Model Interoperability for Performance Engineering: Survey of Milestones and Evolution

Connie U. Smith[1] and Catalina M. Lladó[2]

[1] Performance Engineering Services, PO Box 2640, Santa Fe, New Mexico, 87504-2640 USA, www.spe-ed.com

[2] Universitat de les Illes Balears. Departament de Ciències Matemàtiques i Informàtica. Ctra de Valldemossa, Km. 7.6, 07071 Palma de Mallorca, Spain
cllado@uib.es

Abstract. Next generation Software Performance Engineering tools will exploit a model interoperability paradigm that uses the performance modeling tool best suited to the software/hardware architecture issues and the life cycle stage of the assessment. The paradigm allows the use of existing tools to the extent possible without requiring extensive changes to them. The performance model solution should be transparent to the user. Significant milestones have been accomplished in the evolution of this paradigm. This paper covers key results in the areas of Model Interchange Formats, model transformations, tools, specification of experiments and results, and extensions for real-time and component-based systems. It then offers conclusions on next steps and the future of the model interoperability paradigm.

1 Introduction

Software performance engineering (SPE) is a systematic, quantitative approach to constructing software systems that meet performance requirements [39]. SPE uses performance models to provide data for the quantitative assessment of the performance characteristics of software systems as they are developed. SPE has evolved over more than 30 years and has been demonstrated to be effective during the development of many large systems.

Although sound performance analysis theories and techniques exist, they are not widely used because they are difficult to understand and require heavy modeling effort throughout the development process [41]. Consequently, software engineers usually resort to testing to determine whether the performance requirements have been satisfied. To ensure that these theories and techniques are used, they must be made more accessible–integrated into the software development process and supported with tools.

First generation performance engineering tools were developed for users with modeling expertise. Examples include [7, 17, 40]. To use them for SPE requires performance specialists to work with developers to obtain software design and performance data for evolving software. The principal problem is the gap between software developers who need the techniques and the performance specialists who

have the skill to conduct studies with first generation modeling tools. This limits the ability of developers to explore the design alternatives.

The ideal next generation SPE tool will provide support for many SPE tasks in addition to the obvious requirements for performance modeling. The ideal SPE tool will use the performance modeling tool best suited to the software/hardware architecture issues and life cycle stage of the assessment. It also requires a cost-effective solution that works with existing tools to the extent possible without requiring extensive changes to them. The performance model solution process should be transparent to the user of the SPE tool.

Model interoperability seeks cooperation among existing tools that perform different tasks. Thus, model interoperability makes it possible to create a software specification in a development tool, then automatically export the model description and specifications for conducting performance assessments, then obtain results for considering architectural and design options.

This paper examines the milestones in the evolution from first generation performance modeling tools for SPE to the model interoperability framework. Note that our focus was initially on tools for SPE, however the model interoperability framework supports general performance engineering tasks for a variety of modeling tools and application domains. The following sections address the significant milestones. We then offer conclusions on next steps and the future of the model interoperability paradigm.

2 Version 1 Model Interchange Formats

A model interchange format (MIF) is a common representation for performance model data that can be used to move models among modeling tools. A user of several tools that support the format can create a model in one tool, and later move the model to other tools for further work without the need to laboriously translate from one tool's model representation to the other. For example, an analyst might create a model of a server platform to conduct several studies, then move the model to a tool better suited to network analysis.

MIFs require minor extensions to tool functions (import and export) or creation of an external translator to convert file formats to/from interchange formats. They enable easy comparison of results from multiple tools, and the use of tools best suited to the task. Without a shared interchange format, two tools would need to develop a custom import and export mechanism. Additional tools would require a custom interface to every other tool resulting in a N(N-1) requirement for customized interfaces. With a shared interchange format, the requirement for customized interfaces is reduced to 2N.

Related, earlier work in this area is limited. Beilner advocated hierarchical and modular descriptions of models and showed how multiple solution techniques can be used with these model descriptions. The HIT environment demonstrated the feasibility of this approach [7]. Likewise, the Esprit Integrated Modelling Support Environment, IMSE, integrated several individual modeling tools and

showed the potential of interchanging tools for performance studies [21]. We seek a mechanism for a loose connection among a variety of modeling tools.

