Performance Model Interchange Format: Semantic Validation

Daniel García, Catalina M. Lladó
Dep. de Cien. Mat. i Inf.
Universitat de les Illes Balears
07071, Palma de Mallorca, Spain.
danielgcou@gmail.com, cllado@uib.es

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

Ramon Puigjaner
Dep. de Cien. Mat. i Inf.
Universitat de les Illes Balears
07071, Palma de Mallorca, Spain.
putxi@uib.es.

***Abstract*— A Performance Model Interchange Format (PMIF) provides a mechanism whereby system model information may be transferred among queueing network model (QNM) based modeling tools. The PMIF allows diverse tools to exchange information and requires only that those tools provide importing/exporting mechanisms from/to the PMIF. The XML specification of the PMIF allows implementers to use widely available tools to parse the XML file, check the syntax, and simplify the translation to/from the XML format. Those tools, however, do not know the semantics of a QNM so they cannot check the XML to ensure that it contains a valid QNM. This paper presents the study of the validations needed to carry out such a semantic analysis, and the development of a semantic validation tool that can be used by any developer who wants to implement PMIF import/export mechanisms.**

Keywords. Interchange Format, Performance Models, Queueing Network Models, SPE, Semantic validation, Tool interoperability, XML.

I. Introduction

A Performance Model Interchange Format (PMIF) is a common representation for system performance model data that can be used to move models among modeling tools. The PMIF allows diverse tools to exchange information and 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.

Previous work defined a PMIF meta-model for system performance models that use a queueing network model (QNM) paradigm and an XML (Extensible Markup Language) [1] schema for the meta-model. A proof of concept was provided with a prototype implementation in which the QNM export was implemented in the *SPE·ED* [2] tool and the importing tool was Qnap [3]. A set of examples was provided for validation, see [4].

The XML specification of the PMIF allows implementers to use widely available tools to parse the XML file, check the syntax, simplify the translation to/from the XML format, and other common tasks. Those tools, however, do not know the semantics of a QNM so they cannot check the XML to ensure that it contains a valid QNM.

For example, standard XML tools can check to confirm that arcs in the QNM model are connected to elements that have been specified in the model, but they cannot determine that those elements are actually Nodes (they could be Workloads). Moreover, they cannot determine other conditions that a QNM must have such as the existence of a valid path from a source node to a sink node.

It is best to implement the semantic analysis of a PMIF model in one tool so that importing tools do not have to duplicate the error checking. This makes it easier to implement a PMIF import function. It is also helpful for testing a PMIF export function to confirm that it generates proper models. Moreover, the semantic analysis can be offered as a Web service so that it can be developed, installed, and maintained once for all users who wish to use this capability.

Related work is covered in the next section. Then the remainder of the paper is organized as follows: in order to make the paper self-contained the PMIF XML schema is described in Section III. The types of semantic validation are described in Section IV. Section V describes the design and implementation of a tool for performing these semantic validations. Section VI presents a case study and Section VII offers some conclusions.

II. Related Work

Related work specifically on semantic validation of QNM has not been reported. Before PMIF, modelling tools either implemented error checking internally, or in some cases, just prevented the error with their input language or Graphical User Interface (GUI). For example, a GUI could prevent a user from drawing a QNM model that does not have a path from the source node to the sink node. Some command line tools that use input/output files, as for example Qnap, produce error messages when the models are not semantically correct, for example if a class arrives to a node which does not have a corresponding service specified for this class.

There has been some work on semantic validation of XML in general. The three primary approaches for specifying and checking semantic properties are: domain-specific custom programs, XSLT stylesheets and constraint specification languages.

This paper reports the semantic conditions that must hold for a PMIF to be valid. It also defines the order for checking the conditions. Both of these contributions are needed regardless of the implementation technology used. We have chosen to use

custom programming code to implement the proof of concept because:

- it is easier for us to check conditions that must hold in the model topology (such as open workloads must have a valid path from a source node to a sink node) with programming logic
- it is easier for us to debug the code than the complex constraints that would be required
- there currently is no ”standard” constraint language
- we do not have a commercial XML product for processing constraints
- we have defined some conditions to generate warnings rather than errors so we would need a constraint-checking tool that allows that possibility.

It is possible to use a constraint-based tool or an XSLT stylesheet for checking the ”easy” conditions. However, it does not seem worthwhile in this case because we construct internal data structures for the complex tests anyway, so we might as well use them to test the ”easy” conditions. This avoids the complexity of using multiple tools with tests divided between them.

