



D13.6v0.1 WSMX Use Cases

WSMX Working Draft 05 October 2004

This version:

<http://www.wsmo.org/2004/d13/d13.6/v0.1/20041005/>

Latest version:

<http://www.wsmo.org/2004/d13/d13.6/v0.1/>

Previous version:

<http://www.wsmo.org/2004/d13/d13.6/v0.1/20040816/>

Editor:

Armin Haller

Author:

Armin Haller

This document is also available in a non-normative [PDF](#) version.

Table of contents

1. Introduction

[1.1. Overview](#)

[1.2. Purpose of this document](#)

[1.3. Document Overview](#)

2. WSMX architecture components

3. Trading Partner Integration

3.1. B2C

[3.1.1. What is B2C](#)

[3.1.2. Use case foundation](#)

[3.1.3. Actors, Roles and Goals](#)

[3.1.4. B2C system architecture](#)

[3.1.5. How WSMX can be utilized](#)

[3.1.6. Advantages WSMX exposes for the B2C use case](#)

3.2. B2B

[3.2.1. What is B2B](#)

[3.2.2. Use case foundation](#)

[3.2.3. Actors, Roles and Goals](#)

[3.2.4. B2B system architecture](#)

[3.2.5. How WSMX can be utilized](#)

[3.2.6. Advantages WSMX exposes for the B2C use case](#)

4. Application Integration

4.1. A2A

[4.1.1. What is A2A](#)

[4.1.2. Use case foundation](#)

[4.1.3. Actors, Roles and Goals](#)

[4.1.4. A2A system architecture](#)

[4.1.5. How WSMX can be utilized](#)

[4.1.6. Advantages WSMX exposes for the B2C use case](#)

5. Conclusions and Further Directions

References

Acknowledgement

1. Introduction

1.1 Overview

The Web Services Execution Environment (WSMX) [[Oren et al., 2004a](#)] is an execution environment for dynamic discovery, selection, mediation and invocation of semantic web services. WSMX is based on the Web Services Modeling Ontology (WSMO) [[Roman et al., 2004a](#)] which describes all aspects related to this discovery, mediation, selection and invocation. WSMX is a reference implementation for WSMO. The goal is to provide both a test bed for WSMO and to demonstrate the viability of using WSMO as a means to achieve dynamic inter-operation of semantic web services.

1.2. Purpose of this document

This document exemplifies several usage scenarios of WSMX. In the current version of this document we identify three possible use cases, namely a Business-to-Consumer (B2C), a Business-to-Business (B2B) and an Application-to-Application (A2A) scenario, and showcase how WSMX can be utilized for them. The process steps within these scenarios are only presented at a conceptual level, a more detailed description of already implemented parts of the execution semantics of WSMX is given in [[Oren, 2004](#)].

Following we briefly comprehend the architecture and the scope of WSMX and give a review on related knowledge needed for exemplifying the specific use cases. The current architecture, represented in a preliminary framework, is aligned to WSMX_O [[Cimpian et al., 2004](#)], a conceptual model based on the WSMO-Lite [[Roman et al., 2004b](#)] ontology (a minimal subset of WSMO-Standard [[Roman et al., 2004a](#)]), plus some additional concepts. The final scope of WSMX is the domain defined by WSMO-Standard. The first version of WSMX provides a complete architecture for dynamic discovery, mediation, selection and invocation and a simple but complete implementation of these components. In the current status it is not applicable to fulfill the all aspects of the execution requirements mentioned in these use cases. Subsequent versions of WSMX will incorporate the ongoing research of the WSMO and WMSL working group and will lead WSMX to a status where it fulfills all the Semantic Web Services support

described in this document.

Hence this deliverable is intended to serve as an input for further improvements and providing valuable insight for adapting or including components to deal with real world scenarios for Web Services. On the other hand this deliverable is intended to evolve in accordance to the ongoing development of WSMX itself and further use cases will be included in the future if necessary.

1.3. Document Overview

The use cases we will carry out are divided into two business domains. The B2C and the B2B use case are abstracted to Trading Partner integration. Both deal with the same purchasing process, but with different actors and roles. Application Integration contrariwise exemplifies a use case within a company which raises the demand for a so called intra-Enterprise integration [Bussler, 2003].

The document is organized as follows. [Section 2](#) briefly describes the WSMX architecture components and in particular figures out which components will be addressed during the use cases. Additionally the section points out which components have to be considered to be added to fully satisfy all revealed needs in the use cases. [Section 3](#) describes the purchase of a bike, firstly in a B2C and secondly in a B2B collaboration. Both scenarios are first described in general, whereas the actors, roles and goals are defined, then the overall system architecture is depicted and finally the execution flow of WSMX is outlined. [Section 4](#) describes an Application Integration scenario which exemplifies an A2A use case. It depicts a purchase of stocks which is automatically invoked by an application running in one of the banks division and processed and executed by another division within the same entity. [Section 5](#) finally concludes the document.

2. WSMX architecture components

The following chapter gives only a very brief description of the components. This is solely indented to give the reader the chance to follow the process steps in the use cases without knowledge of other WSMX deliverables. Given the fact that some components are very specific and cover some technical requirements, they are not exemplified in the use case and therefore not specified in this chapter. For further information on all components we point the interested reader to the following deliverables [Oren et al., 2004] [Moran/Zaremba, 2004].