MIFs originated in the VLSI community with the Electronic Design Interchange Format (EDIF) for exchanging VLSI design information among design tools. Subsequently the software community adopted this paradigm for the Case Data Interchange Format (CDIF) standard [4]. In the CDIF standard, the information to be transferred between two tools is known as a model. The contents of a model are defined using a meta-model. A meta-model defines the information structure of a small area of CASE (such as data modeling) known as a Subject Area. Each meta-model is, in turn, defined using a meta-meta-model. The original meta-meta-model is based on the Entity-Relationship-Attribute (ERA) approach.

MIFs for performance modeling were first introduced at the 1995 Tools Conference in Heidelberg. The Software Performance Model Interchange Format (S-PMIF) defined the SPE information requirements for CASE tools with a meta-model and a transfer format for interchange among software performance modeling tools [37]. The Performance Model Interchange Format (PMIF) for the interchange of queueing network models (QNM) was proposed in a panel session and subsequently published in [38]. The session had a lively debate about the concept of model interoperability. It was generally agreed that the concept was good, but the debate centered on the content of the proposals and whether they were ready for technology transfer.

The initial work by Smith and Williams was funded by the National Science Foundation. Without additional funding and without tools to facilitate implementation, the work remained dormant for several years.

3 Version 2 PMIF

With the advent of XML (Extensible Markup Language) tools, MIF implementation became feasible. In 2004, Smith and Lladó developed Version 2 of PMIF [35]. Version 2 converted the meta-model to UML and implemented the XML transfer formats. Modifications were required to exchange QNM among unlike tools. The primary difference was the addition of routing probabilities because the number of visits alone could be insufficient information for some QNM topologies. Probabilities are specified with Transit elements for routing among nodes and for workload entry. Number of visits was retained for tools that must specify the service time per visit rather than total service demand. Other minor changes improved convenience and eliminated redundancy.

We implemented a prototype export mechanism from the *SPE·ED* software performance modeling tool [40] into pmif.xml, and a prototype import mechanism from pmif.xml into the Qnap system performance modeling tool [28]. Subsequent work implemented the reverse exchange exporting from Qnap and importing to *SPE·ED* further demonstrating the soundness of PMIF 2 [34].

The CDIF strategy is export everything you know and provide defaults for other required information; import the parts you need and make assumptions if

you require data not in the meta-model. Everything you know is not necessarily everything you use. For example, *SPE·ED* uses visits to specify routing, but it knows about probabilities, and it is relatively easy to calculate them.

We used the prototypes to study several examples. Our use of unlike tools helped us find limitations in the meta-model and find a general way to resolve them. Example solutions confirmed that the pmif.xml transfer was successful. The examples provided a set of models that are well documented, with reproducible results, that may be used by others who wish to explore the pmif.xml approach to model interoperability.

4 Flurry of Related Work

After establishing the viability of performance model interchange formats, there was a flurry of activity among researchers building on this work. It is described in the following sections.

4.1 Design Models to Performance Models

Cortellessa, et.al. collaborated on an unified approach using S-PMIF [32], derived from the original SPE meta-model. They demonstrated the viability of a complete path from software design to performance validation, based on existing tools that were not designed to interact with each other.

This overall process is beneficial because no single monolithic tool is good for everything. Early in development one needs to quickly and easily create a simple model to determine whether a particular architecture will meet performance requirements. Precise data is not available at that time, so simple models are appropriate for identifying problem areas. Later in development, when some performance measurements are available, more detailed models such as Queueing Network Models (QNM), Stochastic Petri Nets (SPN), or Process Algebra (PA) models can be used to study intricacies of the performance of the system. At that time, tools that provide features not in the simpler models are desirable. These industrial strength models are seldom appropriate earlier in development because constructing and evaluating them requires additional time and expertise that is seldom justified when performance details are sketchy at best.

Thus S-PMIF is an intermediate representation between design models and system performance models. It is useful for studying software architecture and design options. Other researchers go directly from software models to system performance models [19, 10, 25, 5, 12]. These approaches use XML to transfer design specifications into a particular solver.

Lopez-Grao et al. propose a method to translate several UML diagram types to analyzable GSPN models where performance requirements are annotated according to the UML SPT (Schedulability, Performance and Time) profile [23]. As a slightly more general approach, Gu and Petriu present in [18] a process to transform UML models to Performance Models that can be later specialized for different performance modeling formalisms.