III. PMIF Overview

PMIF was first defined using an EIA/CDIF (Electronic Industries Association/CASE Data Interchange Format) paradigm that calls for defining the information requirements for a QNM with a meta-model [5], that is, a model of the information that goes into constructing a QNM. A transfer format was then created from the meta-model and used to exchange information. The PMIF allows diverse tools to exchange information and 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.

A new version of the PMIF meta-model and its XML schema specification (PMIF 2.0) was presented in [4], [6]. In order to comprehend this paper, it is necessary to understand how the different elements of a QNM and their relationships are specified in the PMIF. Therefore, the PMIF meta-model is shown in Fig. 1.

The diagram shows that a *QueueingNetworkModel* is composed of one or more *Nodes*, and one or more *Workloads*. A *Server* provides service for one or more *Workloads*. A *Workload* represents a collection of transactions or jobs that make similar *ServiceRequests* from *Servers*. There are two types of *Workloads*: *OpenWorkload* and *ClosedWorkload*.

A *ServiceRequest* specifies the average *TimeService*, *DemandService* or *WorkUnitService* for each *Workload* that visits the *Server*. A *TimeServiceRequest* specifies the average service time and number of visits. A *DemandServiceRequest* specifies the average service demand (service time x number of visits). A *WorkUnitServiceRequest* specifies the average number of visits requested by each *Workload* that visits a *WorkUnitServer*. Upon completion of the *ServiceRequest*, the *Workload Transits* to other *Nodes* with a specified probability.

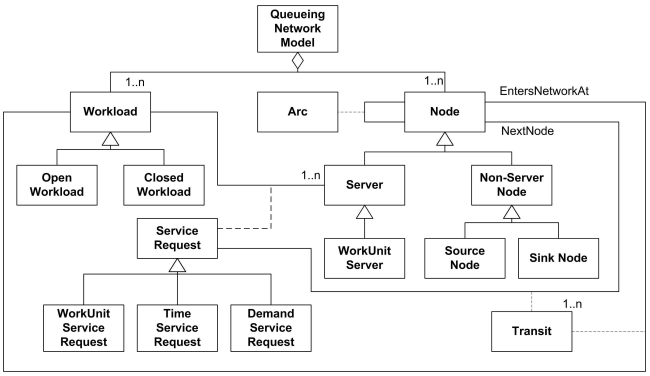

Fig. 1. XML schema for PMIF 2.0

TABLE I
PMIF elements and its attributes

Element	Sub-element	Attributes
Node	SourceNode	Name
	SinkNode	Name
	Server	Name, Quantity, SchedulingPolicy
	WorkUnitServer	Name, Quantity, SchedulingPolicy TimeUnits, ServiceTime
Arc	None	FromNode, ToNode, Description
Workload	OpenWorkload	WorkloadName, ArrivalRate TimeUnits, ArrivesAt DepartsAt
	ClosedWorkload	WorkloadName, TimeUnits NumberOfJobs, ThinkTime ThinkDevice
SR	DemandSR	WorkloadName, ServerID, TimeUnits ServiceDemand, NumberOfVisits
	TimeSR	WorkloadName, ServerID, TimeUnits ServiceTime, NumberOfVisits
	WorkUnitSR	WorkloadName, ServerID NumberOfVisits
	Transit	To, Probability

The attributes of the PMIF elements are shown in Table I[1].

More detailed information and the PMIF schema specification can be found in [4], [6]. The schema itself is at www.perfeng.com/pmif/pmifschema.xsd.

IV. PMIF Semantic Validation

The PMIF XML schema enables PMIF users to syntactically validate their models. However, the models might be semantically incorrect. In this section the semantic validations we consider necessary are described. The validations are grouped into three categories as follows:

1) Error generation: some conditions cause the PMIF model to be incoherent. In fact, most queueing network modelling tools (for example QNAP [7]) would generate an error if these conditions are found. Therefore, this group of semantic validations generates an error because the model cannot be solved as specified.
2) Warnings: other situations may occur in which, even though the model is, in general, still correct it might not actually produce the intended model, or it might be

[1]the Transit sub-element shown is a sub-element of all Workload sub-elements and all ServiceRequest sub-elements. SR stands for ServiceRequest

specified in a way that can cause confusion. Two levels of warnings have been defined. Some validations will produce a significant warning while others will generate a non-significant one, depending on the importance of the problem.

