An Ontology-Based Approach to the Description and Execution of Composite Network Management Processes for Network Monitoring

José María Fuentes, Jorge E. López de Vergara, and Pablo Castells

Departamento de Ingeniería Informática, Escuela Politécnica Superior,
Universidad Autónoma de Madrid, Campus de Cantoblanco, 28049 Madrid, Spain
{Chema.Fuentes, Jorge.Lopez_Vergara, Pablo.Castells}@uam.es

Abstract. Web service technology has been proposed to implement management interfaces of managed resources. These web services can usually be combined to perform composite processes. These composite processes can be defined with service ontologies such as OWL-S, which allows their formal description. However, other technologies, including the Web Services Business Process Execution Language (WSBPEL), provide more mature execution engines. This paper presents an approach to define and execute composite network management processes with existing technology. For this, a use case is developed in which a set of web service interfaces are defined for a network probe, and a composite process is specified using OWL-S to monitor the network load. Then, this specification is later translated to WSBPEL and interpreted by a real execution engine.

Keywords: OWL-S, WSBPEL, Composite Process, Network Management, Network Monitoring.

1 Introduction

Integrated management frameworks have traditionally provided a way to use homogeneous procedures to access managed resources. However, the evolution of the networks and the services deployed on them have implied the necessity of new management mechanisms [1]. Currently, new technologies compete in the network management arena, where web services and ontologies can be used respectively for the exchange of management information and the definition of management information itself. Web services provide a maximum decoupling among components and abstraction of the inner complexities with well defined interfaces. Ontologies provide a way to formally describe the management information, avoiding misinterpretations.

Web service composition is another technology with application in network management. A set of web services can be called in a sequence to accomplish the tasks of a management application. The composition of web services can be defined formally by using service ontologies such as OWL-S that describe by a set of processes how and when to invoke these web services. However, current semantic web service tools are not mature enough to interpret such process descriptions. Then, in the meantime,

R. State et al. (Eds.): DSOM 2006, LNCS 4269, pp. 86 – 97, 2006.

another approach is needed to execute such descriptions in a similar manner, albeit less expressive than a proper ontology-based representation. For instance, Web Services Business Process Execution Language (WSBPEL) definitions can be used instead, as there exist process engines that can interpret this language.

This paper presents an approach to define and execute composite network management processes based on semantic web service technologies. For this purpose, web services and the semantic web technologies are introduced in next section. Then, the representation of composite processes with the OWL-S service ontology is presented, showing a case study for network monitoring. Later on, an approach is proposed to cope with the lack of a semantic web service execution environment by redefining the process with WSBPEL, tackling the translation issues. Finally, some conclusions are given.

2 Web Services and the Semantic Web

This section briefly describes the technologies that support this proposal to execute composite network management processes. For this a short introduction to web services is given, followed by a review of ontology-based technologies and an analysis of the confluence of both areas, in the scope of the so-called semantic web services.

2.1 Web Services in Network Management

A web service, as described in [2], is a software system designed to support the interoperable machine-to-machine interaction through a communication network. To achieve this goal, web services describe their functionality with the Web Service Description Language (WSDL) and they interact with each other by exchanging SOAP messages serialized in XML and sent over a transport protocol, usually HTTP.

The benefits of introducing a web service layer to encapsulate basic functionalities that are useful for network management have already been studied in several works [3, 4, 5], which analyze both service granularity and performance aspects. The last one of these papers points out a fundamental aspect in our study: the benefits of obtaining a common and interoperable interface to access a set of basic functionalities for network management, which can be used to build further, more complex processes.

2.2 Semantic Web and Ontologies in Network Management

The semantic web area [6] comprises a set of technologies to change current web from a network of contents and services interpreted and used by humans to a network in which such contents and services can be exploited by software agents. Among these technologies it is especially relevant in this work the use of ontologies.

