A Semantic Service Orientated Architecture for the Telecommunications Industry

Alistair Duke, John Davies, Marc Richardson, and Nick Kings

BT Exact, Next Generation Web Research, Adastral Park, Martlesham Heath, Ipswich IP5 3RE, United Kingdom
{alistair.duke, john.nj.davies, marc.richardson, nick.kings}@bt.com

Abstract. A Service Orientated Architecture will allow organisations to enhance interoperability and encourage reuse of components and interfaces. In this paper, the application of semantic descriptions to services is advocated with the aim of further improving the SOA and enabling scalability. An application of Semantic Web Services for the Telecommunications Industry is described. It shows how services components forming part of a Service Orientated Architecture can be described semantically in terms of shared data and process ontologies. The potential benefits of this approach are explored. A use case is presented that illustrates how the efficiency of a telecommunications system designer can be improved with the use of Semantic Web Services.

1 Introduction

The Service Oriented Architecture (SOA) has emerged as a way in which organisations can enable interoperability and encourage reuse, thereby reducing cost. The greater agility it affords will also allow organisations to respond to the needs of the market more quickly and in ways that are more attractive to the customer. The SOA is particularly applicable to the Telecommunications market where customer and operational support costs are high and customer satisfaction is a key differentiator. However, industries such as Telecommunications with complex internal organisations and supply chains are finding that a scaleable SOA is not achievable without semantic descriptions of services that can aid service discovery and integration.

The Telecommunications Industry is also seeking ways to encourage interoperability at a business-to-business level. One such approach is the New Generation Operations Systems and Software (NGOSS) initiative from the TeleManagement Forum [1].

This paper examines an approach to combine the SOA with the models of the NGOSS by creating semantic descriptions of services and system interfaces expressed in terms of data and process ontologies derived from NGOSS. The intention is to create explicit links between service components and a commonly understood view of the industry allowing improved service discovery and service integration. A scenario based around a solution designer carrying out a product assurance integration task is presented. The paper examines how Semantic Web Services can be used to enhance

A. Aagesen et al. (Eds.): INTELLCOMM 2004, LNCS 3283, pp. 236–245, 2004.

the efficiency of the designer. Finally, an analysis of the approach is presented which examines the applicability of NGOSS as a domain ontology and the capabilities of existing Semantic Web Services initiatives for supporting a Service Oriented Architecture in the telecommunications industry.

2 Semantic Web Services for OSS

Standard information models are a key element of flexible and low cost integration of Operational Support Systems. Those developing and adopting these models will benefit from consideration of emerging semantic web standards which can make explicit the semantics of the data to aid integration and understanding. A common information model with explicit semantics is a key element to a Service Oriented Architecture, since only with semantic descriptions of services will a degree of automation be achievable for service discovery and composition.

This section will explain the proposed benefits of web services described semantically in the context of a common information model for the OSS domain. In order to do this, the limitations of current web services are first considered.

Web Services are generally described using XML-based standards namely WSDL [1] (which allows one to describe a web service in terms of what it does and what its inputs and outputs are), UDDI (which is a centralised registry allowing one to discover web services) and SOAP (which is a protocol allowing one to execute services). In addition to these low-level standards, work is on-going it create standards that allow services to be combined into a workflow e.g. WS-BPEL[2] (Web Services-Business Process Execution Language) and also to define permissible message exchange patterns and contents e.g. ebXML[3]. However, none of these standards provide a means to describe a web service in terms of explicit semantics. For a given service you might want to describe what kind of service it is, what inputs and outputs it requires and provides, what needs to be true for the service to execute (pre-conditions), what becomes true once the service has executed (post-conditions) and what effect the service has on the state of the world (and/or the data it consumes and provides).

The first of these requirements is partly addressed by UDDI in that a category and human readable description can be assigned to a web service in a registry to aid discovery. This provides only limited support for automated discovery since a computer will not understand the description or what the category means. The second and third of these requirements are partly addressed by WSDL in that XML tags can be attributed to inputs and outputs. A computer can easily match these but again has no notion of their meaning or relationship to other pieces of data.

Services can be described semantically by relating them to ontologies. The explicit relationship between services and ontologies is the key element for Semantic Web Services. It is envisaged that this will enable:

---

[1] http://www.w3.org/TR/wsdl

[2] http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/

[3] http://www.ebxml.org/

- Improved Service Discovery

Semantic Web search technology allows users to search more precisely on ontological concepts or concept value rather than by keyword.