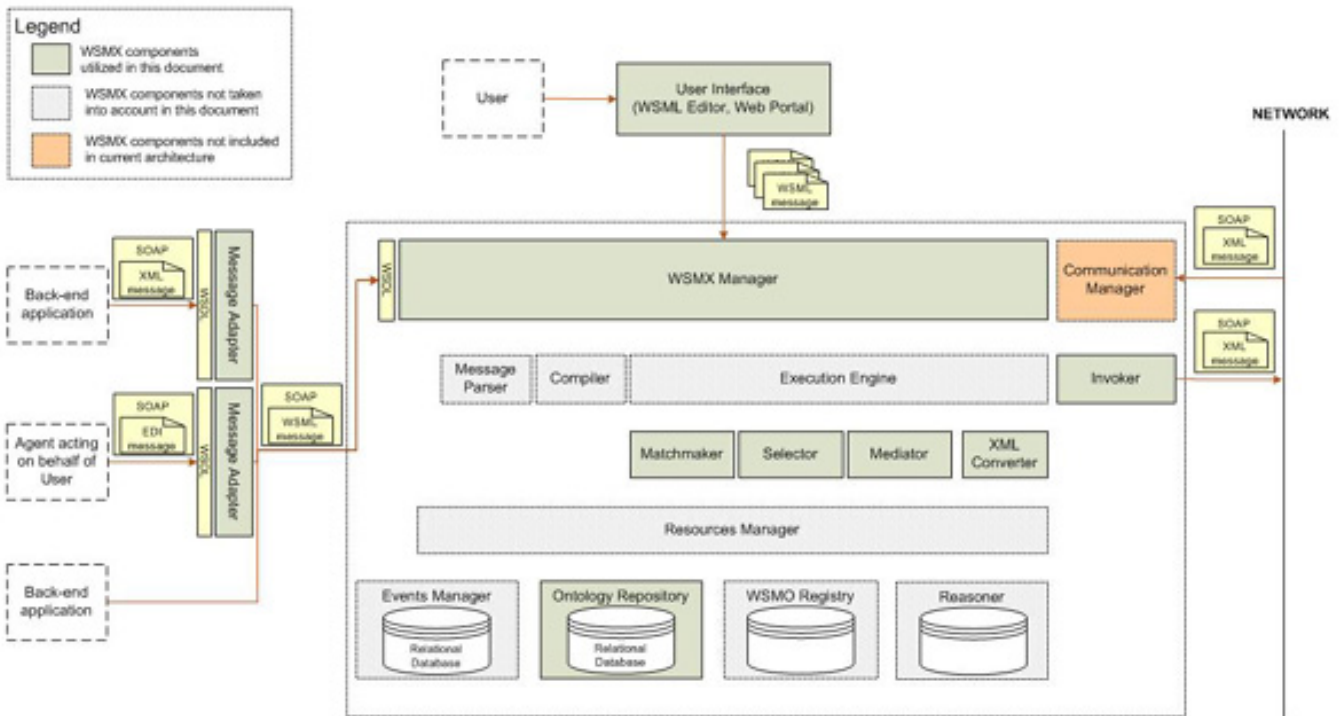


Figure 9: WSMX Architecture (click to enlarge)

- Message Adapters:** These components are in fact externally to the WSMX architecture, but as long as back end applications don't offer a WSML API they are essential for connecting any kind of back-end application. The Message Adapters allow transforming from any message format (e.g. RosettaNet, UBL, EDIFACT, ANSI X12, xCBL etc.) to the expected WSML message format.
- User Interface:** This component is used to create new WSML definitions. Since the user (customer) will not provide their goal in logical terms there has to be some tool support. In the following B2C use case we will differ between two possible architecture variants how the user can provide the input; either by means of a web portal or in a specially designed WSMX editor. More information on these two approaches can be found in [Section 3.1.4](#).
- WSMX Manager:** This component is the coordinator within the architecture. Although it is not an essential component to describe the use cases, it is mentioned here because it denotes the Service Oriented Architecture (SOA) approach of WSMX. All data handled inside WSMX is represented internally as an event with a type and state. The WSMX manager manages the processing of all events by passing them to other components as appropriate.
- Ontology Repository:** WSMX will offer the management of capability descriptions stored in a repository. This repository could be centralized or decentralized, whereas this use case and the current architecture only scopes with a decentralized repository in each WSMX. These repositories are designed to store, search, retrieve and manage WSMO descriptions of semantic web services. Within this document the name Capability Repository is synonymously used for Ontology Repository.
- Matchmaker:** WSMX will offer a set of usable semantic web services by matching capabilities stored in the repository with the goal provided by the user. It is even possible that the fulfillment of the goal is achieved by the composition of several semantic web services. In both cases the result of the matchmaker can be zero, one or many web services.
- Selector:** WSMX will offer dynamic selection of the discovered semantic web services

during the matchmaking process based on specified preferences of non functional criteria, including properties like quality of service, reliability, etc.

- **Mediator:** WSMX will offer mediation of communicated data. The mediation component tries to determine a mediator for a request in case this is necessary. This mediation can be between two or more ontologies in the matchmaking process and the opposite way after invocation to mediate between the instance data of a known ontology provided by the executed web service to the required data in the invoking application. Another application of mediators would be to map between the data provided in the input of the goal to the actual required input of the web service.
- **XML Converter:** This component can be invoked to translate the results of the Mediator (a message in logical terms) into XML. This is necessary as the web service invocations are via SOAP and the message format for SOAP messages is XML.
- **Invoker:** WSMX will offer invocation of the selected semantic web service. These web services are bound to SOAP in their WSDL file. Hence the invoker has to send a SOAP request to these services. It receives the mediated data between the goal described in logical terms and the actually required XML it receives from the XML Converter.
- **Communication Manager:** This component is not included in the current framework, but it is considered in the next WSMX release. To allow a request-reply message flow, the Communication Manager is inevitable. It is responsible for picking up every message produced after invocation of the web service. As long as this information does not denote an error during invocation the data is wrapped with the post-conditions stored in the Capability Repository and passed to the adapter. There the mapping to the required format of the requesting application takes place.