An ontology is defined in [7] as “a formal specification of a shared conceptualization”. In practical terms, an ontology is a hierarchy of concepts with attributes and relations that defines a terminology to define in consensus semantic networks of interrelated information units. An ontology provides a vocabulary of classes and relations to describe a domain, stressing knowledge sharing and knowledge representation.

The use of ontologies to represent information related to the network management scope has been addressed to a significant extent in recent research [8, 9, 10, 11, 12, 13]. In the work presented here, the line started in [8] is extended by using a common representation ontology to formalize a set of specifications for network traffic monitoring. In this way, those definitions can be used to obtain a uniform access to a set of basic functions, common to different network management protocols, which will be used as a base to define a set of composite management processes based on these definitions.

2.3 Semantic Web Services in Network Management

Semantic web services are a particularly thriving area within semantic web technologies. Their objective is to provide a set of functionalities that can be understood by software systems to exploit (discovery, composition, invocation) these functionalities in an automatic or semi-automatic manner.

In this way, a set of ontologies have been defined that allows the description of these functionalities to achieve this goal. Among these proposals, the most relevant are OWL-S (OWL Services), WSMO (Web Service Modeling Ontology), and SWSO (Semantic Web Services Ontology) [14]. Although all of them share a similar semantics (they describe in the same terms of inputs, outputs, preconditions and effects the information about a functionality), the tools and methods provided by each representation are not so similar. In this paper OWL-S is used to represent the set of basic functionalities to be later exploited to obtain composite processes based on these functionalities, resuming the work described in [15]. For this, OWL-S process description is used, as detailed later. Other work [16] also proposes OWL-S for the description of network management processes. Using these OWL-S descriptions, for example, a generic management application could manage resources based on Web Services, even if it does not know *a priori* how to do it, which can be very useful in autonomic environments.

Up to this point, semantic descriptions have been introduced, but not how to implement described functionalities. A common practice is to ground the semantic descriptions on web services. Thus, a grounding between the semantic description and the WSDL description of the web service is set up, so that when a semantic web service is used, a traditional web service is finally invoked.

3 Composite Processes Representation

Starting from the perspective described in the previous section, the objective of this work is to illustrate a set of techniques to allow the description of web services related to network management. For this, a set of composite processes relevant for network management is specified. OWL-S is used for the process description, as it is presented in next subsection.

3.1 OWL-S Process Representation

OWL-S [17] allows the representation of a service as a set of interactions with other services. To represent this interaction, the ServiceModel class and its subclass Process

have been defined. They are based on existing techniques for workflow and process modeling to describe a service as a process. In this context, two kinds of processes can be distinguished: atomic processes and composite processes.

An atomic process receives an input message and returns an output message. Thus, this type of processes can be executed directly. To make it possible, each AtomicProcess class has a Grounding information associated to it, allowing a client to build and interpret the messages interchanged with the service.

A composite process is expressed as a composition of other processes (atomic or composite). This composition can be expressed by the following control structures: sequence, split, split and join, any-order, choice, if-then-else, iterate, repeat-while, and repeat-until. Other specific characteristic of these processes is the data flow. Whereas in an atomic process inputs are generated by a client and outputs are generated by the process, in a composite process, inputs can come from a client or another process, and outputs can be generated by different processes. OWL-S provides constructs to manage the control structures as well as the information flow in composite processes.

Both atomic and composite processes can have two purposes:

1. Change the environment, represented as preconditions and effects.
2. Process data (transform a certain input into a concrete output), represented as process inputs and outputs.

In OWL-S, preconditions and effects are represented as logic formulas. OWL-S does not define a default language to represent such logic formulas. However, it recommends and provides some facilities to work with the Semantic Web Rule Language (SWRL) [18], and gives a mechanism to represent those formulas in other languages. Service inputs and outputs have to be typed with a class of the related domain ontology.

With these tools, it is possible to achieve the objective of creating a complex and interoperable description, based on less complex services, to represent a composite process, which is useful in the network management scope.

