Simulation-based experimentation for designing reliable and efficient Web service orchestrations in supply chains

Tamrat Tewoldeberhan a,*, Marijn Janssen b,1

a Section of Systems Engineering, School of Technology, Policy and Management, Delft University of Technology, Jaffalaan 5, 2628BX Delft, The Netherlands

b Section of Information and Communication Technology, School of Technology, Policy and Management, Delft University of Technology, Jaffalaan 5, 2628BX Delft, The Netherlands

Received 2 April 2006; received in revised form 19 November 2006; accepted 19 November 2006
Available online 21 December 2006

Abstract

The world today is witnessing a growing interest in conducting supply chain business processes electronically using Web services orchestration technology. Fast adoption is often hampered by the need for experimentation to make efficient use of this technology in supply chain processes. In this paper, a simulation-based approach supporting experimentation with the efficiency and reliability of Web service orchestrations in supply chains prior to implementation are presented. The approach simulates the de facto business process specification standard for Web services, BPEL4WS, using Java-based simulation building blocks. A supply chain case study is used to demonstrate and evaluate the approach.

© 2006 Elsevier B.V. All rights reserved.

Keywords: BPEL4WS; Business process re-engineering; Decision support; Service-oriented architecture; Simulation; Web services; Web service orchestration

1. Introduction

The ability to create flexible alliances with partners to form supply chains or business networks becomes more and more important for businesses [30,35]. When products or services are followed through the network, the successive steps of value addition in the network are often described by the term supply chain [31]. The choice of the term supply chain is unfortunate because it might result in confusion. The term “chain” is often associated with linear invocations and static images. The term supply is inappropriate and suggests mass production rather than the creation of customized business processes. Better names like demand network or customer driven Webs have been proposed. Yet, the name supply chain holds its own [15]. Strictly speaking, a supply chain is a network of multiple businesses and relationships [18]. The creation of flexible, temporarily arrangements result in the creation of business processes that are no longer self-contained within a single organization. The reliability of business processes depends more and more on the performance of external partners that are often unknown and viewed as black boxes. In addition, the dynamic nature of supply chains, for example, the changing number and/or types of partners, and the involvement in several networks, increases the difficulty and complexity to understand the dependencies in the supply chain. Consequently, the re-design of the business processes in supply chains becomes difficult, as the impact of changes cannot easily be determined. More insight into business processes spanning multiple organizations is needed.

Nowadays, one of the most promising technologies for ensuring interoperability at a process level is Web service
Web service orchestration (WSO) [39]. Web services are self-contained, Web-enabled applications capable of performing some kind of functionality, and possessing the ability to engage with other Web services to complete higher-order business transactions [38]. WSO coordinates the sequence of Web service invocations. WSO and Web services technology enable loosely-coupled business processes and applications with much less effort compared to previous technologies. However, acquiring the true value of efficient and reliable orchestration of the various Web services based business processes remains a challenge. There is a lack of experience reports, simulation experiments and case study material concerning WSO [12]. Moreover, in practice, it often requires a lot of experimentation to leverage the benefits of WSO to support businesses processes [e.g., 9].

Inefficient and unreliable WSO can easily result in inordinate waiting times for multi-step business processes, a lack of synchronization of events, limited coordination and cooperation, and excessive control at high operational costs. Consequently organizations are often in need of instruments they can use to speed up the design process and improve application effectiveness [9]. Supply chain performance can significantly benefit from decision-making processes that dynamically evaluate design options [2]. Discrete-event simulation is a suitable decision-support instrument, as it allows decision-makers to understand the essence of business systems, to identify opportunities for change, and to evaluate the effect of proposed changes on key performance indicators [4,19]. In discrete-event simulation models the passage of events during time plays a crucial role. Chandrasekaran et al. [5] simulated Web service compositions and concluded that discrete-event simulation provides many benefits if experimentation in practice is not feasible or too costly. They did not investigate the use of Web service orchestration technology nor did they research the use of Web services in organizational settings such as supply chains.

