Chapter XXXVIII
Towards Process Mediation in
Semantic Service Oriented
Architecture (SSOA)

Tariq Mahmoud
Carl von Ossietzky University Oldenburg, Germany

Jorge Marx Gómez
Carl von Ossietzky University Oldenburg, Germany

ABSTRACT

Nowadays, it becomes very hard for anybody in the digital world to search and find suitable Web Services fit into his/her needs, since there is a huge amount of data on the Web caused by the enormous increasing of the Web providers and Web Services widespread in this digital community, and one of the most difficulties Web Services have to overcome, in the attempt to use the contents of the World Wide Web, is heterogeneity which is caused by the nature of the Web itself, and has two origins: data or public process heterogeneity. So it is highly required in such environment to have an intelligent mechanism in which every user can search according to his/her needs and later on can fulfill it in a semantic way. The authors will focus in this chapter on the public process heterogeneity which describes the behavior of the participants during a conversation, and propose a solution for dealing with it, explaining the functionality of the process mediator developed as a part of the Web Service Execution Environment (WSMX) and its mediation scenario, and will also apply this proposed solution on Federated Enterprise Resource Planning (FERP) system to get the semantic extension from it.

1. INTRODUCTION

Web Development

The World Wide Web (Web) (Berners-Lee & Calliau, 1990) is a system of interlinked hyper-text documents accessed via the internet. With a Web browser, user can view Web pages that may contain text, images, videos, and other multimedia and navigates between them using hyperlinks; the World Wide Web was created in 1989 by Tim Berners-Lee.
Towards Process Mediation in Semantic Service Oriented Architecture (SSOA)

According to the extreme growth of information available over the Web, and the powerful development achieved on the basis of World Wide Web, the Web 2.0 was born.

In this new version of interlinked hypertext network, it becomes possible that somebody can have the benefit from the experiments of the others in the same domain, which means that in such an environment like Web 2.0 there is a huge network of information which has the responsibility of enhancing creativity, information sharing capabilities, and most notably, the collaboration among users. These concepts have led to the development and evolution of Web-based communities and hosted services, such as social-networking sites, wikis, blogs, and folksonomies.

Some technology experts, like Berners-Lee, had a lot of reservations on the phrase Web 2.0; Lee had an interview with IBM developerWorks about the differences between the conventional Web (World Wide Web) and Web 2.0, and the discussion was like follows: “Web 1.0 was about connecting computers and making information available, and Web 2.0 is about connecting people and facilitating new kinds of collaboration. Is that how you see Web 2.0?” his point of view was fairly described as follows: “Web 1.0 was all about connecting people. It was an interactive space, and I think Web 2.0 is of course a piece of jargon, nobody even knows what it means. If Web 2.0 for you is blogs and wikis, then that is people to people. But that was what the Web was supposed to be all along. And in fact this ‘Web 2.0,’ it means using the standards which have been produced by all these people working on Web 1.0” (Berners-Lee, 2006).

And according to that, digital world needs a new way in which the people can interact in a semantic manner, to involve machines support side by side to the human interactions, and this is the main objective of the Semantic Web.

Semantic Web is an evolving extension of the existing Web in a way that the semantics of information and services on the Web must be defined, making it possible for the Web to understand and satisfy the requests of people and machines to use the Web content (Berners-Lee, Hendler, Lassila, 2001).

We are trying to describe in this chapter how we can involve semantic process mediation between machines and humans in order to have benefits from this knowledge in a semantic way by using Semantic Web Services as part of Semantic Web.

The Need of Process Mediation in Semantic Web

If we generally consider that the market is an institution where demands and offerings are coming together, markets also can be seen as a channel to manage the problems of the negotiation between required and available software components (as services) in a very large information system landscapes inside one enterprise or among enterprises.

One of the most significant factors in Enterprise Application Integration (EAI) within the information system world is enterprise integration. This integration can be applied in various management systems like: Enterprise Resource Planning (ERP), Customer Relationship Management (CRM) and Supply Chain Management (SCM).

Service-Oriented Architecture (SOA) is one of the concepts involved in EAI, and this means that there must be a human interaction in order to use Web Services, whereas, if machines have to be involved in this scenario, Semantic SOA (SSOA) will be one of the best solutions.

SSOA is the concept that supports the use of Semantic Web Services, one of its duties is to overcome Web resources heterogeneity problems, since in such digital environment there is a need to deal with the differences in both ways; the way in which the requester wants to consume the functionality of a Web Service, and the way in which this functionality is made available by the Web Service back to the requester, in this chapter we
Towards Process Mediation in Semantic Service Oriented Architecture (SSOA) will propose a solution handles these issues.

One of SSOA frameworks is the Web Service Modeling Ontology (WSMO) (Bussler et al., 2004) which initiated the standards related to Semantic Web Services. The objective of this framework is to define a consistent technology for the Semantic Web Services by providing means for (semi-) automatic discovery, composition and execution of Web Services based on logical inference mechanisms.

Web Service Execution Environment (WSMX) (Cimpian et al., 2005), the reference implementation of WSMO, is aiming to provide reference implementations for the main tasks related to Semantic Web Services as suggested by WSMO.