4.2 MIF Extensions

Lately, efforts have developed intermediate models between various design models (different UML diagrams, different UML versions, non UML approaches, etc.) and various system performance modeling tools. Woodside et al. [11] propose the Core Scenario Model (CSM), claiming that it captures the essence of performance specification and estimation as expressed in the UML SPT profile. Grassi et al.[15] focus on the construction of software systems according to the component-based development (CBD) approach. They propose KLAPER (Kernel Language for Performance and Reliability analysis) as an intermediate language between software specification models and performance models. KLAPER is defined as a MOF (Meta-Object Facility) metamodel, which can also be transformed to/from KLAPER models.

Cortellessa investigated the possibility of building an unified ontology for software performance modeling [8]. His study originates from the comparison of three existing metamodels, (i.e. CSM, SPT, PMIF), identifies intersecting areas among the metamodels, and at the same time provides a roadmap to work towards merging metamodels into a unifying ontology.

Harrison et al. [20] generalized the PMIF specifications by considering more abstract collections of interacting nodes using concepts compatible with the Reverse Component Agent Theory (RCAT). The interactions are more general in that they synchronize transitions between a pair of nodes rather than describing traffic flows.

4.3 MIF Tools

Lladó and colleagues developed a Web Service for solving QNM using PMIF [30]. This established the feasibility of a plug-and-play approach for model interoperabilty and provided a foundation for other implementations. Cortellessa and colleagues built a Web Service to a collection of QNM tools along with a tool for creating a PMIF specification from a GUI based tool [31].

The PMIF schema supports syntactic validation of models: it confirms that the XML file contains everything it is supposed to, that IDREFS point to a declared ID, etc. Even if the PMIF is syntactically valid, however, the models might be semantically incorrect. Lladó and colleagues developed a specification for semantic validations and a tool to confirm that a PMIF model is correct before attempting to convert it to a tool-specific file [14, 34].

Woodside and colleagues developed a tool architecture, PUMA (Performance by Unified Model Analysis) as a framework into which different kinds of software design tools can be plugged as sources, and different kinds of performance tools can be plugged as targets [11].

5 Automation of Experiments and Results

Interchange formats have been defined for queueing network models (PMIF), software performance models (S-PMIF), layered queueing networks (LQN), UML,

Petri Nets and other types of models. In each interchange format, a file specifies a model and a set of parameters for one run. Automating experiments requires 3 steps: specifying the experiment; executing one or more model runs and collecting output from those runs; and converting the output into meaningful results for the experiment. Each is addressed in the following sections. Figure 1 shows these steps. The performance model MIF is in the top-left while the experiment specification (Ex-SE) is shown at the top-right. These files are combined and used as input for one or more performance modeling tools. Each tool generates the performance metric output as specified for each experiment. The last step transforms the output into desired results as specified by the Results-SE.

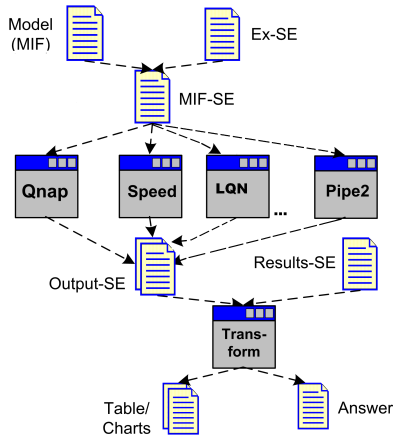

Fig. 1. Model interoperability framework

5.1 Experimentation

Some general approaches to experimentation have been proposed:

- Zeigler [43] has proposed a framework for modeling and simulation that included an experimental frame of the conditions under which the system or its model are to be exercised.
- The IMSE Experimenter tool facilitates performance modeling studies within the Integrated Modeling Support Environment (IMSE) [21] .
- The James II simulation modeling framework [22] allows any experiment definition that has a suitable plug-in for reading and converting the experiment definition (i.e., an instance of IExperimentReader).

Some modeling tools provide an ability to solve models multiple times while varying parameter values. Most current performance modeling tools have a Graphical User Interface (GUI) that leaves it to the user to conduct model

experiments. Tools developed before GUIs were prevalent, however, provided experimentation features as part of their user interface: Qnap [28], LQN [3], HIT and Hi-Slang [2].

