UML-PSI: the UML Performance SImulator

Moreno Marzolla Simonetta Balsamo
Dipartimento di Informatica
Università Ca' Foscari di Venezia
via Torino 155, 30172 Mestre (VE), Italy
marzolla@dsi.unive.it, balsamo@dsi.unive.it

Abstract

In this paper we describe UML-Ψ, a software performance evaluation tool based on process-oriented simulation. The tool can be used to evaluate performances of software systems described as annotated UML diagrams. UML-Ψ transforms the software model into a performance model based on process-oriented simulation, executes the performance model and collects results. Performance results are inserted into the software model as tagged values associated to the relevant UML elements.

1. Introduction

Quantitative analysis of software systems is being recognized as an important issue in the software development process. Performance evaluation should be highly integrated in the software development process [7]. The software engineer should be provided with integrated performance and specification environments where the performance evaluation tools should have the following characteristics:

- *No special performance modeling expertise required*: the tool should be easy to use, requiring little or no specific performance modeling knowledge and skills from the user.
- *Automation*: the integrated software performance tool should be mostly automatic; the user should not be required to perform actions or computations by hand.
- *Feedback*: the performance results should be automatically reported at the software architecture level.

UML-Ψis a software performance evaluation tool which uses UML for software specification, and process oriented simulation as the performance model. The approach integrates UML software specification given by a set of annotated diagrams, with a discrete-event simulation model whose solution gives a set of average performance indices providing automatic feedback at the software architectural level. The UML-Ψ tool can be used by the software designer and the performance modeler to generate and evaluate the performance model.

The software model is drawn, annotated and modified using a UML CASE tool which must be capable of exporting the model in XMI format [6]. UML-Ψ considers the following UML diagrams to derive the performance model: Use Case, Activity and Deployment diagrams. UML-Ψ parses the annotated UML model, which must be exported in XMI format from the CASE tool, and builds a process-oriented simulation model of the software system. Simulation parameters are derived from the annotated UML diagrams. The simulation program is finally executed and computes a set of performance indices of the software system under study: resources utilization and throughput, and the mean execution time of actions and Use Cases. Simulation results are reported back into the original software model as UML tagged values associated to the relevant elements. This allows us to give a user-friendly feedback at the software design level. Figure 1 illustrates a schematic representation of UML-Ψ usage.

Figure 1: Using UML-Ψ

2. Performance Modeling

The UML-Ψ tool builds a process-oriented simulation model [2] from UML specifications. The performance model is composed by a set of concurrent, interacting simulation processes. We define three types of simulation processes: workloads, actions and resources. Workload processes generate sequences of requests to the system; work-

loads may be open or closed. An open workload represents an infinite stream of requests being generated from outside the system, while a closed workload is made of a fixed number of requests circulating through the system. Each request, upon arrival, triggers the execution of a sequence of actions. An action is a request of service from an active resource (e.g., processor) or acquisition/release of a passive resource (e.g., memory).

UML-Ψ parses Use Case, Activity and Deployment diagrams in order to build an internal representation of the UML model from which it derives a process-oriented simulation model. The simulation model has the same structure as the software specification, due to an almost one-to-one mapping between UML elements and simulation processes. Use Case diagrams represent workloads, Activity diagrams represent the actions being triggered by requests, and Deployment diagrams represent active and passive resources. The UML model has to be annotated according to a subset of the Profile for Schedulability, Performance and Time Specification [5] as described in [4, 1]. Annotations are inserted into the UML model as stereotypes and tagged values; such annotations must be provided by the user before the performance model generation phase.

The UML-Ψ tool executes the simulation model by using both user-supplied parameters, that are given as tagged values associated to UML elements, and the parameters included in a configuration file. Simulation parameters include, but are not limited to, the number of times an action is repeated, the service demand of actions, expressed as random variables with a given distribution, scheduling policies of active resources, and others. We consider the specification of tag values by the Tag Value Language (TVL), a subset of the Perl language [8] proposed in [5]. This is motivated by the need to express such values in a complex way, for example by using expressions such as arithmetic or boolean ones. The configuration file is a Perl program which is executed before evaluating tag values. In this way it is possible to define variables in the configuration file and use them inside tag values.

The simulation model is implemented as a C++ program, using the facilities provided by the general-purpose simulation library described in [3]. The simulation model is eventually executed and the computed results are inserted into the XMI document as tagged values associated with the UML elements they refer to. Therefore the results of performance analysis are available to the user which can open again the UML model by using the CASE tool, so allowing the software designer to check whether the software architecture meets the performance goal and possibly repeat the process for further analysis of modified software systems. Simulation results are computed with steady state analysis and with confidence intervals [2].

3. Conclusions

In this paper we briefly introduced UML-Ψ, a simulation-based performance evaluation tool for early assessment of software performances. UML-Ψ transforms annotated UML diagrams into a simulation model, implements the model using process-oriented simulation and evaluates the performance model. Simulation results are inserted back into the UML model as new tagged values associated to the relevant UML elements.

Currently we are extending UML-Ψ in several directions. First, we are planning to use a larger subset of the annotations from [5], in order to allow the modeler to describe the software in more detail using different kinds of UML diagrams. We are also trying to integrate UML-Ψ with other software performance tools based on different performance models derived from the same UML model.The ultimate goal is to integrate different kinds of quantitative software analysis techniques into a general framework allowing different kinds of quantitative and qualitative analysis, e.g., reliability, on the same software specification. Finally, we are evaluating how the UML-Ψ tool, and the associated software performance evaluation approach, can be extended to cope with the forthcoming UML 2.0.

Acknowledgments This work has been partially supported by MIUR research project FIRB “Performance Evaluation of Complex Systems: Techniques, Methodologies and Tools”.

References

[1] S. Balsamo and M. Marzolla. Simulation modeling of UML software architectures. In D. Al-Dabass, editor, *Proc. of ESM’03, the 17th European Simulation Multiconference*, pages 562–567, Nottingham, UK, June 9–1 2003.

[2] A. M. Law and W. D. Kelton. *Simulation Modeling and Analysis*. McGraw–Hill, 3rd edition, 2000.

[3] M. Marzolla. `libcppsim`: a Simula-like, portable process oriented simulation library in C++. In G. Horton, editor, *Proc. of ESM’04, the 18th European Simulation Multiconference*, Magdeburg, Germany, June13–16 2004.

[4] M. Marzolla. *Simulation-Based Performance Modeling of UML Software Architectures*. PhD Thesis TD-2004-1, Dip. di Informatica, Università Ca’ Foscari, Venezia, Italy, Feb. 2004.

[5] Object Management Group (OMG). UML profile for schedulability, performance and time specification. Final Adopted Specification ptc/02-03-02, OMG, Mar. 2002.

[6] Object Management Group (OMG). XML Metadata Interchange (XMI) specification, version 1.2, Jan. 2002.

[7] C. U. Smith and L. Williams. *Performance Solutions: A Practical Guide to Creating Responsive, Scalable Software*. Addison-Wesley, 2002.

[8] L. Wall, T. Christiansen, and J. Orwan. *Programming Perl*. O’Reilly & Associates, third edition, July 2000.