WSMO choreography (Fensel & Stollberg, 2005) will help us to describe the expected behavior of the two parties during a conversation, and it is in fact a formalization of their public business processes. By using these descriptions we will be able to introduce the process mediator, and apply this proposed mediator to the FERP system (one of the Web-Service-enabled Service-Oriented Architecture (SOA) solutions) (Brehm & Marx-Gómez, 2005) to make it possible to adjust the two parties' behavior and to make the communication between them more autonomous, and later on to apply the necessary data mediation scenarios on the system (in our future work). However, the major focus in this chapter is to point out the potentials which process mediator solution offers to enterprise architectures, and to demonstrate the applicability of it inside FERP system to exemplify the activities within the lifecycle of such architecture, and to make the first step towards Semantic FERP (SFERP).

2. BACKGROUND INFORMATION

Enterprise Application Integration (EAI)

In EAI, there are two types of integration: internal and external integration. Internal integration, often referred as intra-EAI, specifies the automated and event-driven exchange of information between various systems within a company, another commonly used term for it is “Application to Application”-Integration (A2A). External integration, referred as inter-EAI, specifies the automated and event-driven information exchange of various systems between companies, it is commonly referred to as “Business to Business”-Integration (B2B) (Bussler, 2003).

EAI solutions can be categorized into three basic layers that make up the majority of technologies common in today’s integration solutions:

---

**The EAI Functional Model**

- Business Process Management Services Layer (Process Layer)
- Transformation & Routing Services Layer (Transformation Layer)
- Core Integration Middleware Services Layer (Transportation Layer)
Towards Process Mediation in Semantic Service Oriented Architecture (SSOA)

Process, Transformation, and Transportation Layers, and Figure 1 illustrates these main layers within EAI.

We can summarize and put some characteristics for each layer mentioned in Figure 1, as follows (Brown, Maginnis, Ruh, 2000):

1. **Business Process Management Services Layer (Process Layer):** It contains tools and components that allow the modeling of discrete business processes across multiple applications. Within this layer there are components for process modeling and process representation support. The purpose of process modeling component is to produce an abstraction of a process model called workflow type that can be either used for improved human understanding of operations within a specific domain or to serve as the basis for automated process representation (Bussler & Jablonski, 1996), in the case of EAI it is used for the latter. Process support refers to the proactive control of the entire process from instancing a predefined workflow type, all the way, to its completion. And when we are talking about process models in EAI, there is a need to make a clear separation between private and public processes, because this separation has the significant role to support the necessary isolation and abstraction between the internal processes within enterprise and processes across enterprises.

2. **Transformation & Routing Services Layer (Transformation Layer):** It includes tools to manipulate data contained in messages between applications. As an application generates a message, components in the transformation layer receive, review, revise, and reroute the message based on a set of rules predefined within the environment. By providing these services, applications do not need to include message queuing, data type matching, and application routing functionality. Instead, application developers can use the same mechanisms across all applications through the use of solutions that fall within the transformation layer. Also this layer addresses the data mismatch either at the lower-level of data type representation or at the higher-level of mismatched data structures. Mismatched data types may arise when two services for example use different binary representation for some data type. Dissimilar data structures on the other hand involve two different structures to represent the same body of data.

3. **Core Integration Middleware Services Layer (Transportation Layer):** It represents the scenarios that allow multiple methods of application-to-application communications. Core middleware techniques and methods can be incorporated directly into the applications that need to communicate with each other. Core middleware, as the name implies, makes up the foundation of most EAI solutions, and it includes database access routines, message-oriented-middleware (MOM), transaction processing middleware (TP), remote procedure calls, and distributed objects. This layer is also responsible for the system- and platform-independent communication between the integration tool and the involved applications, it consists of a common protocol layer and adapters that transform external events in messages and vice versa.

**Service Oriented Architecture (SOA)**

A Web Service as defined by the W3C consortium is “a software system designed to support interoperable machine to machine interaction over a network” (Booth et al., 2004) and it is the main unit inside SOA.

In concept, there are three main components
in SOA architecture (Mahmoud & Marx-Gómez, 2008a):

- **Web service provider**: It creates a Web Service and possibly publishes its interface and access information to the service registry.
- **UDDI-registry**: Also known as service broker, it is responsible for making the access information of both Web Service interface and implementation available to any potential service consumer, and categorizing the results in taxonomies. The Universal Description Discovery and Integration (UDDI) registry, defines the way to publish and discover information about Web Services.
- **Web service consumer**: The service consumer (requester) or Web Service client locates entries in the UDDI registry using various searching operations and then binds to the service provider in order to invoke one of its Web Services.

And Figure 2 illustrates the mechanism of publishing, discovering and binding Web Services in the SOA environment:

So, the role of SOA is to provide a specific Web Service through the internet and make it accessible from a client interface. And a typical example for this scenario is a booking system that uses SOA architecture in order to book a room in a hotel, to buy a flight ticket from an airline company, and to rent a car from a car rental agency...

By extending the concept of SOA with semantics, a formal description of the Web Service functionality will be provided in order to make it understandable by all the involved entities (both humans and machines).