Tools for other types of modeling have similar capabilities:

- Probabilistic Symbolic Model Checker (PRISM) [24], provides a GUI that finds undefined constants in the model and queries the user for a value.
- The Network Simulator [29] has an object-oriented script language TCL that lets the user set up the model topology, place agents, inject events, etc.
- the Möbius modeling environments [9] supports higher level experimental design with two types of two-level experimental design strategies: Plackett-Burman and two-level factorial design, and two types of response surface designs (central composite and Box Behnken).

Smith, Lladó and colleagues incorporated elements from these approaches to provide a comprehensive model interoperability solution for experimentation. We defined the Experiment Schema Extension (Ex- SE) for specifying a set of model runs and the output desired from them [36, 33]. The Ex-SE has the expressive power to specify iterations, alternations, assignments of values, actions based on model results and more. This schema extension provides a means of specifying performance studies that is independent of a given tool paradigm.

We use the term *Schema Extension* because it is used in combination with a host schema and customized to refer to solutions and outputs provided by the host. For example, we have specific instances of Ex-SE for Queuing network models and Petri net models. Figure 2 shows the schema extension instance for PMIF models, PMIF-Ex.

Experiment specifications can be created with an XML editor, with a GUI tool for describing experiments and generating the XML, or by exporting an experiment definition along with the model interchange format from a tool with experimentation functionality. Figure 3 shows a screenshot of the experimenter editor for PIPE2 [1], a Petri net modeling and analysis tool.

5.2 Model Output

An Output schema extension Output-SE is added to the host schema to specify the XML format to be used for output from the experiments. Like the Ex-SE it is customized to the type of output supported by the host schema.

The PMIF Output-SE includes:

- a solution ID for relating the output to the experiment
- the value used for *Ranges* or other variables used in that solution
- the OutputWorkload (overall results by workload)
- OutputNode (overall results by Node)
- OutputNodeWorkload (results by Workload for Nodes).

Tools without experimentation need an Experimenter tool to interpret the experiment, invoke the tool for each `<Solve>`, and return the output. This can be

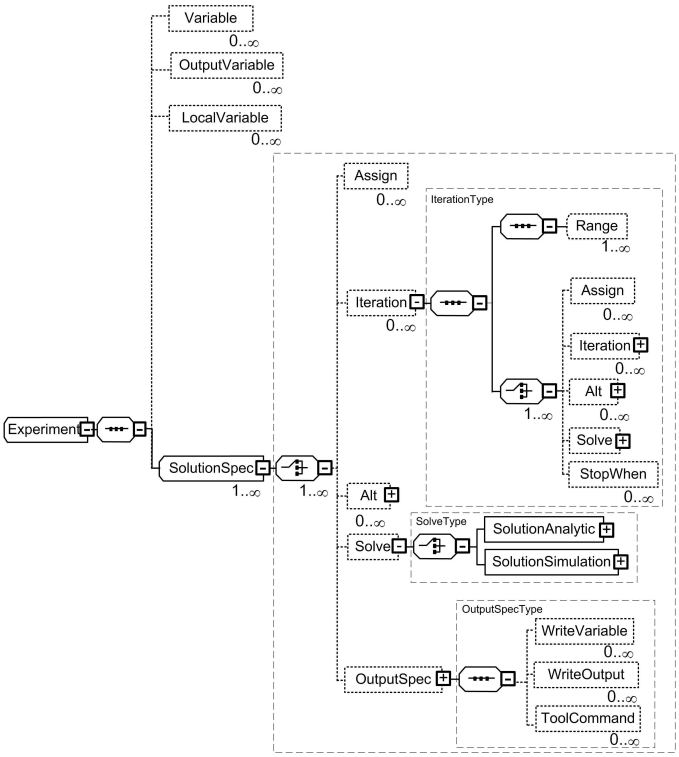

Fig. 2. Experiment Schema Extension, PMIF instance

a general tool that can work with multiple solvers, or tools with experimentation can adapt their Import mechanism to generate the tool specifications for the experiment. This is a more efficient implementation because the tool only needs to be invoked once, and the model does not need to be parsed multiple times.

Lladó and colleagues have developed a tool, EXperiment Output to SpreadSheet (EXOSS), that takes the XML output from one or more experiments and produces a spreadsheet (xls) file for easily viewing the performance output. The tool works with PMIF queueing network models and Petri net models. It has also been integrated into the PIPE2 Petri net modeling tool.

