Transformation of UML Design Model into Performance Model: A Model-Driven Framework

Ramrao Wagh

Indian Institute of Technology Bombay
Powai, Mumbai, India 400076
Tel: 912225764966
E-mail: ramrao@it.iitb.ac.in

PhD Supervisors
Bernard Menezes[1], Umesh Bellur[1],

[1] Indian Institute of Technology Bombay, Powai, Mumbai, India 400076
{bernard, umesh}@it.iitb.ac.in

Abstract. Software Performance Engineering is receiving increasing attention in today's software dominated world. Compared to research work in performance evaluation in hardware and networks, this field is still in its nascent stage. Many methods have been proposed but majority of them are unable to adapt in the software development life-cycle dominated by professionals without substantial performance engineering background. We propose UPE - a Model Driven Software Performance Engineering Framework to facilitate performance engineering within software development life cycle, based on OMG's MDA initiative.

Keywords: Software Performance, Model Transformation, UML, MDA.

1 Introduction

For a large and complex software systems such as enterprise systems designed using distributed technology, it is of paramount importance that the stakeholders are satisfied that the proposed system will meet necessary functional as well as non functional requirements. As development of such systems is costly it is necessary to get a realistic estimate of the level of quality requirements likely to be achieved, before the resources are committed for the project. The best possible time to make reliable estimates that are dependable, especially for performance, is at architectural design stage since performance in terms of response time of software, throughput and device utilization is dependent on the behavioral as well as structural aspects of the software design model.
Since UML[5][13] is being increasingly used to express design, our overall research work is aimed at devising performance analysis methods applicable to software design models expressed using UML artifacts.

This paper is structured as follows. In section 2 we look at the related work and the limitations of this work; Section 3 then presents our solution framework UPE that is based on MDA, section 4 will present the conclusion.

2 Related Work

Over the last decade there has been a steady rise in the number of performance prediction methods for software systems at design stage [14]. Majority of these methods follow an approach where a performance model is constructed for the target software and later evaluated. All these methods are broadly based on at least one of the following performance models: Queuing Network Model, Petri Nets, Process Algebra, Markov Chains, Simulation [2]. One of the major drawbacks of majority of these methods is that one has to manually construct the performance model by studying the software system. As average software developer is not conversant with performance modeling, a specialist may be needed to carry out the performance modeling.

Although number of UML based methods have been proposed for the performance estimation from software design [2], they still suffer from the following major drawbacks in general:

- Restricted to use of particular set of UML diagrams
- There is no proper way of specifying performance parameters
- No provision to represent the performance results back into the model
- Transformation methods are not based on standards
- They are tailored towards use of specific, often proprietary performance analysis tools
- Make use of UML designs that are stored in lengthy and verbose XML form.

We will briefly mention three recent important methods.

The Core Scenario Model (CSM) [9] extracts the relevant performance information scattered implicitly and explicitly around various UML diagrams into a single CSM which will then facilitate the conversion into any desired performance model that can be executed by performance tools. Two transformations are required i.e. from UML model to CSM (U2C) and from CSM to Performance model (C2P). Authors propose to use Query, View, Transformations (QVT) [10] for these transformations but due to non-availability of tools are currently using XSLT based transformations.

[11] apply the Model driven performance analysis to a distributed middleware based enterprise application developed using EJBs. They use MDA supported lightweight extension mechanism using different UML profiles and merging them to represent the design model. Specifically UML profile for SPT and EJB are merged to define a joint profile. A simple profile to represent queuing network model is specified as Analysis profile. Lack of tool support has been cited as the reason for not automating the transformations defined.

[4] proposes a comprehensive framework for achieving interoperability amongst various tools based on MDA approach. Considering the lack and shortcomings of existing tools and specifications, they prescribe a three layer approach comprising of a

technology independent metamodeling layer (MML), technology specific model implementation layer (MIL) and tool layer (TL) to exploit use of available technologies and tools built on MDA and XML. Due to lack of proper tool support for implementing QVT transformations, XML based technologies such as XSLT/XQuery is used for transformation.
Many of the drawbacks mentioned earlier are still retained in these three representative methods. More specifically, Out of these three methods, the one proposed by [4] looks most promising as a complete but complex framework is defined by them to achieve transformation by using proposed standards as well as combining it with existing technologies. However, their target performance model is not generic. On the other hand the idea of representing an intermediate performance model in CSM is also promising as it combines all the performance related information in one place before a complex transformation is applied. Approach by Skene is more tied to a particular platform, i.e, EJB, but could be applied to any other problem domain by merging suitable domain information.

