UPM- An Eclipse Plug-in to Incorporate Performance Requirements into UML Models

Ramrao Wagh
Department of Computer Science
& Technology
Goa University
SPO Goa University
918322451937, 919423882964
ramrao@unigoa.ac.in
ramrao@it.iitb.ac.in

ABSTRACT
In this paper, we describe the design and implementation of UPM- an Eclipse Plug-in that enables specifying the performance requirements directly within the UML design model. We have successfully implemented the most often used diagrams of UML in our editor and also discuss a case study on which the tool was used upon. The limitations as well as future enhancements to make it fully functional are also discussed.

Categories and Subject Descriptors
D.2.2 [**Software Engineering**]: Design Tools and Techniques-*Object-oriented design methods*

General Terms
Measurement, Performance, Design.

Keywords
Performance Model, Model Transformation.

1. INTRODUCTION
Performance requirements constitute a major source of nonfunctional requirements that must be met by the final software system that is to be delivered. Majority of the requirements and design specification methodologies and techniques focus on facilitating the representation of functional requirements and ignoring nonfunctional requirements specification. As a result, these requirements are either not represented or represented in a separate document. This leads to problem such as difficulty to modify the performance requirements when other requirements change, difficulty in imposing quality assurance measures so that these requirements are also ensured to be complied during the software development lifecycle. Moreover, with performance playing a major role in determining the success or failure of the deployed system in the enterprise and web applications it is pertinent that a systematic method is needed to carry performance requirements as a part of functional requirements specifications throughout the development lifecycle.

ISEC'08, February 19–22, 2008, Hyderabad, India.
ACM 978-1-59593-917-3/08/02.

Our research work[10] is focusing on this issue of specifying the performance requirements in the software architecture and design models. We have based our approach on the MDA philosophy in which all the design models carrying performance annotations are transformed into performance models that can generate the necessary data to work with the performance modeling tools.

The advantages of this approach are manifold. Firstly, performance requirements are incorporated as part of the design model thereby avoiding the inconsistency problem. Secondly, as design is refined successively, we can also fine tune the performance requirements and as a result the estimates also will be more and more accurate. Thirdly, it would be possible for us to convert the design model into any desired performance model of our choice.

We have chosen UML to represent the design models and used the standard extension mechanisms of UML to allow the design models to hold the performance requirements.

2. RELATED WORK
Numerous tools [1][3] are present that permit to specify performance requirements and then to estimate the desired performance objective but all of them require that you follow the modeling notation that is unique to these tools. Moreover, automatic transformation is not available. The developer has to leave the deign view and construct these models separately. Developers are also not well-versed with the formalism that is the basis of these tools.

In order to overcome these difficulties, we have developed UPM as an Eclipse plug-in that aims to provide full functional UML modeling capability with the added advantage of being able to represent performance requirements.

3. DESIGN OF UPM
The UPM is developed using UML[4][5] and various model driven development techniques based on OMG's MDA[6] initiative. We created the necessary metamodels for UML diagrams and performance model and then generated the Editor for each of these diagrams.

The UML 2.0 metamodel is described using a meta-metamodeling language called MOF[7] by OMG. We have used Ecore model, which is a popular implementation of MOF 2.0 specification to

generate the metamodel of UML diagrams of interest to us. Particularly, we used EMF(Eclipse Modeling Framework) plug-in within Eclipse[2] which allows one to create Ecore models that represent the metamodels for UML diagrams. The ability to represent performance related information on each of these diagrams was added by adding appropriate modeling elements that have been described in UML Profile for SPT[8] to represent performance models. Figure 1 shows a metamodel created by us to represent class diagram structure.

