Semi-Automatic Distribution Pattern Modeling of Web Service Compositions using Semantics

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

Claus Pahl
School of Computing
Dublin City University
Dublin 9, Ireland
cpahl@computing.dcu.ie

Abstract

Enterprise systems are frequently built by combining a number of discrete Web services together, a process termed composition. There are a number of architectural configurations or distribution patterns, which express how a composed system is to be deployed. Previously, we presented a Model Driven Architecture using UML 2.0, which took existing service interfaces as its input and generated an executable Web service composition, guided by a distribution pattern model. In this paper, we propose using Web service semantic descriptions in addition to Web service interfaces, to assist in the semi-automatic generation of the distribution pattern model. Web services described using semantic languages, such as OWL-S, can be automatically assessed for compatibility and their input and output messages can be mapped to each other.

1. Introduction

Enterprise systems are often 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.

Model Driven Architecture (MDA) is an emerging approach for building software [5]. 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 [4]. Web service compositions can be modeled, using an MDA based approach, from a number of aspects. In [13], service and workflow modeling aspects of Web service compositions are investigated. Service modeling considers interfaces and their operations, while workflow modeling considers control and data flows from one Web service to another. Our previous work [1], introduced an additional aspect, distribution pattern modeling, which expresses how the composed system is to be deployed using UML. Two well known distribution patterns are centralised and decentralised. Distribution patterns address a considerable shortcoming of fixed centralised coordination identified by Siren et al. [14].

In this paper we introduce the use of Web service semantics to assist in the generation of a distribution pattern model. This approach assumes that all the Web services to be composed, are semantically annotated using OWL-S, and have already been discovered by a system. These semantics enable the automated matchmaking and subsequent sequencing of the order in which the pre-selected Web services will be composed. They also enable the automatic integration of inputs and outputs for each Web service. This automation effort, using semantics, reduces the manual modeling workload of the system architect.

The paper is structured as follows: section two provides background material on our approach and the technologies underlying it; section three introduces our modeling and transformation technique; section four investigates our tool implementation; section five presents related work; finally, section six considers future work and concludes the paper.

2. Background

Ontologies are often used to define the vocabulary of a domain. These definitions must be sharable, interoperable and standards compliant to enable the expression of machine interpretable semantics. Creating an ontology is analogous to domain modeling, where the different discrete components that make up a system are investigated. The Web Ontology Language (OWL), is a language for capturing the conceptual data of a domain and their interrelationships, for use in the description of resources [11]. This technology enables the semantic description of Web resources such as Web pages and Web services. Semantic descriptions enable unambiguous, computer interpretable documentation of resources. OWL-S is an ontology based on OWL which is used for defining the properties and capabilities of Web services [10].

Distribution patterns express, using models, how a composed system is to be assembled and subsequently deployed. In MDA terms, distribution pattern models are a form of platform-independent model (PIM), as the patterns are not tied to any specific implementation technology. These patterns are considered compositional choreographies, where only the message flow between services is modeled. As such, a choreography can express how a system would be deployed. The workflow logic between these services are not modeled here, as there are many approaches to modeling the branch flows of such services [3, 6].

There are three pattern categories, core patterns, auxiliary patterns and complex patterns. Core patterns are the fundamental distribution patterns, most commonly encountered in Web service compositions. Auxiliary patterns are distribution patterns which by themselves cannot facilitate Web service compositions, and are often used in conjunction with core patterns to create complex patterns. Finally, complex patterns combine two or more core or auxiliary patterns. Complex patterns often resolve fundamental problems evident within core patterns. The patterns, organised by category, are listed below.

- Core patterns
  - Centralised Dedicated-Hub
  - Centralised Shared-Hub
  - Decentralised Dedicated-Peer
  - Decentralised Shared-Peer
- Auxiliary patterns
  - Ring
- Complex patterns
  - Hierarchical
  - Ring + Centralised
  - Centralised + Decentralised
  - Ring + Decentralised

UML & UML Profiles: Distribution patterns are modeled using a UML activity diagram in association with our novel distribution pattern UML profile, DPLProfile. UML is a standards based graphical language for the modeling of software systems [12]. Activity diagrams illustrate the sequential flow of actions within a system, capturing actions and their results [4].

These diagrams consist of actions, which are the basic unit of behaviour within an activity, and control flows, which illustrate the transitions through the system. Activity partitions, also known as swim lanes, are often used to group actions together. Rich models, such as those necessary for modeling distribution patterns, can make use of a particular type of UML action to model Web service operations defined in a WSDL interface. These actions, called CallBehaviorActions, model process invocations along with the flow of control through the system, using ControlFlow connectors. CallBehaviorActions have an additional modeling constructs called pins. There are two types of pins, InputPins and OutputPins, which map directly to the parts of the WSDL message parts going into and out of a WSDL operation. ObjectFlow connectors are used to connect pins together. Figure 1 illustrates the use of an activity diagram used to model the decentralised distribution pattern. The UML pins and associated connectors have been omitted from the diagram for clarity.