- Re-use of Service Interfaces in Different Products / Settings

Services that are described semantically can more easily be discovered, understood and applied thus reducing the need to create new services that serve the same purpose.

- Simpler Change Management

One example of how semantics can help here is when a proposed change is made to a data element, those services or interfaces that employ that data in some way can be dynamically discovered and appropriate action taken.

- A browseable, Searchable Knowledge Base for Developers (and Others)

This would allow developers and solution providers to perform queries relating to the data and processes they were concerned with, for example to determine the origin piece of data or its destination.Semi-automatic service composition

Given a high level goal which we wish a service or set of services to achieve, expressed in terms of an ontology, it should be possible to carry out decomposition into components parts and then match these components with appropriate services.

- Mediation between the data and process requirements of component services

Often there is need for two or more services to interact even though their communication requirements are semantically the same but syntactically different. In this case it should be possible to automatically construct a translation between message data elements that allows the services to communicate.

- Enterprise Information Integration

The Semantic Web should afford universal (or at least enterprise-wide) access to semantic descriptions of services (or information). One advantage of this is the ability to answer complex queries without having to consider how to access the various systems where the data required for the answer is held.

3 Use Case Scenario

The project scenario is based around a solution designer who, given a high-level requirement, wishes to compose a set of web services that will allow the requirement to be met. The scenario assumes that a set of services exist and that they are described semantically and related to a common information model. However, there is no clear approach for forming explicit links that will allow some degree of automation when moving from the model to the service description and vice-versa. Figure 1 illustrates how an explicit link can be created with the adoption of a set of technologies or approaches. These are now briefly described.

The work of the TeleManagement Forum in developing a framework for Next Generation OSS can be seen as ontology building in that NGOSS provides a level of shared understanding for a particular domain of interest. NGOSS [1] is available as a toolkit of industry-agreed specifications and guidelines that cover key business and technical areas including Business Process Automation delivered in the enhanced Telecom Operations Map (eTOM™) [2] and Systems Analysis & Design delivered in the Shared Information/Data Model (SID) [3].

The eTOM and SID have been considered in this work as ontologies in that they can provide a level of shared understanding for a particular domain of interest. The eTOM provides a framework that allows processes to be assigned to it. It describes all the enterprise processes required by a service provider. The SID provides a common vocabulary allowing these processes to communicate. It identifies the entities involved in OSS and the relationships between them. The SID can therefore be used to identify and describe the data that is consumed and produced by the processes.

Fig. 1. Semantic Service Descriptions create an explicit link

OWL-S is a OWL-based Web service ontology, which supplies Web service providers with a core set of mark-up language constructs for describing the properties and capabilities of their Web services in unambiguous, computer-interpretable form. OWL-S mark-up of Web services will facilitate the automation of Web service tasks, including automated Web service discovery, execution, composition and interoperation [4].

The OSS through Java (OSS/J) initiative provides a 'standard set of Java technology-based APIs to jump-start the implementation of end-to-end services on next-generation wireless networks, and leverage the convergence of telecommunications and Internet-based solutions' [5]. It is used here because it provides a set of telecommunications interfaces close to those that would be provided by services in a SOA.

The application area of the use case is trouble ticketing. The scenario is that given a service alarm, the service problem should be resolved while keeping the customer informed of progress. The goal is met by designing a composed service from a number of component services. In the scenario, these component services will first be discovered and then integrated in an appropriate manner according to their descriptions. The scenario will illustrate the benefits of ontological support by making use of process and data ontologies. In order to satisfy the goal it will be necessary to employ services that will create a trouble ticket to manage the resolution of the problem and create a task in a workforce management system to ensure that the problem is addressed. In the first part of the scenario consists of the following five steps:

1. A network problem results in an alarm being triggered. This is captured by the process manager.
2. The process manager reads the alarm to determine the affected resource. It then carries out a request to an inventory manager to determine the customer affected by the resource.
3. The inventory manager responds with details of the customer.

4. The process manager requests that the Trouble Ticket system creates a new trouble ticket and provides details of the problem.
5. The Trouble Ticket system creates a new ticket (and informs the customer of the problem) then responds to the Process Manager with an ID for the created ticket.