3 Unified Performance Engineering Framework

We propose a Model Driven Framework based on various MDA[6] standards. **Figure 1** depicts the overall UPE framework. All the metamodel as well as transformations are based on Meta Object Facility-MOF[7] which is a meta-metamodeling language from OMG. We claim that basing the design model as well as analysis models on same meta-metamodel i.e. MOF will facilitate easy transformation.

The main advantage of this approach is that performance estimation can be done continuously during the design stage. It will also maintain the performance parameters separate from the design model so as not to clutter the design model during development. Another advantage will be that transformation will be based on metamodel level that will be expressed in a high-level model transformation language compared to complex XML based transformations.

We present the UPE approach in more details in the following sub-sections by describing the various models and the required transformations:

3.1 Design Model

As design evolves during the development process, any of the UML features deemed fit by the developers are to be used. Organizations usually follow some process standards such as Unified process, which prescribe the set of diagrams to be used at different stages. However, in the initial stage, we will focus on Use case diagram to represent the workload, activity diagram to represent the scenarios, and deployment diagram to represent the resource allocations. In particular we will be using UML 2.0 notations and diagrams as well as any platform specific model (PSM) based profiles such as CORBA or J2EE.

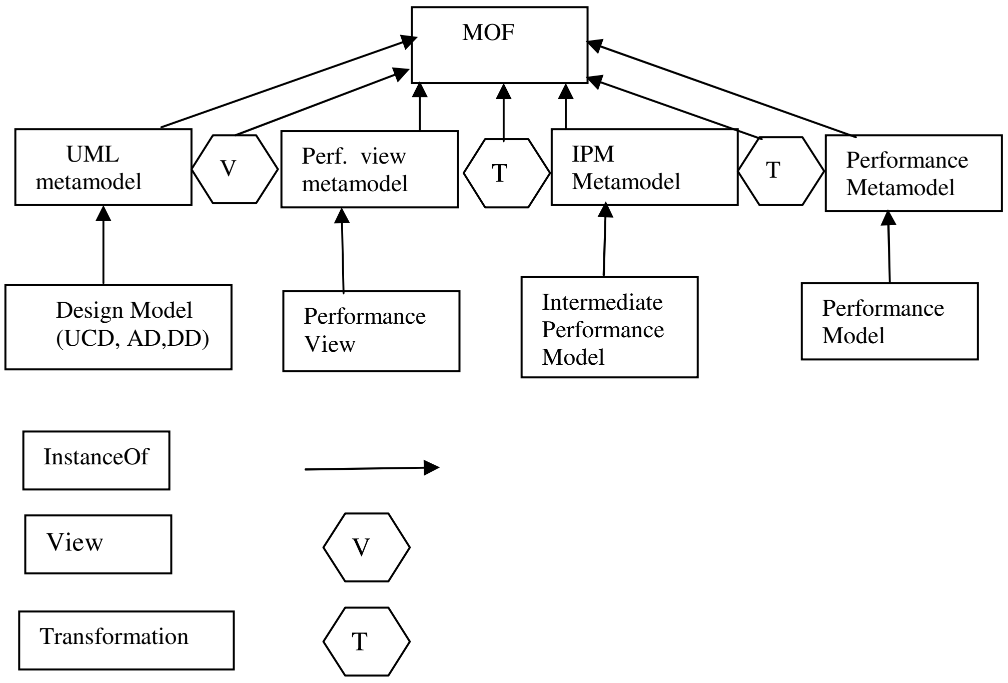

Figure 1 Unified Performance Engineering Framework

3.2 Performance View

We propose a novel idea of defining performance views to represent performance related input as well as results based on the concept described in MOF QVT [10]. This mechanism will allow any other view, say, security model, to be described without interfering with performance model on the same design model. This will be an editable view that will allow the selected UML diagrams from the design model to be annotated with performance data by using an extension of UML profile for Schedulability, Performance and Time (SPT) [12].

