How to Automatically Execute Performance Models and Transform Output into Useful Results

Connie U. Smith
Performance Engineering Services
PO Box 2640 Santa Fe
New Mexico, 87504-2640 USA
www.spe-ed.com

Catalina M. Lladó, Ramon Puigjaner
Dep. de Cien. Matemàtiques i Informàtica
Universitat de les Illes Balears
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 present a standard approach to define an experiment consisting of a set of model runs and the output desired from them. We also present a mechanism for automatically transforming the tool output into useful results. A proof of concept example demonstrates the framework.

1. Introduction

The concept of performance model interoperability was first introduced in 1995 [10]. 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 [7].

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.

In each interchange format, a file specifies a model and a set of parameters for one run. Since modeling studies typically require multiple runs of the same model using different parameters (e.g., different workload mixes), this requires preparing and exchanging multiple interchange files. In addition, interchange formats do not specify the output metrics that are to be returned after model execution, typically resulting in either a default set of metrics or all possible metrics.

To address these issues, this paper presents an Experiment Schema Extension (Ex-SE) for defining a set of model runs and the output desired from them. This schema extension provides a means of specifying performance studies that is independent of a given tool paradigm. It requires only that a tool support the Ex-SE or have an interface that is capable of reading/writing extended interchange files. This schema extension was developed for use with an interchange schema (e.g., PMIF) when exchanging models among performance modeling tools. However, it may also be used in a stand-alone mode to specify studies for the tool in which the model was created. It may also be used to specify measurement as well as modeling studies.

To illustrate the use of the Ex-SE, this paper defines and uses it with the PMIF. Thus, this instance of the extension is known as PMIF-Ex.

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 software design models into performance models [2, 11]. 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 [5]. They may also be exported by one modeling tool in order to compare results to other modeling tools.

The experiment specification file is shown at the top-right of Fig. 1. These two files are combined and used as input for one or more performance modeling tools. Each tool generates the performance metric

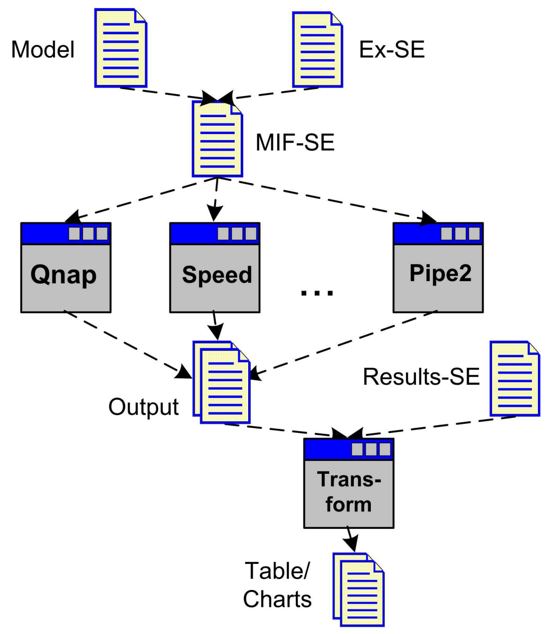


Figure 1. Model interoperability framework

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 also 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 (see [9] for its description) has recognized the need for this last step. Our work develops the concept and provides a concrete realization of it.

This paper describes the results of recent research projects and some prototype software that demonstrates the proof of concept. Hopefully we will see new tools introduced that provide these capabilities. Tool developers can use these results by incorporating support for the model interchange formats into their tools. Performance analysts can use all or part of these results with tools that provide a file interface even if they do not support the model interchange formats. A section at the end of the paper discusses how to do this.

The next section presents the experiment schema and provides examples to illustrate how to use it to express experiments. Section 3 presents the transformation of tool output into results. Section 4 discusses a prototype implementation, and Section 5 presents a proof of concept example. Section 6 discusses how to make use of this work. Summary and conclusions are in section 7.

2. PMIF-Ex

This section describes the Experiment Schema Extension for PMIF. We present the schema and provide some examples. This schema definition is included in the host schema (PMIF). An OutputFormat schema (described later) is also needed in the host schema to specify the XML format to be used for output from the experiments.