3) Excluded: other validations are described, which were also considered but have not been included in the development of the tool. The reason for their exclusion is also explained.

A. Coherent Identifiers

In XML schemas the ID attribute is used to specify a unique identifer, and the attribute IDREF is a reference to an ID. In the PMIF schema, two groups of entities have an ID attribute, Nodes and Workloads and the IDREF's are used to relate those to other entities. The XML schema (syntactic) validation checks that attributes of type IDREF have a value that has been specified as an ID. However, such an ID might be incorrectly used in the sense that every IDREF attribute of elements (or sub-elements) has some semantic restrictions on which elements or sub-elements may be referenced. For example, the sub-element OpenWorkload has an attribute of type IDREF which is ArrivesAt. This IDREF needs to be a reference to an element of type Node (it cannot be of type Workload) and moreover it needs to be of type SourceNode (sub-element of Node).

The valid groups and subgroups of identifiers for the attributes of the different elements and sub-elements can be found in [8]. This paper omits these details due to lack of space. The reference shows, for example, that the attribute To of a Transit element must reference a Node. Additionally, the Transit element is a sub-element of type Workload and of ServiceRequest, and for both of them it is always related to a specific Workload. Depending on whether this workload is open or closed the To attribute of the Transit element can only reference some sub-elements of Node. The allowable references are also shown in [8].

All the validations described in this section belong to the first group, so when problems are found, an error will be generated.

B. Elements Specified but not Referenced

Elements of type Node and Workload do not play a roll in a model unless they are referenced in another element. More specifically, Workload, Server (unless it is a ThinkDevice) and WorkUnitServer elements need to be referenced in some ServiceRequest element. SourceNode and SinkNode elements need to be referenced in an OpenWorkload element as ArrivesAt and DepartsAt attributes respectively. Otherwise a significant warning will be generated because the model may not be the one intended.

C. Duplicates

The ID attribute of Node and Workload elements makes sure that those are unique. However, ServiceRequest and Transit elements might be specified more than once and the PMIF XML file would still be valid against the PMIF schema. Therefore the semantic validation needs to do the following:

- For each couple Node and Workload only one ServiceRequest can be specified.
- For each Workload element only one Transit element can be specified with the same value for the attribute To (i.e., transiting to the same node).
- For each ServiceRequest element only one Transit element can be specified with the same value for the attribute To (i.e., transiting to the same node).

An error is generated if these conditions do not hold because the intended behavior is ambiguous.

D. Multiple Servers

The Quantity attribute specifies the number of parallel servers that a Node of type Server or WorkUnitServer has. The PMIF schema specification allows this value to be zero. Hence, the semantic validation needs to verify that a node with Quantity equals zero is not associated to any Workload in a ServiceRequest. Otherwise, an error will be generated.

On the other hand, if the attribute Quantity of an element of type Node is equal to zero but this element is not associated to any Workload in a ServiceRequest a significant warning will be produced because the model can be solved but may not be the intended model.

E. Think Device

When a Node is referenced as ThinkDevice in a ClosedWorkload element, this server cannot be referenced by any other ServiceRequest element. Moreover, such a Node needs to have a SchedulingPolicy that is IS (Infinite Server). A violation of these conditions results in an error.

F. Coherent Workload Chains

As described in Section III a Workload can be a ClosedWorkload or an OpenWorkload. The XML schema validation makes sure that the attributes specified for each Workload sub-element are the appropriate ones for that sub-element. However, the workloads specified as closed might not really form a closed chain and the ones specified as open workloads might not be really open.

Firstly, each element containing elements of type Transit (i.e. ServiceRequest or Workload elements), needs to have the sum of the probabilities of its Transits greater than zero, otherwise, an error will be generated. We are not assuming that each of the probabilities needs to be different from zero due to the following: A queueing network modelling tool that internally represents the routing probabilities as a matrix would, in general, have many of the positions of the matrix with value 0. If such a tool automatically generates a PMIF XML file, it would probably generate many Transit elements with probability zero. Therefore, this case would only give a significant warning.

Secondly, if a workload chain arrives to a specific Node, this Node needs to be referenced by a ServiceRequest that relates it to the specific Workload. Otherwise an error will be

generated. On the other hand, the reverse case, i.e. if there exists a ServiceRequest that relates a Workload and a Server but the Workload chain never gets to that Server, a significant warning is produced.

