Model Driven Design of Distribution Patterns for Web Service Compositions

Ronan Barrett
School of Computing
Dublin City University
Dublin 9, Ireland
rbarrett@computing.dcu.ie

ABSTRACT

Increasingly, distributed systems are being constructed by composing a number of discrete components. This practice, termed composition, is particularly prevalent within the Web service domain. Here, enterprise systems are built from many existing discrete applications, often legacy applications exposed using Web service interfaces. There are a number of architectural configurations or distribution patterns, which express how a composed system is to be deployed. However, the amount of code required to realise these distribution patterns is considerable. In this paper, we outline our novel Model Driven Architecture, which takes existing Web service interfaces as its input and generates an executable Web service composition, guided by a distribution pattern modeled by the software architect. The creation of the model can be partially automated when semantic representations, such as OWL-S descriptors, of the discrete Web services are provided. Additionally we consider a novel mechanism which enables the automated deployment of the generated executable Web service composition.

1. INTRODUCTION

Web services are pieces of software functionality that can be invoked using Web based technologies. The basic technology stack for Web services is WSDL, SOAP and UDDI. In addition to the basic technology stack, additional technologies have emerged. Semantic technologies, such as OWL-S, enhance Web service descriptions and enable automated service provision and consumption, by classifying the elements and characteristics of a Web service [8].

Often enterprise systems are built by combining a number of Web services together to realise some novel functionality. This practice of combining Web services together is termed composition. Web service composition is often ad-hoc, where no architectural models are drawn, and considerable low level coding effort is required for realisation [1]. In addition, there are many different ways in which these novel applications can be assembled.

These issues are addressed by a semi-automated modeling approach, an executable system generator, and a solution for dynamic deployment of the generated composition. Our novel approach combines a semantically assisted Model Driven Architecture using UML 2.0, for modeling and subsequently generating Web service compositions, with a method for achieving decentralised communication amongst services, if required. We also provide a Web service based facility for enabling the dynamic deployment of compositions.

2. BACKGROUND

Model Driven Architecture (MDA) is an emerging approach for building software [4]. In MDA, the model is the primary software artifact, and is used to generate the program code. Rich, well specified, high level models, often defined in the Unified Modeling Language (UML), allow for the auto-generation of a fully executable system based entirely on the model [10].

Semantics can be used to assist in the creation of a model. The Web Ontology Language (OWL), is a language for capturing the conceptual data of a domain and their inter-relationships, for use in the description of resources [9]. OWL-S is an ontology based on OWL which is used for defining the properties and capabilities of Web services [7, 8]. Semantic descriptions enable unambiguous, computer interpretable documentation of resources.

Web service compositions can be modeled, using an MDA based approach, from a number of aspects. In [12], Grønmo et al. investigate the service and workflow modeling aspects of Web service compositions. Service modeling considers interfaces and their operations, while workflow modeling considers control and data flows from one Web service to another. We consider an additional aspect, distribution pattern modeling [14], which expresses how the composed system is to be deployed using UML.

Distribution patterns address a considerable shortcoming of fixed centralised coordination identified by Siren et al. [13]. Our approach provides a high level model which intuitively expresses, and subsequently generates, the system's distribution pattern using a UML 2.0 based Activity diagram, in association with our novel distribution pattern UML profile, DPLProfile [3]. UML profiles are a standard extension mechanism of UML [4]. Profiles define stereotypes and tagged values that extend a number of UML constructs. Each time one of these derived constructs is used in a model it may have attributes assigned to its tagged values.

Web services, as passive participants within a composition, cannot interact with each other without mediation. However, the mediation or interaction logic, can be modeled centrally, and subsequently deployed to participants, provided that the runtime infrastructure of the participants supports enactment of the interaction logic[11]. Such characteristics, as well as the additional deployment overheads of decentralisation, are considered here when attempting to enable decentralised compositions.