In this paper, a simulation-based approach is presented that is aimed at improving the efficiency and reliability of Web service orchestration in supply chains. This paper is structured as follows. In Section 2 Web service orchestration, service-oriented architecture, Web service technology and simulation are discussed. Thereafter, we present our simulation-based approach. In Section 4 this approach is demonstrated and evaluated using a case study. In Section 5 conclusions and recommendations for future research are drawn.

Our main finding from the research is that our simulation approach provided a suitable instrument to experiment with Web service orchestration in supply chains prior to implementation. Using the approach it was possible to gain insight into the dynamics of Web service orchestrations, to assess the efficiency of the Web service orchestrations, and to conduct “what if” analyses to estimate the reliability and the effect of a service failure on to the overall orchestration.

2. Background

2.1. Web service orchestration

Supply chain process partners need to share and access information such as stock levels and inventory. Different middleware technologies have been researched and applied to enhance and support the supply chain business processes operations [22]. Common object request broker architecture (CORBA), remote method invocation (RMI), and electronic data interchange (EDI) are just a few of the examples. Though these technologies were used for integration purposes, they do not address interoperability issues that are keys in supply chains consisting of several partners each using different types of applications and information systems infrastructures [33]. Web service orchestration promises to solve these problems and can be used to address interoperability issues in supply chains. It facilitates interactions and sharing of information in heterogeneous environments. Using Web services, it can “glue together” applications running on two different platforms, enable database information to be accessible to others, and enable internal applications to be made available over the Internet [17].

Papazoglou and Georgakopoulos [26] stressed the importance of Web services with regards to business processes like supply chains. They discussed the need for coordination, monitoring, conformance, and quality of service. Meredith and Bjorg [24] pointed out how contracts and trust can be reached between partners using Web services in their business processes. Current architectures can cover quality of services, security, management, business level agreements, service level agreement, and so on [17]. Curbera et al. [6] stated that XML and Web service technology are becoming the standard interoperable programming language.

Enterprise resource planning (ERP) systems can also be used for supply chain automation. Although these systems are already used widely, they are not based on standards and their main focus is neither on process orchestration nor on enhancing interoperability. This gives Web services and particularly business process execution language for Web service (BPEL4WS) a unique position in addressing interoperability issues and enabling process orchestration in supply chain.

2.2. Service-oriented architecture and Web service technology

A service-oriented architecture (SOA) is essentially an architecture that describes the communication patterns between interacting Web services, their functions and their operations [26]. These architectures provide the ability to register, discover, and use services, where the architecture is dynamic in nature.

In an SOA, a service is a function that is well-defined, self-contained, and does not depend on the context or state
of other services. SOAs do not have any specific technology or a description language. Services are often created in the environment of one programming language. SOA defines how computing entities can interact in such a way that it enables one entity to perform a unit of work on behalf of another entity. Each interaction is self-contained and loosely-coupled, so that each interaction is independent of other interactions.

Web service technology uses a loosely-coupled integration model that allows flexible integration of heterogeneous systems in a variety of domains [21]. Web service technology is an Internet technology, which allows applications to communicate with other applications to offer business data or functional services programmatically. Using a layered structure, Web service technology includes all the Internet-based network layers and specific layers can be added on to. These layers include a Web service description layer and a process flow layer [6]. The description layer uses a language called Web service description language (WSDL) [26]. The process flow layer provides the orchestration of activities to be performed within a node and between nodes in the network. The process flow layer has several modeling languages available including business process execution language for Web services (BPEL4WS), business process markup language (BPML), Web service choreography interface (WSCI), extensible language (XLANG), and Web service flow language (WSFL). Among these process flow languages BPEL4WS has become the de facto standard [21]. BPEL4WS is developed by a consortium led by Microsoft, IBM, and BEA, and unifies two older languages from Microsoft and IBM: XLANG and WSFL. BPEL4WS promises many operational advantages, including increased efficiency, reduced development, operating and personnel cost, reduced processing times and reduced delay/transport times [11].