The second part of this validation is different for open and closed workloads, but both result in an error when the following conditions do not hold. For the OpenWorkload case:

- There exists a path from the SourceNode specified by the attribute ArrivesAt to the SinkNode specified by the attribute DepartsAt, which includes at least one intermediate node of type Server or WorkUnitserver.
- A SinkNode can only be used in a workload chain if it is referenced by the attribute DepartsAt of the OpenWorkload specification.

Finally, the validation for ClosedWorkloads implies the following:

- There exists a loop going from the ThinkDevice node and back to it which includes at least one intermediate node of type Server or WorkUnitserver.

G. Time Units

TimeUnits is an optional attribute that appears in many elements. Since it is optional, a model will be syntactically valid when this attribute appears in some elements but not in all of the elements. If none of the elements have TimeUnits specified it implies that they all have the same time unit whatever that is (some modelling tools, for example Qnap, use this paradigm). On the other hand, when this attribute is specified for some elements and not specified for others, a non-significant warning is given with a message that the default value for time units is seconds. Moreover, we also consider it useful to give a non significant warning if different values for the attribute TimeUnits are used throughout the model because it might not be correct.

H. Attribute Values Equals Zero

The PMIF schema is defined allowing many attributes to have a value of zero which can be useful for some specific cases, such as temporarily ”disabling” a server to investigate the impact. However, in general, models will not have a zero value for the following attributes: ServiceTime for WorkUnitServer, ArrivalRate for OpenWorkload, NumberofJobs for ClosedWorkload, NumberofVisits for ServiceRequest, ServiceTime for TimeServiceRequest, ServiceDemand for DemandServiceRequest. A significant warning will be given if any of these attributes have a zero value since the model might not be the one intended.

I. Routing Probability Equations (and Number of Visits)

In the PMIF XML schema routing probabilities are required and number of visits are optional since the number of visits can always be calculated from the routing probabilities as shown in [4]. Many tools use visits, others use probabilities, so both are included to make the PMIF easier to import. Therefore when the number of visits are also specified in a model, their values need to be consistent with the routing probability values using the well-known routing probability equations (see for example [9]). An error is produced otherwise (referenced as *b* in Fig.2).

Additionally, since the number of visits is an optional attribute, it can happen that for the same Workload, in some ServiceRequest this attribute is filled and in some others it is not. A significant warning will be given in that case (referenced as *a* in Fig.2).

J. First Come First Served Servers

In a model with multiple workloads, First Come First Served (FCFS) servers need to have the same service time for all the workloads in order for that model to be solved analytically. However, such a model could be simulated. Therefore this validation will produce a significant warning.

K. Excluded Validations

Routing probabilities for each Node and Workload should add to one. However, that does not need to be validated since many tools (for example Qnap and *SPE·ED*) normalize the probabilities and therefore the values given by the user do not need to follow that rule.

When the scheduling policy of a server is of type infinite server the quantity value is not normally used since, in general, it is not meaningful. A warning could have been produced in this case but some tools (as for example *SPE·ED*) use the quantity value for special calculations. Another possible validation would be to check whether a model is trap free, that is, it does not contain a sub-chain that allows a client to get in and never get out again. We thought this would be a duty for the queueing network modelling tool to do.

V. Validation Design and Implementation

This section describes how the validations explained in the previous section will be performed. Due to their inter-related nature the validations cannot be carried out in a random order since the checking of most conditions only makes sense if other conditions have already been validated. Therefore, our validation process will have different phases that need to happen sequentially. Fig. 2 shows these phases and the validations performed in each of them (each validation is represented by the number of the subsection where that validation is described). Since one of the advantages of having the PMIF meta-model specified as an XML schema is that the PMIF files can easily be syntactically validated against that schema, the semantic validation starting point is an XML file that is syntactically valid. The validation dependencies are shown in Fig. 2, so that if there is an arrow that goes from validation *a* to validation *b* it means that validation *b* only makes sense if validation *a* has already been confirmed. Moreover, we have grouped the validations in three sets depending on whether they generate errors, significant warnings or non significant warnings (see Section IV). These groups are also distinguished in Fig. 2.

