Adaptive technologies to address operational complexity in highly configurable value chains

Ray Richardson[1], Aidan Boran[1], Tomas Vitvar[2], Paavo Kotinurmi[2]
David Lewis[3], John Keeney[3], Declan O'Sullivan[3]

Bell Labs, Ireland[1],
firstnamelastname@lucent.com
Digital Research Institute, Galway[2],
firstname.lastname@deri.org
Knowledge and Data Engineering Group, Trinity College, Dublin[3]
firstname.lastname@cs.tcd.ie

Abstract Existing B2B infrastructure is primarily focussed on the secure, reliable, and scaleable transfer of information between business partners. The time and effort taken to establish these B2B connections has meant that the resulting business relationships tend to be long-term and rigid in nature. More recently, however, the configurability of the value chain connecting business partners has been seen as a key to competitiveness. There is increasing pressure to establish more transient ad-hoc relationships whereby dynamic decisions can be made to, for instance, exchange one partner with a more competitive alternative or to introduce new trading partners to improve the robustness of an organisations business model. This new dynamic business model introduces considerable complexity both in the need to deal with heterogeneous partner interfaces and the need to support dynamic decision-making. In this paper we explore how semantic web service technology can be combined with policy-based management to infuse adaptivity into existing B2B infrastructure. The discussion focuses on the use of a high performance policy decision engine to deal with operational complexity introduced as a result of dealing with new business partners.

1. Introduction

In today's business world outsourcing, off-shoring, and partnering have combined to create the virtual organisation where value-adding activities from multiple organisations are connected to form value chains, [1]. There has been considerable investment in efforts to optimise the operational efficiencies of these value chains. Heavyweight e-business frameworks such as EDI, RosettaNet, cXML, and ebXML have been deployed to allow each organisation in the chain to view and modify information inside their own systems rather than have buyers, sellers, assemblers, and engineers work over the phone, fax, or e-mail, [3]. A consequence of these considerable investments to connect businesses is the fact that partnerships, once established, tend to be rigid and long term. More recently, however, the configurability of the value chain has been seen as a key to competitiveness. For many organisations there is increasing pressure to establish more transient ad-hoc relationships whereby decisions can be made to, for instance, exchange one partner with a more competitive alternative or to introduce new business partners in an effort to lessen an organisations dependence on one or more trading partners [1,5].

This dynamic business model introduces considerable complexity both in the need to deal with heterogeneous partner interfaces and the increased operational complexity associated with dynamic decision-making, [7]. Organisations participating in these fluid value chains need a highly adaptive B2B infrastructure to address the additional complexity. In our research we are investigating the use of an integrated semantic web service (SWS) and policy-based management approach to infusing adaptivity into existing e-business framework implementations. Semantic web service technology is used to mediate process and data conflicts within partner interfaces. Policy based management techniques are used to automatically or semi-automatically enforce human governance on dynamic decisions relating to service selection, pricing, levels of service, and so on. Specific contributions of the paper include:

- an overview of a system architecture which incorporates an integration of semantic web services (SWS) technology, widely deployed e-business frameworks, and policy engineering techniques (section 3.1).
- an account of how the approach delivers adaptivity in a specific use case scenario (section 3.2)
- an insight into how a high performance decision engine can be utilised to to enforce organizational policies in decision making related to service selection, parameter setting, and constraint enforcement (section 4).

2 Problem domain - Use Case Scenario

To illustrate the problem domain we consider a use case in which a small-scale supplier, organisation A, supplies widgets to a considerably larger electronics manufacturer, organisation B. Organisation A and B have a long term B2B relationship with fixed terms (pricing, service level, shipping locations, etc..)

and IT support through a RosettaNet e-business framework. Organisation A would like to lessen its dependence on organisation B as its main customer and has recently been approached by other electronics manufacturers requesting quotes for its widget product.

The nature of this new business is somewhat different insofar as requests are more ad-hoc and terms more variable, i.e. pricing, service level, shipping locations, and so on can vary greatly. Specific difficulties exist in that the B2B interfaces to each of the electronics manufacturers are different and additional overheads exist due to the requirement to make on-the-fly decisions on pricing, levels of service, etc..