**Towards Semantic Service Oriented Architecture (SSOA)**

**Web Service Modeling Ontology (WSMO)**

Web Service Modeling Framework (WSMF) (Bussler & Fensel, 2002) consists of four different main elements for describing Semantic Web Services (see Fig. 3): ontologies that provide the terminology used by other elements, goals that define the problems which should be solved by Web Services, Web Services descriptions that define various aspects of a Web Service, and mediators that bypass interpretability problems.

WSMO is a formal model for describing various aspects related to the world if Semantic Web Services. And it is based on the WSMF concepts. WSMO applies Web Service Modeling Language.
Towards Process Mediation in Semantic Service Oriented Architecture (SSOA)

Figure 3. The main elements of WSMF

(WSML), based on different logical formalisms, as the underlying language (Bruijn et al., 2005). Like we mentioned above about WSMF, WSMO defines four main modeling elements to describe several aspects of Semantic Web Services (Bruijn et al., 2007):

- **Ontologies**: Ontology is a formal explicit specification of a shared conceptualization (Gruber, 1993), WSMO ontologies give meaning to the other elements (Web Services, goals and mediators), and provide common semantics, understandable by all the involved entities (both humans and machines).

- **Goals**: They represent the objectives of the service requester that have to be accomplished when consulting a Web Service. They provide the means to express a high level description of a concrete task. A goal can import existing ontologies to make use of concepts and relations defined somewhere else, either by extending or simply by reusing them. The main advantage of using goals is that requesters only have to provide declarative specifications of what they want in order to find the services that providing the appropriate functionality suit into their requirements.

- **Web services descriptions**: Similar to the way that the requester declares his goals, every Web Service capability can be declared. Additionally non functional properties must be defined, and the interface used mediators. Only if the service requester and provider use the same ontology in their respective service description, the matching between the goal and the capability can be directly established. Unfortunately, in most cases the ontologies will differ and the equivalence between a goal and a capability can only be determined if a third party is consulted for determining the similarities between the two ontologies. For this, WSMO introduces the fourth modeling element: the mediator.

- **Mediators**: There are four different types of mediators: ooMediators, ggMediators, wwMediators and wgMediators.
  - **ooMediators**: mediators that have the role of resolving possible representation mismatches between ontologies.
  - **ggMediators**: mediators that have the role of linking two goals. This link represents the refinement of the source goal into the target goal or states equivalence if both goals are substitutable.
Towards Process Mediation in Semantic Service Oriented Architecture (SSOA)

- **wgMediators** mediators that link Web Services to goals, meaning that the Web Service can fulfill the goal to which it is linked.
- **wwMediators** mediators that are used for linking two Web Services in the context of automatic composition of Web Services.

And as a result, in SSOA there is a need to redefine the three main concepts of the traditional SOA as follows:

- **Service provider:** In it, WSDL can still be in use as a universally accepted interface language, but additionally there is a need to provide another Web Service description like: WSMO, OWL-S or WSDL-S… compliant service description. By doing so Web Service requester will be able to discover services based on formally-defined goals instead of searching only through the directory service and later on selecting the suitable service.
- **Service registry:** The functionality of the service registry (broker) remains the same. The only difference between it and traditional SOA service registry is that it stores semantic service description instead of WSDL description.
- **Service requester:** Service requesters have to publish their desired functionality as semantic goals instead of the traditional way in SOA.

**Web Service Modeling eXecution Environment (WSMX)**

WSMX is the reference implementation of WSMO and it is an execution environment for dynamic discovery, mediation and invocation of Web Services, it offers also a complete support for interacting with Semantic Web Services. WSMX supports the interaction with non-WSMO Web Services (classical ones), ensuring that the interaction with existing Web Services is totally possible.

The internal language used within WSMX is WSML, where the capabilities of the Web Services can be described semantically and the Web Service requesters can invoke these capabilities based on semantic goals described also with WSML, and the role of WSMX is to make a matchmaking between semantic capabilities/goals, show the most related Web Services, mediate between Web Service’s and requester’s ontologies heterogeneity, and invoke the selected Web Service.

And the main components are depicted in Figure 4 that shows the WSMX core manager inside WSMX architecture.

In WSMX core manager there are the following component interfaces (Han et al., 2005):

- **Communication manager:** It has to handle the various invocations that may come from requesters and also to invoke Web Services and retrieves the results of these invocations back to WSMX.
- **Resource manager:** It manages the repositories to store definitions of any WSMO and non-WSMO related objects.
- **Parser:** It checks if the syntax of received WSML descriptions is correct.
- **Discovery:** It has the role of matching the service requester’s goal with a service capability stored in any known repository.
- **Selector:** It provides a dynamic selection of the discovered Web Services in the matchmaking process. The selection process is currently based on a limited set of non-functional properties based on the user needs.
- **Data mediator:** It transforms the incoming data from the means of the requester’s conceptualization (source ontology) into the means of the provider’s conceptualization (target ontology) (Cimpian & Mocan, 2005a).
Towards Process Mediation in Semantic Service Oriented Architecture (SSOA)

• **Process mediator**: The role of the process mediator (Cimpian & Mocan, 2005b) is to make the necessary runtime analyses of two given choreography instances and to solve the problem of the possible mismatches that may appear; taking into consideration that we will use a similar process mediator in our new proposed SFERP system.

