How to Automatically Transform Performance Model Output into Useful Results

Connie U. Smith[1], Catalina M. Lladó[2], and Ramon Puigjaner[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, putxi@uib.es`

Abstract. This paper presents a performance model interoperability framework that brings together performance model interchange formats and experiment specifications with the automatic generation of performance analysis results for presentation and publication. We define the Use Cases and requirements and survey output and results used in practice. We present the output specification, the issues in the output-to-results transformation, the results specification schema extension, and a prototype implementation. A proof of concept example demonstrates the framework.

Key words: Performance modelling, Tool interoperability, Queueing networks, Results

1 Introduction

The concept of performance model interoperability was first introduced in 1995 [18]. Methods and tools supporting model interchange formats have evolved rapidly since 2004 with the introduction of XML as a viable mechanism for supporting model interchange [16].

Performance model interchange formats (PMIF) provide a mechanism for automatically moving performance models among modeling tools. Use of the PMIF does not require tools to know about the capabilities of other tools, internal data formats, or even existence. It requires only that the importing and exporting tools either support the PMIF or provide an interface that reads/writes model specifications from/to a file. Interchange formats have also been defined for layered queueing networks (LQN), UML, Petri Nets and other types of models.

Fig. 1 shows the model interoperability framework for creating and evaluating performance models. The model interchange format specifying a performance model is in the upper left. The formats may be created by translating design models into performance models [2, 15]. They may be created by a tool that provides a graphical user interface for specifying the model topology and parameters then creates model interchange files [13]. They may also be exported by one modeling tool in order to compare results to other modeling tools.

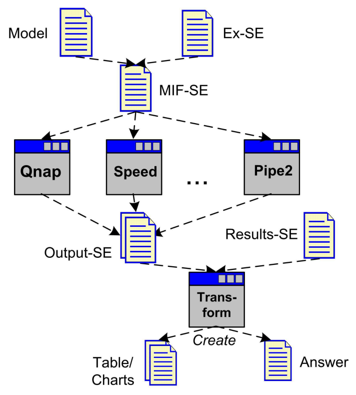


Fig. 1. Model interoperability framework

Another interchange format, the Experiment Schema Extension (Ex-SE), allows the user to define a set of model runs that vary parameters. The Ex-SE provides a means of specifying performance studies and the output desired from them that is independent of a given tool paradigm. Ex-SE was developed for use with a performance model interchange schema (e.g., PMIF); however, it may also be used in a stand-alone mode to specify studies for the tool in which the model was created, and it may be used to specify measurement as well as modeling studies. It was shown to work with QNM (Queueing Network Models) and LQN (Layered Queueing Networks) [17] and with Petri net models [10]. An experimental framework following this schema extension has also been developed for the Petri net modeling tool, PIPE2 [11].

The experiment specification file is shown at the top-right of Fig. 1. These two files (Model and Ex-SE files) are combined and used as input for one or more performance modeling tools. Each tool generates the performance metric output, such as response time, throughput, utilization, etc., specified for each experiment. A performance analyst typically studies this output to form conclusions about the results of the experiments, then prepares a presentation and/or report to explain the results.

This paper streamlines this last step in the model interoperability framework by defining a Results Schema Extension (Results-SE) that enables a user-customized transformation from the output of an experiment into the desired results.

Other work described in the next section has recognized the need for this last step. Our work develops the concept and provides a concrete realization of it.

The contributions of this work are:

- A definition of the most frequent Use Cases for this model interoperability framework
- A review of the typical types of output and results produced for queueing network based performance models for these Use Cases

– Definition of a modeling-paradigm independent schema for specifying the output of experiments and the transformation to results
– Implementation of a prototype demonstrating the feasibility of the approach
– Demonstration that the approach works.

The next section reviews some related work. Then we present the Use Cases for this framework, the results of a survey of typical output and results for these Use Cases, and an overview of our selected approach. Section 4 presents the model transformation approach, the supporting schemas for the Use Cases, and discusses the prototype implementation. Section 5 describes a proof of concept experiment. The summary and conclusions are in section 6.

2 Related Work

Extensive work has been done on performance model interchange formats and on the transformation of models into interchange formats. See [17] for an overview of this work. The following work has addressed the framework for modeling studies, output, and results.