2.3. Business process simulation

The dynamics associated with supply chain processes, the complexity associated with the interdependencies between individual parts, and the stochastic elements associated with the randomness and unpredictable behavior of transactions complicates the design of efficient and reliable Web service orchestrations. Simulation models are capable of capturing the interdependencies between systems, stochastic elements and showing the time-ordered dynamics [34]. Simulation models do not rely heavily on mathematical abstraction and are therefore suitable for modeling complex systems [27]. They can be used for generating quantitative output on various parameters that influence the performance of the orchestration. Statistical representation of real-world uncertainties is an integral part of simulation models [19]. In addition, simulation models can easily be updated to reflect changes in the real-world. Discrete-event simulation models can be used to mimic the stochastic and unpredictable behavior of Web service orchestrations. Therefore, discrete-event simulation models seem to be able to model Web service orchestrations in supply chains.

Shannon [29] defines simulation as the process of designing a model of a concrete system and conducting experiments with this model to understand the behavior of a concrete system and/or to evaluate various design options for the operation of the system. To date, much research has been conducted in the field of business process simulation research [10,19,20]. Business process simulation approaches focus on simulation of business processes, and the subsequent execution of activities, often for the purpose of designing information systems supporting the execution of these processes. A business process simulation approach can help to test and analyse different orchestration scenarios to understand their impact on a broader system or to provide proof-of-concept evidence before moving forward with implementation plans. In this way it enables managers to determine the likely consequences of investments, operational decisions and process changes before they are actually implemented.

Fig. 1 shows the steps taken in a simulation-based inquiry approach [4]. The adoption of supply chain systems has been limited, as their design often neglects existing organizational realities [14]. Therefore, first an “as is” model is developed of the current situation. Next, based on a diagnosis, one or more “to be” models are developed, and compared with each other. An important aspect of simulation is experimentation. An experiment embodies a number of simulation runs utilizing the same treatment. A treatment consists of the specification and collection of exogenous or input data, initialization conditions, run control conditions and specification of endogenous or output data [32]. An experiment with a simulation model of the “to be” situation is a set of replications conducted under the same treatment. A replication consists of a collection of output data regarding a certain treatment [32]. The simulation steps shown in Fig. 1 will be followed to construct simulation models of Web service orchestration in this research.

3. A simulation approach

The translation of Web service orchestration constructs into a simulation model is a complicated endeavor. There are three main ways to construct simulation models: (1) develop each simulation model from scratch, (2) develop a simulator plug-in for a Web service orchestration engine, or (3) develop a simulation building block library. The first approach requires that a specific simulation model is developed for every process orchestration instance. This approach is time-intensive and complex, as each element needs to be developed from scratch, and results in difficult to maintain, adapt, and reuse simulation models. Moreover, it does not enable reuse of already developed pieces of simulation programming code.

The second approach concerns the development of a simulator plug-in for a Web service orchestration engine.
This approach is similar to the many simulators developed for workflow systems. In this approach, a process orchestration instance developed in a Web service orchestration language does not have to be mapped nor converted to a simulation model. Existing flows are automatically interpreted by the simulator and then executed. The main problem with this approach is that there is limited flexibility for conducting experiments, merging and adding other process orchestration models, linking/embedding external software systems, and reusing simulation models partially or fully. In addition, it is not possible to use BPEL4WS, or most of the current Web service orchestration languages, directly for simulation [28]. First, the dynamics of discrete-event simulation cannot adequately be represented, as entities and resources are not represented in this language. Second, a time parameter is not defined explicitly in BPEL4WS and is a necessarily ingredient for simulating the time-ordered sequence. Moreover, BPEL4WS does not include any quality or service metrics.
Therefore, we opted for the last option and decided to develop a simulation building block library for BPEL4WS, the de facto standard Web service orchestration language. Several simulation research studies claim the following benefits of building blocks [7,16,37]: improved efficiency, reusability, maintainability, less need for low-level programming and even automatic generation of models, extensibility of models, and flexiblity in experimentation.

Most authors consider the most important benefits of simulation building blocks the efficiency improvement. Our approach adopts the building blocks concept, which solves the problems and issues mentioned for the previous two approaches. In comparison to the second approach, it has the drawback that a simulation model of the “as is” situation needs to be developed.

The simulation building blocks should have a close resemblance of the Web orchestration constructs. Therefore we followed the three steps depicted in Fig. 2 to develop BPEL4WS based simulation building blocks.

