Supporting Systems QoS Design and Evolution through Model Transformations *

Amogh Kavimandan and Aniruddha Gokhale
Dept. of EECS, Vanderbilt University
{amoghk,gokhale}@dre.vanderbilt.edu

Abstract

We describe Quality of service pICKER (QUICKER), a model- driven QoS mapping toolchain for supporting the QoS design and evolution of systems software. QUICKER automates the mapping of QoS requirements onto middleware-specific QoS configuration options by (1) choosing appropriate subset of QoS options for given QoS policies and (2) assigning values to each of these selected QoS options. QUICKER also provides support for validating the generated QoS configurations and resolving any dependencies between them using model checking.

Categories and Subject Descriptors D.1.2 [*Automatic Programming*]: Program Transformation

General Terms Middleware, Configuration, Design

Keywords Model-driven development, model tansformations, component middleware, web services, SOA, CCM

1. Introduction

Service Oriented Archictures (SOA), such as Web Services (WS), and component middleware technologies, such as Enterprise Java Beans (EJB) and CORBA Component Model (CCM), have raised the level of abstraction for the application developers by separating functional and non-functional aspects during the application software development lifecycle. In an effort to support a wider range of target application domains, these *systems software* technologies have evolved into a highly configurable and customizable systems software platforms that provide a number of configuration mechanisms to satisfy non-functional requirements of applications in each of these target domains. Owing to such a flexibility, however, the size of the system software configuration space (*i.e.*, the configuration options suited for an application and their appropriate value set) becomes large. A comprehensive knowledge of various configuration options, their inter-dependencies, and how they impact application-level requirements is critical to correctly configure the systems platform. Failure to correctly map policies to low-level configuration options will lead to a sub-optimal (or wrong) system software configuration degrading the overall application performance, or worst run-time errors that are costly and difficult to debug.

This poses a significant challenge for application developers who are seldom experts at performing optimal configuration of the systems platform for their application.

Our Solution. We have developed QUality of Service pICKER (QUICKER) [1], a model-driven engineering (MDE) QoS mapping toolchain to support QoS design of the implementation systems software(s). QUICKER provides a system composition language to enable application developers to annotate applications with QoS policies *i.e.*, QoS requirements of applications. The current implementation of QUICKER supports QoS configuration of applications implemented using CCM and WS systems software platforms. It also defines model transformations to automatically map these QoS policies to a set of platform-specific QoS configuration options that are required to satisfy the specified QoS policies. Finally, QUICKER uses generative techniques to synthesize model-checking input for the application to verify various QoS options generated through model transformations.

2. Model-driven QoS mapping for Systems Software Platform

Figure 1 shows the model-driven QUICKER toolchain. We discuss its capabilities in this section.

Challenge 1: Capturing QoS policies. Application developers are domain experts with a thorough understanding of the application business logic but often lack the knowledge about the QoS configuration space of the systems platform to optimally configure the systems platform for their appli-

* This work is sponsored by grant from Lockheed Martin Advanced Technology Laboratories.

OOPSLA '07 October 21-25, 2007, Montreal, Canada.

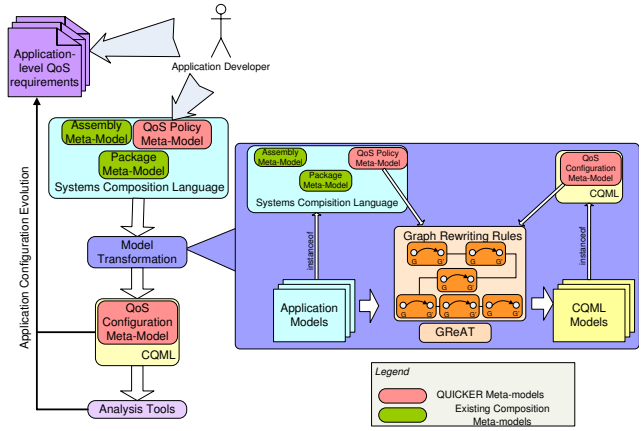

Figure 1. QUICKER Toolchain

cations.

