12]. They instead employ informal descriptions of the protocol that can be imprecise, ambiguous, or incomplete; they often fail to consider unusual cases and therefore make implementation of these protocols an error prone process. In order to raise the level of confidence of implementers, it is important that these protocols are unambiguous, operate correctly and describe completely the allowed behavior of the protocols' participants.
The need for precision and completeness in a standard naturally suggests the use of formal methods. Such methods use a well-defined language with a precise semantics for writing formal specifications of a protocol's allowed behaviors. Tools can be applied to analyze those

behaviors and help check the correctness of the protocol. There is no generally accepted method of formally specifying a WS protocol. State tables that are given in some specifications are a step in that direction, but they are usually not written as precisely as formal specifications [13]. TLA+ [14] is a language for writing high-level specifications of concurrent systems. Unlike most specification languages, it is based on mathematics rather than programming-language constructs. This makes it extremely expressive. Because it is so mathematical, TLA+ seems foreign to most engineers. However, we have found that engineers can very quickly learn to read TLA+ specifications. Learning to write TLA+ specifications seems to be about as hard as learning a new programming language. Our experience shows that writing and model checking a TLA+ specification can help eliminate errors and ambiguities in a protocol. TLA+ can be used to help achieve the reliability required in the design of a WS protocol standard. We also believe that writing formal specifications can aid in the process of designing WS protocols. The TLA+ specification therefore describes the part of the protocol that is generally left imprecise in current specifications, and ignores those aspects of the protocol that are already specified quite precisely. TLA+ thus complements the approach taken in most existing standards. Having a TLA+ specification can help prevent incompatibility among different implementations. We have specified and checked some WS protocols in [12] and [13].

4.5. Protocols implementation

Web Services protocols implementation (step 5, figure 1) can be done using the available languages, frameworks and execution platforms. The two leading runtime execution platforms are Java 2 Enterprise Edition (J2EE) and Microsoft Windows/.NET. While both of these runtime platforms make excellent underpinnings to enable Service execution, they aren't in and of themselves an SOA platform. For instance, while J2EE is well-suited for running platform-dependent Services, it is not Service infrastructure. J2EE wasn't developed to be a Service-oriented platform, but rather, the best distributed, virtual machine-based, object-oriented platform around for runtime code execution. In an SOA, declarative, metadata-driven composition trumps portable code, and that's just not what J2EE is ideally suited for [15]. Microsoft upcoming Windows/.NET/Indigo combination is claimed to be the first programming model built from the ground up for building service-oriented applications with the support of a broad range of Web services standards as well as advanced standards and specifications that comprise the WS-* architecture [16]. However we think that it will take time to verify these claims and have an intuitive service-oriented programming model. Hence people will continue to follow pieces-part approach using existing platforms to implement SOA and hence doing more work that may be necessary. We have used .NET framework 1.0 for our prototype system with SQL Server 2000 as a content repository.

4.6. System validation

Normally newly implemented systems have to under go extensive testing which is usually manual and a number of standard and exceptional situations have to be simulated against which the application is tested later on. Simulation of test situations and testing itself requires additional time and man power and there is always a risk of nonconformance due to limited number of test cases. Moreover in case of misinterpretations of specifications the conformance cannot be ensured. In our case implemented protocols are exposed to permanent automatic validation under real environment conditions with the help of Specifications using XQuery [17] expressions on Traces (SXQT) as suggested in [18]. SXQT specification technique allows detecting non-conformances in the observed message sequences, when compared against protocol specification. The flow of real message exchange traces is used for the validation purposes to see whether the protocols work the way that is intended. The permanent validation is not limited to a number of simulated situations as the system is validated with each appearing exception. It does not require involvement of developers and therefore excluding mistakes because of human factor. It is also independent of developers understanding of specifications and informs developers whenever some deviations appear and hence supporting developers in creation of conforming implementations. Validation using SXQT can be implemented for early prototypes, for final implementations or in real world working environments.