Hillston [4] describes the IMSE Experimenter, a tool designed to facilitate performance modeling studies within the Integrated Modeling Support Environment (IMSE). The IMSE Experimenter allows the user to specify how the values of parameters in a model's input specification vary during an experiment. The experimental plan must include 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). In IMSE, *output* is produced by each run, *results* are produced by the experiment, and a *Reporter* tool creates and collates results and reports. There is no documentation of the capabilities of the Reporter tool. Our approach is not one reporter tool, but a framework for producing results. The IMSE work also recognizes the need for output metrics, analysis, and results, but does not elaborate on how that should work.

The Software Performance Experimenter (SPEX) is a tool for managing performance studies using LQN models [5]. SPEX also has provisions for input, output and control specifications. Input parameters may be specified using any legal Perl expression. *Observation indicators* are used to indicate output metrics of interest (e.g. utilization, throughput). These metrics may then be written to the result file. This approach addresses the output of the experiments, but not the analysis and results.

The results of an experiment enable us to determine whether a given system meets its performance requirements. Another way of tackling the same problem is to express performance-related queries in terms of requirements and measures. This is done by Performance Trees [20], a graphical formalism for the specification of performance requirement- and quantitative measure-based queries on stochastic system models. We believe that our interoperability framework can also integrate Performance Trees in the future.

3 Requirements for Producing Results

QNM may be used in a variety of fields from computer performance evaluation to traffic management or any other field that is interested in the behavior of queues and servers. This paper addresses computer performance evaluation; other applications of QNM may require extensions to the analysis and results.

The most common reasons that performance analysts build and analyze QNM models are to:

1. Monitor and report on operational system performance
2. Analyze capacity requirements for future workload volumes
3. Evaluate problematic systems, identify causes and study options
4. Compare model results to measurements
5. Conduct technical investigations to compare results from: multiple tools, different solution algorithms, or even different types of solutions.

The next step is to determine the output metrics and results that are most often desired for these Use Cases.

The Proceedings of the Computer Measurement Group are the main source of papers written and presented by practitioners about the results of their performance modeling studies [3]. We examined a sample of papers from the 25th anniversary edition of the proceedings (1974 through 1999) for papers that had the terms "performance model results." We found three general types of results: *tables*, graphs or *charts* as they are called in spreadsheet tools, and metric values that are embedded in the text of the paper.

Some examples of common tables are:

- Response times on 3 servers and the overall sum
- Response times for 2 workloads from 2 different workload intensities
- Columns with number of processors, relative throughput (0-1) and instructions per second; Rows for scalability data with 4,8 and 12 processors.
- Measured Response time, CPU Utilization, Database Utilization, and Wait time for multiple workloads, and another table with predicted values for the same metrics
- Measured Transactions per hour, Response time, CPU Utilization for several workloads; the same metrics for different transactions per hour, and another table with predicted values for the same metrics

The "metrics embedded in the text" are usually conclusions of the modeling experiments, and may or may not be supported by tables or charts. Some examples are: necessary number of processors, response time values, server response time, and CPU utilization.

These samples represent practitioner and researcher studies published in the CMG proceedings. Research results in other publications tend to be similar. The primary difference is that research papers often compare results from multiple solution algorithms and/or tools, and explore more ranges of values of input parameters (looking for model sensitivity).

Our conclusion is that the primary results are tables and charts. Charts are derived from tables, so they can be combined into one “result.” Since there are many common combinations of both tables and charts, the specifications for those should be streamlined.

The most common format for tables and charts is xls [1] as in spreadsheet tools such as Excel and OpenOffice Calc, and imported by most presentation and word processing packages. However, the most common document preparation system for research publications is LaTex. Our approach transforms the output metrics to tables and charts in xls and LaTex.

Additionally, we support two transformation modes: create a new table/chart and update an existing one. The update mode is convenient because it is unlikely that final results will be produced with one pass. It is also convenient when tables involve output from multiple tools. More importantly, it is easier to define table and chart formats by typing column and row headings or chart specifications directly into the spreadsheet rather than specifying transformation commands to create them.

This work does not address the metric values that are embedded in text. They have no tedious formatting requirements, and they might be best suited to the performance tree question/answer approach [20].

4 Model Transformation Approach

This section covers our approach to the framework steps for transforming the output of the performance model solutions into the desired results. The first section addresses the output. The next section explains the transformation by first describing the key issues and decisions, the update specifications, the create specifications, the approach for simplifying the generation of standard results, and some implementation issues.