1. **Extract**: Choose the appropriate simulation formalism to map BPEL4WS elements.
2. **Map**: Make a representation the BPEL4WS elements in the chosen simulation formalism.
3. **Implement**: Program the BPEL4WS based simulation building blocks in discrete-event simulation software that supports the chosen formalism.

### 3.1. Extract

There are a number of different types of approaches that can be used to describe the dynamic system structure of discrete-event simulation models, including event scheduling, activity scanning, and process interactions [8]. These approaches are also called conceptual frameworks, world-views or Weltansicht, simulation strategies, or formalisms by Balci [3]. He further defines a formalism as a structure of concepts and views under which the simulation developer is guided when constructing a simulation model. Formalisms describe a system in terms of objects, attributes, events, activities, or processes depending on the relevance.

We selected event-scheduling to develop the BPEL4WS based simulation building blocks, as it focuses on the events happening at a certain time and it is not necessary to have knowledge of the next movement of entities. The event scheduling description approach meets the characteristics of Web service orchestration. Entities that are orchestrated also do not have any knowledge about the next steps.

### 3.2. Map

In this step, the representation the BPEL4WS elements is mapped to the representation of the chosen simulation formalism. We choose to map the BPEL4WS elements and simulation constructs to each other in one-to-one manner and decided to use the same name and functionality. All the BPEL4WS elements were mapped to simulation constructs in this way. In Fig. 3, the BPEL4WS elements are shown on the left side of the figure and the resulting simulation building blocks on the right side of the figure. The meta-view on the BPEL4WS elements (left) and simulation object classes (right) are shown at the top of the figure. The complete mapping of all elements can be found in [25].

### 3.3. Implement

Now that the BPEL4WS have been mapped, they need to be implemented. For this purpose, a discrete-event simulation tool that supports an event-scheduling formalism is required. There are a number of commercial tools that support event-scheduling formalism [36]. However, we only investigated simulation tools supporting XML parsing and processing, because BPEL4WS processes are specified in XML. The availability of such features and their utility facilitates the automatic generation of simulation models from BPEL4WS based process instances. In this way the drawback of the simulation building blocks approach, that a simulation model of the “as is” situation needs to be developed, is minimized. Java is a high-level programming language that contains a rich XML processing and parsing library [23]. Simulation tools based on Java can benefit from this library. Therefore, we further investigated...
Java-based discrete-event simulation tools supporting event-scheduling formalism. We selected the distributed simulation object library (DSOL) simulation suite because it is freely available, easy-to-extend and can be used to develop simulation building blocks for a particular domain of application. DSOL is a Java-based simulation tool that support event-scheduling formalism. It is an object-oriented simulation tool that supports Web-enabled execution, multi-formalism, and distributed modeling [13].

After developing BPEL4WS-based simulation building blocks, they can be used to create simulation models of BPEL4WS process instances. In the following section we show how the BPEL4WS (receive) element is translated into Java code, demonstrate the building blocks approach using a case study, and conduct two experiments.

4. Case study: faplin electronic supplies

To demonstrate and evaluate the usefulness and usability of our simulation-based approach, we applied it to a fictitious but realistic supply chain case study. The case study is based on the supply chain case of the Web service interoperability organization [1].

4.1. Case description

Faplin electronic supplies is an electronics retailer with around 80 stores in several countries. The company sells over 12,000 electronics products ranging from resistors and communication components to computers and telecom accessories. The company already has a transactional Web site, which is regularly used by thousands of customers. Faplin wanted to investigate the potential of Web service orchestration technology to connect with its upstream and downstream partners. For the case study, we looked at the supply chain processes with its three main warehousing partners and electronics suppliers as shown in Fig. 4. These suppliers manufacture various types of electronics products, but we only included three product types, televisions, DVDs and video cameras, in our model.

For the pilot study we analysed their three main business processes: (1) catalog access, (2) purchase goods and (3) replenishment stock. The catalog access process involves customers sending a request to Faplin and getting a response consisting of a list of items available. The purchase good process consists of two sub-processes: (2a) the ordering process where a customer sends an order to Faplin and later receives the response for the order and (2b) the source goods process in which Faplin finds a suitable warehouse to fulfill the order. The replenishment stock process involves interactions between Faplin, manufacturers and warehouses. The process aims to control the stock levels by providing updates regularly and handling the ordering of new products.