5. Conclusions

The paper presents the six steps approach for successful Service Oriented Architecture and its implementation by using Web Services protocols. The presented model with the validation following every stage enables exclusion of systems errors at early stages and allows saving time and costs due to the reduction of man hours required for testing and maintenance of the system. The investigations are being further backed up by the development of solution for Innovint Aircraft Interiors GmbH's cooperation with its customers and suppliers, especially concentrating on the observed interaction with the Airbus. It is hoped that this work will demonstrate how companies in the Aviation industry which are affected by the tendency of mass customization can create flexible and extensible systems while still making use of their IT investments by reusing current implementations. The case study aims at contributing to creation of more advanced applications that greatly improve cross enterprise business process efficiency, transparency and flexibility by integrating a number of enterprise applications, unifying data and stored information.

References

[1] Don Box: A Guide to Developing and Running Connected Systems with Indigo. Published in MSDN Magazine, January 2004, available at: http://msdn.microsoft.com/msdnmag/issues/04/01/Indigo/default.aspx

[2] Christensen, E., F. Curbera, G. Meredith and S. Weerawarana: Web Services Description Language (WSDL) 1.1 (2001), W3C Note, available at: http://www.w3.org/TR/2001/NOTE-wsdl-20010315.

[3] Karsten Januszewski: The Importance of Metadata: Reification, Categorization, and UDDI, MSDN Library, September 2002, available at: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnuddi/html/impmetadata.asp

[4] Mitra, N. (2003, June 24). SOAP version 1.2. part0: Primer, technical report, World Wide Web Consortium (W3C), available at: http://www.w3.org/TR/2003/REC-soap12-part0-20030624/

[5] Microsoft: Web Services Specifications, Microsoft Corporation, Redmond, 2003. http://msdn.microsoft.com/webservices/understanding/specs/default.aspx

[6] Tigris.org: Open Source Software Engineering Tools: ArgoUML at: http://argouml.tigris.org/

[7] Object Management Group (OMG): Unified Modeling Language (UML) at: http://www.uml.org/

[8] Microsoft: BizTalk Server 2004 at: http://www.microsoft.com/biztalk/default.mspx

[9] THATTE, Satish; et. al.: *Business Process Execution Language for Web Services, Version 1.0*. BEA Systems, International Business Machines Corporation, Microsoft Corporation, 2002. http://www-06.ibm.com/developerworks/library/ws-bpel/

[10] Jurga Kazlauskaite: Information Flows in Logistics Networks. Master Thesis. Hamburg University of Technology, September 2004.

[11] Thompson, H. S., D. Beech, M. Maloney and N. Mendelsohn: XML schema part 1: Structures (2001), W3C Recommendation, available at: http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/.

[12] James E. Johnson, David E. Langworthy, Leslie Lamport, Friedrich H. Vogt: Formal Specification of a Web Services Protocol. Published in: Electronic Notes in Theoretical Computer Science, Volume 105, 10 December 2004, Pages 147-158.

[13] Friedrich H. Vogt, Simon Zambrovski, Boris Gruschko, Peter Furniss, Alastair Green: Implementing Web Service Protocols in SOA: WS-Coordination and WS-Business Activity, IEEE International Workshop on Service oriented Solutions for Cooperative Organizations (SoS4CO '05), Munich, Germany

[14] Lamport, L.: Specifying Systems, Addison-Wesley, Boston, 2003.

[15] Jason Bloomberg: Tracking the Elusive SOA Platform, ZapFlash, June 15, 2005 at http://www.zapthink.com/

[16] David Chappell: Introducing Indigo: An Early Look, MSDN Library, February 2005 at: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnlong/html/introindigov1-0.asp

[17] BOAG, Scott; et. al.: *XQuery 1.0: An XML Query Language*. W3C Working Draft. World Wide Web Consortium (W3C), November 2003. http://www.w3.org/TR/2003/WD-xquery-20031112/

[18] Marcus Venzke: Specifications using XQuery Expressions on Traces. Published in: Electronic Notes in Theoretical Computer Science, Volume 105, 10 December 2004, Pages 109-118.