5.3 Experiment Results

The IMSE Experimenter has an experimental plan with at least one *analysis* specification that describes how experimental *results* are obtained from model outputs. Thus, outputs from multiple runs can be used to obtain overall measures (e.g., mean, standard deviation). The PRISM GUI also provides the ability to specify a graph of the results and to export it in several formats (png, jpg, eps).

We examined Use Cases for building and analyzing performance model results then studied the output metrics and results that are most often desired for the

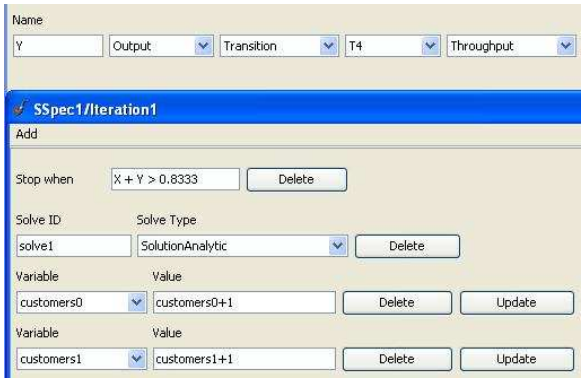

Fig. 3. PIPE2 experiment editor

Use Cases [33]. A review of published papers found some examples of common tables of results, such as response times for workloads with varying workload intensities, measured versus model results for response time and utilization for multiple workloads, etc. We also found common charts with line plots, columns or bars showing throughput versus response time; response time versus service time; number of users versus response time, etc.

Our conclusion for both QNMs and Petri net models is that the primary results are tables and charts. In the future, more sophisticated techniques for visualization of results may be desired, such as images of multidimensional colored surfaces. We did not address those types of results – we will leave that to the tools that render these types of images. Those tools usually can import the performance values to be displayed from a file and it should be straightforward to transform tool output into such a file format.

The most common format for tables and charts is xls as in spreadsheet tools such as Excel and OpenOffice Calc. However, the most common document preparation system for research publications is LaTeX. Our approach supports transforming the output metrics to tables and charts in xls and LaTeX. The results for an experiment are transformed into one or more tables/charts. Several tools have been implemented for experiment results [33].

5.4 Multiple Domains

In [36], Ex-SE was illustrated with an instance of the extension in which the interchange schema was the PMIF and also LQN. The Ex-SE is compatible with a large number of interchange formats including: PMIF, S-PMIF, LQN, GPMIF, and Petri Net xml specifications. In general, one appends the Experiment type definition into the other XML schema, and changes the Variable type specifications to match new schema if necessary.

Petri nets are different from QNM and other performance model paradigms because they provide additional representation and analysis capabilities in addi-

tion to performance analysis. Examples include constraints on tokens in places, invariant analysis, reachability analysis, and so on.

Lladó and colleagues presented a specific, extended instantiation of the Ex-SE for Petri nets (PN-Ex) [26] and analogous extensions to the Output and Results specifications. Both the Output-SE and the Results-SE required specification of an array of values instead of a single value.

The Experiment Schema Extension is not limited to use with models. It could be used with a measurement experimenter. Some of the terminology should be customized, such as changing Solve to Run, and changing the SolutionTypes to something such as ToolSpecification.

6 Real-time and Component-Based Systems

Some work has addressed the performance analysis of component-based systems. Wu and Woodside use an XML Schema to describe the contents and data types that a Component-Based Modeling language (CBML) document may have [42]. CBML is an extended version of the Layered Queuing Network (LQN) language that adds the capability and flexibility to model software components and component-based systems. Becker et al. address components whose performance behavior depends on the context in which they are used [6]. They address sources of variability such as loop iterations, branch conditions, and parametric resource demand, and then use simulation to predict performance in a particular usage context. Grassi et al. extend the KLAPER MOF meta-model to represent reconfigurable component-based systems in [16]. It is to be used in autonomic systems and enable dynamic reconfiguration to meet QoS goals. These approaches are performance-centric in that they create/adapt a model of component based systems specifically for performance assessment. We prefer to work with generally accepted architecture representations, and use a common interchange format (S-PMIF) that allows the use of a variety of performance modeling tools to provide performance predictions for architecture and design alternatives.