• **Choreography**: WSMO choreography describes the expected behaviour of the Web Service, and it is in fact a formalization of its public business processes, we will also use WSMX choreography in the proposed SFERP system.

• **Orchestration**: It specifies what each Web Service will actually do during a conversation.

In discovery, selection and invocation functionalities offered by WSMX, mediation can be needed at both data and process level. In this chapter we will focus only on the process mediation.

**Abstract State Machine (ASM) and WSMX Choreographies**

Yuri Gurevich proposed in the mid-1980s the concept of ASMs, and according to him, ASM consists of states and guarded transition rules. In definition: a state $S$ of a Vocabulary (signature) $V$, defined as a finite collection of function names, is a non-empty set $X$ together with the interpretations of these function names in $V$ on $X$ (Gurevich, 1995). So ASM states are not only mere points within the state space, but more precisely an ASM state is a structure in the sense of mathematical logic and is a nonempty set together with a number of functions (operations over the set) and relations.

Related to ASM, we have to make close look at “ground” and “unground” terms in order to have better understanding about ASM concepts. In general, if we take a mathematical term which does not contain any variable symbol, then we can call it “ground term”, otherwise we called it “unground term” (Gurevich, 1995).

Also Egon Börger denoted that [we used the term ground model (primary model) for such formulations of “the conceptual construct” or “blueprints” of the to-be-implemented piece of “real world” which “ground the design in the reality”] (Börger, 2003).

Based on ASM methodology, WSMX choreography (Haller & Haselwanter, 2005) interface specification in concept is composed of state signatures and guarded transitions. Generally, the signature in WSMX is defined by an ontological schema of the information interchanged in choreography interface by specifying the used concepts, relations and functions of it. Conceptually, a state signature in WSMX choreography is defined also by its attribute role, and there are two types of roles: grounded and ungrounded roles. Also role is defined by an attribute called mode which determines whether it is grounded or not depending on the mode that it performs. Within the choreography only an abstract grounding
Towards Process Mediation in Semantic Service Oriented Architecture (SSOA)

is referenced (the actual physical grounding is resolved in the grounding specification of a Web Service (Kopecký et al., 2007)).

Since choreography interface specification describes the dynamic behavior of the Semantic Web Service, state signature can be described using WSMO ontology. And by checking the mode attribute of state signature’s role, the process of who has the right to modify the instances of the concept (the service or the environment) can be controlled. And the mode attribute can take these following values (Fensel, 2006):

- **In**: The extension of the concept, relation, or function can only be changed by the environment. And it represents a grounded role which has to reference a mechanism that implements write access for the environment.
- **Out**: The extension of the concept, relation, or function can only be changed by the service’s owner. It also represents a grounded role which has to reference a mechanism that implements read access for the environment.
- **Static**: The extension of the concept, relation, or function cannot be changed. Static modes can only be ungrounded since neither the service nor the environment can change any of the ontology elements.
- **Controlled**: The extension of the concept, relation, or function can only be changed by the service’s owner. The difference to the out mode is that it represents an ungrounded role.

In the other hand, guarded transitions are used to express the changes of states by means of rules; and it can be expressed in the following form:

*If condition then rules*

*Where condition is an arbitrary WSML axiom, refers to the state changes regard*

*ing to the information space evolution throughout the consuming of Web Service functionality.*

And the rules actually represent a set of transition rules, consist of arbitrary WSMO ontology instances (add, delete, update).

In SSOA spectrum, each action can be represented as one or more transitions, and by using ontologies to model the concepts we can handle the state of the Semantic Web Service. This assumption leads to the issue that we can control the runtime exchanging process schema by using the WSMX choreographies within the process mediator component as a first step towards implementing the SFERP environment.

**Federated Enterprise Resource Planning (FERP) System**

FERP System Virtues over Conventional ERP System Problems

Before defining FERP system, there is a need to describe the conventional ERP system functionality, list its disadvantages that led to the idea of FERP system.

An ERP system is a standard software system which provides functionality to integrate and automate the business practices associated with the operations or production aspects of a company. The integration is based on a common data model for all system components and extents to more than one enterprise sectors (Boudreau, Robey, Ross, 2002) (Rautenstrauch & Schulze, 2003). The main disadvantages of conventional ERP systems are that: in most cases not all of the installed components are needed; and high-end computer hardware is required and the customizing of such systems is very expensive which means that only huge enterprises can apply such complex ERP system to provide business logic of all its sectors.
Towards Process Mediation in Semantic Service Oriented Architecture (SSOA)

According to these problems, there is a need to a dedicated system in which the enterprise can fulfill its requirements, and one of these systems is FERP. It allows the separation of local and remote functions whereby no local resources are wasted for unnecessary components. Furthermore, in FERP, single components are executable on small computers which subsides the installation and maintenance costs by decreasing the degree of local system complexity (Brehm, Lübke, Marx-Gómez, 2007).