3.2 Case Study: Network Monitoring

To illustrate the concepts described above, a detailed case study is provided. In this case, a network traffic monitoring process has been defined to analyze the network load. This process creates a report about network traffic for those interfaces of a probe that have a load with a value higher than a given threshold. For this, it is necessary to define the following set of elements:

- A domain ontology developed in OWL that represents the network traffic management domain. For this purpose, we have used the work in [9], whereby RMON-MIB (RFC 2819) is translated into OWL as a set of classes and properties.
- A set of web services that encapsulate the functionality provided by the RMON-MIB. One service has been generated automatically for each object group of the MIB, defining configuration functions needed to create, modify and delete monitoring tasks, and information retrieval functions needed to obtain the results of

the monitoring tasks. The semantics of the defined tables has been extracted to distinguish between a configuration table, that includes read-create objects and an EntryStatus (or RowStatus) column, and a results table, which includes read-only objects. Fig 1 shows an example of the operations generated for the tables hostControlTable and hostTable, in pseudo-code, of the RMON-MIB host object group.

- Finally, these web services are used as a grounding for a set of OWL-S descriptions. These descriptions represent the services, and relate them with the concepts contained in the domain ontology defined before. Also, SWRL rules are defined, as described in [11], in order to establish how the represented service interacts with the real world.

```
hostControlIndex createHostControlEntry(
          hostControlDataSource, hostControlOwner)
void removeHostControlEntry(hostControlIndex)
void modifyHostControlDataSource(
          hostControlIndex, hostControlDataSource)
void modifyHostControlOwner(
          hostControlIndex, hostControlOwner)
HostControlEntry[] getAllHostControlEntry()
HostControlEntry getHostControlEntryByHostControlIndex(
           hostControlIndex)
HostControlEntry[] getHostControlEntryByHostControlOwner(
           hostControlOwner)
HostEntry[] getAllHostEntry()
HostEntry[] getHostEntryByHostIndex(hostIndex)
HostEntry[] getHostEntryByHostAddress(hostAddress)
```

Fig. 1. Operations generated for the RMON-MIB host object group

Then, the monitoring process can be described by using these elements. Fig. 2 shows the modeled process. This process takes the following steps:

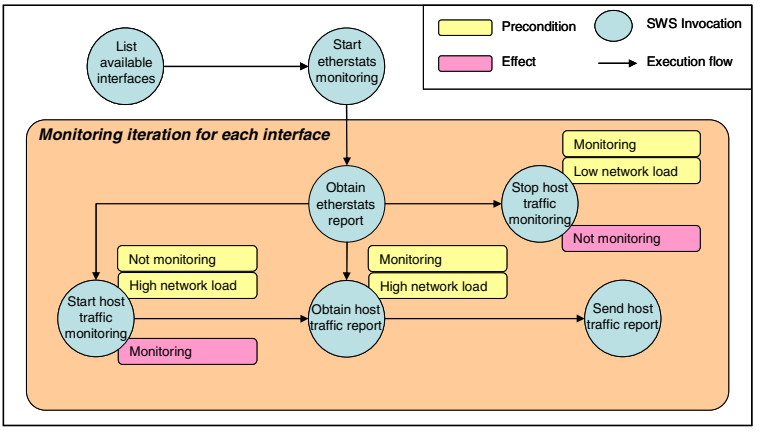

Fig. 2. Conceptual representation of the traffic-monitoring OWL-S process

1. Call the service operation “List available interfaces”, based on IF-MIB (RFC 2863) ifEntry. This service takes a void input, and offers an output with information about all available interfaces in the network probe.
2. Call the service operation “Start etherstats monitoring”, based on RMON-MIB etherStatsEntry. This service takes as an input the interface list to monitor, and starts the monitoring task, obtaining Ethernet statistics for each interface.
3. For each interface:
   a. Call the service operation “Obtain etherstats report”, based also on RMON-MIB etherStatsEntry.
   b. If the preconditions “high network load” and “not monitoring” are met, call the service “Start host traffic monitoring”, based on RMON-MIB hostControlEntry. This service starts the monitoring of each host in a concrete interface of the probe. If it is correctly invoked, call the service operation “Obtain host traffic report”, described below.
   c. If the “high network load” and “monitoring” preconditions are met, call the service operation “Obtain host traffic report”, based on RMON-MIB hostEntry. This service obtains the report of traffic by host in a concrete interface of the probe. If it is correctly invoked, call the service operation “Send host traffic report”, in charge of sending reports to a network manager.
   d. If the “low network load” and “monitoring” preconditions are met, call the service operation “Stop host traffic monitoring”. This service stops the monitoring of hosts in a concrete interface of the probe.