Organisation A's plans to diversify its customer base and grow production is good for business but requires significantly increased adaptivity in its B2B function. The organisation needs to ensure that profits from its new business are not wiped out by the overhead associated with managing the additional operational complexities.

2.1 Interface heterogeneities

The option to standardise B2B interfaces across its customers is not realistic for Organisation A, primarily due to the company's size relative to its customers. B2B interface heterogeneities occur across three layers – network, data, and process. The network layer is perhaps the most straightforward to deal with as many of the manufacturers are willing to provide smaller suppliers with B2B software clients which can handle the encoding and transportation of messages according to their standard protocol.

Data conflicts are more troublesome. Even for manufacturers sharing the same e-business standard it's quite common for data conflicts to arise. For instance one manufacturer may expect contact phone numbers that include area and country codes whereas another simply expects phone numbers to have an area code. Perhaps more serious would be a situation where two manufacturers have different standard units of measure for the same product (e.g. one uses a 5 pack whereas another uses 10 pack). These mismatches are only apparent by inspecting the content of messages or even worse as a result of an investigation following unexpected events.

Process heterogeneities relate to differences in the specific message exchange sequences. One manufacturer may issue multi-line orders as a series of individual requests whereas another may bundle them together into a single request. Again even within the same e-business standards differences can arise.

2.2 Dynamic decision making

In its current business Organisation A has long term agreements with its customer base (Organisation B). Decisions on pricing, levels of service, order volumes, and so on are agreed up front for a period of 12 to 24 months. These 'variables' can be plugged into existing supporting B2B infrastructures. In the extended business model relationships are more ad hoc in nature and operational decisions are much more dynamic.

Human involvement in all decision making is an expensive solution which does not guarantee consistent policy enforcement and fails to scale well. Hard coded application logic or the use of configurable parameters leads to static policies that can be difficult to extend and often result in sub-optimal decisions due to poor models of the real world or poor use of all available relevant information. There is a requirement to support truly dynamic adaptive decision making.

3 Highlevel Architecture Overview

Our approach to addressing the particular demands of the use case scenario is to implement an integration of SWS technology, policy based management techniques, and existing e-business frameworks.

3.1 Overview of Solution

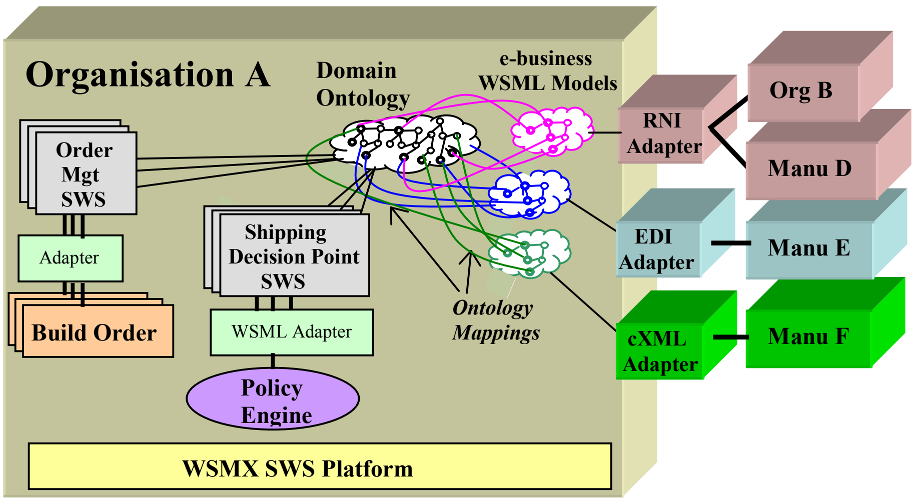

Fig 1: High-level view of architecture

Key elements of the architecture include:

- WSMX - we make use of the Web Services eXecution Environment (WSMX) as our SWS platform, [2]. It provides core support for semantic service discovery, data meditation, process mediation, and service invocation. WSMX resides entirely within organisation A. This allows us to avoid making any assumptions on the semantic technology capability of partner organisations. WSMX makes use of the Web Services Modelling Language (WSML) [8] for all internal processing.
- e-business framework ontologies – these ontologies are flat WSML representations of e-business framework (RossettaNet, ebXML, EDI, etc.) message contents. XSLT is used to automatically construct these ontologies from an XML schema representation of the e-business standard.

```
…
<ProductLineItem>
   <UnitOfMeasureCode>12-pack <UnitOfM../>
   <LineNumber>1</LineNumber>
   <requestedQuantity>
       <ProductQuantity>10</ProductQuantity>
   </requestedQuantity>
…
```

```
…
concept productLineItem
 nonFunctionalProperties
   dc#title hasValue "…"
 endNonFunctionalProperties
 lineno ofType (0 1) _integer
 unitcode ofType UnitOfMsre
 qty ofType  (0 1) Quantity
```

- e-Business Adapters – these exist to translate or 'lift' e-business standard messages into a WSML format, making use of concepts defined in the e-business framework ontologies. For messages going in the opposite direction the WSML concepts are 'lowered' to become e-business standard messages. These adapters are also responsible for creating WSML goals on receipt of 'kick-off' messages.

- Semantic web service descriptions representing back-end information systems (Order Mgt., Shipping, etc..) within organisation A. The descriptions eliminate semantic ambiguity by binding input and output parameters to specific concepts within an accompanying domain ontology. Another adapter exists to 'lift' and 'lower' messages received and sent between back office systems and their corresponding semantic web services.

```
webService OrderMgt

importsOntology { _"http://www.orgA.com/OM" }

capability OrderMgtSWSCapability
 sharedVariables {?request}

 precondition
  definedBy
   //A request to create an order
   ?request memberOf mn#createOrderRequest or
   //A request to add a lineitem to an order
     …

 postcondition
        …
 interface OrderManagementInterface
 choreography OrderManagementChoreography
   stateSignature
       …
```

- Design time ontology mappings - these mappings identify equivalences and relationships between concepts in each of the e-business ontologies and the domain ontology. A data mediation tool exists to support the process.
- The Policy decision engine is used to enforce organisational policies in decision making processes. Decision points are exposed as semantic web services using concepts from the domain ontology (or possibly another ontology linked to the domain ontology via mappings). The policy decision engine is discussed in greater detail in section 4.

3.2 Simple walk through

In this section we provide a simple walkthrough of the architecture described previously. For the walkthrough we assume we are dealing with a purchase order request received from a customer who utilises RosettaNet. We further assume there are backend services to both build an order (*ProcessOrder*) and to deal with shipping. As part of the decision to broaden its business activities org A has introduced a range of external shipping functions that can be used in place of its internal shipping function for certain situations. The basic flow of activity for processing a purchase order is shown below.

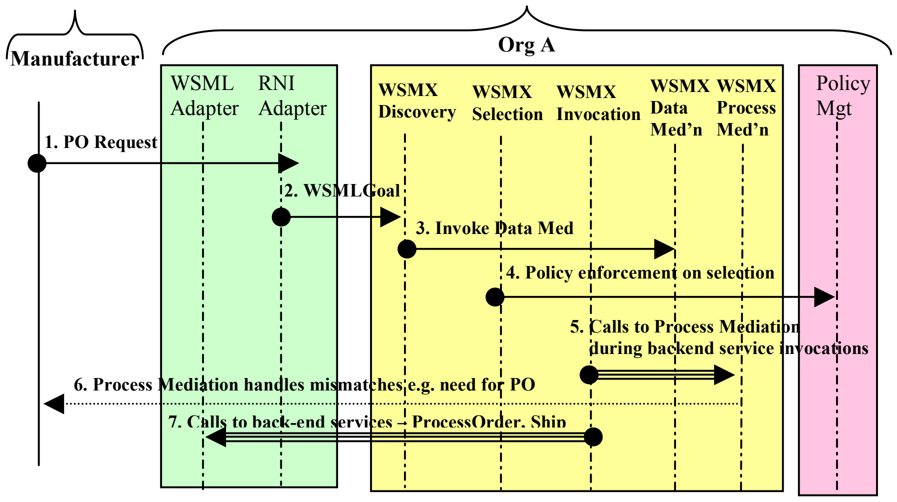
Manufacturer
Org A
WSML Adapter
RNI Adapter
WSMX Discovery
WSMX Selection
WSMX Invocation
WSMX Data Med'n
WSMX Process Med'n
Policy Mgt
1. PO Request
2. WSMLGoal
3. Invoke Data Med
4. Policy enforcement on selection
5. Calls to Process Mediation during backend service invocations
6. Process Mediation handles mismatches e.g. need for PO
7. Calls to back-end services – ProcessOrder, Ship