The scenario allows the services and the messages they require to be determined. For example the Inventory Manager might expose a service getCustomerID which requires and input message of a resourceID and provides an output message containing either a customerID or an error code.

4 Domain Ontologies

One of the aims of the use case is to make use of existing ontologies that exist for the telecommunication sector and understand how they can be used to enhance service descriptions. This section describes the modelling work of the TeleManagement Forum then illustrates how this can be converted to OWL for use in the case study.

In order to make use of the eTOM and SID within the project, it was necessary to express them in a formal ontology language i.e. OWL [6].

The eTOM and SID are subject to ongoing development by the TMF. The current version of the eTOM (3.6) is expressed in a set of documents although there are plans to provide a clickable HTML version (a previous version is already available in this form) and an XML version. The SID is also expressed in a set of documents but is also available as a set of UML class diagrams.

The eTOM can be regarded as a Business Process Framework, rather than a Business Process Model, since its aim is to categorise the process elements business activities so that these can then be combined in many different ways, to implement end-to-end business processes (e.g. billing) which deliver value for the customer and the service provider. [2]. The eTOM can be decomposed to lower level process categories e.g. 'Customer Relationship Management' is decomposed into a number of categories, one of which is 'Problem Handling'. This is then decomposed further into categories including 'Track and Manage Problem'. It is to these lower level categories that business specific processes can be mapped. Each category is attributed with metadata giving a name, a unique identifier, a brief and extended description and a set of known process linkages (i.e. links to other relevant categories).

The SID [3] is much more complex than the eTOM in both its aims and form. It provides a data model for a number of domains described by a collection of concepts known as Aggregate Business Entities. These use the eTOM as a focus to determine the appropriate information to be modelled. The SID models entities and the relationships between them. For example a 'customer' is defined as a subclass of 'role'. It contains attributes such as 'id' and 'name'. It is linked to other entities such as 'CustomerAccount' with an association 'customerPossesses'.

5 Semantic Service Descriptions

5.1 Service Grounding

The case study makes use of OSS/J interfaces. Although OSS/J has yet to be adopted commercially, extensive work by the OSS/J consortium has gone on to ensure that it meets the requirements of product vendors and consumers in delivering interfaces at the appropriate level. In order to make use of the interfaces in this case study, it was necessary to wrap them as WSDL web services since OWL-S only supports a grounding to WSDL.

In OWL-S, a service grounding creates a link between the semantic description of a service and the service itself which is described in WSDL. One aim of the use case is to illustrate discovery and composition at the level of WSDL operations e.g. getCustomerID. For this reason, the decision has been made to model operations as OWL-S services. This is because a service has only one service profile, which is the means by which discovery is carried out. If a WSDL service had been modelled as an OWL-S service, then the profile would not allow advertisements of the operations within the service to be made.

5.2 Service Model

The Service Model describes what happens when the service is executed. The Process Model is a subclass of the Service Model and gives a detailed perspective of the service. The Process Model describes a service in terms of inputs, outputs, preconditions, effects, component sub-processes, and aims at enabling planning, composition and agent/service interoperation.

The service getCustomerID identified in section 3 is an example of an atomic service as it takes a number of inputs and returns a number of outputs. It maps directly to a WSDL description and can be invoked directly.

Inputs (e.g., resourceID), outputs (e.g., customerID), preconditions and effects are described separately. For brevity, inputs, outputs, preconditions and effects (iopes)

```
<process:AtomicProcess rdf:ID="getCustomerID">
  <process:hasInput rdf:resource="#resourceID"/>
  <process:hasOutput rdf:resource="#customerID"/>
</process:AtomicProcess>
```

In OWL-S, the service model allows inputs and outputs to be related to ontological concepts thus providing a frame of reference for the data requirements of the service.

Preconditions and conditional effects are described analogously to inputs and conditional outputs. Unfortunately, there is no standard way to express preconditions within OWL-S although placeholders for these have been provided in the OWL-S ontology. The proposed Semantic Web Rule Language[4] and related initiatives will provide such expressions.

[4] http://www.daml.org/2003/11/swrl/

In order to specify preconditions for the atomic services in the use case it is first necessary to consider the scenario in terms of the states that can exist between receiving a service alarm and closing a trouble ticket. These states can be characterised by the things that must be true for that state to exist. In the scenario, these things are embodied by the existence of data in variables or the value of those variables. Naturally, the variables are exactly the input and output data that is consumed and produced by the atomic processes. The conditions can be seen as postconditions of the preceding process and preconditions of the following process.