4 Implementation Approach: Use of WSBPEL

Although the formal approach has been introduced, it is necessary to make an extra effort when working with semantic web technologies, because current tools are still under development. Then, first of all, a revision of currently available OWL-S tools has been done. Among them, only Mindswap’s OWL-S API[1] and CMU OWL-S VM[2] provide some support to execute semantic web services from an OWL-S description, although with important limitations. Neither the if-then-else and repeat-while control structures, nor conditional outputs and effects are supported by the OWL-S API, unless custom extensions are introduced. The CMU OWL-S VM is not sufficiently documented to assess the level of support provided by this tool for the execution of complex semantic web service descriptions. Other tools also exist, as stated in [15], but they are just devoted to the edition of OWL-S instances.

Due to these limitations, and given that the defined semantic web services are grounded on a conventional web services, other existing technologies for web service composition have been studied. In this way, if the semantic web services are grounded on a traditional web service, process descriptions can also be grounded on traditional web service composition technologies. In this scope, there are three main approaches: WSBPEL (Web Services Business Process Execution Language), WSCI (Web Services Choreography Interface), and BPML (Business Process Modeling Language).

[1] http://www.mindswap.org/2004/owl-s/api/
[2] http://projects.semwebcentral.org/projects/owl-s-vm/

However, only WSBPEL currently provides a sufficient mature set of tools, including graphical process editors, execution engines, deployed process managers, process debuggers, etc. Moreover, being an OASIS standard, WSBPEL is highly accepted, and has the support of a large community of users.

4.1 WSBPEL Process Representation

WSBPEL [19] defines a model and a grammar to describe the behavior of a business process based on the interactions among the process and its partners. This interaction is achieved by means of web services. Moreover, WSBPEL allows defining how the partners and the process are coordinated to achieve a goal, as well as the state of the interaction and the logic needed to make this coordination possible. Finally, WSBPEL provides a mechanism to describe the way in which some activities have to be compensated or undone if any error occurs in the business process. Then, WSBPEL provides a language to generate process descriptions, independent of the platform, and supporting the definition of all the fundamental aspects of processes.

As it can be observed, a WSBPEL process implementation externally consists of a web service, which defines a set of operations to let other systems interact with the process. Internally, however, a WSBPEL process consists of a complex business process description, which includes variables, partners, error handling and business flow definition.

The variables section is composed of the variable descriptions used by the process, providing its definition in terms of WSDL messages, XSD (XML Schema Data type) types or XML Schema elements. These variables are useful to maintain data and information related to the process status, based on the exchanged messages at a certain time. To access these variables, XPath expressions can be used.

The partners or partnerLinks section describes the behaviour of each web service that interacts with the process. Each partner is defined by a type and a role. This information represents the functionality that a partner has to provide so that the process performs correctly.

The error handling section allows the definition of the actions to be done when an error occurs during the execution of a business process.

The definition of a business flow allows the description of the set of activities to be done in order to achieve the goals defined for the business process. For this purpose, WSBPEL offers a wide set of primitives to deal with data, message reception and transmission, service invocation, conditional expressions and other control structures.

Finally, WSBPEL can be considered a sufficiently expressive language to be used for the execution of the composite processes described in OWL-S. Nevertheless, there are some aspects that WSBPEL cannot cover. The next subsection studies the viability of using WSBPEL to support the execution of OWL-S definitions.

