A Software Performance Engineering Tool based on the UML-SPT *

Elena Gómez-Martínez and José Merseguer
Dpto. de Informática e Ingeniería de Sistemas.
Universidad de Zaragoza, Spain.
{megomez,jmerse}@unizar.es

1. Introduction

Software performance engineering [13] (SPE) proposes methods to evaluate performance of software systems early in the development process. A SPE accepted approach consists in deriving performance models from UML specifications, usually annotated according to the OMG Profile for Schedulability, Performance and Time Specification [12] (UML-SPT). Performance models use to be based on simulation environments or a given modelling formalism: (layered) queuing networks, stochastic Petri nets or stochastic process algebras. Some tools are being developed under this SPE approach [9, 4, 6, 5]. OMG has defined a framework for SPE tools, which is a standard for communicating information between tools, see Figure 1.

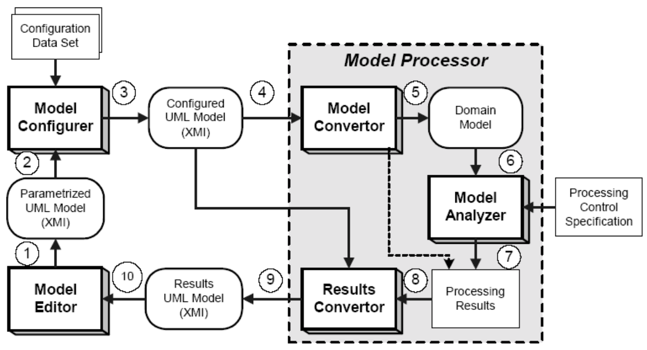

Figure 1. Tool architecture (from UML-SPT).

We introduce here a new SPE tool [1] that fits in the OMG framework and implements most of the features given in [10, 8]. The tool allows to design UML diagrams annotated according to the UML-SPT, and automatically generates a performance model in terms of Generalized Stochastic Petri nets (GSPN), using the GreatSPN [7] file format. The input of the tool constitutes a software model designed as a set of UML state machines, whose activities can be modelled using UML activity diagrams. The class diagram specifies system population and the deployment diagram models some inter-nodes characteristics (i.e. network speed).

The tool comprises several performance queries useful to analyze system properties. Query results are obtained running the GreatSPN programs and reported back to the tool.

* This work was supported by the project TIC2003-05226 of the Spanish Ministry of Science and Technology.

2. Software Performance Tool

The OMG framework establishes a division of components based on its functionality: the Model Editor, the Model Configurer and the Model Processor. The tool has been designed as a set of Java modules, that are plugged into the ArgoUML [2] tool. Each Java module implements a module of the framework.

Model Editor This functionality is provided by the diagram editor of ArgoUML, that assumes the metamodels of the 1.3 version of UML. Then it allows to create and modify the input of the tool (the software model) and to annotate these diagrams according to the UML-SPT, using the Tag Value Language (TVL).

TVL, a subset of the Perl language, allows to specify performance requirements and execution parameters. The annotations currently supported by the tool are summarized in Table 1. As an example, `<<PAstep>>{PAprob=0.8}`, could be used to annotate a message with its probability to occur.

UML models are exported into XMI files, allowing the standard exchange of information with other tools.

Model Configurer Its functionality consists in converting a UML model, in XMI format, into a configured UML model using a configuration data file. Then, it substitutes in the XMI file, the tagged values written in TVL with evaluable expressions, for the equivalent evaluated expressions, according with the actual values in the configuration file.

Annotation	Stereotype	Tag	Diagram
Act. duration	PAstep	PArespTime	SM, Act
Message probability	PAstep	PAprob	SM, Act
Message size	PAstep	PAsize	SM
Network speed	PAcommunication	PAspeed	Deploy
Initial number of objects	PAclosedLoad	PApopulation	Class
Initial state	PAinitialCondition	PAinitialState	SM
Resident classes	GRMcode	GRMmapping	Deploy

Table 1. Performance annotations

Model Convertor This module takes as input the configured UML model in XMI format to produce a GSPN model that represents the behaviour of the modelled system. This translation is performed according to [10] for the state machines and to [8] for the activity diagrams.