3. Trading Partner Integration

The following use cases are high level descriptions of business collaboration. Hence we currently ignore issues like communication security, transaction management, error handling and compensation. Furthermore we imply in these scenarios that all participants, exposing business functionality as a web service, agree to fulfill the real world interactions. Hence we imply that these business partners trust each other, although they maybe conduct the first business case. Precisely the business partners don't rely on existing Service Level Agreements (SLA) between them. We will take this into consideration in further versions of this document as well as in improvements in the WSMX architecture. It is currently planned that SLAs between two partners will be taken into consideration in the non-functional properties of the goal and capability description.

The choice to describe a purchase of bicycles in the following use cases has been made deliberately, mainly because they represent non-commodity goods. Commodity goods on the other hand are traded primarily on the basis of price and not on differences in quality or features. Manufactured goods are said to be commodity goods if purchasing decisions are made almost solely on the price of the product. By choosing the bicycle in our use cases we obviously have to deal with selection criterias beyond the price of a good.

3.1. B2C

3.1.1. What is B2C

B2C stands for "business-to-consumer" and applies to any business or organization that sells its products or services to consumers over the internet. The most commonly known B2C e-commerce market participant is the book retailer Amazon.com, which launched its website in 1995. However, in addition to solely online based retailers, also bricks-and-mortar companies have included many B2C services on their websites, such as online banking, travel services, auctions, betting and accommodation search.

3.1.2. Use case foundation

Let us imagine a Bicycle retailer, further called BCR, which provides the disposal of several kind of bikes, including mountain-, racing-, city- and juvenile ones, over internet. The following scenario is not depicted by using a traditional retailer with bricks-and-mortar branches, a fixed number of business relations with manufacturers and wholesalers and tightly coupled applications, but with one that enhances its online business model by applying Semantic Web Services to it. By utilizing WSMX as the means of achieving this enhancement, the BCR can enlarge its formerly limited number of bicycle types and brands offered for sale. In some respect the BCR migrates to a central knowledge repository regarding purchases of bicycles. If this BCR also operates real world branches this has no influence on the use case itself.

Naturally the business model we describe below does not necessarily have to be offered by a bicycle retailer. It can also be some company provideing a web service repository covering a broader domain than only the bicycle one. To simplify matters and in respect to basic economic effects of specialization we assume that a retailer will offer this supplementary functionality first.

Fundamentally we can differ between two service models the BCR can offer to its customer.

1. The first one is coherent with current approaches in E-Commerce. The BCR runs a **portal website**, but unlike ordinary B2C portals, this website differs by utilizing semantic web services. The user is not only able to search for different bicycles from different vendors and browse through the resulting offers to decide, if one fits to their needs, but they are also able to provide the BCR a specific goal in formal semantics. How the user can provide the goal via the portal is discussed in [Section 3.1.4](#). The goal represents the user's tangible needs, based on which WSMX will find web services capable of fullfilling the user's goal.
2. The second one would be that the BCR only **hosts a server running WSMX** and the customer uses a client side version of a WSMX editor. This editor allows the user to define a goal and connect to a WSMX server afterwards. WSMX tries to match this user goal with web services offering the desired service.

Excursus: In the latter service model the BCR could in fact become dispensable. WSMX will be able to connect to capability repositories of other WSMX's to search for their capability store. This means for our example, if all bicycle vendors utilize WSMX as their web service execution environment or at least provide a WSMO compliant description of their web service in any WSMX or central repository, the customer's client goal editor would be able to find this web service by connecting to any arbitrary WSMX. For further descriptions of our use cases we assume the existence of a BCR which hosts capability descriptions of several services from different manufacturers. This will be the most likely scenario in the near future, as far as a central repository of semantic web services (like Google is for websites) or a central trading platform for semantic web services (like Ebay is for consumer goods) is not established so far.

3.1.3. Actors, Roles and Goals

Within the B2C view we focus on two different actors. The following listing defines them in detail, why they are interacting (goal), with whom and in what way (role) and what assumptions are essential.

- **Customer:**
 - *Goal:* buys a city-bike
 - *Role:* the customer interacts with the BCR for service usage, payment and acceptance of the bike in the requesting role
 - *Assumption:* either uses a WSMX editor or utilize the functionality offered by the portal website of the bicycle retailer
- **BCR:**
 - *Goal:* provides services to customers by aggregating the product range of different bike manufacturers; maximizes profit by selling as many bikes as possible at a profitable price.
 - *Role:* acts as an intermediate between the customers and different bicycle manufacturers interacting with its customers via a user interface (online based portal) or by providing a WSMX repository accessible over internet.
 - *Assumption:* either provide web portal allowing customers expressing their goals or running WSMX as a standalone server for direct access by the customers editors.

3.1.4. B2C system architecture

Since the BCR can offer its service in one of the two ways mentioned above and the fact that the user's goal has to be described in WSML [Oren et al., 2004b], we have to differ between two system architectures as described below. The initialization of these goals in a formal description like WSML has to be as user friendly as possible. This should be reflected in the utilized end user tools.