Based to this brief comparison we can notice that the FERP overall functionality is provided by an ensemble of allied network nodes that all together appear as a single ERP system to the user. Different ERP system components can be developed by different vendors (Brehm, Marx-Gómez, Strack, 2007) (Brehm & Marx-Gómez, 2005).

Figure 5 shows how ERP system is supplied by a network of FERP business logic components to provide the needed business functionality.

FERP Reference Architecture

FERP architecture is based on the concepts of SOA, and it is an execution environment for discovery and invocation of enterprise Web Services, it offers also a complete support for interacting with Web Services within the enterprise space or among enterprises. The architecture consists of several subsystems which are interconnected.

Figure 5. Vision of a FERP system landscape where ERP software components can be developed and provided by different software vendors
Because one of the main objectives of the FERP system is to integrate business components of different vendors, all components have to comply with standards which are described as XML schema documents (Brehm & Marx-Gómez, 2007).

Figure 6 illustrates the reference architecture of FERP system.

And the main subsystems in the FERP architectures are:

- **FERP database system**: Includes the required functions to interact with FERP relational database by exchanging XML messages.

- **FERP Web service consumer system**: Contains XML schema definitions and functions needed for the processes of Web Services discovery and invocation provided by different service providers.

- **FERP Web service provider system**: Contains functions required for providing Web Services compatible with FERP Web Service standard and dealing with HTTP incoming and outgoing user’s requests, and has a connection to FERP Web Service directory in order to allow the publication of Web Services.

- **FERP Web service directory**: Its interface has the responsibility of the publication and searching for FERP Web Services based on UDDI standard.

In the SFERP system, Semantic Web Services will be involved within it and the system will be responsible of the matchmaking, filtering results

*Figure 6. Reference architecture of FERP System*
processes; furthermore it will enable dynamic Web Service composition process by including several Web Services at runtime, and there are two possible scenarios to achieve this: locally inside the enterprise domain by receiving this process from public Web Services directories and implement it inside, or by implementing this process remotely on those directories.

And our focus in this chapter will be on the process mediating issues that deal with the differences in the client and Web Services communication patterns in order to have an equivalent public processes inside the system.

3. PROBLEM DEFINITION

SSOA Advantages over SOA
Problems

SOA Service Discovery and Invocation
Problems

Within traditional SOA, the problems in discovery and invocation phases can be described as follows:

Since Web Services appear to be more widely adopted, allowing much broader intra- and inter-enterprise integration, the developers will require automated systems for service discovery, enabling further Web Services interaction with even less human effort (UDDI exists precisely for this reason). However, unless the service requester knows the exact form and meaning of a service’s WSDL in advance, the combination of UDDI with WSDL and coarse-grained business descriptions is not enough to allow fully automated service discovery and usage.

The public business difference between the requester and the provider processes might arise in the invocation phase of traditional SOA, and there is a need to transformation mechanism between both entities at runtime binding to enable fully dynamic scenario (Brehm et al., 2008).

SSOA Service Discovery and Invocation Advantages

Within SSOA, the discovery and invocation phases can be described as follows:

As we mentioned earlier, the main components in SOA architecture are: provider, registry, and requester, in order to add semantics to these components’ interactions, (Li, Lin, Qiu, 2007) proposed a scenario to add semantic matchmaker system to existing UDDI registry, in the other hand, the service provider will add semantics to the descriptions of his Web Services; as well as, the requester will send his request as a semantic goal, and this will be done by using ontologies. In this way, the UDDI will receive a semantic goal and make a matchmaking process with the available Web Services to send the most related results back to the requester based on his needs in semantic and autonomous manner.

In the other hand, using a powerful process mediator between requester and provider at runtime will solve the heterogeneity problems which might arise between their public processes at the invocation phase in order to have a unique and identical public process in the both entities business logic (Brehm et al., 2008).

Definition of a Process, and Process Types

The standard definition of a process is: collection of activities designed to produce specific output for a particular customer, based on specific inputs (Bussler & Fensel, 2002); an activity could be a function, or a task that occurs over time and has recognizable results.

Dealing with business processes, and depending on considered level of granularity, we can consider that each process can be seen as a composition of different transitions or activities.

Also we can differ between two types of processes: private processes and public processes. Private processes, which an organization can
execute them internally, and they are used only inside its domain and are not visible to any other entities. While public processes are the processes that represent the behaviour of the organization in collaboration with other entities. In this chapter we are interested only in the public processes; the private processes are used only within the enterprise and nobody has to do with them except enterprise’s developers.

Process Heterogeneity Problem

In FERP, every Web Service can be seen as a set of processes and these processes composed of actions. By applying semantic framework like WSMX on it, Semantic Web Services will be involved, and therefore we can use WSMX choreography to represent the public business processes of these services in a way that its actions can be represented as state signatures controlled by guarded transitions rules.

In this new system, WSMX choreography determines the constraints on the ordering of messages sent between service consumer and service provider systems. However, the constraints alone are not enough to determine exactly which message is sent when, this is the role of an orchestration. An orchestration is a specification of which message should be sent when. Hence, the choreography specifies what is permitted for both parties, while an orchestration specifies what each party will actually do (Mahmoud & Marx-Gómez, 2008b); we will use WSMX choreography to apply the mediation algorithm because it describes the behaviour of each partner within a conversation.