Where conditions are not provided, it is up to the designer to ensure that the correct input is provided to the generic WDSL operation. With Semantic Web Services it should be possible to specify the process to the extent that the designer no longer has the ability to add services to a composition in such a way that their preconditions or input requirements cannot be met.

Similarly, preconditions on the state of variables can be expressed. For example, in the scenario, it may be the case that the state of the trouble ticket can only be set to 'CLOSED' if the current status is 'CLEARED'. The following represents this requirement:

```
<process:AtomicProcess rdf:ID="closeTT_Process">
  <process:hasPrecondition rdf:resource=
    "#updateTroubleTicketStatusOutput_State_Out_CLEARED"/>
</process:AtomicProcess>
```

This requires the output from the previous process (which included an output State_Out) to be set to the required value.

In addition to preconditions, OWL-S has the notion of effects. These are the things that are true once a process has completed. For example, the effect of updating the trouble ticket once a job complete notice has been received is that the TTState is set to 'CLEARED'. That is of course if everything is correct with process e.g. that the trouble ticket ID sent is correct. The underlying WSDL operation contains an error flag that could be set if anything was wrong. Obviously, it would not be wise to set the TTState to 'CLEARED' under those circumstances. For this reason, the effects are conditional upon certain facts. In this case that the error flag is false.

The aim of the use case is to illustrate how a designer can compose services together to satisfy a high-level goal. The output of this activity will be a composed service. OWL-S allows atomic process to be composed together using a number of different constructs such as sequence, split-join, etc. The following example considers the composite process of following an alarm, collect details from the inventory manager then create a trouble ticket.

There are four possible states in this part of the process i.e. 'start', 'alarm received', 'got customer data' and 'trouble ticket queued'. The process is simplified in that it does not contain any error handling states. In the following example, if errors are received then there will be no state transition i.e. the process will return to the state that was current at the start of the attempted transition.

The following fragment shows the top level description of the composed process. The designer would name this and refer it to a service profile, allowing it to be adver-

tised. The description also includes a pointer to the start state. All other states are encapsulated within the description of the start state so this is the only reference to the actual composition that is required.

```
<process:ProcessModel rdf:ID=
     "handleAlarmWithTroubleTicket_Process">
  <service:describes rdf:resource=
      "&service;#handleAlarmWithTroubleTicketService"/>
  <process:hasProcess rdf:resource="#StartState"/>
</process:ProcessModel>
```

The start state is described below as a composite process.

```
<process:CompositeProcess rdf:ID="StartState">
  <processComposedOf>
    <process:Sequence>
      <process:components rdf:parseType="Collection">
        <process:AtomicProcess rdf:about=
            "#getAlarmResource"/>
        <process:CompositeProcess rdf:about=
            "#AlarmReceivedState"/>
      </process:components>
    </process:Sequence>
  </processComposedOf>
</process:CompositeProcess>
```

The composition for this state is a simple sequence of two processes. The first is the atomic process 'getAlarmResource' which as described earlier takes the alarm as input and outputs the resource on which the alarm has occurred. The second process is the next state in the composition i.e. the 'AlarmReceivedState'. This is another composite process which includes a selection which determines the state transition based upon the output from the getCustomerID atomic process.

OWL-S provides the <process:sameValues> construct to allow the data flow to be constructed. This allows the output from one atomic process to be aligned to an input from another. For example the troubleTicketID which is an output of createTroubleTicket can be related to the corresponding input of populateTroubleTicket.

Although useful in this simple example, the construct is limited where more complex data flow is required such as when two outputs should be combined to form one input or where an output is a complex data type from which only a portion is required.

The output of the design process would be an OWL-S composed service along the lines of that described above. The composed service could then be advertised and discovered using its own process model without regard to the atomic processes that form it.

5.3 Service Profile

The service profile describes the service in terms of what it does. It is intended to advertise the capabilities of the service allowing it to be discovered. The profile includes non-functional and functional descriptions. Non-functional descriptions cover

areas such as descriptions of the service provider, the quality rating of the service, etc. The most interesting non-functional description is classification of the service according to a domain ontology. This allows the services described in the case study to be classified according to the eTOM. The service profile exists as an instance of this class allowing it to be discovered by a matchmaking process that used the eTOM ontology.