Solution: Domain-specific QoS policy specification using QUICKER. We have developed a QoS Policy domain-specific modeling language (DSML) that allows specification of desired QoS policies. The QoS Policy DSML meta-model shown in Figure 1 has the semantics of systems QoS requirements. By focusing on *what* is expected from the application (*i.e.*, systems QoS requirements) rather than *how* the QoS may be achieved (*i.e.*, low-level platform-specific QoS options), QUICKER enables easier and intuitive QoS specification.

Challenge 2: Identifying the set of platform-specific QoS options from application policies. Once systems QoS policies are captured using the QoS Policy DSML, these policies still need to be mapped onto the correct platform-specific QoS options. Current solutions to resolve this challenge are ad-hoc, *i.e.*, manually identifying the QoS options from the given systems QoS policies. An application typically goes through several iterations during its software development cycle (and possibly, during its maintenance cycle, in order to incorporate new requirements). Without automated tool support, particularly for large-scale applications, it is time consuming, error prone and in some cases infeasible for application developers to correctly configure the systems software for a given QoS policy set.

Solution: Automated QoS policy mapping through model transformations. We have defined transformation algorithms using the GReAT toolchain that translate the QoS policies into detailed, platform-specific QoS configuration options. QUICKER model transformations define rules that perform the following activities: (1) choosing an appropriate subset of QoS options that high-level systems QoS policies map to, and (2) choosing valid values for each of these QoS options to perform *QoS configuration* of the systems platform. As shown in Figure 1, transformations are defined in terms of meta-models, and thus can be used repeatedly for any application models that conform to the QoS Policy DSML. The generated QoS options are themselves models that are amenable to further analysis/transformations.

Challenge 3: Validating platform-specific QoS options. QoS options for an application may be associated at various levels of granularity of the systems platform. For example, RT-CCM configuration options have component-level associations, RT event channel service options have asynchronous connection-level associations, and WS Reliable Messaging options have port-level (*i.e.*, event source and/or event sink) associations. Depending on their associations, QoS options are often dependent on each other and hence a change in value of one QoS option may affect many other QoS options. Thus, such dependencies must be resolved before the application can be prepared for deployment for which manual approaches to resolve these challenges have significant limitations.

Solution: Model-checking to validate (generated) QoS options. QUICKER extends the Bogor Input Representation (BIR) [2] with new constructs that enable specification and model-checking of system properties more closely to the domain of implementation platform. Using these extensions, a systems platform-based application can be expressed in terms of BIR and the properties of the application(*i.e.*, QoS options) can be validated using the model-checking framework. QUICKER uses generative techniques on the models of QoS options in order to synthesize: (1) input to the Bogor model-checking framework in order to model-check the QoS options, and (2) descriptors in middleware-specific format that are required to configure application QoS before deployment.

3. Concluding Remarks

In this paper we discussed QUICKER, a model-to-model transformation toolchain that provides an automated, scalable, and reusable approach to resolve the QoS mapping challenge. QUICKER provides intuitive modeling abstractions to facilitate application QoS policy specification and model transformation algorithms that map these QoS policies to the platform-specific QoS options that will ultimately achieve the desired application QoS. QUICKER is available as open-source from `www.dre.vanderbilt.edu/CoSMIC/`.

References

[1] Amogh Kavimandan, Krishnakumar Balasubramanian, Nishanth Shankaran, Aniruddha Gokhale, and Douglas C. Schmidt. QUICKER: A Model-driven QoS Mapping Tool for QoS-enabled Component Middleware. In *Proceedings of the 10th IEEE International Symposium on Object/Component/Service-oriented Real-time Distributed Computing*, Santorini Island, Greece, May 2007.

[2] Robby, Matthew Dwyer, and John Hatcliff. Bogor: An Extensible and Highly-Modular Model Checking Framework. In *Proceedings of the 4th Joint Meeting of the European Software Engineering Conference and ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE 2003)*, Helsinki, Finland, September 2003.