In process mediation, behind any interaction, each party has some internal process which manages the resources necessary to perform it (in many domains of application, this will correspond to a business process). In some cases, even though the two parties are able to interact via some protocol, there might be some differences between their processes which mean that this interaction will not succeed. And since the choreographies are built on the scenario of ordering the messages, the process mediator functionalities can be categorized as follows (Cimpian & Mocan, 2005b):

- **Stop an unexpected message**: One of the partners sends a message that the other one does not intend to receive, the mediator should just retain and store it. And it can be sent later, if needed, or it will just be deleted after the communication ends.
- **Inverse the order of messages**: One of the partners sends messages in a different order that the other partner expects them. The messages which are not yet expected will be stored and sent, when it will be needed, in a correct order.
- **Split a message**: One of the partners sends much information in a single message that the other one expects to receive in different messages.
- **Single message**, containing information sent by the other one in multiple messages.
- **Send a dummy acknowledgement**: One of the partners expects an acknowledgement for a certain message, and the other partner does not want to send it, even if he receives the message.

4. TOWARDS PROCESS MEDIATION IN SEMANTIC FERP

Process Mediation Functionality within SFERP

If we take a look on the FERP system architecture, we can notice that it relied on standards in the business process level, and to make the system functionality more extensible and semantic to the consuming system (in addition to keep the conventional behaviour), we will suggest the mechanism of applying WSMX concepts on it through the SOA main phases: discovery, selec-
Towards Process Mediation in Semantic Service Oriented Architecture (SSOA)

We will also apply WSMX process mediator (PM) as a solution to the problem of business process heterogeneity described in the previous paragraph.

WSMX process mediation will deal with the case of how two public processes can be combined in order to provide certain functionality (how two business partners can communicate, considering their public processes), and in the case of FERP, the partners are: FERP Web Service consumer and FERP Web Service provider systems.

The mediation transactions can be described as follows: when WSMX Core Manager receives a message, either from the consumer or from the Web Service, it has to check if it is the first message in the conversation. If it is the first, copies (instances) of both the sender and the targeted business partner choreographies will be created, and stored in a repository, together with the id of the conversation. Otherwise, the conversation id must be determined (in case that it isn’t the first message). These computations performed on the message are done by two WSMX core manager components: Communication Manager and Choreography Engine.

After the conversation id had been obtained, the PM receives it, together with the message which consists of concepts’ instances from the sender’s ontology. Based on the id, the PM loads the two choreography instances from the WSMX Repository, by invoking the Resource Manager component. All the transformations performed by the PM will be done on these instances. In the case that different ontologies formulations have been used for modeling the two choreographies, PM must invoke an external data mediator for transforming the message in terms of the target ontology and this is done by checking value of the attribute mode, if it is set to in, for a certain concept, this means that the instance of this concept maybe needed in a future transaction, and all the instances that expected by the targeted partner will be stored in an internal repository (Cimpian & Mocan, 2005c).

PM has to check if these instances (obtained from internal repository) are needed at any part of communication by evaluating the transitions rules using conditions. The evaluation procedure will check if the instances for that rule are expected (an instance is expected when it can trigger an action by changing a state or eliminating one condition for changing a state), only the instances which return a condition that cannot be fulfilled are unexpected by the targeted partner. For the expected instances (it is also possible that set of instances can compose a single instance expected by the targeted business partner), the PM sends the instance (message), deletes it from the repository, and updates the targeted business partner choreography instances; and then reevaluates all the rules, until no further updates are needed. Also one of the PM responsibilities is to check if the sender business partner expects an acknowledgement which the other partner does not intend to send, if so it generates a dummy acknowledgement and sends it back to him.

And for better understanding to these transactions, Figure 7 illustrates the overall functionality in the SFERP system and shows how the process mediation takes place between FERP WS consumer and FERP WS provider systems.

**Example**

In this paragraph and in order to illustrate our idea well, we will give a general example about using choreographies within SFERP, a concrete ERP example will be implemented in our future work, so in this section we will consider some FERP Web Services involved in an electronic shop in order to purchase mobile phones, those Web Services are able to afford mobile phone by providing the name of manufacturing company, model, main specification and price; also the client who wants to invoke such those service.

To keep the example as simple as possible, we will present only some parts of the proposed consumer’s and provider Web Service’s choreog-
Towards Process Mediation in Semantic Service Oriented Architecture (SSOA)

Figure 7. Semantic FERP Overall Functionality

FERP WS Consumer and Provider’s Choreographies

We will consider that the both parties have the following concepts in their internal ontologies:

- **manufactoryCompany**: The instance of this concept represents the manufacturing company name of a specific mobile phone;
- **modelNumber**: The instance of this concept represents the model number of a specific mobile phone;
- **actualSpecifications**: The instance of this concept represents the manufacturing company’s main specifications of a specific

In the following paragraphs we will explain firstly the structure of the proposed choreographies and secondly the step by step process mediation scenario that takes place during the conversation between partners, taking into consideration that all the concepts and axioms used in this example are written using WSML language.
mobile phone like camera, internet connectivity, video, music etc;
• userSpecifications: The instance of this concept is very similar to the one in the previous concept, but the difference is that the specifications here are represented from the user’s point of view (he can make mixture of specifications) while the previous specifications represent the actual specifications of a certain mobile phone.
• price: The instance of this concept represents the price of a specific mobile phone.