2.1. Schema

As shown in Fig. 2, the Experiment schema has two well-differentiated parts. The first part is the variable specification. This allows the user to specify different types of variables that can be used anywhere in any of the solution specifications. Variables refer to an attribute of an element in the model (for example the ArrivalRate of a specific OpenWorkload). They can be used to assign different values to that attribute, iterate over it, and so on. Expressions combining variables can be assigned to LocalVariables. Finally, an OuputVariable specifies a concrete result that will be used in the solution specification (for example, OutputVariable UCPU represents the Utilization of the node named CPU).

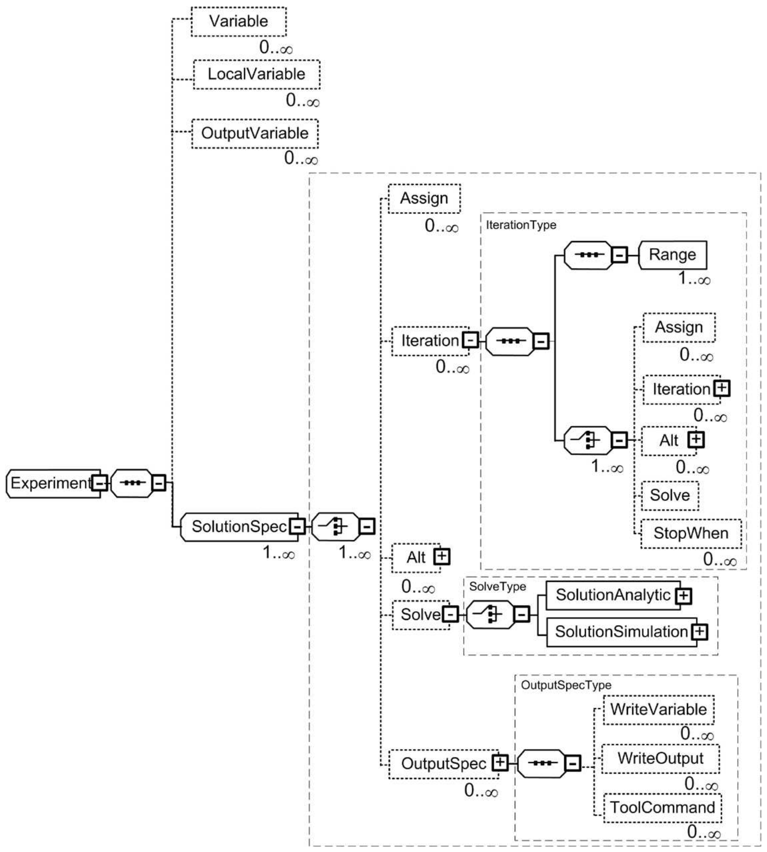


Figure 2. Experiment schema

The second part is the solution specification, which indicates the experiment and the output desired. That is, the variables to be written in the output, the results (e.g., throughput or utilization) and possibly tool specific output.

The ToolCommand element allows specification of control parameters that are not included explicitly because they depend on the tool.

The ExperimentType allows for assignments, iterations, alternations and a specific Solve command to specify the point where the model is to be solved. The Solve command also specifies the type of solution (analytical, simulation). Fig. 2 shows in detail the Iteration (but not Alternation due to lack of space). The Iteration requires at least one Range of values and one Solve statement. It can also have one or more StopWhen conditions that may stop the iteration earlier than specified by the Range(s). It can also include assignments, iterations, and alternations.

Thus, the Ex-SE allows specification of:

- Changes in parameter values from one execution of a model to the next
- Specification of control in performing model studies, including iteration and alternation
- Variables that are local to the experiment to be used in computations and output
- Model-results dependent execution
- Use of previous output as input to subsequent runs
- Specification of the output metrics to be returned
- Solution type specifications.

The schema specifies the syntactic characteristics of the Experiment. Additional semantic constraints and assumptions used in PMIF-Ex are provided at: `www.spe-ed.com/pmif/`.