Fig 2: High-level walk through

Key points include:

- RosettaNet purchase Order request message, PIP 3A1, is 'lifted' by the RNI adapter into an equivalent WSML format. The receipt of this message triggers the adapter to formulate a WSML goal from a set of pre-defined templates. The goal seeks to have a purchase order processed.

- The WSMX Discovery component matches the WSML goal against existing service capabilities. There may be capabilities to quote a price, to create an order, to ship an order, etc.. This match may be quiet simple in practice with inputs/outputs simply checked – data mediation is invoked where necessary. In some cases it may be the case there is no direct match between a goal and the available service descriptions. For instance in this example there is no single service to process an order. However by composing the Build Order and one of the Ship Order services the goal can be achieved.
- The service selection step, in particular which shipping service to utilise can be partially or fully controlled by policy based management. Further detail on how this may take place can be found in section 4. The use of policy based management in this way introduces a significant level of adaptivity into operations management and provides a consistent scaleable solution to the increasing operational complexities found when dealing with additional business partners.
- The mappings defined between the base and domain ontologies are executed as part of the SWS invocations. The mappings ensure RosettaNet concepts/attributes such as *AddressLine1, CityName,* etc., are appropriately translated into their backend equivalents, e.g. *Street, City,* etc..
- Process mediation is a further service offered by the WSMX environment. WSMX is capable of analysing the process choreographies of the goal and the individual service descriptions to identify and reconcile process heterogeneities. For instance the RossettaNet message process purchase order expects to receive an acknowledgement following the issue of a purchase order message (PIP 3A1). This acknowledgment may not be provided by the back-end BuildOrder service, the process mediator is thus responsible for auto-generating the acknowledgement message. Abstract state machines are used at runtime to keep track of process executions.
- During the service execution WSML individuals will be lowered into a message format that can be consumed by the Order Mgt and Shipping back-end services.

4. Policy based management

In our research we propose policy-based management to support dynamic decision making. Declarative rules are used to enforce organizational policies in decision making related to service selection, parameter setting, and constraint enforcement. The Vortex rules engine [10] is being used as the decision engine. Vortex is a high performance, acyclic, forward chaining rules engine that supports reasonably rich policy management for real time environments.

Policies, in their simplest form, are event-condition-action rules. Correspondingly Vortex rules have a simple :

```
If(Condition) then
        action1,
        action2,
        action3,
         ...
    end
```

format. The rules language is strongly typed with support for both atomic and complex typed variables. Vortex is packaged with an extensible range of support functions that can be called from any rule condition or action. Permitted actions include assigning a value to a variable, appending a value to a list variable and removing a value from a list variable. Rules are organized into what are known as rule sets, i.e. the set of rules that should be used for a given 'decision request'. Each rule set has an explicit input/output signature. From an architectural perspective we expose rule sets as individual services with ontologically bound input/output parameters. In some cases data mediation may be necessary as part of decision request processing.