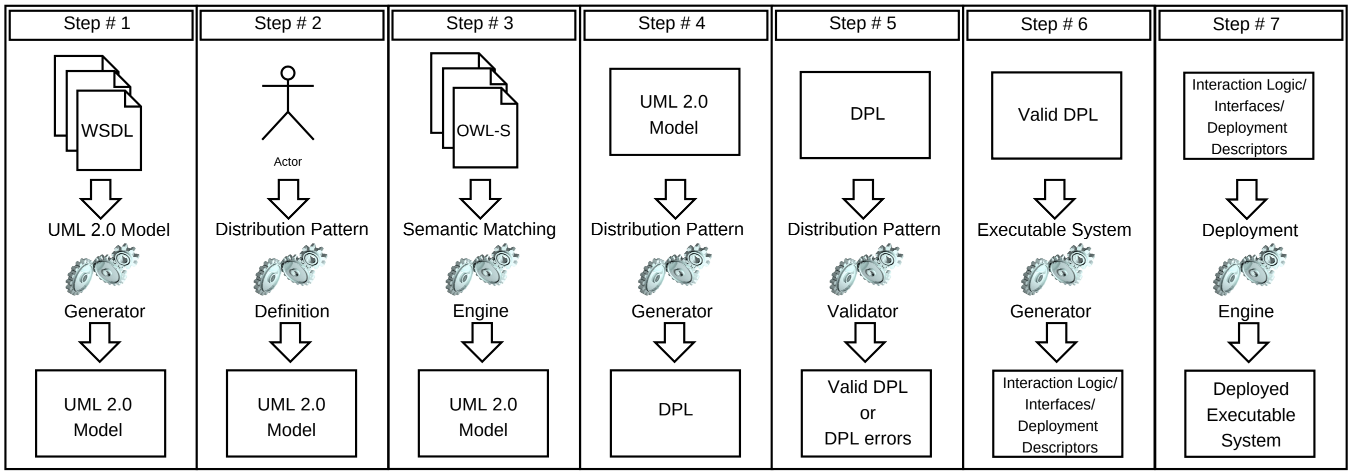

Figure 1: Overview of modeling approach

3. DISTRIBUTION PATTERNS

We consider three pattern categories, core patterns, auxiliary patterns and finally complex patterns. The three pattern categories encompass a total of seven distribution patterns, which are briefly outlined below.

- Core Patterns - Fundamental distribution patterns, most commonly encountered in Web service compositions.
  - Centralised - Composition is managed from a single location.
  - Decentralised - Composition management is distributed amongst participants.
- Auxiliary Patterns - Patterns which are often used in conjunction with core patterns to create complex patterns.
  - Ring - Composition participants or controllers are mirrored.
- Complex Patterns - Combine two or more core or auxiliary patterns and often resolve fundamental deficiencies within core patterns.
  - Hierarchical - Composition is managed locally from a number of locations in a tree structure.
  - Ring + Centralised - Composition management is mirrored at a single location.
  - Centralised + Decentralised - Composition is managed locally from a number of locations in a peer-to-peer structure.
  - Ring + Decentralised - Distributed compositional participants are mirrored.

4. MODELING AND TRANSFORMATION TECHNIQUE

In this section, we briefly describe our approach to distribution pattern modeling, Web service composition generation and subsequent automated deployment. There are five specific techniques listed below and elaborated in the seven specific steps that follow, as illustrated in Figure 1. Further details are available in [2].

- UML activity diagram/Profile extension (step 1,2)
- Semantic matching engine (step 3)
- Generators (step 1,4,6)
- DPL validator (step 5)
- Deployment Engine (step 7)

4.1 Step 1 - From Interface To Model

The initial step takes a number of Web service interfaces as input, and transforms them to the UML 2.0 model, using the UML 2.0 model generator. These interfaces represent the Web services which are to be composed. The model generated is based on the web services inputted, however, each service is logically separated as no composition has yet been defined. Finally, our novel distribution pattern profile, DPLProfile, is automatically applied to the model by the generator.

4.2 Step 2 - Distribution Pattern Definition

The UML model produced in step 1, requires additional modeling. The architect must select a distribution pattern and then assign appropriate values for the tagged values of the distribution pattern profile, which was applied automatically to the model in the previous step.

4.3 Step 3 - Semantic Matching

We assume all of the Web services to be composed, are semantically annotated using OWL-S. The semantic documents for each service are passed to the semantic matching engine for processing. These semantic descriptions enable the automated sequencing and connection of Web services to one another, in the model. Services are matched together based on their level of compatibility. If a sufficiently similar match is found, a connection is created between the two compatible services in the model. Subsequently the inputs and output parts of these matched services can be mapped. Without semantic annotation this entire step would have to be completed manually by the software architect. At this stage the model is complete and fully expresses the distribution pattern selected by the software architect.