[Include graph with: Box 1: customer - wsmx editor; Box 2: retailer - portal site - wsmx; Box 3: manufacturer wsmx/wsdl - back end applications

Three Tier Architecture: Client/Browser - Web Server (Application Server) - WSMX Server
[include graph]

This architecture type reflects the requirements of the first service model in 3.1.2. Within this architecture we can differ between two possible implementations of portal site in the middle tier:

- The first and unlikely case is that the retailer only provides the means to input the goal in WSML. Practically we can rule out this solution in the long run, because the end user (customer) should not be bothered with expressing their goal in a formal language. This might be applicable as a second choice for experienced users, but not as the only means to describe the goal.
- The second implementation would allow the user to change the postConditions and effects patterns of predefined goals. The task for the user would simple changes to some variables in the hidden logical description of the goal according to their requirements. This can be either goals provided by the BCR retailer or goals already included in the release version of the WSMX repository itself. Both are represented on the portal website.

Two Tier Architecture: Client Editor (WSMX editor) - WSMX Server [include graph]

This architecture type reflects the requirements of the second service model in 3.1.2. The user can easily define their goal with a GUI based editor. This editor provides the means to graphically represent logical expressions in a way that the user only has to deal with some kind of symbols. These symbols can be linked with other ones with several kinds of relationships and provided with properties.

3.1.5. How WSMX can be utilized

The following paragraph describes the collaboration between the user and the BCR retailer and specifies which specific component in WSMX is utilized to perform the required task. The collaboration between the BCR and the manufacturer is not presented here, since it is discussed in detail in the following B2B scenario.

The above mentioned customer has a goal to buy a bicycle. Let us assume an average skilled biker, with no preference regarding the manufacturer of the bike. Their only conditions on the effects in the sense of WSMO are as follows:

- Buy a city bike,
 - with a price less than 400 €,
 - with dirt deflectors at the front and rear tire,
 - and with at least 21 gears.
1. First the customer has to express this goal, as mentioned above in one of the following three ways:
 - The customer composes the formal description directly in WSML. Hence the BCR has to provide some API to its WSMX. A simple textbox in their portal for adding WSML goals would perfectly be enough.
 - The BCR offers predefined goals from which the user is able to choose one to align it to the actual requirements. This could be realized with a form on their portal website describing every box in a way that it expresses logical terms in a natural language.
 - Another solution would be a client side editor tool connecting remotely to any known WSMX environment providing the means to describe the user's goal. This tool can be designed in different ways; it should include a way to use WSML directly, reusing goals (e.g. by means of templates) and it should avoid using any formal logics by providing a symbol/icon based composition process.
 2. The message containing the goal described in WSML is picked up by WSMX. The way this message is sent, depends on where the user defined their goal. This can be in the case of the portal the web server generating a message picked up by an adapter providing the WSML message or in the case of the client side editor this component directly invokes an API of the manager component.
 3. Ignoring the parsing, compilation, persistent storage etc. of the message, the WSMX manager now invokes the matching component to find all the web services that provide the capability that satisfies the user's goal. In the current specification the matching component just selects all the capabilities that are syntactically identical to the goal. Further developments will take logical reasoning into account and therefore provide the means to find as many matching web services as possible [Keller et al., 2004]. For web services that use a different ontology than the ontology used by the goal, mediation is

needed. The required interaction between the matchmaker and the mediator is disregarded in this high level use case.

4. If the outcome of the matchmaker is not nil or exactly one web service the selector component has to find on the basis of the non-functional properties and the user preferences web service fulfilling the user's goal best. In this B2C scenario and in case of a non-commodity good like the bicycle, this will most likely not happen without human interaction, because a user can't describe for example their preferred design of the bicycle in a formal way.
5. The selector component starts an interaction via the manager component with the user providing them the list of web service, rated and ordered by measuring the non-functional properties and taking the user preferences into account. For each web service the provided capability description is passed through the adapter. How this information is displayed at the end user interface (again either the Browser or the Client side Editor) is at the responsibility of the transformation process in the adapter.
6. The user selects the desired web service, the new goal is created and the message is again passed through the adapter to the manager component and further on to the selector.
7. Selector is now able to select the desired web service and passes this information to the Invocation component. Finally the invocation component tries to invoke the selected web service by using the originally provided input. Since in the B2C scenario we can expect a user interaction and the matching process has been conducted without matching the input with the pre-conditions this input will normally be insufficient or even empty. Hence the invoker has to start an interaction with the user. Again the message is passed by the manager to the adapter and then displayed at the end user GUI.
8. The user provides the input; the message is passed through the adapter and the manager component back to the invoker.
9. The invoker could now perform the execution of the web service and picks up the result. Given the fact that the output should comply to the logical expressions given in the capability description the adapter can perform a mapping between this output and the required one again in the GUI.

3.1.6. Advantages WSMX exposes for the B2C use case

Advantages for the customer: The main advantage for the customer is the fact that they don't have to search through several on- or offline catalogues to find bicycles fulfilling their goal. WSMX will present them only web services which exactly offer them the possibility to order the desired bicycle. If exactly the same bike is offered by different providers WSMX will select according to the user preferences the cheapest one. Hence the customer can be sure that there is no provider on the market (at least one who exposes their web service capability in a WSMX repository) who offers the bicycle at a cheaper price.

TBC

Advantages for the BCR: The advantages for the BCR are even more manifold. First the BCR could avoid to tightly coupling their end user shopping portal (again either the web portal or the Capability Repository) to specific manufacturers/wholesalers. WSMX dynamically orders the bicycle from the chosen provider. Under the above mentioned assumption that the providers of business functionality via a web service agree to fulfill the real world transactions independently who invokes it (even without having a SLA), this can be even one, the BCR was not aware of before. The customer or in fact the BCR (as mentioned above they should provide the means to

describe a goal in logical terms) can use different ontologies for describing goals. WSMX will find even capabilities using different ontologies than used in the goal as long as there is a mediator available mapping the two ontologies. Again this allows a loose coupling and the BCR does not have to use predetermined ontologies impressed by a strong wholesaler or manufacturer.