Functional properties describe the service in terms of their iopes. These are intended to aid discovery by allowing goal services to be described in these terms. There are no encoded logical constraints between the inputs in the process model and the inputs in the profile model, therefore, at least in theory, the two sets may be totally unrelated. This is a major current deficiency of OWL-S since during matchmaking knowledge of how the iopes are used by the service would be of great benefit.

6 Conclusion

This document has considered the applicability of Semantic Web Technologies to OSS information modelling. It considers how Semantic Web Services can be applied to an OSS related scenario and which aims to create an explicit link between information models and low-level service / interface descriptions. These services have been described using WSDL wrapped OSS/J interfaces. Semantic annotations have been provided by OWL-S. Within these annotations, references are made to OWL ontologies which have been generated from the TMF's NGOSS models (eTOM & SID).

This process has allowed a number of observations to be made regarding the applicability of the Semantic Web in this field. The Web Ontology Language is in general flexible enough to capture the semantics of the TMF NGOSS models. Tool support for this process is poor. None of the major UML vendors support any Semantic Web languages.

The semantics of UML and OWL differ. One of the key barriers to the adoption of the Semantic Web is likely to be a shortage of skills. Database modellers and information architects could help solve this problem but in order to utilize them efficiently, methodologies for creating ontologies and a clear understanding of the differences between the closed world model and the open world model are required.

The TMF NGOSS initiative will provide process and data models for the Telecommunications industry. These are under development but it is clear that they are at a high level and require further modelling within a particular context if they are to perform as a domain ontology for Semantic Web Services. There is currently a mismatch between these model and underlying service components. Having said this, the eTOM provides a useful process framework for categorising processes or service functions. The SID provides a useful starting point when constructing a canonical data dictionary and/or exchange model for a particular environment.

OWL-S is an approach to allow the semantics of services to be expressed. It is currently the most concrete of the emerging initiatives in this area. OWL-S in its current form provides good support for mapping services and their data requirements (i.e. inputs and outputs) to ontological concepts. This can improve service discovery and promote a better understanding of the capabilities of a service within a wider

domain. There are a number of outstanding issues with OWL-S. Firstly, support is required for expressing rules. This will allow the preconditions and effects of a service to be expressed. Secondly, the OWL-S' process model is too simple. The minimal set of control structures provided do not have formally specified semantics. Thirdly it does not distinguish between public and private processes. Fourthly, it only supports grounding to WSDL web services. Finally it has little in the way of tool support.

Alternative approaches are emerging that are attempting to overcome these shortfalls. The Semantic Web Services Initiative[5] (SWSI) is an ad hoc initiative of academic and industrial researchers. The Web Service Modelling Framework[6] (WSMF) is a major element of the EU-funded projects, SWWS[7] & DIP[8]. DIP will develop tools and trial WSMF on three major case studies within three years (one of which is based on B2B in the ICT sector). The coupling of OSS/J to web services promises a significant set of benefits for a telecom service provider, but the maturity of the underlying technologies are insufficient at this moment in time. Currently, there is no standardised mapping from OSS/J services to web services, which is crucial when inter working between two, or more, companies. Also, WSDL does not currently define a standardised, agreed way to describe and implement asynchronous services. Both of these features, however, are under development.

Much remains to be done but Semantic Web technologies have a key role to play in the development of efficient e-Business integration.

References

1. TeleManagement Forum, "NGOSS Overview Document". Available on the web at: http://www.tmforum.org/
2. TeleManagement Forum, "Enhanced Telecom Operations Map" (eTOM) data sheet. Available on the web at: http://www.tmforum.org/
3. TeleManagement Forum, "Shared Information/Data Model (SID)". Available on the web at: http://www.tmforum.org/
4. "OWL-S Technical Overview", http://www.daml.org/services/OWL-S/1.0/OWL-S.html
5. Sun Microsystems, Inc., "OSS through Java Initiative Overview", http://java.sun.com/products/oss/overview.html
6. D. L. McGuinness and F. van Harmelen (eds), "OWL Web Ontology Language Overview". Available on the web at: http://www.w3.org/TR/owl-features/

[5] http://www.swsi.org/
[6] http://www.wsmo.org/
[7] http://swws.semanticweb.org/
[8] http://dip.semanticweb.org/