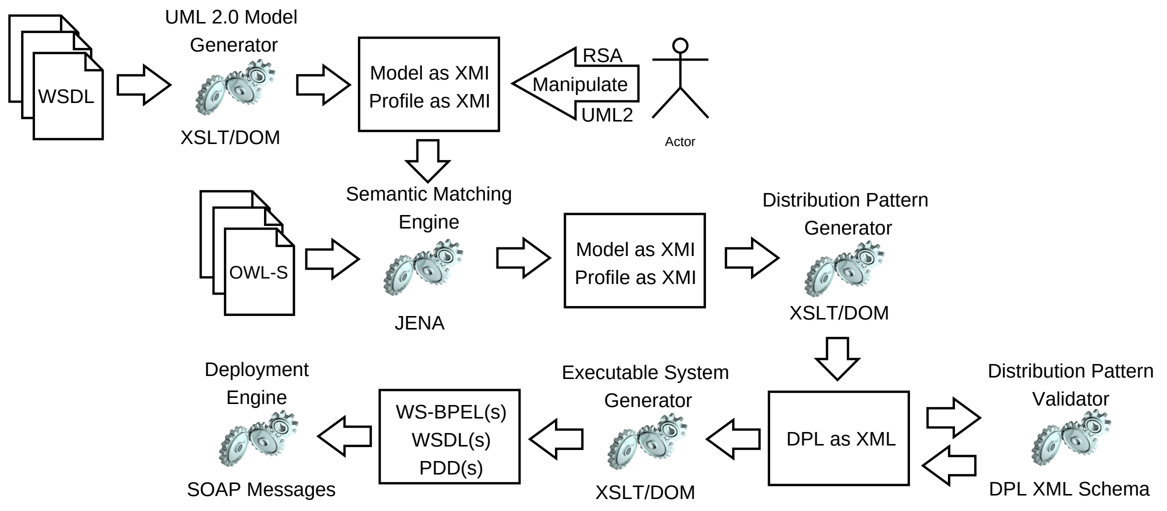
UML 2.0 Model Generator
WSDL
XSLT/DOM
Model as XMI Profile as XMI
RSA
Manipulate
UML2
Actor
Semantic Matching Engine
OWL-S
JENA
Model as XMI Profile as XMI
Distribution Pattern Generator
XSLT/DOM
Deployment Engine
SOAP Messages
WS-BPEL(s) WSDL(s) PDD(s)
Executable System Generator
XSLT/DOM
DPL as XML
Distribution Pattern Validator
DPL XML Schema

Figure 2: Overview of TOPMAN tool

4.4 Step 4 - From Model to DPL

Using the model generated in step 3 as input, the model is transformed to a distribution pattern instance, using the distribution pattern generator. This pattern instance, represented in XML using our novel specification language Distribution Pattern Language (DPL), is called a DPL document instance. The DPL specification, written in XML Schema, has no reliance on UML and so any number of modeling techniques may be used as an input.

4.5 Step 5 - Model Validation

The DPL document instance is verified at this step by the distribution pattern validator, to ensure the values applied in steps 2/3 are valid. If incorrect values have been entered, the architect must correct these values, before proceeding to the next step. Validation of the distribution pattern instance is essential to avoid the generation of an invalid system.

4.6 Step 6 - DPL to Interaction Logic

The executable system generator takes the validated DPL document instance and generates all the code and supporting collaboration document instances required for a fully executable system. A deployment descriptor document, describing the participants of the composition is also created for each participant. This system will realise the Web service composition using the distribution pattern applied by the software architect. All that remains is to deploy the generated artifacts and supporting infrastructure to enable the enactment of the composed system.

4.7 Step 7 - Interaction Logic to Deployed Executable System

In the final step, the documents generated in the previous step are automatically passed to the participant services, by the deployment engine. We consider a novel enhancement to the container of each participant called Interaction Logic Document Processor (ILDP). ILDP enhanced participants are exposed as Web services, capable of receiving, processing and deploying these documents. These enhanced participants can receive documents from the deployment engine. Subsequently the documents are processed by ILDP to ensure they are valid before storing them on the participant. Finally the stored documents are deployed by ILDP on the participant and exposed for composition by a composition runtime interface. The ILDP is also responsible for enacting the interaction logic and subsequently invoking the participant services, enabling decentralised interaction amongst the participant services, if required by the distribution pattern chosen. This approach negates any requirement of manually deploying documents to participant services.

5. IMPLEMENTATION

TOPMAN (TOPology MANager) is our solution for distribution pattern modeling using UML 2.0 and subsequent dynamic Web service composition generation. The only technologies required by the tool are the Java runtime, the Jena framework and both an XML and XSLT parser. The tool implementation is illustrated in Figure 2.

The UML 2.0 model generator uses XSLT to transform the WSDL interfaces of the Web services participating in the composition, to a UML 2.0 activity diagram, which generates, using XML DOM, an XMI 2.0 document [5]. XMI is the XML serialisation format for UML models. The model generated includes a reference to our predefined UML profile, DPLProfile, which is also serialised to XMI 2.0.

A number of tools may be used by the software architect to manipulate the distribution pattern model. IBM's commercial tool Rational Software Architect (RSA) is compatible with XMI 2.0 and supports many of the UML 2.0 features.