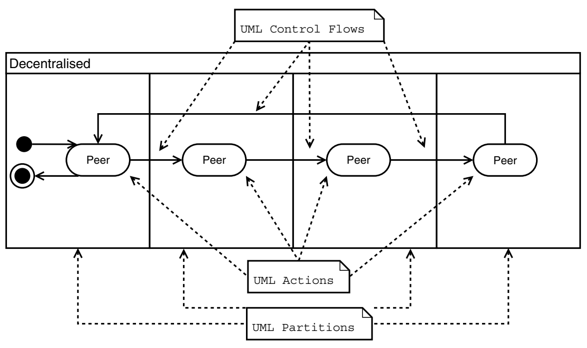

Figure 1. Decentralised distribution pattern

UML profiles are a standard extension mechanism of UML [5]. 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. Our distribution pattern profile, called DPLProfile, is outlined in detail in [2].

3. Modeling and Transformation Technique

Our semi-automated distribution pattern modeling and subsequent executable system generation, comprises four steps, each is illustrated in Figure 2.

Step 1 - From Interface To Model: The first step involves taking a number of Web service interfaces, WSDL documents, as input to a generator. These interfaces rep-

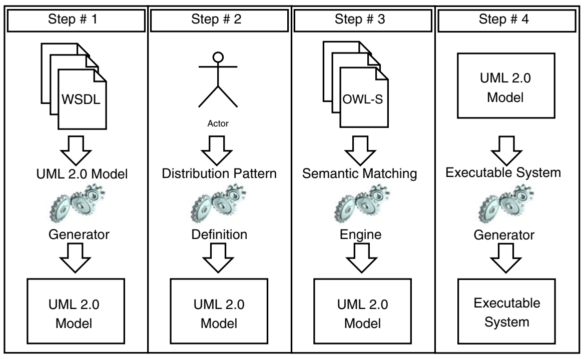

Figure 2. Overview of modeling approach

resent the services which are to be composed. These interfaces are transformed automatically, using the UML 2.0 model generator, into a UML 2.0 activity diagram. Finally, our novel distribution pattern profile, DPLProfile, is automatically applied to the model by the generator. The software architect does not need to manipulate the model in any way at this step. The output from the UML 2.0 generator can be seen in Figure 3. The figure illustrates that the DPLProfile, UML profile, has been applied to the model.

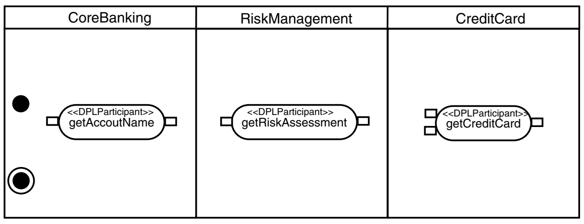

Figure 3. Output from the model generator

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. The complete list of tagged values are outlined in [2]. Values that must be assigned include Web service namespaces, participant roles, choice of collaboration language and distribution pattern type. Previously at this step, the architect had to connect CallBehaviorActions to one another, and also connect up UML InputPins and OutputPins together. However, this requirement is now negated by the use of semantics to automate the connections in the following step.

Step 3 - Semantic Matching and Integration: 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. Each service must have an atomic process model describing, using an ontology, the message input and output parts. OWL-S atomic process models are analogous to WSDL operations. These semantic descriptions enable the automated sequencing of actions and connection of CallBehaviorActions to one another, in our distribution pattern model, using UML ControlFlow connectors. Services are matched together based on their level of compatibility. Each service is checked against every other participant service to assess if their process models are compatible. Compatibility here is defined as one participant having output message part(s) which match the input message part(s) requirements of another participant. If a sufficiently similar match is found, a UML ControlFlow connector is created between the two compatible services in the model. Subsequently the inputs and output parts of these matched services can be mapped. This integration results in the connection of UML InputPins and OutputPins in the model, using UML ObjectFlows connectors, so data can flow through the composition. In some cases, additional pins may be added automatically to the output of CallBehaviorActions, to meet data input requirements of other services. Existing services are wrapped to support the new connections. 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. Sample output from the matching engine can be seen in Figure 4.