The FERP WS consumer’s ontology contains the concept myMobilePhone, which has the following state signature:

concept myMobilePhone
nonFunctionalProperties
dc#description hasValue “concept of a mobile phone requested by the user, containing the manufacturing company’s name, model number, and main specifications of the mobile phone”
mode hasValue out
endNonFunctionalProperties
xCompany ofType manufacturingCompany
nNumber ofType modelNumber
ySpecifications ofType userSpecifications

Actually, the mode attribute within the FERP WS consumer’s choreography has the value out for the concept myMobilePhone, which means that an instance from this concept will be sent to the environment by the consumer. Whereas, for the concepts: actualSpecifications and price, the mode attribute has the value in since the client expect that he will receive information from the Web Service about the phone specifications with its price. And the mode attribute for the concepts: manufacturingCompany, modelNumber and userSpecifications has the value controlled because only the consumer can take the decision regarding these information about the mobile phone.

The FERP WS consumer choreography (ferpcc) has the following rules:

?q x | xCompany hasValue ?xCompany_, nNumber hasValue ?nNumber_, ySpecifications hasValue ?ySpecifications_ ] memberOf myMobilePhone<-
?q x Company memberOf manufacturingCompany and
?q nNumber memberOf ferpcc#modelNumber and
?q ySpecifications memberOf ferpcc#userSpecifications.

The previous rule creates an instance of myMobilePhone when the instances from manufacturingCompany, modelNumber and userSpecifications are created. And the consumer does not expect any input to create an instance of myMobilePhone because the mode attribute of the concepts: manufacturingCompany, modelNumber and userSpecifications have been set into controlled.

When the myMobilePhone instance had created and sent to the service, instances of actualSpecifications and price have been expected by consumer, and the following rules describe that:

?q x memberOf actualSpecifications <-
?q myMobilePhone memberOf ferpcc#myMobilePhone.

?q x memberOf price <-
?q myMobilePhone memberOf ferpcc#myMobilePhone.

So, and as an overall scenario, the consumer will firstly send an instance of the concept myMobilePhone to the service, and expect instances of actualSpecifications and price, regardless the order of receiving them from it.
And as a response from the service, there are two concepts in the service choreography expected from the FERP WS consumer system which are: mobilePhone and phoneOnUserSpecifications, and they have the following state signatures:

**concept mobilePhone**
nonFunctionalProperties
dc#description hasValue “concept of a mobile phone, having two attributes of type manufacturing company’s name and its model number which contains the company name and the mobile phone model”
mode hasValue out
endNonFunctionalProperties
xCompany ofType manufacturingCompany
nNumber ofType modelNumber

**concept phoneOnUserSpecifications**
nonFunctionalProperties
dc#description hasValue “concept of mobile phone based on the user specifications, containing information about mobile phone, user specifications, actual specifications and price of the mobile”
mode hasValue out
endNonFunctionalProperties
forMobilePhone ofType mobilePhone
ySpecifications ofType userSpecifications
zSpecifications ofType actualSpecifications
forPrice ofType price

Within FERP Web Service choreography, the previous concepts have different values for their mode attribute; it takes the value in for manufacturingCompany and modelNumber concepts, and the value out for mobilePhone and phoneOnUserSpecifications concepts, while it takes the value static for the other concepts because their instances’ values can not be changed within the communication process.

The service choreography (ferpsc) has the following rules:

?x [ xCompany hasValue ?xCompany_, nNumber hasValue ?nNumber_ ]
   memberOf mobilePhone <-
   ? x C o m p a n y _ m e m b e r O f ferpsc#manufacturingCompany and
   ? n N u m b e r _ m e m b e r O f ferpsc#modelNumber.

The previous rule creates an instance of mobilePhone only when the instances from manufacturingCompany, modelNumber already exist. The service expects that an input (company’s name, model’s number) has to be provided by the environment in order to create an instance of mobilePhone, because the mode attribute of the concepts: manufacturingCompany, modelNumber have been set into in, so it sends an instance of mobilePhone to the consumer to fulfill his request and send it back to the service.

Another rule states that an instance of phoneOnUserSpecifications can be created only when the instances from mobilePhone, userSpecifications, actualSpecifications, and price are already exist. The service expects that only one input (userSpecifications) has to be provided by the environment in order to create that instance, because the mode attribute of the concept userSpecifications has been set to in, while the instances of the other concepts have been already exist and none of them is expected from the environment. This rule is described as follows:

?x [ forMobilePhone hasValue ?forMobilePhone_,
   ySpecifications hasValue ?ySpecifications_,
   zSpecifications hasValue ?zSpecifications_,
   forPrice hasValue ?forPrice_ ]
   memberOf phoneOnUserSpecifications <-
   ? forMobilePhone _ memberOf ferpsc#mobilePhone and
Towards Process Mediation in Semantic Service Oriented Architecture (SSOA)

Later on, as an extension of this example, and according to his agreement, the consumer wants to purchase the selected mobile phone, so he will send an instance of the concept creditCard and receive an instance of the concept creditCard-Verification, and then he will send an instance of the concept person (containing only the name of the person) to receive an instance of the concept purchaseDone.