TBC

3.2. B2B

3.2.1. What is B2B

Business-to-Business (B2B) and B2C differs only in one thing. The customers are different - B2B customers are other companies while B2C customers are individuals. Overall, B2B transactions are more complex and have higher security needs. Beyond that, there are two big distinctions:

- **B2B sales require negotiation**

Unlike B2C scenarios, where a company only has to publish its catalog online, selling to another business is much more sophisticated and involves negotiation over prices, delivery and product specifications.

- **Integration**

Automation is the main advantage of online B2B. Therefore companies selling to businesses have to integrate their systems or provide a standardized interface to allow communication with their customers without human intervention.

3.2.2. Use case foundation

As mentioned in point 3 we ignore the negotiation between two business partners and imply that everyone can use the web service with the same conditions. But the second differential between B2C and B2B, "Integration", changes the interaction flow between the participating parties, namely the BCR and the manufacturing companies, in comparison to the flow between the customer and the BCR fundamentally. Since applications are interacting with each other and any user interaction should be avoided, the matchmaking process and further on the selection and invocation have to fulfill completely other requirements.

In this scenario we will change the role of the "Bicycle retailer" from the only service oriented view in the B2C use case, where they only act as an intermediary party to the more common role of a Bicycle retailer as a selling point with warehousing. In contrast to the above described B2C use case, for the B2B one, it does not matter, if the BCR only sells their bikes traditionally offline or also make them available online, either as a semantic web service or only via an ordinary online portal.

3.2.3. Actors, Roles and Goals

Within the B2B view we focus on the two following actors. The following listing defines the actors, why they are interacting (goal), with whom and in what way (role) and what assumptions are essential.

Bicycle retailer: acts as a retailer with bricks-and-mortar branches. Provides high-quality services to customers by offering them a wide variety of bicycles, most likely available in their stores.

- *Goal:* provide reliable and high-quality services; maximize profit by selling as many bikes as possible at a profitable price.
- *Role:* consult customers in its branches, (optionally provides online based portal or provides a WSMX repository accessible over the internet - see 3.1.)
- *Assumption:* either WSMX installed in its system architecture or ensure a tightly coupling of their back end applications with WSMX adapters provided by manufacturers

Manufacturer: a commercial company that produces different kind of bicycles

- *Goal:* maximize profit by selling as much bicycles as possible, at or more likely above break even price
- *Role:* provides purchasing service to other companies
- *Assumption:* exposes service either as WSDL compliant web service or even has WSMX installed in their system architecture with determined interfaces

3.2.4. B2B system architecture

For the fact that a bicycle retailer is at the far end of a supply chain, normally their business relations are mainly based on pull interactions. This means they rather invoke services via interfaces at their closest business partner (e.g. wholesaler, manufacturer) than to expose services to them. Hence, except in sophisticated supply chain management (SCM) scenarios, the bicycle retailers have not to offer these interfaces to their business applications e.g. for automatic purchase ordering, delivery inquiries etc.. In excluding these SCM scenarios and assuming that the manufacturer has installed WSMX by any means (normally manufacturers expose several web services, hence it is easier to register them once in their own repository) we have to consider two architecture designs for our BCR.

Architecture ERP system - WSMX - back-end-application: [include graph]

If the retailer exposes no web services itself (only uses the ones from the wholesalers and manufacturers) and don't participate in an automated Supply Chain as mentioned above, there is no explicit need to have WSMX installed. Hence they would obviously lose the advantages offered by WSMX for internal integration as described in [Section 4](#). If they choose to set WSMX aside they have to tightly couple their ERP-system with different WSMX adapters of different manufacturers.

Architecture ERP system - WSMX - WSMX - back-end-application: [include graph]

If they do offer their services as a web service to their end consumers (simplified this scenario is equivalent to offer services like the above mentioned ones to other business partners) or want to profit from the advantages WSMX provides even in A2A scenarios there arises the need for WSMX installed at the BCR.

3.2.5. How WSMX can be utilized

Since the two above mentioned system architectures differ fundamentally, we have to describe them separately. Starting with the first architecture (no WSMX installed at the BCR) the use case looks as follows.

1. To best serve their customers needs and to have the broadest available range of bicycles available in stock, the BCR operates an inventory management system with automatic inventory replenishment. Let's assume the BCR operates an SAP R3 system with the Material Management (MM) module installed. The inventory management system now initiates a purchase order every time the stock of inventory under-run the recorder point. Because the BCR retailer has not installed WSMX, every purchase in SAP R3 is directly linked to the WSMX adapter interface of a certain manufacturer or wholesaler, which sells the specific bike. This means SAP R3 sends a message containing the goal to purchase a specific bike to the default linked WSMX.
2. WSMX installed as the online front-end in the system architecture of the manufacturer receives the message and acts as an automatic message delivery system based on the goal defined in the received message. WSMX finds based on the capabilities defined in the repository, which application is responsible and capable of picking up and processing the purchase order. Because the BCR does not make use of WSMX and tightly couple its own SAP R3 system with an adapter provided by the manufacturer, the message type has to comply with the requirements specified by the manufacturer.
3. The matchmaker picks up the logical expression and tries to find a matching capability. Due to the tight coupling between the two parties and the a priori agreement which capabilities the manufacturer can offer the BCR, the matchmaker will find none or exactly one service fulfilling the BCR goal. This mainly because one functionality within a company is (or should) usually only be offered by one application. The case that no web service fulfilling the capability can only occur if the manufacturer has deleted the agreed capability from its WSMX repository. Since there will also be an agreement on the inputs that the BCR has to provide to invoke the certain web service, the matchmaking process will not restrict the results on basis of the input - precondition match.
4. The selection is quite straightforward based on the assumption that one capability is only provided by one application within the manufacturer's system architecture.
5. After the selection the invoker sends the SOAP message with the given inputs to the web service of the selected application determined by WSMX.
6. The Communication Manager picks up the result of the invocation and sends it back to the adapter. Since the inputs to WSMX have to be known by the BCR and the outputs comply to the logical expressions given in the capability description the adapter to the SAP R3 MM tool can process this data.