It is not possible to conduct all validations in one pass of the PMIF XML file. Therefore, intermediate data structures are

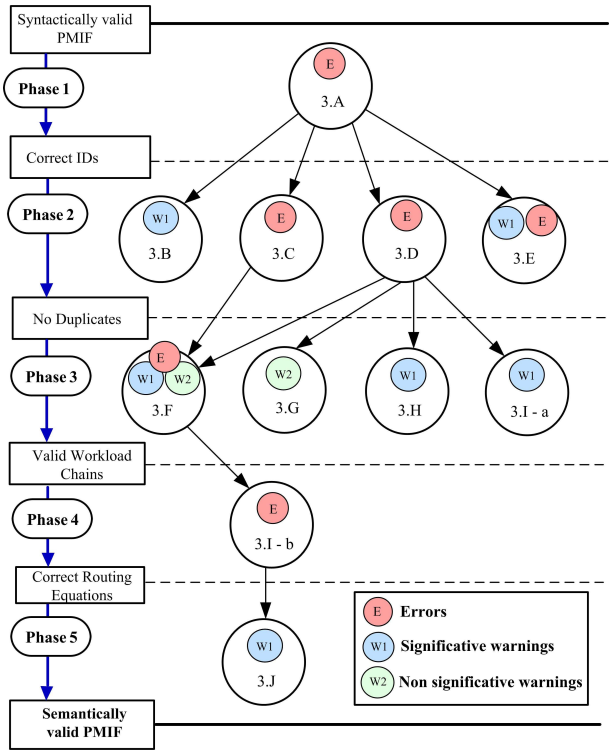

Fig. 2. Validation phases and dependencies (the number inside each circle corresponds to the subsection where the validation is described)

used to facilitate the PMIF validation. The PMIF document is parsed once and these structures are built. Similarly, the errors and warnings are stored in a second data structure so that they can be collected and inserted in the output at the appropriate location.

Thus, the functions that our PMIF validation application performs are:

1) PMIF file analysis and construction of model data structures.
2) Validation using data structure analysis and construction of error data structure.
3) Output generation with errors and warnings found.

We would like to execute the validation application in three different ways, single invocation from a command line, using a graphical user interface, and through a web service. This way the validation application is independent from the interface and the different interfaces are developed separately. The graphical user interface is shown in Fig 3.

The PMIF semantic validation tool has been implemented in Java since it is a platform independent language and it provides Application Programming Interfaces (APIs) that facilitate the manipulation of XML documents. We have used the Java API for XML Processing (JAXP) [10] and as part of it the Simple API for XML (SAX). The Xerxes library [11] is also used with SAX to perform the syntactic validation of the XML document against the PMIF schema.

The implemented classes have been grouped into 2 packages as follows:

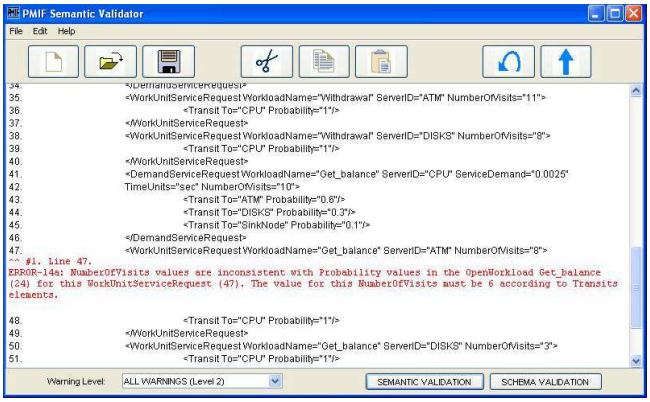

Fig. 3. Graphical user interface

- *Information*, which contains the classes used to build the memory data structures.
- *Modules*, containing the classes that carry out the tasks related to the semantic validation itself.

The details for all the classes can be found in the API documentation available at `http://dmi.uib.es/~cllado/pmif/validation`.

VI. Case Study

This section describes the validation of a case study step by step. The example used is based on the ATM model from [5]. The PMIF XML file corresponding to this model can be found at `http://dmi.uib.es/~cllado/pmif/ATM.xml`. This file has been modified so that even though it is syntactically valid, it has many semantic errors, or inconsistences. In fact, most of the semantic validations described in Section IV fail in one way or another when this example is checked. The modified ATM PMIF specification is shown in [8], where the non-valid elements (or the ones that will produce warnings) are commented with the name of the subsection in which the related validation is described.