4.2. Simulation model construction

The purchase order process, consisting of the purchase order and source goods sub-processes, is shown in Fig. 5. The figure visualizes the Web service invocations necessary for executing the process. Invocations are either synchronous or asynchronous. Synchronous invocations are handled immediately and give a direct response. Invocations to processes such as “Validate Order” are asynchronously, as the processing takes a while. The process logic is shown in the middle of the figure and the Web services of external organizations are visualized on the right and left.

We used the BPEL4WS simulation building blocks to construct the simulation model of the purchase goods and source goods processes. Each BPEL4WS element
was translated to the simulation model in a one-to-one manner. The BPEL4WS \(<\text{receive}>\) element part of the “purchase order” process is shown below.

\[
\text{<receive partnerLink = "client" portType = "Faplinreply" operation = "receiveresponse" variable = "PO" createInstance = "yes/no"? standard-attributes> /<receive>}
\]

Using the BPEL4WS building blocks, the \(<\text{receive}>\) element is translated into the Java-based simulation code shown below.

\[
\ldots \text{_receiveFaplin = new receive( devSimulator, "receive", new partnerLink("client", "Ins:Faplin", "provider", "requester")),}
\]
new variable(“PO”, new messageType(“order”, new part(_element)));
new portType(“ins:Faplinreply”, new operation(“receiveresponse”, “order”, “response”)),
new operation(“receiveresponse”, “order”, “response”));
_receiveFaplin.setMessageDelayTime (messageDelayTime);

The service time, the time a Web service takes to perform its task, of a Web service consists of various parts. Chandrasekaran et al. [10] use three indicators to calculate: the service time ($S$); the message delay time ($M$), the time it takes to send and receive messages by the invocation call; and the waiting time ($Q$), the service invocation delay caused by the load on the system where the service is deployed. Thus the total invocation time ($T$) for a Web service $j$ is the sum of the three time values.

$$T(j) = S(j) + M(j) + Q(j)$$

In our model the message delay time ($M$) has the same distribution for all Web services. Warehouse A processing time is normally distributed with an average of 15 min and 2 min of standard deviation, Normal (15, 2) minutes; Warehouse B processing time is triangular (2, 17, 25) minutes; and Warehouse C processing time is normal (30, 5). The message delay time for all Web services is uniform (0.0, 0.1) minutes. The inter arrival time of orders is exponential (10) minutes.

### 4.3. Orchestration experiments

The output of the “as is” experiment called sequential invocation is shown in Fig. 6. Experimentation with this simulation model showed that the average cycle-time for orchestration was 60 min with a standard deviation of 26 min. The simulation revealed that even though the average was 60 min, at peak periods the processing time reached over 104 min. In addition, there were times when orders were processed within 13 min. Some orders were fulfilled by a single warehouse, others by more than one.

The evaluation of the fluctuations in the total orchestration cycle-time resulted in the development of a second experiment called parallel invocation. In this experiment there is no exception handling mechanism in place. In contrast to the initial experiment, this experiment invoked all warehousing Web services parallel instead sequential.

A comparison between the initial experiment and the “parallel invocation” experiment revealed that the second experiment was more efficient; having a minimum cycle-time of 4 min and a maximum cycle-time of 42 min. The
fluctuations between the minimum and maximum are high due to the differences in order types and amounts. The “parallel invocation” scenario has a lower reliability than the initial experiment. In the initial experiment, clients regularly received the response that their orders could not be fulfilled. Often there was no stock available in the warehouses. The results indicate that the number of successful orders in the second experiment were 34 out of 81, a 41% success rate, whereas in the first experiment, the number of successful orders were 62 out of 81, a 76% success rate. In short, the experiments show that both the “sequential invocation” and “parallel invocation” scenario have their strengths and weaknesses. The “sequential invocations” experiment has relatively long cycle-times, thus a low efficiency, and resulted in many successful orders, thus a high efficiency. Often there was no stock available in the warehouses. The results indicate that the number of successful orders in the second experiment were 34 out of 81, a 41% success rate, whereas in the first experiment, the number of successful orders were 62 out of 81, a 76% success rate. In short, the experiments show that both the “sequential invocation” and “parallel invocation” scenario have their strengths and weaknesses. A higher efficiency results in a lower reliability and vice versa. The “what if” analyses stress the need to have alternative Web services available in case of failure of one of the services to ensure the smooth and reliable functioning of the supply chain.