2.2. PMIF-Ex Example

Fig. 3 illustrates how to express a typical experiment with PMIF-Ex. It shows an excerpt from the specification for the case study presented in [9]. This

```
<Variable AttributeToChange="NumberOfJobs" WorkloadName="Forms" Name="NForms" />
<Variable AttributeToChange="NumberOfJobs" WorkloadName="Apply" Name="NApply" />
<Variable AttributeToChange="NumberOfJobs" WorkloadName="Store" Name="NStore" />
<Variable AttributeToChange="NumberOfJobs" WorkloadName="Convert" Name="NConvert" />
<LocalVariable Name="TotTput" InitialValue="0.0" />
<OutputVariable ResultToUse="Throughput" WorkloadName="Forms" Name="TForms" InitialValue=".016" />
<OutputVariable ResultToUse="Throughput" WorkloadName="Apply" Name="TApply"  InitialValue=".005"/>
<OutputVariable ResultToUse="Utilization" ServerName="CPU" Name="UCPU" InitialValue="0" />
<OutputVariable Name="SumTFormTApp" Expression="TForms+TApply" />
<SolutionSpec>
  <Iteration>
    <Range VariableName="NForms" Start="18" End="36" Step="9" />
    <Range VariableName="NApply" Start="32" End="64" Step="16" />
    <Range VariableName="NStore" Start="50" End="100"Step="25" />
    <Range VariableName="NConvert" Start="30" End="60" Step="15" />
    <StopWhen OutputVariableName="UCPU" Test="GE" Value=".99" />
    <StopWhen OutputVariableName="Tform+TApp" Test="LE" Value="TotTput" />
    <Solve SolutionID="Run1">
       <SolutionAnalytic/>
     </Solve>
    <Assign VariableName="TotTput" Value="(TForms+TApply)" />
 </Iteration>
 <OutputSpec>
    <WriteVariable VariableName="TotTput" />
    <WriteOutput Metric="ResponseTime" />
    <WriteOutput Metric="Throughput" />
    <WriteOutput Metric="Utilization" />
 </OutputSpec>
</SolutionSpec>
```

Figure 3. PMIF-Ex example

is a specification used in an actual complex, large-scale model of a system with 4 workloads (Forms, Apply, Store, and Convert). The details of the model are relatively unimportant for this example. Using a model of an actual, complex, large-scale model, however, does show that the EX-SE can represent the types of studies that are often conducted in practice and shows that they can be conducted faster with EX-SE. This example illustrates the use of a local variable (TotTput), results testing (the StopWhens), setting multiple values in one iteration (the four Range specifications), the use of multiple StopWhen clauses, the comparison of the results of one iteration to the results of the previous iteration (throughput of TForms+TApply) and other features. The complete PMIF-Ex file for the case study is at `www.spe-ed.com/pmif/`.

3 Transforming Output to Results

First we discuss the typical situations, or *Use Cases,* for conducting modeling experiments and analyzing results. Next we identify typical output and results that are needed for those Use Cases. Then we present our approach to providing the output and results.

3.1 Requirements for Producing Results

QNM may be used in a variety of fields from computer performance evaluation to 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. Is there a typical set of output and results, or are they unique to each situation? Are they the same for the Use Cases or do they have significant differences? How are they typically presented? There has been a lot of interest lately on visualization; how is visualization of performance model results used in practice?

The Proceedings of the Computer Measurement Group are the main source of papers written by practitioners about the results of their performance modeling studies [3]. We examined a sample of papers from the Computer Measurement Group 25th anniversary edition of the proceedings (1974 through 1999) [3] - the main source of practitioner modeling papers. Research results in other publications are similar.

We found three types of results: *tables*, graphs or *charts* in spreadsheet tools, and metric values embedded in the text of the paper. Some combinations of performance metrics occur frequently; examples are: service times and response times for several workloads; and throughput, response time and CPU utilization for several workloads.

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.

This approach is simplistic. Nevertheless, the evidence shows that the majority of published papers containing performance model results use these three simple approaches. Visualization techniques are emerging, but are currently used mainly with performance measurements rather than performance model results. As additional visualization techniques for performance model results are developed they can be incorporated into the Results-SE.

The most common format for tables and charts is xls [1] as in spreadsheet tools such as Excel and OpenOffice, 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 [12].

3.2 Model Transformation Approach

This section covers our approach 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 approach for simplifying the generation of standard results, and some implementation issues.

The Output Schema Extension is in Fig. 4. The "ValueUsed" applies to *Ranges* or other variables used in the experiment specification and reports the value used for that particular solution.

The metrics that may be produced are in the OutputWorkload (overall results by workload), OutputNode (overall results by Node), and OutputNodeWorkload (results by Workload for Nodes). For more information see `www.spe-ed.com/pmif/`.

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 custom reports, visualization output, etc. particular to the tool, see [9].

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.