Moreno, Smith and Williams have extended the S-PMIF to include features necessary for evaluating real-time, component-based systems [13]. They subsequently presented a substantially modified S-PMIF, version 2.0, that adapted the meta-model so that it can be expressed in terms of the Ecore meta-meta-model– the core meta-model of the Eclipse Modeling Framework. Several changes were made to better align S-PMIF 2.0 with other performance-related meta-models (PMIF, LQN, and ICM), to clarify terminology, and simplify the M2M transformations [27]. This work demonstrated the use of the model interoperability approach for component-based real-time systems, and implemented a M2M prototype to transform an intermediate constructive model (ICM) to S-PMIF.

We found that preserving the type hierarchy and associations of the S-PMIF meta-model in the schema facilitates the implementation of S-PMIF interchange support by tools using strongly typed modeling technologies to generate the XML such as EMF or ATL. This work opened a door to allow the performance analysis of CCL specifications with other analysis tools without the need for

additional integration effort. This means that standard SPE models can easily be used for analysis of systems specified in CCL. We demonstrated the ease with which the S-PMIF can be employed to transform additional design notations into software performance models, thus building on the previous UML-based approaches. In the future, it may be possible to unify the various interchange formats as suggested by [8]. In the meantime, it makes sense to extend the meta-models as necessary to create a superset of the necessary information for performance assessment.

7 Conclusions

This paper introduced the model interoperability approach for Software Performance Engineering. It presented significant milestones and stepping stones in the evolution of the approach: Version 1 Model Interchange Formats; Version 2 PMIF; Flurry of work in model transformations, MIF extensions and MIF tools; Automation of experiments and results; and Extensions for real-time and component-based systems. Our focus was initially on tools for SPE, however the model interoperability paradigm supports general performance engineering tasks for a variety of modeling tools and application domains.

MIFs provide a mechanism for exchanging performance models among tools. The exporting and importing tools can either support the MIF or provide an interface to read/write model specifications from/to a MIF file. The comparison of multiple solution techniques across tools can lead to a wide range of benefits. It is a significant result that MIFs can be used by anyone to exchange information relatively easily between two tools that provide a file input/output capability. It does not require the tool developer to modify code to be of use.

The basic version of PMIF 2 supports queueing network models that may be solved using efficient, exact solution algorithms. It was appropriate to restrict the domain for initial research. Our future work will create an extended PMIF and the supporting tools to cover model features that are supported by most simulation-based tools.

S-PMIF 2 is substantially different from the 2005 version. So, for instance, prototypes developed for the earlier version would have to be modified if they are to support the additional features. Model interchange formats and interfaces, however, must be relatively stable to be viable. S-PMIF was based on concepts embodied in two earlier model interchange formats: EDIF for VLSI designs and the CDIF for software design interchange. Creators of EDIF envisioned the stability problem and addressed it by using a concept of levels that add functionality at each successive level and giving ownership to a standards organization that managed changes. S-PMIF 2 adds a level for analyzing Real-time systems; future levels may add features for additional types of analysis. Tools can continue to support a lower level without change, or may opt to modify interfaces to support additional functionality and/or other changes. Future work should use the newest version, even for the basic level. Using a standard organization to manage the contents of model interchange formats should be considered.

The Experimental Schema Extension (Ex-SE) allows specification of multiple model runs along with the results that are desired from them. The schema extensions provide a means of specifying performance studies that is independent of a given tool paradigm.

Our future work will address a general purpose Experimenter tool with its corresponding editor. We also plan to develop templates for the most frequent results. We will extend the framework to support models that provide simulation solutions and other analysis techniques including real time systems analysis. An interesting extension will include creating rules for specifying threshold values and highlighting results in tables that exceed the threshold. Moreover, we plan to develop a GUI based transformation tool from output to results which will simplify the specification of the most common tables and graphs.

Model interoperability enables the use of the best tool(s) for the particular performance assessments. Early work demonstrated the benefit of comparing results from multiple tools; errors were found in several published case studies. Additional work is needed on model to model transformations among CSM, S-PMIF, PMIF, eDSPN, and others to enable comparison of results from multiple model paradigms. One approach is to unify the various interchange formats as suggested by [8]. Another is to ensure that sufficient modeling power is in each MIF to enable the M2M transformations.