In order to provide an insight into how the decision engine is utilised we build on the scenario developed in previous sections. In this scenario one of the decisions required is to select the most appropriate shipping options. Organisational policies should be adhered to in compiling these options. Information from the purchase order and candidate shipping services are forwarded with the decision request – it itself being a WSML goal. The decision engine is capable of issuing requests to external sources to retrieve additional information required to evaluate the conditions of all rules. The information returned from the decision service consists of a set of shipping options that comply with organisational policies pertaining to the shipping request. A human administrator may make the ultimate shipping decision from this short list of valid choices. Alternatively the selection may be based on some simple criteria such as the cheapest conforming shipping service.

Rule sets begin with a declaration of input, output, and intermediate variables. In our simplified *shipping decision* rule set input variables include the list of concrete shipping candidates, the name of the purchasing organisation, the time the shipment will be available for pickup, the shipment destination, etc..

```
variables:
          purchasingOrgName : string;
         availableForPickup : string;
         shipmentDestination :  list Record of  { location : string };
         shippingCandidates  : list of Record { identity : String;
                                                pickup_Time : string;
                                                pickup_Date : string;
                                                cost : string;
                                                setdown : string; };
```

An adapter takes care of lowering WSML concepts to become input variables. In some cases the set of input variables are extended as a result of additional domain knowledge held in the ontology. For example a single shipmentDestination of Kista would have the additional locations of Stockholm and Sweden added as Kista is located in Stockholm which is in turn located in Sweden. This expanded list of locations results in more robust rule sets.

Intermediate variables are used to store temporary values during the rule set execution. In some cases these temporary values are populated as a result of rule actions to retrieve information from external sources.

```
shipmentChannel : string;
internalShippingCapacity : string
filteredFromInHrs, filteredFromPerf, filteredFromPickup,
filteredFromPerferred, blacklisted :
                list of Record { identity : String;
                                 pickup_Time : string;
                                 pickup_Date : string;
                                cost : string;
                                setdown : string;
                                 priority : string; };
  onTimePerf  : list of Record {identity : string;
                                              channel : string;
                                              perf_Rating : string; };
 preferredVendorList  : list of Record {identity : string;
                                              channel : string; };
```

The single output variable in this case is the list of shipping options that adhere to all organizational policies.

```
validShippingServices  : list of Record { identity : String;
                                           pickup_Time : string;
                                           pickup_Date : string;
                                          cost : string;
                                          setdown : string;
                                           priority : string; };
```

The actual rules are typically organized into groups with the initial group setting intermediate variables, e.g. :

```
shipmentChannel = "lane1";
if(shipmentDestination[$i] == "USA" || shipmentDestination[$i] == "UK") then
            shipmentChannel = "lane2"
end
```

The *shippingDestination[$i]* syntax leads to an evaluation of the rule for each member of the *shippingDestination* list value.

Subsequent rule groups actually enforce the organizational policies. In our simplified scenario we assume the following policies exist :

1. A preferred vendor list exists for each shipping lane. Company policy states for any given shipment the selected shipper must be on the preferred list for the shipments shipping lane.
2. A shipper must have an on time performance of greater than 95% for the the shipping channel in question
3. Shippers are required to make pickups within regular hours
4. The pickup cannot be more than 4 days after *availableForPickup* date
5. Shipments for organization B take priority in the case of the internal shipment service

Generally speaking policies act to filter the allowable list of shipment services. The corresponding rules for each of the policies are presented below :

```
rule: Rule_1
if(shippingCandidates[$i].identity == PreferredVendorList[$j].identity &&
      PreferredVendorList[$j].ShippingChannel == shipmentChannel)
          filteredFromPreferred += ShippingCandidates[#i];

rule: Rule_2
if(filteredFromPreferred [$i].identity == onTimePerf[$j].identity &&
      onTimePerf [$j].channel == shipmentChannel && onTimePerf [$j] > 0.95)
          filteredFromPerf  +=  filteredFromPreferred[#i]

rule: Rule_3
if(Time::between(filteredFromPerf[$i].pickupTime, "08:00", "18:00"))
        filteredFromInHrs += filteredFromPerf[#i];

rule: Rule_4
if(Time::numberOfDaysBetween(filteredFromInHrs[$i].pickupDate,
                                              availForPickup) < 4)
      filteredFromPickup += filteredFromInHrs[#i];
```

```
rule: Rule_5
if(filteredFromPickup[$i].identity == "Internal" &&
            internalShippingCapacity < 0.2 && requestingOrg != "Org B")
    blacklisted += filteredFromPickup[#i]

rule: Rule_6
if(! (filteredFromPickup[$i] in blacklisted ))
    validShippingServices += filteredFromPickup[#i]
```