Some spreadsheet tools, such as OpenOffice, do not yet support xml import and export, and the "standard" schema does not include chart specifications. Option 1 would require an additional schema to specify the transformation. Thus we chose the second option because it provides a specification of tables and charts using familiar notation, e.g., numeric rows and alphabetic columns. Java tools support the creation of a spreadsheet in xls format.

Fig. 5 shows the Results-SE 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 tables go into different worksheets). WriteResult specifies the type of

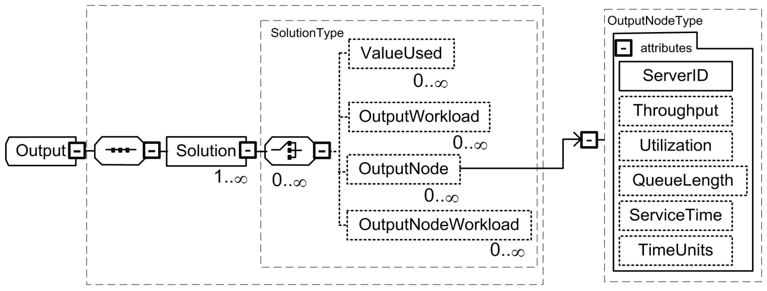


Figure 4. Output Schema Extension

output metric to use (Node, Workload, etc.), the metric (such as response time), and where to place the values in the table (row, column, etc). There is a placeholder for Chart specifications to be added in future work.

4. Implementation

A prototype interface was implemented to demonstrate the feasibility of the PMIF-Ex framework. The implementation requires a mechanism to create an experiment definition, and a mechanism to interpret the experiment, solve the models accordingly, and return the requested output. In this section, implementation alternatives for each of these are described then the interface implementation for executing experiments using Qnap citeModlineQnap is presented. In Section 5, the use of the prototype is illustrated with two case studies.

4.1. Creating the experiment specification

There are three alternatives for creating the experiment specifications:

1. It is relatively easy to use an XML editor, such as XMLSpy, to create experiments for a particular model paradigm.
2. Create a tool with a GUI for describing experiments and generating the XML.
3. Tools that have an experimentation capability could export their experiment definition along with the model interchange format.

We used the first alternative for this proof of concept.

4.2. Executing the experiment

There are two alternatives for interpreting the experiment specifications, solving the models accordingly, and returning the requested output. The choice depends on whether or not the target tool provides its own capability for experimentation.

Tools without experimentation need an Experimenter tool to interpret the experiment, invoke the tool for each `<Solve>`, and return the output. This can be a general tool that can work with multiple solvers.

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. We used this alternative for the proof of concept because Qnap provides direct support for experimentation. Future work will address a general purpose Experimenter tool.

4.3. Prototype implementation

The prototype implementation is based on PMIF and Qnap, a queueing network-based modeling tool. Qnap can be used on its own or accessed via Modline [6]. which provides a graphical, user-friendly interface for model definition and interactive visualization of the results.

The prototype uses Qnap without modification and does not make use of the Modline interface (Qnap2 v.9.3 [6]). Qnap reads the input (QNM and experiment specification) from a file and writes the results to another file that also includes the input.

Ultimately, it would be best for Qnap to have an interface that would read from its standard file OR the pmif.xml file. However, we did not have access to the Qnap source code and we could not implement such an interface directly. Therefore, we translated the pmif.xml file into a file in Qnap's format to demonstrate the proof of concept. We were initially disappointed with this limitation, but later realized that this confirms that this approach can be used with any modeling tools that provide a file interface and do not require any changes to the tool itself. Thus this is a broadly

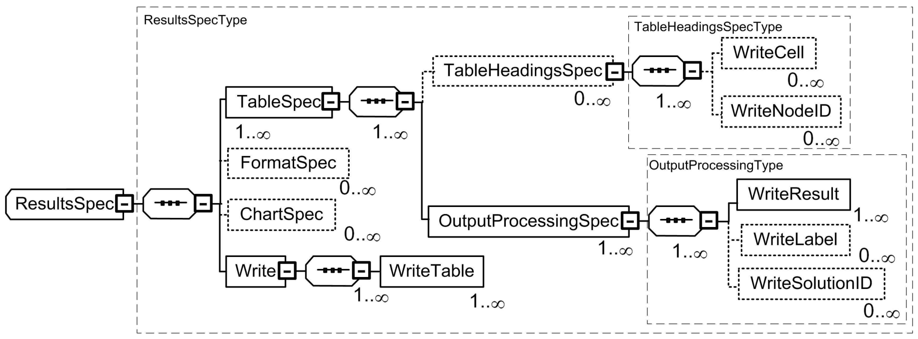


Figure 5. Results-SE schema

applicable approach that can be used with most modeling and even measurement tools.