In the second case - having WSMX installed in the BCR's and the manufacturer's system architecture - we can exploit the full potential of WSMX in B2B scenarios.

1. Again we assume that the BCR operates an SAP R3 system with the Material Management (MM) module installed. But this time the ERP system is not linked to the WSMX adapter interface of a certain manufacturer, it is connected to an adapter of its own WSMX. This means SAP R3 sends a message containing the goal to purchase a specific bike to the BCR's own WSMX.
2. WSMX receives the message and starts the discovery process.
3. The matchmaker tries to find a capability stored in its own WSMX repository. This process would be comparable to the tight coupling to specific manufacturers mentioned above, since the repository is maintained by the BCR and only known web services can be stored. But there are two main differences.
 - o There can be multiple results of the matchmaking process since the BCR can have

included various web services capable of fulfilling the same goal. This means for example that the BCR always uses the cheapest offer to buy the ordered bike, as long as they specify this condition in their preferences. In the tight coupling scenario above the bike would always be purchased from the same manufacturer, even if for example a wholesaler offers the bike cheaper because they want to empty their stocks.

- The even more important distinction is that the BCR's WSMX will remotely consult other WSMX's, if it is not able to find matching web services in its own repository or the retrieved number of web service is smaller than defined in the preferences or in the WSMX settings. This has mainly one big advantage. If the BCR has deleted one of the capabilities offered by a specific manufacturer or wholesaler (bankruptcy, issues led to a cancellation of the business connection, etc.) their SAP system is probably still be able to proceed a automatic replenishment order. The WSMX matchmaker will perform the process of finding capabilities matching the goal not only by matching the desired and provided post-conditions and effects of the web service but also by taking into account the provided inputs and match them with the pre-conditions of the web service. This is crucial to facilitate an automated execution of the whole process; otherwise WSMX would have to ask for additional inputs and an undesired user interaction would follow.
4. This time WSMX can either find none, one or many web services. Therefore the selector has to pick the service which based on the given preferences and the determined non-functional properties fit the agents (ERP application as agent of the BCR) requirements best.
 5. After selecting the matched web service the invoker sends the SOAP message with the given inputs to the selected application determined by WSMX. This can be of course either another WSMX or an application exposing its functionality by means of a WSDL interface. Since the matchmaker has only chosen web services for which the provided input is enough to successfully guarantee the execution, the desired effect and post-conditions are obtained.
 6. The Communication Manager picks up the the results of the state-change of the information space and passes the message to the adapter to the ERP system. Given the fact that the output should comply to the logical expressions given in the capability description the adapter can perform a mapping between this output and the required one by the SAP R3 MM module. If the execution of the web service was achieved remotely by another WSMX the process would change slightly. The input would be picked up by the Communication Manager of the remotely invoked WSMX which again has the description of this data stored in logical terms in the repository, as long as the capability description comply to it. Due to the fact that there exists a mediator between the used ontology of the web service providing back end application, which was invoked by the remotely WSMX, and the used ontology in the goal description (otherwise there would have been no match in the matchmaking process) the adapter of the initiating WSMX can process this data and feed the back end application with it.

3.2.6. Advantages WSMX exposes for the B2B use case

Again we have to break down the advantages regarding the chosen architecture model. Starting with the first one the BCR obviously would not exploit the functionality of WSMX by coupling the BCR's SAP R3 system tightly with one WSMX. Hence there only arise **advantages for the manufacturer:**

- Firstly not any retailer will tightly couple their systems with one manufacturer. Therefore the installation of WSMX supports, under the premise of a registration of its web service capabilities in its own repository, the automatic discovery of the capabilities by any other WSMX.
- Secondly the manufacturer is not bound to provide the functionality by one specific web service. If they change the back-end application, the web service will still be found, as long as the Capability Repository is updated and the inputs remain the same. Even if the inputs change it is still easier for the BCR to adapt their system to the new conditions.

In the second architecture model the advantages for both parties are manifold, but obviously for the manufacturer they are overlapping with the currently described ones. Hence we will only present the advantages now arising for the BCR:

Advantages for the BCR:

- The BCR avoids to tightly coupling their back-end application with a specific manufacturers/wholesalers system. WSMX dynamically finds a web service in other WSMX's to purchase the given good from this provider. The same as in the B2C case applies here; under assumption that the providers of business functionality via a web service agree to fulfill the real world transactions independently who invokes it, this can be even one the BCR was not aware of before.
- Similar to the above mentioned advantages in the B2C use case, the BCR can use different ontologies for describing their goals. WSMX will find even capabilities using different ontologies than used in the goal as long as there is a mediator mapping the two ontologies available. Again this allows a looser coupling and the BCR has not to use predetermined ontologies impressed by a strong wholesaler or manufacturer.
- By using WSMX the BCR is not only able to find capabilities of other companies, it help them also to expose their own web service functionalities. Therefore for example it would be possibly to advertise sophisticated SCM related web services (like the sales of one specific bike in a period, type of the most popular bike etc.) in their own WSMX Capability Repository which are then traceable and executable for downstream companies.