4.2 OWL-S Process Grounding on a WSBPEL Description

The grounding of an OWL-S process on a WSBPEL description is relatively easy to do. WSBPEL offers control structures that are similar to OWL-S structures. At the same time, other functionalities (data flow, variable declaration) are also similar in both descriptions. However, there are some issues to be taken into account: service,

data and logic expression descriptions. Then, this subsection analyzes those points in which both technologies differ, which instruments can be used to solve these differences, or what functionality is lost if WSBPEL is used instead of OWL-S. The inverse approach (i.e. a translation from WSBPEL to OWL-S) can be found in [20].

The first aspect to deal with is related to the types used when defining process data. In OWL-S, data are typed by an OWL class or a basic XSD type. However, in both WSDL and WSBPEL descriptions, data are represented by a basic XSD type or a type described with XML Schema. Thus, a translation from an OWL class instance to an XML Schema element is needed. For this kind of translation, document transformation languages such as XSLT (eXtensible Stylesheet Language Transformation) are commonly used. Nevertheless, this process is usually not trivial, because in OWL and other ontology languages based on description logics, classes can be defined as a set of restrictions, and the form of an instance is not easily known. It is worth mentioning that this problem is not common in network management ontologies, because most ontologies are derived from existing MIB or CIM schema specifications, based on objects and properties. Another consideration is about the unique identification of an instance with a URI, which is lost when transforming it to an XML Schema data type. Once again, this problem is not common in network management ontologies, in which functional properties are usually used to identify a concrete instance of a class.

The next aspect is related to logic expressions and their use in both OWL-S and WSBPEL. Logic expressions in OWL-S are mainly used to define conditions in control structures, preconditions and effects. As stated before, WSBPEL allows the use of control structures, but it does not have preconditions and effects when calling a partner. Then, OWL-S preconditions and effects have to be extracted from each service call, and included in the process flow to achieve a functional correspondence in the WSBPEL process. This extraction cannot be easily automated, so it has to be done by hand. Another relevant issue is the expressiveness of the logic expression languages used in OWL-S and WSBPEL. WSBPEL does not provide such a language, using XPath instead. XPath [21] is a language to manipulate XML with a set of added functions, such as arithmetic comparisons (<, >, =) and simple Boolean expressions (and, or, not). On the other hand, OWL-S proposes the use of SWRL to define logic expressions, which joint with the OWL descriptions provides a higher expressiveness than XPath. Given that current WSBPEL engines do not support SWRL, the logic expressions contained in the OWL-S descriptions have to be limited so that they can be translated to XPath. Then, this translation cannot be done automatically.

Finally, the description of partners has to be analyzed. As commented before, WSBPEL allows the definition of roles for those partners involved in the process. This definition is done based on the set of operations that a partner provides. Semantic Web techniques aim at allowing partner descriptions to be presented in terms of what is going to be obtained instead of describing a communication interface. Given this fact, and keeping in mind that the objective of this work is to obtain a practical result, this kind of partner descriptions have to be avoided. Instead, just operations, inputs and outputs, along with the appropriate XML Schema mappings, should be defined.

4.3 Application to the Case Study: Network Monitoring Process in WSBPEL

Once the WSBPEL process representation and its relationship with OWL-S have been described, this subsection explains the adaptation to WSBPEL of the case study presented in subsection 3.2, where an OWL-S specification was defined for a network monitoring process.

First of all, it is necessary to bind all data. For this purpose, a transformation is performed from the OWL class instances, defined as service inputs and outputs, to the XML Schema data types of the web services, which encapsulate the RMON functionality. There are several ways of doing this binding, among which XSLT transformations are our proposed approach in this work.