The schemas and other information are available at `www.spe-ed.com/pmif/`. Several prototypes with source code are at `dmi.uib.es/~cllado/mifs`.

8 Acknowledgements

This work is partially funded by the TIN2009-11711 project of the *Ministerio de Educacion y Ciencia*, Spain. Smith's participation was sponsored by US Air Force Contract FA8750-09-C-0086.

References

1. Platform Independent Petri net Editor 2. `http://pipe2.sourceforge.net/`.
2. `www4.cs.uni-dortmund.de/{HIT}/`.
3. `www.sce.carleton.ca/rads/lqn/lqn-documentation/`.
4. Electronics Industries Association. CDIF - CASE Data Interchange Format Overview, EIA/IS-106, 1994.
5. S. Balsamo and M. Marzolla. Performance evaluation of UML software architectures with multiclass queueing network models. In *Proc. 5th International Workshop of Software and Performance*, Palma de Mallorca, Spain, July 2005. ACM.
6. S. Becker, H. Koziolek, and R Reussner. Model-based performance prediction with the palladio component model. In *WOSP 2007*. ACM, Feb 2007.
7. H. Beilner, Mäter, and N. Weißenberg. Towards a performance modeling environment: News on HIT. In *Proc. 4th Int. Conference on Modeling Techniques and Tools for Computer Performance Evaluation*. Plenum Publishing, 1988.
8. V. Cortellessa. How far are we from the definition of a common software performance ontology? In *WOSP 2005*. ACM, 2005.

9. T. Courtney, S. Gaonkar, G. McQuinn, E. Rozier, W. Sanders, and P. Webster. Design of experiments within the möbius modeling environment. In *Proc. of the Fourth International Conference on the Quantitative Evaluation of Systems*, pages 161–162, Edingurh, UK, 16-19 September 2007. IEEE Computer Society Press.
10. A. D'Ambrogio. A model transformation framework for the automated building of performance models from UML models. In *WOSP 2005*, pages 75 – 86, July 2005.
11. C.M. Woodside et al. Performance by unified model analysis (PUMA). In *WOSP 2005*, pages 1–12, July 2005.
12. N. Savino et al. Extending UML to manage performance models for software architectures: A queuing network approach. In *Proc. 9th Int. Symp. on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, SPECTS*, 2002.
13. G.A.Moreno, C.U.Smith, and L.G.Williams. Performance analysis of real-time component architectures: A model interchange approach. In *WOSP 2008*. ACM Press, June 2008.
14. D. García, C. M. Lladó, C.U. Smith, and R. Puigjaner. Performance model interchange format: Semantic validation. In *International Conference on Software Engineering Advances*. INRIA, October 2006.
15. V. Grassi, R. Mirandola, and A. Sabetta. From design to analysis models: A kernel language for performance and reliability analysis of component-based systems. In *Proc. WOSP*, pages 25–36, July 2005.
16. V. Grassi, R. Mirandola, and A. Sabetta. A model-driven approach to performability analysis of dynamically reconfigurable component-based systems. In *Proc. WOSP*. ACM, Feb 2007.
17. Adam Grummitt. A performance engineer's view of systems development and trials. In *Proceedings Computer Measurement Group*, pages 455–463, 1991.
18. G. Gu and D. Petriu. From UML to LQN by XML algebra-based model transformations. In *WOSP 2005*. ACM, 2005.
19. G. Gu and D. C. Petriu. XSLT transformation from UML models to LQN performance models. In *WOSP 2002*, pages 227–234, 2002.
20. P. Harrison, C.M. Lladó, and R. Puigjaner. A general performance model interchange format. In *Proc. of the First International Conference on Performance Evaluation Methodologies and Tools (Valuetools)*, 2006.
21. J. Hillston. A tool to enhance model exploitation. *Performance Evaluation*, 22(1):59–74, 1995.
22. J. Himmelspach, M. Rhl, and A. M. Uhrmacher. Component-based models and simulations for supporting valid multi-agent system simulations. *Applied Artificial Intelligence*, 24(5):414–442, 2010.
23. J.P. López-Grao, J. Merseguer, and J. Campos. From UML activity diagrams to stochastic Petri nets: Application to software performance engineering. In *WOSP 2004*. ACM, 2004.
24. G. Norman M. Kwiatkowska and D. Parker. Prism: Probabilistic model checking for performance and reliability analysis. *ACM SIGMETRICS Performance Evaluation Review*, 36(4):40–45, 2009.
25. M Marzolla and S. Balsamo. UML-PSI: the UML performance simulator (tool paper). In *Proc. 1st Int. Conf. on Quantitative Evaluation of Systems (QEST)*. IEEE Computer Society, 2004.
26. M. Melià, C.M. Lladó, C.U. Smith, and R. Puigjaner. An experimental framework for PIPEv2.5. In *5th Int. Conference on Quantitative Evaluation of Systems*, pages 239–240, St Malo, France, Sept. 2008. IEEE Computer Society Press.