4. Application Integration

Application integration is a buzzword that has been widely used in the past several years, but what exactly does it describe? Enterprises use different back-end applications that need to exchange data. Since they are in most cases originally not designed to exchange data, the need to integrate them arises. To avoid manual intervention to transfer data into one system, which is stored in another one, software technology is needed in order to transfer this data automatically. The following use cases should exemplify on a very high level how WSMX can be used as an integration tool. Technical issues like the particular implementation of adapters, the type of message, which communication channels are used etc. are not in the scope of this document.

4.1. A2A

4.1.1. What is A2A

In contrast to the B2B integration above, application-to-application integration is a subset of B2B integration and therefore the foundation for successful interactions between two companies in a B2B scenario [Bussler, 2003].

Bussler defines A2A as an integration which refers to the integration of any number of back-end application systems that are hosted and require integration. This includes the integration of back-end application systems over a network that resides in a remote division of the enterprise. Therefore A2A is every integration of back-end applications where the boundaries of the company as a legal entity are not crossed. WSMX will act in this scenario as a system commonly referred to as an Enterprise Application Integration system. Although WSMX will not offer business process management it still covers the two main functionalities which initially made up the foundation to be called an EAI tool [Stonebraker, 2002]:

- message transformation
- adapters

As mentioned WSMX currently does not exploit the functionality of a business process engine, which is referred as compulsory to be called an EAI system by some authors [Linthicum, 1999] [Ruh et al., 2000]. WSMX will offer a Business Process Management (BPM) engine in further versions. Additionally WSMX will offer goal decomposition. This means that the execution of a web service by matching an agents goal would also succeed if the capability is not matched by one singular web service, but has to be composed by the usage of several ones. For the agent there is no apparent difference if their goal is fulfilled by one or more web services (if it is a single action or a composed one).

Nevertheless the required orchestration and choreography will be part of future WSMX implementations. As soon as these issues are addressed in WSMO and in the architecture we will also include it in this use case scenarios.

4.1.2. Use case foundation

For our application integration use case we will exemplify the purchase of stocks within a bank operating in several countries with different departments dealing with investment funds. A division managing a global hedge fund, further called HFD (hedge fund division), wants to buy stocks. Typically purchases of stocks on different markets within a bank are managed by so called Settlement divisions. In large-scale banks there might be not only one Settlement, but several ones in different countries with overlapping functionality. Similarly the different funds are managed in different departments (e.g. pension funds in different departments than hedge funds), either by outsourced sub companies or independently entities within the bank.

4.1.3. Actors, Roles and Goals

Within the A2A integration we focus on the two following actors. The following listing defines the actors, why they are interacting (goal), with whom and in what way (role) and what assumptions are essential.

- **HFD:** a division managing the banks portfolio of hedge funds
 - *Goal:* maximize the return of the committed money by lowering its transactional costs and very simplified by trying to be long in securities outperforming the market and be

short in the ones underperforming the market.

- *Role*: provides the management of a variety of hedge funds.
- *Assumption*: WSMX is used by every application in this division to communicate with other applications of other divisions within the bank.
- **Settlement division**: Provides high-quality bank internal transaction management to carry out purchase and sale orders on different security markets.
 - *Goal*: provide reliable and high-quality services; minimize transaction costs.
 - *Role*: provide services as web service as well as via offline communication.
 - *Assumption*: WSMX is installed in the settlements system architecture and every application exposes its services via web services. Furthermore these web services are registered in the Capability Repository of WSMX.

4.1.4. A2A system architecture

Architecture: proprietary trading platform - WSMX - WSMX - back-end settlement system
[include graph]

To exploit the full potential of WSMX in A2A scenarios the company has two architecture options. Firstly it can implement WSMX on top of its system architecture in a global scale and introduce a policy that every division has to expose every web service capability in this central repository. Furthermore the adapters to back-end applications are built by the global IT department. Because this approach to introduce integration platforms on a strategic level requires a tightly cooperation between several IT departments and divisions in general, it proved to be not successful in the past in several cases in EAI system implementations. Since WSMX can act in a distributed manner, we assume for our use case that every single division operates its own WSMX.

As the term A2A implies, two applications have to communicate with each other. Additionally to these two applications, in our case a proprietary trading platform in the hedge fund division and a settlement system installed in the homonymous division, as mentioned WSMX runs in both divisions. Hence both divisions use WSMX as their mean to expose functionality either over the intranet or the extranet (internet). This assumption is solely made to utilize the repository of each WSMX. From the architectural point of view the use case would be nearly similar if only the HFD has WSMX installed and the settlement division only exposes the functionality of their back-end application as a web service. To be traceable this web service must be registered in at least one WSMX within the company. Since it is more likely that the division would do this registration in software installed in their own system architecture and as WSMX provides a central point of integration if they have to connect the settlement system to other systems within the division, we assume that WSMX is installed in the HFD as well as in the settlement division.

4.1.5. How WSMX can be utilized

Let's assume that the above mentioned HFD (hedge fund division) wants to buy one million shares of the Bridgestone Corporation, listed on the NIKKEI index and traded at the Tokyo Stock Exchange (TSE).