The performance annotations in the class and deployment diagrams are also considered by this module. They are useful, among others, to obtain the initial marking of the GSPN model or the firing rates of GSPN transitions representing messages delivered through the network.

The GSPN model consists of a set of GreatSPN [7] files, that are available for the tool user. Then s/he can use them to directly feed the GreatSPN tool and to calculate their own defined metrics (queries).

Model Analyzer This module uses the GSPN model to invoke a GreatSPN analysis program. It happens when the user executes one of the predefined performance queries. Currently, the available queries are:

- **Time in a state** [in SM diagram]: mean time spent by an object in a given state. Useful to compute the time needed to perform a complex activity (carried out in such state).
- **Stay time** [in SM diagram]: percentage of time that the objects spend in each one of its states. Useful to compute how long a resource is idle.
- **Transmission speed** [in Deployment]: network connection delay between two physical nodes of the model.
- **Message delay** [in SM diagram]: delay of a message when travelling through the network. Taking as starting point its delivering and end point its acknowledge.

Results Convertor The main function of this module is to convert the results of the analysis (obtained by the Model Analyzer) back to the Model Editor, in a way that a software engineer can interpret them.

3. Further work

A number of new features will improve the tool. Some of them are ongoing development.

- Our SPE method takes into account the sequence diagram [3] (SD), but the current version of ArgoUML does not. We consider to replace the SD with the collaboration diagram, supported by ArgoUML.
- The use case diagram, present in our SPE method and also in ArgoUML, has not been incorporated in the tool yet.
- New performance and dependability queries have to be implemented, e.g. execution time or failure time.
- The tool offers support for a subset of the TVL grammar. It has to be extended, then allowing more complex performance expressions.
- To prevent the user to install GreatSPN, it could be published as a web service. Then, the tool could remotely invoke it.
- The tool could be integrated with other Petri nets analyzers, such as Möbius [11].

Acknowledgments
We would like to thank Juan Pablo López-Grao, Borja Fernández, Isaac Trigo, Alvaro Iradier and Luis Carlos Gallego for their work in the development of the tool.

References

[1] ArgoSPE. `http://argospe.tigris.org`.

[2] ArgoUML project. `http://argouml.tigris.org`.

[3] S. Bernardi, S. Donatelli, and J. Merseguer. From UML SDs and SCs to analysable PN models. In *ACM WOSP 2002*, pages 35–45.

[4] V. Cortellessa, M. Gentile, and M. Pizzuti. XPRIT: An XML-based tool to translate UML diagrams into EG and QN. In *IEEE QEST 2004*, pages 342–343.

[5] S. Distefano, D. Paci, A. Puliafito, and M. Scarpa. UML design and software performance modeling. In *ISCIS 2004*, pages 564–573.

[6] S. Gilmore and L. Kloul. A unified tool for performance modelling and predicition. In *SAFECOMP'03*, pages 179–192.

[7] GreatSPN tool. `http://www.di.unito.it/~greatspn`.

[8] J. P. López-Grao, J. Merseguer, and J. Campos. From UML ADs to SPN: Application to SPE. In *ACM WOSP 2004*, pages 25–36.

[9] M. Marzolla and S. Balsamo. UML-PSI: the UML Performance SImulator. In *IEEE QEST 2004*, pages 340–341.

[10] J. Merseguer, S. Bernardi, J. Campos, and S. Donatelli. A compositional semantics for UML SMs aimed at performance evaluation. In *IEEE WODES 2002*, pages 295–302.

[11] The Möbius tool. `http://www.mobius.uiuc.edu/`.

[12] Object Management Group. *UML Profile for Schedulabibity, Performance and Time Specification*, January 2005.

[13] C. U. Smith. *Performance Engineering of Software Systems.* Addison–Wesley, 1990.