The model and experiment translation from a pmif.xml file into a Qnap input file is done using XSLT. We generate a specific XSLT file that transforms a pmif-ex.xml file into a file that can be directly read and executed by Qnap. Additionally, Qnap allows the programming of any input/output operations and so the generation of an XML output file is possible. Some of the detailed issues in the Experiment Schema transformation are as follows:

1. Qnap variables can only be 8 characters, so a routine does this check, truncates the identifier if necessary and generates unique names if it happens to be a duplicate.
2. Iterations are implemented as While statements that finish when any of the StopWhen conditions occur or when any of the Ranges reaches its End value.
3. Alternations are implemented as If statements.
4. Qnap gives a default output that is written to the standard output (typically the screen or a file). The necessary instructions for Qnap to create another file and write the specified results on the pmif-ex.xml in XML format are also generated by the transformation.

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.

5. Proof of concept

To validate the prototype, we selected a published experiment that provides sufficient data on the model, the output, and the experiment for reproducibility. The experiment is published in [4]. It shows 4 runs of a model presented on page 574-5. The first run is the original model. It is an open model with one CPU and two disks. The second run increases the workload arrival rate. The third explores the advantage of caching by increasing service times, and reducing the demand on one of the disks. The last run uses a lower cost server with only one disk. An excerpt of the experiment specification is in Fig. 5.

This example also 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.

We run the experiment using Qnap to produce the Output file of performance metrics specified in the experiment (the complete experiment specification is at `www.spe-ed.com/pmif/`). We manually create an xls file with the formatting and Jain results taken from his book [4]. We then update the file using the results specification (an excerpt is in Fig. 5) to produce the table in Fig. 8.

Table 8 shows that the results reported in the book match the results derived from the Prototype interchange of the PMIF-Ex model for Runs 1,2, and 4. The results for Run 3, however, differ due to an error in the book. It incorrectly uses 16 CPU visits in calculating CPU demand, rather than the correct value of 12 visits. We only noticed this error after the models produced different results. Even though this is a simple experiment, it demonstrates the successful use of the PMIF-Ex model interchange and automatic generation of xls results.

This example also shows the value of having an automated interface for executing multiple experiments. Automated experiments substantially reduce the time required to run models when tools do not provide an internal experimentation capability. The proof of concept also shows the value of being able to compare

```
<SolutionSpec>
 <SolutionAnalytic/>
 <Solve SolutionID="Run1" />
 <Assign VariableName="C1Arrival" Value="4" />
 <Solve SolutionID="Run2" />
 <Assign VariableName="C1Arrival" Value="3" />
 <Assign VariableName="CPUServiceTime" Value="0.013" />
 <Assign VariableName="DiskBServiceTime" Value="0.033" />
 <Assign VariableName="DiskBVisits" Value="4" />
 <Assign VariableName="DiskAProb" Value="0.58333" />
 <Assign VariableName="DiskBProb" Value="0.33333" />
 <Assign VariableName="OutProb" Value="0.08333" />
 <Solve SolutionID="Run3" />
 <Assign VariableName="CPUServiceTime" Value="0.01" />
 <Assign VariableName="DiskBVisits" Value="0" />
 <Assign VariableName="DiskAVisits" Value="15" />
 <Assign VariableName="DiskAProb" Value="0.9375" />
 <Assign VariableName="DiskBProb" Value="0.00" />
 <Assign VariableName="OutProb" Value="0.0625" />
 <Solve SolutionID="Run4" />
 <OutputSpec>
   <WriteVariable VariableName="C1Arrival" />
   <WriteOutput Metric="ResponseTime" />
   <WriteOutput Metric="Throughput" />
   <WriteOutput Metric="Utilization" />
   <WriteOutput Metric="ResidenceTime" />
 </OutputSpec>
</SolutionSpec>
```

Figure 6. Excerpt of Experiment Specification

```
<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>
```

Figure 7. Excerpt of Results Specifications

results from different tools using PMIF-Ex. With it the models solved are identical - there are no errors due to manual re-entry of the model into a different tool.