Once the software architect has finished assigning values to the distribution pattern profile, the model can be exported back to XMI and passed to the semantic matching engine. This engine uses the Jena semantic web framework to assess the compatibility of the participant Web services input and output messages, based on their OWL-S atomic process models [6]. The engine iterates through each service definition checked to see if its output can be used as input to one of the other participants.

The distribution pattern generator uses XSLT to transform the UML 2.0 model to an XML instance document, where it can be verified by an XML validating parser. Finally the XML is used to drive the executable system generator, resulting in the creation of an executable composition. Within the executable system generator, XSLT and XML DOM are used to generate the interaction logic and interface documents needed by a workflow engine to realise the distribution pattern.

An open source orchestration engine, activeBPEL, is used to realise the distribution pattern. Full details of the implementation of the executable system generator and the dynamic deployment of the executable system are available in [2].

6. CONCLUSION

An engineering approach to the composition of service-based software systems is required. We have introduced techniques based on architectural modeling and pattern-based development, which have already been applied successfully in both object-oriented and component-based systems. Our contribution is a semantically enhanced modeling and transformation approach, technique and implementation for expressing the distribution pattern of a Web service composition. The approach is achieved using five modeling and transformation techniques.

Our novel modeling aspect, distribution patterns, expresses how a composed system is to be deployed, providing for improved maintainability and comprehensibility. Any of the distribution patterns discussed may be used to guide the generation of an executable system, based on the enterprises requirements. A mechanism for the dynamic deployment of decentralised compositions was also introduced. Finally, we presented a tool (TOPMAN) which assists in the generation of an executable system guided by the chosen pattern.

7. ACKNOWLEDGMENTS

This research is supported by the Irish Research Council for Science, Engineering and Technology (IRCSET).

8. REFERENCES

[1] G. Alonso, F. Casati, H. Kuno, and V. Machiraju. *Web Services: Concepts, Architecture and Applications*. Springer Verlag.

[2] R. Barrett, C. Pahl, L. Patcas, and J. Murphy. Model Driven Distribution Pattern Design for Dynamic Web Service Compositions. In *Proc. Sixth International Conference on Web Engineering (to appear)*, Palo Alto, California, July 2006.

[3] H. E. Eriksson, M. Penker, B. Lyons, and D. Fado. *UML 2 Toolkit*. Wiley, 2003.

[4] D. S. Frankel. *Model Driven Architecture: Applying MDA to Enterprise Computing*. Wiley, 2004.

[5] T. J. Grose. *Mastering XMI: Java Programming with XMI, XML, and UML*. Wiley, 2002.

[6] Jena. A semantic web framework for java, 2006.

[7] D. Martin, M. Burstein, O. Lassila, M. Paolucci, T. Payne, and S. McIlraith. Describing Web Services using OWL-S and WSDL. DAML-S Coalition working document., 2003.

[8] D. Martin, M. Paolucci, S. McIlraith, M. Burstein, D. McDermott, D. McGuinness, T. P. B. Parsia, M. Sabou, M. Solanki, N. Srinivasan, and K. Sycara. Bringing Semantics to Web Services: The OWL-S Approach. In *Proc. First International Workshop on Semantic Web Services and Web Process Composition (SWSWPC 2004)*, San Diego, California, USA, 2004.

[9] D. McGuinness and F. van Harmelen. OWL Web Ontology Language Overview. W3C Recommendation 10 February 2004., 2004.

[10] OMG. Unified Modeling Language (UML), version 2.0. Technical report, OMG, 2003.

[11] L. M. Patcas, J. Murphy, and G. M. Muntean. Middleware support for data-flow distribution in web services composition. In *The PhDOOS Workshop and Doctoral Symposium, ECOOP*, Glasgow, UK, July 2005.

[12] R. Grønmo and I. Solheim. Towards modeling web service composition in uml. In *Proc. 2nd International Workshop on Web Services: Modeling, Architecture and Infrastructure (WSMAI-2004)*, pages 72–86, Porto, Portugal, April 2004.

[13] E. Sirin, J. Hendler, and B. Parsia. Semi-automatic composition of web services using semantic descriptions. In *Proc. Workshop on Web Services: Modeling, Architecture and Infrastructure (WSMAI)*, pages 17–24, Angers, France, 2003. ICEIS.

[14] S. J. Woodman, D. J. Palmer, S. K. Shrivastava, and S. M. Wheater. A system for distributed enactment of composite web services. In *Work in progress report, Int. Conf. on Service Oriented Computing*, Trento, Italy, December 2003.