In the other hand, the service will receive an instance of creditCard and an instance of person (the distinction between the credit card owner and the person who makes the reservation is made internally, and there is no need to publish it in the choreography file) and then it will send an instance of purchaseDone to the consumer.

In the previous example we tried to show parts of service and consumer choreographies, where the keywords within the code lines were bold, and the concepts were italic, and the value of mode attribute was bold italic.

Step by Step Mediation Scenario

We explained in the previous example how the FERP Web Service and FERP WS consumer choreographies have different mode attribute values for the same concepts inside. To understand the whole scenario in that example, Figure 8 illustrates the overall interactions between the Web Service and the consumer.

In this Figure the communication process between the FERP WS provider and consumer systems can be explained as follows:

- The consumer starts the conversation by sending an instance of myMobilePhone.
- WSMX process mediator (PM) receives this instance and interprets it in terms of

---

**Figure 8. Example about Process Mediation within SFERP**

![WSMX Core Manager Diagram](image)
the Web Service ontology, acquiring one instance of `manufacturingCompany`, one of `modelNumber` and one of `userSpecifications`; and stores them in its internal repository.

- According to the FERP Web Service’s choreography, all these three instances are expected, but the guarded transitions show that only two of them (instances of `manufacturingCompany`, `modelNumber`) are expected at this phase of communication. PM randomly sends one of them to the service (since the targeted choreography does not specify which one of these instances are expected), and then deletes it from the repository. Then the second instance is sent and then deleted from the repository.

- Based on the two instances that had been received, service creates an instance of `mobilePhone` concept and sends it to the PM.

- After translating this instance in terms of the FERP WS consumer’s ontology, and analyzing both choreographies, the PM discards the instance of `mobilePhone` because nobody is expecting any information from that instance. By evaluating the instances in the repository, PM decides that the provider system expects the previously stored instance of `userSpecifications`; so it sends it to him and then deletes it from the repository.

- PM marks the first rule from the service’s choreography and marks the first rule from the consumer’s choreography (which means that these rules will not be reevaluated at further iterations). And if there are still unmarked rules, the communication is not over yet.

- The provider system creates an instance of `phoneOnUserSpecifications` and sends it to PM which interprets it in terms of the consumer’s ontology obtaining one instance of following concepts: `manufacturingCompany`, `modelNumber`, `mobilePhone` instances will be deleted since nobody expects them anymore, the `price` and `actualSpecifications` instances are then sent to the consumer; the order of sending them is not specified in the requester’s choreography, so the PM randomly selects one of them, sends and deletes it from its repository; and then the corresponding rule had been marked.

- The second instance is sent and deleted at the second evaluation of the instances stored in the repository. By sending this instance the rule had been marked.

- PM checks if all rules are marked; since they are, the communication is over.

- PM the deletes both choreography instances from the repository.

5. CONCLUSION AND OUTLOOK

In this chapter the main focus was to show how we applied WSMX process mediator to FERP system to get the new SFERP, and we showed how the heterogeneity problem at the public business process level can be solved by using WSMX choreographies, which represents the behaviors of the participants within a conversation. And as a conclusion the communication scenario between FERP consumer and FERP provider systems, by applying that kind of semantic process mediation, is represented in autonomous and semantic way in order to involve humans and machines side by side.

Our further research will focus on the development of the data mediation scenarios inside such system, and also a concrete example inside the ERP world has to be implemented after finishing some theoretical researches regarding semantic SOA.

Prototype implementations of the future will have to show the practicability of those concepts.
REFERENCES


Towards Process Mediation in Semantic Service Oriented Architecture (SSOA)


KEY TERMS AND DEFINITIONS

**ASM**: A state-based architecture that represents state by algebra as a non-empty set together with number of functions and relations changing their values by guarded transition rules which ultimately model the changes of the state.

**ERP System**: An ERP system is a highly integrated software system representing different types of business application systems.
**FERP System:** A federated ERP system is an ERP system which consists of system components that are distributed within a computer network, and it is Web-Service-enabled SOA solution.

**Ontologies:** Represent the key element in WSMO, firstly to define the information’s formal semantics and secondly to link machine and human terminologies.

**Process Mediation:** Is the mediation scenario for solving the problems of heterogeneity between two participants’ business processes during a conversation.

**Semantic SOA:** An architectural style that enables the use of Semantic Web Services and within it data structures are expressed in ontologies in order to create a distributed knowledge base.

**SOA:** An architectural style that guides all aspects of creating and using business processes, packaged as services, as well as provisioning the IT infrastructure that allows different applications to exchange data from the operating systems and programming languages underlying those applications.

**WSMO Choreography:** Web Service choreography defines actually its usage interface; choreographies in general define how several Web Services interact in order to perform a unified business goal.