Relating organisational policy semantics to the semantics used to define both back-end systems and partner interfaces has obvious benefits in ensuring policy constraints operate as expected. By enforcing policies in service selection and parameter setting an organisation can flexibly and consistently control how it interoperates with partners. Semantically encoded policies are themselves more adaptable to change and heterogeneity and are considerably easier to encode. For instance a policy that states "*during public holidays pickups must take place between 9:00am and 12:00am*" can take advantage of domain knowledge for what constitutes a public holiday to simplify the encoding. Data mediation further enables policies to adapt to heterogeneity, e.g. a concrete service description might encode a pickup time using a 24 hour format in place of the standard 12 hour clock used internally. A mediator can automatically mediate this

5. Conclusions and future work

In this paper we have presented some of the problems facing organisations attempting to participate in configurable value chain partnerships. Increased adaptivity is required within the B2B function to address interface heterogeneities and operational complexities introduced by the more dynamic business model. An integration of SWS technology and policy-based management are proposed to deliver the required adaptivity. Internally deployed semantic web services are employed to address data and process heterogeneities present in partner interfaces. Semantically encoded policies are used to ease difficulties associated with the dynamic decision-making.

The focus of our work is currently on investigating the options available to integrate the policy management and semantic web services onto a single platform, preparing an evaluation framework for the architecture, and building supporting tools. Specific tools in the policy management space include a GUI tool to support the creation of rules and pre-processors to expand rule sets based on domain knowledge.

References

1. T. Friedman, The World is Flat: A Brief History of the Twenty First Century, Farrar, Straus and Giroux, 2005.
2. Haller, E. Cimpian, A. Mocan, E. Oren, and C. Bussler. WSMX – A Semantic Service-Oriented Architecture. In Proceedings of the 3rd International Conference on Web Services, pages 321 – 328, Orlando, Florida, USA, 2005.
3. Medjahed, B. Benatallah, A. Bouguettaya, A. H. H. Ngu, and A. K. lmagarmid. Business-to-business interactions: issues and enabling technologies. VLDB Journal, 12(1):59–85, 2003.
4. G Olsen, An overview of B2B Integration, eAI Journal, May 2000, p 28-36.
5. Y Sheffi, The Resilient Enterprise - overcoming vulnerability for competitive advantage, The MIT Press, 2005.
6. M. Kerrigan, The WSML Editor Plug-in to the Web Services Modeling Toolkit. In *Proceedings of 2nd WSMO Implementation Workshop (WIW2005)*. Innsbruck, Austria, 2005.
7. C. Preist, J. E. Cuadrado, S. Battle, S. Williams, and S. Grimm. Automated Business-to-Business Integration of a Logistics Supply Chain using Semantic Web Services Technology. In ISWC '05: Proceedings of 4th International Semantic Web Conference, 2005.
8. J de Bruijn, H. Lausen, and D. Fensel, The WSML Family of Representation Languages, http://www.wsmo.org/TR/d16/d16.1
9. E Cimpian, and A. Mocan, Process Mediation in WSMX, http://www.wsmo.org/TR/d13/d13.7/v0.2
10. Richard Hull, Francois Llirbat, Francois Llirbat, Eric Simon, Jianwen Su, Guozhu Dong, Bharat Kumar, and Gang Zhou, *Declarative Workflows that Support Easy Modification and Dynamic Browsing*, International Joint Conference on Work Activities Coordination and Collaboration (WACC) held in San Francisco, February, 1999, pp. 69-78.