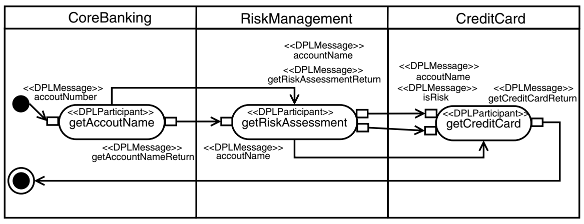
CoreBanking
RiskManagement
CreditCard
<<DPLMessage>> accountName
<<DPLMessage>> getRiskAssessmentReturn
<<DPLMessage>> accoutName
<<DPLMessage>> isRisk
<<DPLMessage>> getCreditCardReturn
<<DPLMessage>> accoutNumber
<<DPLParticipant>> getAccoutName
<<DPLParticipant>> getRiskAssessment
<<DPLParticipant>> getCreditCard
<<DPLMessage>> getAccountNameReturn
<<DPLMessage>> accoutName

Figure 4. Semantic matching engine output

Step 4 - Model to Executable System: The executable system generator takes the finished model and generates all the interaction logic required to realise the distribution pattern. Interaction logic documents describe the message flow between the participants in the distribution pattern as well as input and output variable mappings. The generator also creates interfaces which expose the new interaction logic processing capability as a wrapper to the existing Web service functionality of the participant. A deployment descriptor document describing each participants is also created. Once deployed, these documents will realise the Web service composition, driven by the distribution pattern modeled by the software architect, see [2] for more details.

4. Implementation

TOPMAN (TOPology MANager) is our solution for distribution pattern modeling using UML 2.0 and subsequent Web service composition generation. The tool implementation is illustrated in Figure 5.

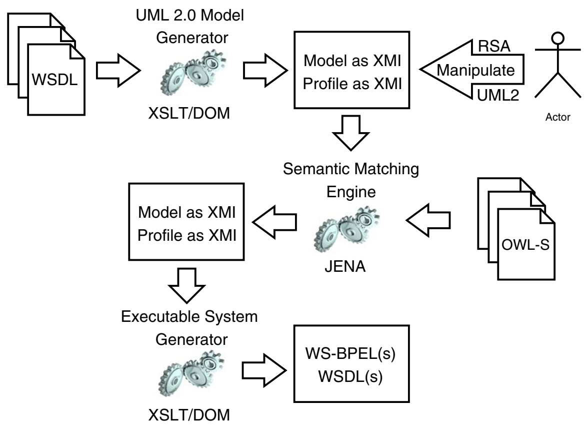

Figure 5. Overview of TOPMAN tool

A non-semantically enabled version of our tool is described in [2]. The only modification made to the tool here is the use of the Jena semantic web framework to assist in the generation of the distribution model. The tool assesses the compatibility of the participant Web services input and output messages, based on their OWL-S atomic process models [8].

5. Related Work

Using semantics to assist in the composition of Web services is considered by Siren et al. [14]. Here semantically annotated services can be combined, semi-automatically based on their input and other non-functional requirements. Two additional systems devised by Timm et al. and Grønmo et al. use MDA based techniques to assist in the creation of ontologies for semantically enriching services which are to be composed [7, 9]. However, these systems do not consider the distribution pattern of the resultant composition, resulting in a fixed centralised distribution pattern.

6. Conclusion

Mechanisms to assist in the generation of Web service-based compositions are desirable. We have combined existing techniques based on architectural modeling and pattern-based development, with the emerging areas of MDA and semantic descriptions. Our contribution is the novel application of semantics to distribution pattern modeling. Web services described using semantic languages, such as OWL-S, can be automatically assessed for compatibility, and their input and output messages can be mapped to each other. Semantics enable the semi-automatic generation of a distribution pattern model, which is subsequently used to guide the generation of an executable system.

7. Acknowledgments

The authors would like to thank the Irish Research Council for Science, Engineering and Technology IRCSET.

References

[1] R. Barrett and C. Pahl. Semi-Automatic Distribution Pattern Modeling of Web Service Compositions using Semantics. In *Proc. Tenth IEEE International EDOC Conference*, Hong Kong, China, October 2006.

[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*, Palo Alto, California, July 2006.

[3] D. Skogan and R. Grønmo and I. Solheim. Web service composition in uml. In *Proc. 8th International IEEE Enterprise Distributed Object Computing Conference*, pages 47–57, Monterey, California, September 2004.

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

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

[6] T. Gardner. UML Modeling of Automated Business Processes with a mapping to BPEL4WS. In *Proc. First European Workshop on Object Orientation and Web Service (EOOWS)*, Darmstadt, Germany, July 2003.

[7] R. Grønmo and M. Jaeger. Model-driven semantic web service composition. In *Proc. 12th Asia-Pacific Software Engineering Conference (APSEC 2005)*, Taipei, Taiwan, 2005.

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

[9] J.T.E. Timm and G.C. Gannod. A model-driven approach for specifying semantic web services. In *Proc. International Conference on Web Services*, Orlando, Florida, USA, 2005.

[10] 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.

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

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

[13] 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.

[14] 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.