6. How to Make Use of These Results

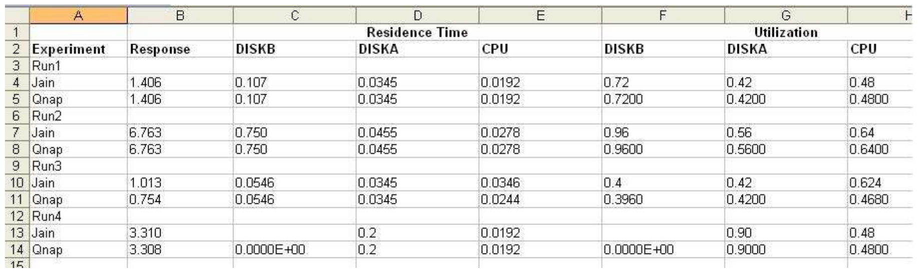
If your performance modeling tools do not yet support the model interchange formats, you can still use all or part of these results to automate your modeling studies. If your tool(s) provide a file interface, you can use XSLT or custom code to convert a PMIF specification to the file format. An example of this conversion is in [8]. If your tool provides the ability to define experiments, convert the experiment definition into the tool's format. If it does not, you an create a tool to interpret the experiment specification, change the model input file accordingly, solve the model, then examine the output when necessary to determine the next action.

We anticipate that tools will be available soon to convert model solutions from the Output-SE format to results in xls format. So, you can also use XSLT or custom code to convert your modeling tool output into the Output-SE format and use those tools to automatically produce reports and presentations. You can also convert results from measurement tools into this format so tables can compare model results to measurements.

Note that it only requires one input and output translator per tool, so if you develop this capability, please share the tools with others.

7. Summary and Conclusions

This paper has described an Experimental Schema Extension (Ex-SE) for defining performance modeling experiments. The schema allows specification of multiple model runs along with the output that is desired from them. This schema extension provides a means of specifying performance studies that is independent of a given tool paradigm. It requires only that a tool support the Ex-SE or have an interface that is capa-



	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
15								

Figure 8. Xls file automatically produced for Jain's case study

ble of reading/writing extended interchange files. The Ex-SE allows specification of:

- Changes in parameter values from one execution of a model to the next
- Specification of control in performing model studies, including iteration and alternation
- Variables that are local to the experiment to be used in computations and output
- Model-results dependent execution
- Use of previous output as input to subsequent runs
- Specification of the output metrics to be returned
- Solution type specifications

The Ex-SE was developed for use with an interchange schema, such as the Performance Model Interchange Format (PMIF), for exchanging models between queueing network based modeling tools. However, it may also be used in a stand-alone mode to specify studies for the tool in which the model was created. It may also be used to specify measurement as well as modeling studies.

This paper has also presented the output-to-results transformation to produce performance analysis results for presentation and publication. We defined the requirements for the transformation by identifing typical Use Cases, surveying output and results found in practice for those Use Cases, and summarizing 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.

This paper has presented a particular instantiation of the Ex-SE, the PMIF-Ex and provided examples of its use. To demonstrate that the concept is feasible, we have also described a prototype implementation of the PMIF-Ex and Qnap. 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. The validity of the prototype was demonstrated with a case study. Future work will also extend the PMIF to include features supported by simulation solvers and incorporate these capabilities into a tool framework.

The PMIF-Ex XML schema is available at `www.spe-ed.com/pmif/`.

Acknowledgment

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 FA8750-09-C-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] R. Jain. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. John Wiley, 1991.

[5] SEAlab Software Quality Group. WEASEL, a web service for analyzing queueing networks with multiple solvers. `sealabtools.di.univaq.it/SeaLab/Weasel/`.

[6] Simulog. *MODLINE 2.0 QNAP2 9.3: Reference Manual*, 1996.

[7] C. Smith and C. 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.

[8] C. Smith and C. Lladó. Performance model interchange format (PMIF 2.0): XML definition and implementation. tecnical report. Technical report, Performance Engineering Services, 2007.

[9] C. Smith, P. R. Lladó, C.M., and L. 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.

[10] C. Smith and L. Williams. Panel presentation: A performance model interchange format. In *Proc. of the International Conference on Modeling Techniques and Tools for Computer Performance Evaluation*, 1995.

[11] 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.

[12] 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.