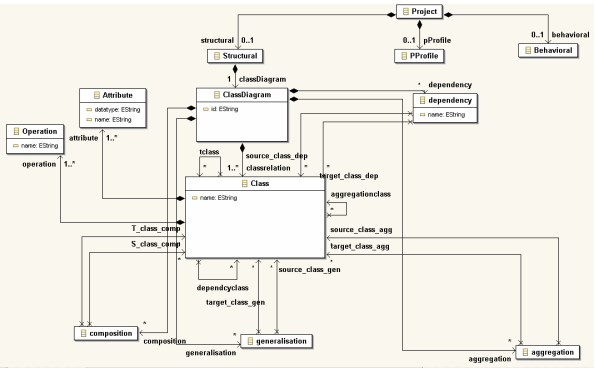

Figure 1: UML Class Diagram Metamodel

Once these metamodels were created, we used GMF(Graphical Modeling Framework) plug-in to create the basic graphical editor that has the necessary capabilities to draw the UML diagrams as well as specify the performance requirements.

4. USING UPM ON A CASE STUDY

We used the UPM Editor to represent a Web Video case study that is used in UML Profile for SPT[9] to describe the use of performance profile. It is seen that we have been successful in representing all the performance modeling notations defined in the performance profile. We show some of the snapshots of the UPM Editor plug-in representing the various diagrams of Web Video case study.

5. FUTURE WORK

The current version of UPM Editor supports five out of 13 UML 2.0 diagrams which are adequate for performance modeling. Support for drawing other diagrams is being worked upon. The major limitation of the present tool is that we have to manipulate the entire five diagrams separately. Current version of GMF does not allow us to integrate the five metamodels under one editor. We are hopeful that future releases of GMF will solve this problem. We are also working toward the mechanism to maintain consistency among the information stored within these models as they represent different views of the same underlying system. We need a consistency mechanism to check the information among the various diagrams as well as consistency among the various refinements of the same diagrams as design proceeds ahead.

We are also building tools to represent various performance models and transformation models to use the information represented by the model created by UPM editor by applying the proposed model transformation QVTP[9] specification of MDA.

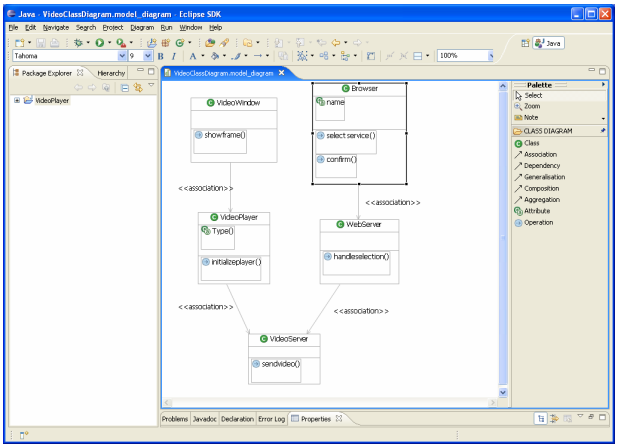

Figure 2: Drawing Class Diagram in UPM

6. ACKNOWLEDGEMENTS

The author of the paper wishes to thank Jelissa Rodrigues, Michelle Coutinho, and Veena Damle, for their help in implementation of this editor.

7. REFERENCES

[1] 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)

[2] Eclipse website: www.eclipse.org

[3] Petriu D, Woodside M: A Metamodel for Generating Performance Models from UML Designs, Proc. of UML Conference, LNCS 3273, pp 41-53 . (2004)

[4] Object Management Group: Unified Modeling Language (UML) Specification: Infrastructure, Version 2.0, ptc/03-09-15, (2003).

[5] Object Management Group: UML 2.0 Superstructure,ptc/03-08-02, (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: UML profile for SPT, OMG Document: ptc/04/02/01(2001)

[9] Object Management Group: QVT-Partners. MOF Query/Views/Transformations, Revised Submission. OMG Document: ad/2003-08-08 (2003)

[10] Ramrao Wagh, Umesh Bellur, Bernard Menezes: Transformation of UML Design Model into Performance Model - A Model-Driven Framework. ICEIS (3) 2006: 576-5