4.1 Output Specification

A survey of the output produced by widely available QNM tools [19, 14, 7, 9, 12, 6] indicates that all of the tools have a notion of “principal results” that correspond to those in the earlier Output specification. Some simulation-based tools can provide additional metrics such as minimum, maximum, and variance of residence and response times, histograms of response times, and other metrics.

The Output Schema Extension is in Fig. 2. There is a solution ID for relating the output to the experiment. The “ValueUsed” applies to *Ranges* or other variables used in the experiment specification and reports the value used for that particular solution.

The principal results are in the OutputWorkload (overall results by workload), OutputNode (overall results by Node), and OutputNodeWorkload (results by Workload for Nodes). OutputNode ones are shown in Fig. 2, for more information see www.spe-ed.com/pmif/pmif-output.xml.

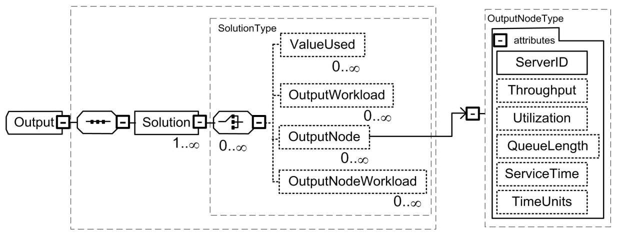


Fig. 2. Output Schema Extension

The output desired is specified in the Experiment specification. For each solution, the user may specify: WriteVariable, WriteOutput, or a ToolCommand that is passed to the tool unchanged. This allows users to print the custom reports or other output particular to the tool, see [17] for details.

4.2 Output to Results Transformation

The next step is to provide for an automatic conversion of the output into the table and chart results. We considered 2 options related to how those tables and charts would be expressed:

1. To use a "standard" xsd schema for spreadsheets for the results specification and transform the output into the xml format that follows such a schema.
2. To develop a transformation specification from output into the standard elements of a spreadsheet: rows, columns, and charts and transform the output into xls or LaTex format.

We explored the first option and discovered that some spreadsheet tools, such as OpenOffice, do not yet support xml import and export. Additionally, the "standard" schema does not include chart specifications. The transformation would also need to be specified, so option 1 would require two schemas. The second option allows us to create a user friendly specification of the tables and charts using familiar notation, e.g., numeric rows and alphabetic columns. Java tools support the creation of a spreadsheet in xls format.

Other issues and situations we considered and handled with our specification are described below:

1. How to handle table values that are expressions of other metrics
2. How to handle tables with results that come from different tools

For the first issue we considered modifying the output specifications to specify expression values to be written directly to the output. The other option is to specify the expressions or functions to be calculated in the cell in spreadsheet format [e.g., `=SUM(A1:A3)`]. Computing the values during output is preferable for a translation to LaTex because it is a document preparation system and has no ability to compute. On the other hand, there are some difficulties in doing the calculation in the output.

For example, we tried a typical computation found in tables: displaying total demand by calculating the product of visits and service time. The visits are specified in the PMIF, so we can define a variable in the experiment for visits. The service time is an OutputVariable, so we tried defining a new local variable with the value equal to the product of these two variables. We discovered a problem: that OutputVariables in the Experiment-SE are meant to be from the *previous solution*. That is so the previous results can be used in an expression to determine what experimental step should be done next. So the current Experiment-SE combined with the Output-SE does not have the capability to produce this particular result.

We could add a new ResultVariable that references results from *this* solution. As stated earlier, we opted to separate the output from the results computation. This is easily handled in a spreadsheet tool because it receives the results from *this* solution, not the previous one.

This strategy has two consequences. The LaTex transformation becomes more complicated because the translator will have to calculate and insert the results into the appropriate table cell. The second consequence is that sometimes the spreadsheet will have extra rows or columns to hold intermediate results used in computations. This can be handled by hiding those rows or columns so they do not appear in the result table.

The second issue is how to handle the output to results transformation when the output comes from separate runs and thus separate output files, but they should both be in the same table. There are two options for handling this situation: one is to concatenate the two output files and have different results specifications that insert the results into the proper rows and columns. The second option is to create the table with the output from the first tool, then update that table for the output from additional tools. The Results-SE that we specify handles both options.

Fig. 3 shows the Results-SE schema. All the results for an experiment go into one file. The transform tool can create this file from scratch or update the file, depending on the value of the Action attribute, defined on the schema. The Output-SE has a collection of outputs for each OutputSolutionSpec (or Solve) in the Experiment-SE. So the Results-SE specifies how to process each of those, and specifies the file/s containing the output. It can specify one or more tables (in xls different tables will go into different worksheets). WriteResult specifies the type of output metric to use, the metric for that type (such as response time), and information about where to place the values in the table (row, column, etc). There is a placeholder for Chart specifications to be added in future work.