1. The proprietary application managing the portfolio of the HFD was configured in a way that it automatically has to order one million shares of the Bridgestone Corporation if they share price drops below 10 Yen. Since the application is configured to avoid tight coupling with specific applications, it sends the goal to buy one million Bridgestone

- shares at the current market price at the TSE to the WSMX adapter.
2. WSMX receives the message and starts the discovery process.
 3. The matchmaker first tries to find a capability stored in its own WSMX repository. Thus this is the first transaction in this stock the WSMX repository will most likely not include a stored capability description. Hence WSMX will try to match with capabilities stored in other WSMX. In contrast to the B2B use case this communication will be restricted to WSMX's running within the entity bank. This restriction is not only set to tailor this use case to an A2A one, but in fact because the bank would strongly prefer to handle the order in a division belonging to itself. Only if the matchmaker could not find any web service capability within the bank which offers to buy this share on the specified market WSMX would enlarge to outside capability repositories. The further steps in this case are the same as described above.
 4. WSMX can similar to the B2B use case either find none, one or many web services. Therefore the Selector has to pick the service which based on the given preferences and the determined non-functional properties fit the agents (proprietary trading platform) requirements best. Mostly even in an intra company interaction the price of the web service will be the most important factor to determine the selection process, because although the money costs for execution remain within the company, it is added to internal cost accounting of the HFD.
 5. After selecting the matched web service the invoker sends the SOAP message with the given inputs to the settlement system. Since the matchmaker has only chosen web services for which the provided input is enough to successfully guarantee the execution, the desired effect and post-conditions are obtained.
 6. Similar to the second scenario in the B2B use case the Communication Manager of the remotely invoked WSMX would pick up the information. Again there exists a mediator between the used ontology of the web service providing settlement system, which was invoked by the remotely WSMX, and the used ontology in the goal description. Therefore the adapter of the initiating WSMX can process this data and feed the trading platform with it.

4.1.6. Advantages WSMX exposes for the A2A use case

- As it is common in large-scale companies that divisions are not aware of all the functionalities other divisions offer computer process able within applications, the first main advantage of WSMX is to find these capabilities as long as they are stored in a WSMX repository. In the same process WSMX can execute them and helps to reduce the time for completing the full process fundamentally by avoiding any user interaction.
- Even if divisions would know where to find a specific functionality (it is quite likely that the HFD would know which settlement division is capable to process this trade), they don't have to know which specific application in the settlement division has to be addressed to invoke the trade. They only have to provide the goal together with the input and let WSMX find the address of the required application.
- Since WSMX don't rely on global ontologies like this is often the case with global EAI platforms, the several division can use their own ontologies in expressing the capabilities of their web services and nevertheless they are still traceable by other divisions using different ontologies describing their goals.
- As mentioned in the B2B use case the settlement division is not bound to provide the functionality by one specific web service. If they change the back-end settlement system, the web service will still be found, as long as the Capability Repository is updated and the

inputs required by the web service remain the same.

5. Conclusions and Further Directions

TBC

References

[Bussler, 2003] C. Bussler: B2B Integration, Concepts and Architecture. *Springer-Verlag*, 2003, ISBN 3-540-43487-9.

[Cimpian et al., 2004] E. Cimpian, A. Mocan, M. Moran, E. Oren, M. Zaremba: WSMX Conceptual Model. *WSMX Working Draft v0.1*, 2004. Available from <http://www.wsmo.org/2004/d13/d13.1/v0.1/>.

[Keller et al., 2004] U. Keller, R. Lara, A. Polleres, H. Lausen, M. Stollberg, M. Kifer: Inferencing Support for Semantic Web Services: Proof Obligations. *WSML Working Draft v0.1*, 2004. Available from <http://www.wsmo.org/2004/d5/d5.1/v0.1/>.

[Linthicum, 1999] D. S. Linthicum: Enterprise Application Integration. *Addison-Wesley*, 1999, ISBN 0-201-61583-5.

[Moran/Zaremba, 2004] M. Moran, M. Zaremba: WSMX Architecture. *WSMX Working Draft v0.1*, 2004. Available from <http://www.wsmo.org/2004/d13/d13.4/v0.1/20040622/>.

[Oren, 2004] E. Oren: WSMX Execution Semantics. *WSMX Working Draft v01*, 2004. Available from <http://www.wsmo.org/2004/d13/d13.2/v0.1/>.

[Oren et al., 2004] E. Oren, M. Zaremba, M. Moran: Overview and Scope of WSMX. *WSMX Working Draft v0.1*, 2004. Available from <http://www.wsmo.org/2004/d13/d13.0/v0.1/20040611/>.

[Oren et al., 2004b] E. Oren (Ed.): BNF grammar for WSML language. *Working Draft v0.2*, 2004. Available from: <http://www.wsmo.org/2004/d16/d16.1/v0.2/>.

[Roman et al., 2004a] D. Roman, H. Lausen, and U. Keller: Web Service Modeling Ontology Standard. *WSMO Working Draft v02*, 2004. Available from <http://www.wsmo.org/2004/d2/v02/20040306/>.

[Roman et al., 2004b] D. Roman, H. Lausen, E. Oren, R. Lara (eds.): Web Service Modeling Ontology - Lite (WSMO - Lite). *WSMX Working Draft v0.1*, 2004. Available from <http://www.wsmo.org/2004/d11/v0.2/>.

[Ruh et al., 2000] W. Ruh, F. X. Maginnis, W. J. Brown: Enterprise Application Integration: A Wiley Tech Brief. *John Wiley and Sons*, 2000, ISBN 0-471-37641-8.

[Stonebraker, 2002] M. Stonebraker: Too Much Middleware. *ACM SIGMOD Record* 31(1): 97-106, 2002.

Acknowledgement

The work is funded by the European Commission under the projects [DIP](#), [Knowledge Web](#), [Ontoweb](#), [SEKT](#), [SWWS](#), [Esperanto](#), and [h-TechSight](#); by [Science Foundation Ireland](#) under the DERI-Lion project; and by the Vienna city government under the [CoOperate](#) program.

The editors would like to thank to all the [members of the WSMX working group](#) for their advice and input into this document.



[webmaster](#)