3.3 Intermediate Performance Model

From this performance view, the intermediate performance model that combines performance related information from all the design artifacts and performance view is generated. This will involve a transformation that will create the Intermediate Performance model by systematically analyzing each of the activity diagrams and combining their information into one performance model. The IPM metamodel is closely related to CSM approach of [9].

3.5 Performance Analysis Model

The transformation of this IPM will result in the performance model of choice such as Queuing Network Model, SPA, Stochastic Petri Nets, etc. These different transformations need to be designed to achieve the desired performance model. This model then will generate the parameters that will be submitted to the Performance tool for solving the model.

3.6 Transformations

Various model transformations will be needed for converting Design Model annotated with performance into Intermediate Performance model and for transforming Intermediate performance Model into Performance Analysis Model.
An editable view generation is required for performance annotations to be specified on the software design model.

3.7 Feedback

The results from the performance tools will be sent back and represented in the performance view. This will also need similar reverse transformations which we are not currently focusing upon.
As QVT based approach will lead to standardization in the model transformation field, we are planning to use one of the suggested approaches for our research work. We propose to use Object Constraint Language(OCL) 2.0 [8] and MTL from QVT Partners [3] that is based on the relational approach [1]. Although some of the required specification standards are not completed by the standards body, we will apply the concepts, not from the point of view of complying with the standards but in a way to validate them for a specific domain such as performance. We are hopeful that this may contribute to the field of model driven development by identifying shortcomings and proposing alternative solutions.

4 Conclusion

We have outlined UPE - a framework for model driven software performance engineering in this paper. The need for such an approach was established; the relevant standards and technologies that are under various stages of completion themselves were described. Since the underlying field of MDD itself is in a high state of flux, it is a challenge to devise an approach that will be adaptive to such changing scenario.
As model driven paradigm is increasingly being embraced by researchers and industry, We are hopeful that we can contribute to this pool of knowledge by applying this approach to the domain of Software Performance Engineering thereby validating

the same and making performance analysis an integral part of the software development activity.

References

1. Akehurst D.H., Kent S., Patrascoiu O.: A Relational Approach to Defining and Implementing Transformations between metamodels, Journal of System and Software, Issue 2 (2003)
2. Balsamo S., Di Marco A., Inverardi P., Simeoni M: Model-Based Performance Prediction in Software Development: A Survey. IEEE Transactions on Software Engineering, vol. 30, n. 5, pp. 295-310 (2004)
3. Appukuttan B., Clark T., Reddy S., Laurence T., Venkatesh R.: A Model driven approach to building implementable model transformations. In Workshop on Model Driven Architecture: Foundations and Applications, University of Twente, (2003)
4. D'Ambrogio A.: A Model Transformation Framework for the Automated Building of Performance Models from UML Models. In Proceeding of ACM Workshop on Software and Performance. (2005)
5. Object Management Group: Unified Modelling Language(UML) Specification: Infrastructure, Version 2.0, ptc/03-09-15, (2003).
6. Object Management Group: MDA-Guide, V1.0.1, omg/03-06-01, (2003).
7. Object Management Group: Meta Object Facility (MOF)2.0 Core Specification, ptc/03-10-04, (2003).
8. Object Management Group: Unified Modelling Language:Object Constraint Language version 2.0, Draft Adopted Specification, ptc/03-08-08,(2003).
9. Petriu D.., Woodside M: A Metamodel for Generating Performance Models from UML Designs, Proc. of UML Conference, LNCS 3273, pp 41-53 . (2004)
10. Object Management Group: QVT-Partners. MOF Query/Views/Transformations, Revised Submission. OMG Document: ad/2003-08-08 (2003)
11. Skene J ,Emmerich W: Model driven performance analysis of enterprise information systems, Electronic Notes in Theoretical Computer Science(ENTCS), Elsevier Science B.V. (2003)
12. Object Management Group: UML profile for SPT, OMG Document: ptc/04/02/01(2001)
13. Object Management Group: UML 2.0 Superstructure,ptc/03-08-02, (2003)
14. Proceedings of ACM Workshop on Software Performance, 1998, 2000, 2002, 2004, 2005
15. Object Management Group: XML Metadata Interchange (XMI) Specification, version 2.0, (2003)