Next, it is necessary to model the OWL-S composite process in WSBPEL. As mentioned before, this translation is complex and cannot be done automatically, so it has to be done manually, taking advantage of the available editing tools. In our work, the ActiveBPEL Designer[3] editor has been used. The translation process has been as follows:

1. Include in the specification all the web service calls needed to complete the process. During this step, it is necessary to define the partner profiles for the process. That is, the set of methods that any network probe has to implement. Given that semantic web services are used, this definition can be done in terms of objectives (preconditions and effects) instead of inputs and outputs. However, due to the problem mentioned above, this description has to include the required operations, as well as their inputs and outputs, in order to obtain an executable WSBPEL process.
2. Define the flow and control structures needed to execute the process. In this step, the control structures used in the OWL-S description are translated to WSBPEL structures. This process also requires the translation of the logic expressions used in the OWL-S specification to those of the WSBPEL description. This is only possible if the expressivity of OWL-S expressions is limited to fit in the accepted WSBPEL expressions.
3. Extract the logic introduced in the preconditions and effects of the OWL-S description, and integrate it in the WSBPEL process definition. This step has to be done again manually for each precondition and effect.

One important aspect to translate the OWL-S description to WSBPEL is the role that performs the reasoner when processing OWL-S descriptions. In OWL-S processes, the definition of memory structures does not exist, because it is the reasoner who takes care of it. However, when describing a WSBPEL process, it is necessary to specify all the data structures to be used. Then, it is possible that during translation, some auxiliary variables have to be declared, and the management of these variables (access, init values, etc.) needs to be specified. If all these facts are taken into account, the result is a WSBPEL process that can be loaded into a BPEL engine and run as shown in Fig. 3.

In this work, ActiveBPEL Engine[4] has been used to run the WSBPEL process. The result of this development is a WSBPEL process definition that implements the functionality contained in the OWL-S process description. This WSBPEL definition

[3] http://www.active-endpoints.com/freebpel/
[4] http://www.activebpel.org/

presents a WSDL service interface that can be used as a grounding for the OWL-S Service description. Thus, this WSBPEL process definition is completely interoperable, so it can be deployed in any WSBPEL engine. The location of the component services implied in the WSBPEL process description can be modified using a WS-Address. In Fig. 4, the process deployed in the ActiveBPEL Engine is shown.

