

Model Driven Design of Distribution Patterns for Web Service Compositions

Ronan Barrett
School of Computing
Dublin City University
Dublin 9, Ireland

Email: rbarrett@computing.dcu.ie

Claus Pahl
School of Computing
Dublin City University
Dublin 9, Ireland

Email: cpahl@computing.dcu.ie

Abstract

Increasingly, distributed systems are being constructed by composing a number of components, often legacy applications exposed using Web service interfaces. There are a number of architectural configurations or distribution patterns, which express how such systems are to be deployed. However, the amount of code required to realise these distribution patterns is considerable. Here, we propose a novel Model Driven Architecture using UML 2.0, which takes existing Web service interfaces as its input and generates an executable Web service composition, based on a distribution pattern chosen by the software architect.

1. Introduction

The development of composite Web services is often ad-hoc and requires considerable low level coding effort for realisation. This effort is increased in proportion to the number of Web services in a composition or by a requirement for the composition participants to be flexible. This paper proposes a modeling and code generation approach to address this requirement. This approach suggests Web service compositions have three modeling aspects. Two aspects, service modeling and workflow modeling, are considered by [3]. We consider an additional aspect, distribution pattern modeling, which expresses how the composed system is to be deployed.

2. Distribution Patterns

Distribution patterns provide a more complete picture of the distributed system at the platform-independent model level. Previously, it was difficult to ascertain how the composed system would be deployed by studying the models, if any, or code of the system. Having the ability to model and alter the distribution pattern, allows an enterprise to con-

figure its systems as they evolve, to realise different non-functional requirements. Two examples of distribution patterns are centralised and peer-to-peer. Collaboration languages, such as Web Services Business Process Execution Language (WS-BPEL), can enable the runtime enactment of distribution pattern based compositions.

3. Modeling and Transformation Approach

Our approach to distribution pattern modeling and Web service composition generation consists of five steps (see Figure 1). Relations are defined at the meta-model level using the recently standardised Query/View/Transformation (QVT) notation, to verify the preservation of semantics between related meta-models.

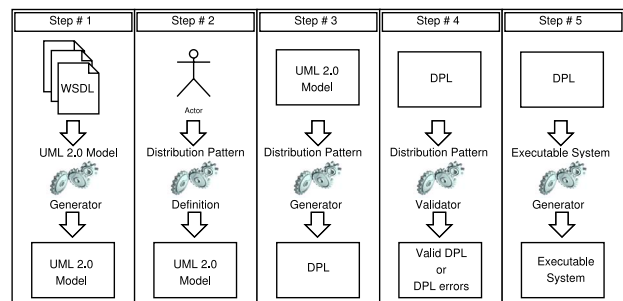


Figure 1. Overview of modeling approach

Step 1 - From Interface To Model: The initial step takes a number of Web service interfaces as input, and transforms them to a UML model. A UML activity diagram is chosen to model the distribution pattern as it provides features which assist in clearly illustrating the distribution pattern, while providing sufficient information to drive the generation of the executable system. UML ActivityPartitions, or swim-lanes are used to group a number of actions within an activity diagram. In our model, these actions will represent WSDL operations. Any given interface has one or more

ports that will have one or more operations, all of which will reside in a single swim-lane. CallBehaviorActions model the operations of the WSDL interface. Each CallBehaviorAction has two types of pins, InputPins and OutputPins, which map directly to the parts of the WSDL messages. We use a standard extension mechanism of UML, called a profile, to provide a more descriptive model than the standard UML constructs allows. Profiles define stereotypes and tagged values that extend UML constructs, enabling distribution pattern metadata to be applied to the constructs.

Step 2 - Distribution Pattern Definition: The UML model produced in step 1, requires additional modeling. Guided by a chosen distribution pattern, and restricted by the UML meta-model/DPL Profile, the architect must manipulate the UML model by defining connections between individual Web services and map the messages from one service to the next. First the architect selects a distribution pattern and then assigns appropriate values to the tagged values of the stereotypes. Based on the chosen distribution pattern, the architect defines the sequence of actions by connecting CallBehaviorActions to one another, using UML ControlFlow connectors, each of which is assigned an order value. The architect then connects up the UML InputPins and OutputPins of the model, using UML ObjectFlows connectors, so data is passed through the composition. Partial automation of this step is considered in [1].

Step 3 - From Model to DPL: The UML model from step 2 may now be transformed to a Distribution Pattern Language (DPL) document instance. The transformation and resultant pattern instance are restricted by the DPL meta-model. This document, which is at the same level of abstraction as the UML model, is a representation of the distribution pattern which can be validated. The use of this new language allows non-MOF compliant description frameworks, such as Architectural Description Languages, to be used in place of UML as the transformation source.

Step 4 - Model Validation: The DPL document instance, is verified at this step to ensure the values entered in step 2 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. Although this validation may be considered redundant as the pattern definition has already been restricted by the QVT relations, we envisage supporting non-QVT compliant modeling languages as set out in the previous step.

Step 5 - DPL to Executable System: The verified DPL document instance is now used to generate all the interaction logic documents and interfaces required to realise the

distribution pattern. The generator, restricted by the appropriate platform specific collaboration meta-model, creates interaction logic documents based on the UML profile settings. The generated artifacts and supporting infrastructure are now ready for deployment. Dynamic deployment of the executable system is considered in [2].

4. Tool Implementation

TOPMAN (TOPology MANager) is our solution to distribution pattern modeling using UML, and subsequent composition generation. We utilise XSLT to transform the Web services interfaces, to a UML 2.0 activity diagram. A GUI based UML tool can be used by the software architect to define the distribution pattern, by importing the generated model. Upon completion, the model can be transformed to a DPL instance document using XSLT. The document is then verified by an XML validating parser. Finally the document is transformed using XSLT and XML DOM, resulting in the generation of interaction logic and interface documents needed to realise the distribution pattern. Each transformation is written to implement a previously defined QVT relation between source and target meta-models.

5. Conclusion

An engineering approach to the composition of service-based software is required. We have presented a modeling and transformation approach, along with an implementation for expressing distribution patterns. Our novel modeling aspect, distribution patterns, expresses how a composed system is to be deployed, providing for the documentation and realisation of various non-functional requirements.

6. Acknowledgment

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, USA, July 2006.
- [3] 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.