As an example, the OutputProcessing specifications for the case study of section 5 is shown in Fig. 5:

The Transform prototype has been implemented in Java, using the Document Object Model (DOM) to read and validate the xml files (Output and ResultsSpec). We have also used the Apache POI APIs for manipulating MS Excel and OpenOffice file formats using pure Java.

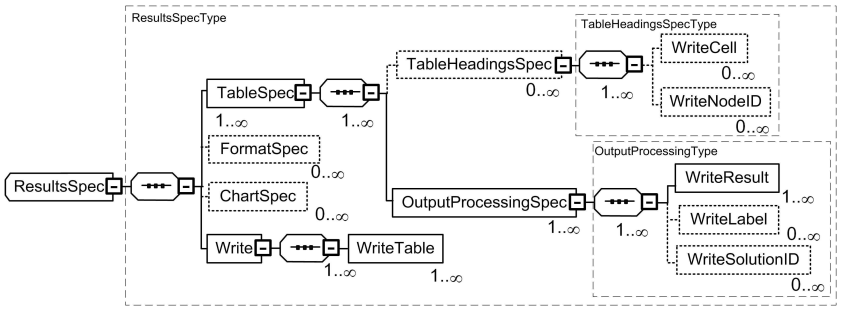


Fig. 3. Results-SE schema

5 Proof of Concept

The proof of concept uses a case study previously published and replicates it with the model interoperabilty experimental framework. It is a technical paper (Use Case) that compares a published solution to solutions derived automatically from experiment specifications.

The example was published in Jain's book [8] and subsequently used as an example of the experimental framework in [17]. It shows how to manually create a table, specify formats, enter the results from another source, then update the remaining values with the output from the experiment. The demonstration seeks to replicate the table in [17].

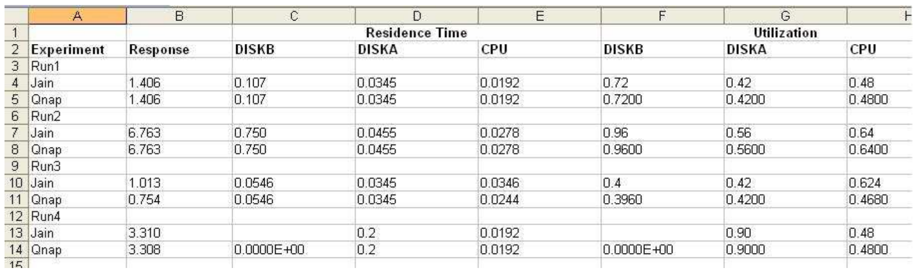
We run Qnap to produce the Output file of performance metrics specified in the experiment in [17]. We manually create an xls file with the formatting and Jain results taken from [8]. We then update the file using the results specification (an excerpt is in Fig.4) to produce the table in Fig. 5.

```
<OutputProcessingSpec RowIncrement="3"FileToProcess="Jain574.xml">
  <WriteSolutionID Format="5" Row="3" Col="1" />
  <WriteLabel Value="Qnap" Row="5" Col="1" />
  <WriteResult Type="OutputWorkload" Metric="ResponseTime" Row="5" Col="2"/>
  <WriteResult Type="OutputNodeWorkload" Metric="ResidenceTime" Row="5" Col="3"
               ColIncrement="1"/>
  <WriteResult Type="OutputNode" Metric="Utilization" Row="5" Col="6" ColIncrement="1"/>
<OutputProcessingSpec>
```

Fig. 4. Excerpt of Results Specifications

6 Conclusions

This paper has tied together previous work on performance model interchange formats and experiment specifications. It adds an output-to-results transformation to produce performance analysis results for presentation and publication.