Our research shows that Web service orchestration is a viable technology for automating supply chains and can be used to create alliances with partners and connect to the loosely-coupled business processes of network partners. Web service orchestration results in new forms of coordination among network partners. The case study highlight that there is more to successful Web service orchestration than merely considering it as implementing a technological application. We used efficiency and reliability metrics for evaluating the experiments, however, the real pay-off of orchestration might come from other areas. We did not look at operational measures, such as, quality, agility and adaptability of supply chains, nor did we investigate organizational and strategic benefits, disadvantages and risks. Orchestration requires the management of relationships with other organizations, the coordination of processes, and the handling of assets and capabilities of the network organizations to deliver value to the customers. Orchestration of networks is a continuous learning process, not only aimed at optimizing the supply chain, but also at balancing the interests of all organizations involved in the supply chain. Our simulation approach can help to take some steps in this journey, but it does not take the changing conditions and the stakeholders’ interests into account.

Our approach simulated Web service orchestration in supply chains using pre-defined Java-based simulation building blocks. The traditional way for modeling would be to use existing simulation languages and accompanying constructs to model Web service orchestrations. Instead we developed building blocks for modeling Web service orchestration constructs in a Java-based, open source simulation language. We used three steps, extract, map and implement to create the building blocks. Using these steps, the BPEL4WS based XML elements were mapped to the simulation building blocks in a one-to-one manner using the same name and functionality. We found that the building blocks provided an appropriate representation of Web service orchestration elements, it was fairly easy for non-simulating experts to construct new models. The names of the simulation blocks are the same as the BPEL4WS elements, as they were mapped in a one-to-one manner.

In our research, we opted for the design of simulation building blocks and did not evaluate the other two options, develop a simulation model from scratch and develop a
simulator plug-in for a Web service orchestration engine. We made the assumptions that the use of building blocks would result in improved efficiency, reusability, maintainability, less need for low-level programming and even automatic generation of models, extensibility of models and flexibility in experimentation. Using our case study, we were able to illustrate that the building blocks were reusable for modeling a number of situations, there was no need for low-level programming, we were able to extend the models and used to building blocks to develop alternative models, which enabled flexibility in experimentation. We did not evaluate the efficiency improvements and improved maintainability of the models, as suggested by the literature.

One of the motivations for using pre-defined building blocks derived from BPEL4WS based XML elements was that this would allow for the automatic generation of models from BPEL4WS implementation. This intuitive appealing approach not only supports the modeler to create models more quickly, but it should also enable the automatic generation of simulation models from the existing situations. We did not investigate this possibility and further research is necessary to test this assumption. Moreover, the building blocks are directly derived from the BPEL4WS based XML elements. Organizations and technologies are dynamic and might have changed or updated elements, which might block the automatic generation of simulations model. In our case study all the data were available, however, in many supply chains this might not be a valid premise. Many supply chain members will be reluctant to share data about their mission critical business processes with others. Thus, the adoption of these technologies will be hampered by the fact that organizations often demand that their data and inner business processes remain hidden from the other organizations in the supply chain. In reality, orchestrations might consist of both standard and customized parts. In our testing case study, we made the assumption that the supply chain consists of only standard elements. The inclusions of customized part to simulate might be complicated and deserves future research attentions.

Future research might extend the proposed simulation approach and should also include the modeling of Web service orchestrations for situations where the supply chain is not completely known and limited data are available. Moreover, more research is necessary to systematically evaluating the three options, building a model from scratch, use a simulator plug-in and building blocks. Each option might have some disadvantages and benefits. The business process specification language used, BPEL4WS, lacks quality of service metrics. This was a major setback in retrieving quality of service metrics automatically from the Web service-based infrastructure. We recommend that in the future BPEL4WS designers incorporate relevant quality of service metrics in the language. Web service orchestration result in new forms of coordination and as a result there is a need for investigate orchestration as a new organizational arrangement and to evaluate the implications from a broad perspective.

References