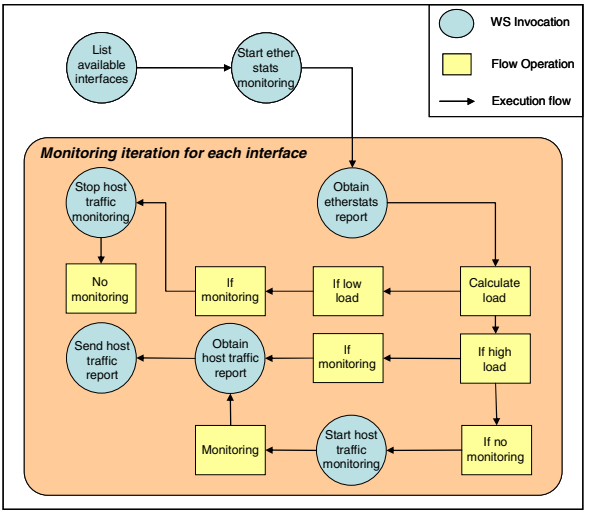

Fig. 3. Conceptual representation of the traffic monitoring WSBPEL process

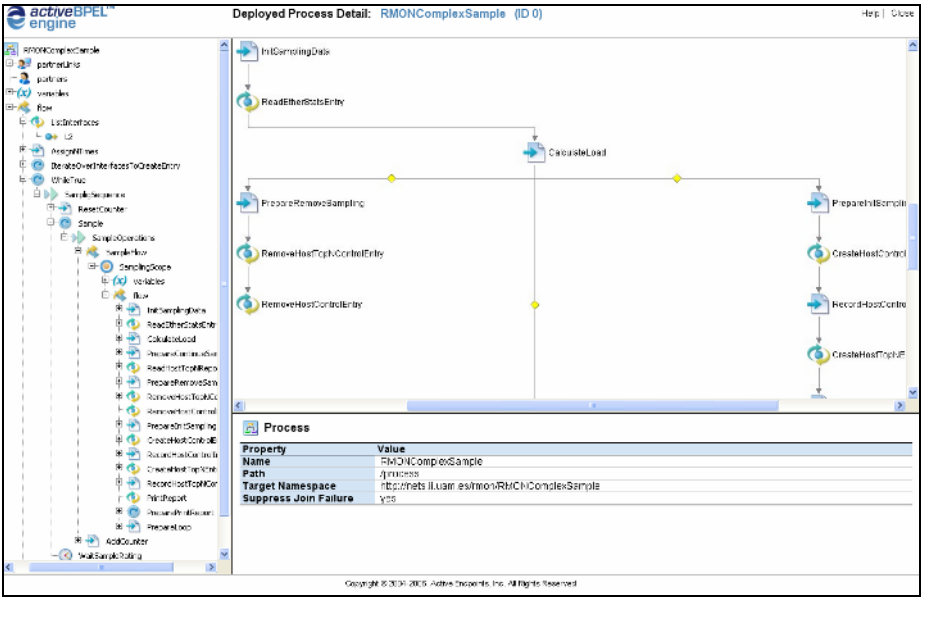

Fig. 4. Load-based Network Management process deployed in the ActiveBPEL Engine

5 Conclusions

Web service technology allows the definition of network management interfaces to be deployed on the network resources. These services are usually combined to perform a management task, but WSDL specifications only provide the information related to each interface. To address this problem, service ontologies, such as OWL-S, are useful to define the relation among different web services in a management process. This definition can be interpreted by a manager, which calls the services following a sequence with control structures. The advantage of this approach is the shift of the application development workload to a process definition, aided by graphical editors which directly generate that definition from a flow diagram. This paper has presented a case study in which OWL-S has been used to describe the composite process to monitor the traffic load of a network.

Due to the necessity of using an execution engine to interpret such definitions, this work has also studied how to translate an OWL-S definition to WSBPEL. Thus, until future OWL-S engines make this task unnecessary, the defined composite process has been translated to WSBPEL and loaded into an execution engine, performing the network monitoring previously described. Using currently available WSBPEL engines has several benefits, including the use of BAM (Business Activity Monitoring) technologies [22] to monitor and assess the correctness and quality of the deployed processes. When OWL-S engines are available and related technologies like BAM can work with such engines, a future task shall be to load the defined semantic process and check if they perform as foreseen.

Given this approach, one may think that WSBPEL can be used directly in most of cases to combine web services for a management application. However, WSBPEL is somehow limited, as web services must comply with a set of defined inputs and outputs. On the other hand, the semantics of OWL-S enable the future definition of autonomic systems that can interpret the semantics of the processes to achieve their goals. Future process execution engines will either use OWL-S Process descriptions or should improve current WSBPEL, importing some of OWL-S semantics key points identified in this work.

In our envisioned future work we shall also study the application of these technologies to other management functional areas, following the FCAPS (Fault, Configuration, Accounting, Performance and Security) model, to assess the feasibility of such management architecture.

References

1. J. Schönwälder, A. Pras, J.P. Martin-Flatin: On the Future of Internet Management Technologies. IEEE Communications Magazine, Vol. 41, Issue 10 (2003) 90-97.
2. H. Haas, A. Brown: Web Services Glossary. W3C Working Group Note (11 February 2004)
3. G. Pavlou, P. Flegkas, S. Gouveris, A. Liotta: On Management Technologies and the Potential of Web Services. IEEE Communications Magazine, Vol. 42 Issue 7 (2004) 58-66.
4. A. Pras, T. Drevers, R. van de Meent, D. Cuartel: Comparing the Performance of SNMP and Web Services-Based Management. eTransactions on Network and Service Management, Vol. 1, No. 2 (2004) 72-82.