	A	B	C	D	E	F	G	H
1				Residence Time			Utilization	
2	Experiment	Response	DISKB	DISKA	CPU	DISKB	DISKA	CPU
3	Run1							
4	Jain	1.406	0.107	0.0345	0.0192	0.72	0.42	0.48
5	Qnap	1.406	0.107	0.0345	0.0192	0.7200	0.4200	0.4800
6	Run2							
7	Jain	6.763	0.750	0.0455	0.0278	0.96	0.56	0.64
8	Qnap	6.763	0.750	0.0455	0.0278	0.9600	0.5600	0.6400
9	Run3							
10	Jain	1.013	0.0546	0.0345	0.0346	0.4	0.42	0.624
11	Qnap	0.754	0.0546	0.0345	0.0244	0.3960	0.4200	0.4680
12	Run4							
13	Jain	3.310		0.2	0.0192		0.90	0.48
14	Qnap	3.308	0.0000E+00	0.2	0.0192	0.0000E+00	0.9000	0.4800

Fig. 5. Xls file automatically produced for Jain's case study

We began by defining the requirements for the experimental framework: we identified typical Use Cases, surveyed output and results found in practice for those Use Cases, and gave an overview of our approach for satisfying those requirements. We then presented the output specification, the issues in the output-to-results transformation, and rationale for decisions made. The results specification schema was then presented followed by a description of our prototype implementation. Finally, we presented a proof of concept example to demonstrate the generation of the performance results.

Our general purpose approach was demonstrated with PMIF, however it also applies to other modeling paradigms, tools, and even measurement tools. It supports the automation of model studies from the creation of the performance model specification, the experiments to be conducted with the model, the execution of models and transformation of output to tables and charts for presentation and publication. It supports multiple Use Cases (tracking operational system performance, analyzing capacity requirements for future workload volumes, evaluating problematic systems, comparing results to measurements, and technical investigations of the model technology). It streamlines typical tasks such as exploring output and identifying results for presentation. It is a standard format that can be used by multiple tools.

Future work will develop additional templates for the most frequent results and implement additional prototypes for updating tables and creating charts. We will apply the framework to other tools, and extend it to apply to real time systems. An interesting extension might include creating rules for specifying threshold values and highlighting results in tables that exceed the threshold. We also envision the integration of Performance Trees in the interoperability framework by relating the queries to the output in order to produce results.

Acknowledgments. This work is partially funded by the TIN2006-02265 QUASIMODO project of the *Ministerio de Educacion y Ciencia*, Spain. Smith's participation is sponsored by US Air Force Contract FA8750-C-09-0086.

References

1. Microsoft office binary (doc, xls, ppt) file formats. `www.microsoft.com/interop/docs/OfficeBinaryFormats.mspx`.
2. S. Balsamo and M. Marzolla. Performance evaluation of UML software architectures with multiclass queueing network models. In *Proc. of the Fifth International Workshop of Software and Performance (WOSP)*, July 2005.
3. CMG. Computer Measurement Group. `www.cmg.org`.
4. J. Hillston. A tool to enhance model exploitation. *Performance Evaluation*, 22(1):59–74, 1995.
5. A. Hubbard. SPEX: The software performance experiment driver. Technical report, Real-time and Distributed Systems Lab, Dept. of Systems and Computer Engineering, Carleton University, Ottawa, 1997.
6. IBM. Best1 capacity planning tool. `publib.boulder.ibm.com/iseries/v5r1/ic2924/books/c4153411.pdf`.
7. Mesquite Software Inc. `www.mesquite.com`.
8. R. Jain. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling.* John Wiley, 1991.
9. Metron Tecnologiy Limited. `www.metron.co.uk`.
10. M. Melià, C.M. Lladó, C.U. Smith, and R. Puigjaner. Experimentation and output interchange for Petri net models. In *Proc. of the Seventh International Workshop on Software and Performance (WOSP)*, pages 133–138, June 2008.
11. Marc Melià, C.M. Lladó, C.U. Smith, and R. Puigjaner. An experimental framework for PIPE2. In *Proc. of the Fifth International Conference on Quantitative Evaluation of Systems*, pages 239–240, 2008.
12. Opnet. `www.opnet.com`.
13. SEAlab Software Quality Group. WEASEL, a web service for analyzing queueing networks with multiple solvers. `sealabtools.di.univaq.it/SeaLab/Weasel/`.
14. Simulog. Modline 2.0 qnap2 9.3: Reference manual, 1996.
15. 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 on Software and Performance (WOSP)*, pages 87–98, July 2005.
16. 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, September 2004.
17. 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, September 2007.
18. 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*, 1995.
19. SPE-ED. LS Computer Technology Inc. Performance Engineering Services Division. `www.spe-ed.com`.
20. T. Suto, J. T. Bradley, and W. J. Knottenbelt. Performance trees: A new approach to quantitative performance specification. In *Proc. 14th Intl. Symp. on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS06).* IEEE Computer Society, September 2006.