27. G.A. Moreno and C.U. Smith. Performance analysis of real-time component architectures: An enhanced model interchange approach. *Performance Evaluation*, 67:612–633, May 2010.
28. D. Potier and M. Veran. QNAP2: A portable environment for queueing systems modelling. In D. Potier, editor, *First International Conference on Modeling Techniques and Tools for Performance Analysis*, pages 25–63. North Holland, May 1985.
29. The VINT Project. *The ns Manual*. UC Berkeley, LBL, USC/ISI, and Xerox PARC, 2010.
30. J. Rossello, C.M. Lladó, R. Puigjaner, and C.U. Smith. A web service for solving queueing networks models using PMIF. In *Proc. WOSP 2005*, pages 187–192, Palma de Mallorca, Spain, July 2005. ACM.
31. SEAlab Software Quality Group. WEASEL, a web service for analyzing queueing networks with multiple solvers. `sealabtools.di.univaq.it/Weasel/`.
32. C. U. Smith, V. Cortellessa, A. Di Marco, C. M. Lladó, and L. G. Williams. From UML models to software performance results: An SPE process based on XML interchange formats. In *Proc. of the Fifth International Workshop of Software and Performance*, pages 87–98, Palma de Mallorca, Spain, 12-14 July 2005. ACM.
33. C. U. Smith, C. M. Lladó, and R. Puigjaner. Automatic generation of performance analysis results: Requirements and demonstration. In Jeremy T. Bradley, editor, *Computer Performance Engineering, 6th European Performance Engineering Workshop*, volume 5652 of *LNCS*, pages 73–78. Springer, Berlin, 2009.
34. C. U. Smith, C. M. Lladó, and R. Puigjaner. Performance Model Interchange Format (PMIF 2): A comprehensive approach to queueing network model interoperability. *Performance Evaluation*, 67(7):548 – 568, 2010.
35. C.U. Smith and C.M. Lladó. Performance model interchange format (PMIF 2.0): XML definition and implementation. In *Proc. of the First International Conference on the Quantitative Evaluation of Systems*, pages 38–47, Enschede, The Netherlands, September 2004. IEEE Computer Society Press.
36. C.U. Smith, C.M. Lladó, R. Puigjaner, and L.G. Williams. Interchange formats for performance models: Experimentation and output. In *Proc. of the Fourth International Conference on the Quantitative Evaluation of Systems*, pages 91–100, Edingurh, UK, 16-19 September 2007. IEEE Computer Society Press.
37. C.U. Smith and L.G. Williams. Panel presentation: A performance model interchange format. In *Proc. of the International Conference on Modeling Techniques and Tools for Computer Performance Evaluation*, Heidelberg Germany, 20-22 September 1995. Springer, Berlin.
38. C.U. Smith and L.G. Williams. A performance model interchange format. *Journal of Systems and Software*, 49(1):63–80, 1999.
39. C.U. Smith and L.G. Williams. *Performance Solutions: A Practical Guide to Creating Responsive, Scalable Software*. Addison-Wesley, 2002.
40. SPE-ED. LS Computer Technology Inc. `www.spe-ed.com`.
41. C.M. Woodside, G. Franks, and D.C. Petriu. The future of software performance engineering. In *International Conference on Software Engineering (ICSE)*, pages 171–87. IEEE Computer Society, May 2007.
42. X. Wu and C. M. Woodside. Performance modeling from software components. In *WOSP 2004*, pages 290 – 301, January 2004.
43. B. P. Zeigler, H. Praehofer, and T. G. Kim. *Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems, Second Edition*. Academic Press, 2000.