5. T. Fioreze, L.Z. Granville, M.J. Almeida, L. Tarouco: Comparing Web Services with SNMP in a Management by Delegation Environment. In. Proc. 9th IFIP/IEEE Intl. Symp. on Integrated Network Management (IM 2005), Nice, France, (May 2005) 601-614.
6. T. Berners-Lee, J. Hendler, O. Lassila: The Semantic Web. Scientific American, Vol. 284, No. 5 (2001) 34–43
7. T. R. Gruber: A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition, Vol. 5, No. 2 (1993) 199-220.
8. J.E. López de Vergara, V.A. Villagrá, J.I. Asensio, J. Berrocal: Ontologies: Giving Semantics to Network Management Models. IEEE Network, Vol. 17, Issue 3, (2003) 15-21.
9. J.E. López de Vergara, V.A. Villagrá, J. Berrocal: Applying the Web Ontology Language to management information definitions. IEEE Communications Magazine, Vol. 42, Issue 7 (2004) 68-74.
10. S. Quirolgico, P. Assis, A. Westerinen, M. Baskey, E. Stokes: Toward a Formal Common Information Model Ontology. WISE'2004, Lecture Notes in Computer Science, Volume 3307, Springer Verlag (2004) 11-21.
11. A. Guerrero, V.A. Villagrá, J.E. López de Vergara, J. Berrocal: Ontology-based integration of management behaviour and information definitions using SWRL and OWL. DSOM'2005, Lecture Notes in Computer Science, Vol. 3775, Springer Verlag (2005) 12-23.
12. A.K.Y. Wong, P. Ray, N. Parameswaran, J. Strassner: Ontology Mapping for the Interoperability Problem in Network Management. IEEE Journal on Selected Areas in Communications, Vol. 23, Issue 10 (2005) 2058-2068.
13. J. Keeney, D. Lewis, D. O'Sullivan, A. Roelens, A. Boran, R. Richardson: Runtime Semantic Interoperability for Gathering Ontology-based Network Context. In Proc. 10th IFIP/IEEE Network Operations and Management Symposium (NOMS'2006), Vancouver, Canada (April 2006)
14. M. Burstein, C. Bussler, T. Finin, M.N. Huhns, M. Paolucci, A.P. Sheth, S. Williams, M. Zaremba: A semantic Web services architecture. IEEE Internet Computing, Vol. 9, Issue 5 (2005) 72-81.
15. J.E. López de Vergara, V. A. Villagrá, J. Berrocal: Application of OWL-S to define management interfaces based on Web Services. MMNS'2005, Lecture Notes in Computer Science, Vol. 3754, Springer Verlag (2005) 242-253.
16. J. Keeney, K. Carey, D. Lewis, D. O'Sullivan, V. Wade: Ontology-based Semantics for Composable Autonomic Elements. In Proc. Workshop on AI in Autonomic Communications at the 19th International Joint Conference on Artificial Intelligence (IJCAI), Edinburgh, Scotland. (July 2005)
17. D. Martin, M. Burstein, J. Hobbs, O. Lassila, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, B. Parsia, T. Payne, E. Sirin, N. Srinivasan, K. Sycara: OWL-S: Semantic Markup for Web Services. W3C Member Submission (November 2004)
18. I. Horrocks, P.F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, M. Dean: SWRL: A Semantic Web Rule Language Combining OWL and RuleML. W3C Member Submission (May 2004)
19. A. Arkin, S. Askary, B. Bloch, F. Curbera, Y. Goland, N. Kartha, C.K. Liu, V. Mehta, S. Thatte, P. Yendluri, A. Yiu, A. Alves: Web Services Business Process Execution Language Version 2.0. OASIS Consortium Committee Draft (December 2005)
20. D. J. Mandell, S. A. McIlraith: Adapting BPEL4WS for the Semantic Web: The Bottom-Up Approach to Web Service Interoperation. ISWC'2003, Lecture Notes in Computer Science, Vol. 2870, Springer Verlag (2003) 227-241.
21. J. Clark, S. DeRose, eds.: XML Path Language (XPath) Version 1.0. W3C Recommendation (November 1999)
22. Alan Joch: Containing Business Processes. Oracle Magazine (March/April 2005)