As described in Section V, the PMIF XML file is parsed once and intermediate data structures are built.

Using these structures, the validations are carried out following the phases described in Fig. 2 in such a way that a phase only starts if the validation of the previous one has finished without errors (though it could have generated warnings), see Section V.

Given this example, the first phase does not validate, so the rest of them cannot be carried out. The output obtained is 1 error which is written in the output file as: *Line 23. ERROR-02f: In the OpenWorkload test (23), the attribute ArrivestAt must be a SourceNode.* When this error is corrected and the validation is executed again, the second phase can start and the output obtained is *2 errors and 3 warnings* (phase 2 has given errors and so phase 3 does not start). The output file contains the following errors/warnings:

- *Line 7. ERROR-06b: The Server CPU (7) has 0 servers (attribute quantity 0) but it is associated with a De-*

mandServiceRequest or a TimeServiceRequest or is a ThinkDevice of ClosedWorkload.

- *Line 12. WARNING-08c: The Server CPUNEW (12) is not referenced by any TimeServiceRequest or DemandServiceRequest or ThinkDevice of a ClosedWorkload.*
  WARNING-06a: The Server CPUNEW (12) has 0 servers (attribute quantity 0) but it is not used.
- *Line 23. WARNING-08e: The OpenWorkload test (23) is not referenced by any ServiceRequest (23).*
- *Line 45. ERROR-04a: This combination (CPU, Withdrawal) is a repetition of a previous ServiceRequest (38).*

The errors are corrected (not the warnings since they do not cause the interruption of the validation process) and the new errors/warnings found corresponding to phase 3 only, are:

- *Line 2. WARNING-09e: Different units in the TimeUnits are used. The used units are: Sec: WorkUnitServer (9) / OpenWorkload (22) / OpenWorkload (27) / OpenWorkload (31) / DemandServiceRequest (37) Ms: WorkUnitServer (8)*
- *Line 22. WARNING-11b: The OpenWorkload test (22) has a value 0.0 in its ArrivalRate attribute.*
  ERROR-13c: A ServiceRequest does not exist for CPU and test.
  ERROR-13e: A transition does not exist to the SinkNode for the OpenWorkload test.
- *Line 31. ERROR-13c: A ServiceRequest does not exist for CPU and Get-balance.*
  ERROR-13e: A transition does not exist to the SinkNode for the OpenWorkload Get-balance.
  WARNING-13f: The WorkUnitServer ATM with ServiceRequest associated with OpenWorkload Get-balance, is unreachable by this Workload.
  WARNING-13f: The WorkUnitServer DISKS with ServiceRequest associated with OpenWorkload Get-balance, is unreachable by this Workload.

Again, we correct the errors found so far and the validation results in 1 error corresponding to phase 4 execution only: *Line 47. ERROR-14a: NumberOfVisits values are inconsistent with Probability values in the OpenWorkload Withdrawal (27) for this WorkUnitServiceRequest (47). The value for this NumberOfVisits must be 8 according to Transit elements.*

When this last error is corrected, in the subsequent execution of the validation, phase 5 is also carried out and the ATM PMIF file is semantically valid since there are no other errors.

VII. Conclusion

The Performance Model Interchange Format (PMIF) allows diverse tools to exchange QNM performance models and 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. The XML specification of the PMIF allows implementers to use widely available tools to parse the XML file, check the syntax, simplify the translation to/from the XML format, and other common tasks. Those tools, however, do not know the semantics of a QNM so they cannot check the XML to ensure that it contains a valid QNM.

This work describes a companion tool that checks the PMIF XML to ensure that it contains a semantically correct QNM. This tool makes it possible for modelling tools that import the PMIF to omit these error checking features. The tool is also helpful for testing a PMIF export function to confirm that it generates proper models. It is possible to use these features via a Web service, then it is not necessary to install or maintain the tool. PMIF validation updates can thus be handled once and reused by other tools.

This approach is a general one that applies to PMIF models produced by many different types of tools, such as software modelling tools, analytic tools and simulation tools. Some validation cannot be done because things invalid in one tool may be fine in a different tool. For example, Qnap requires a separate source node for each open workload, but other tools may only need one source node. Therefore, tools may need to supplement this validation with some additional model-checking steps. A constraint language or XSLT may be a good way for tools to supplement our tests with their own custom checks

The number and type of validations are easily extensible so a tool provider could use what we have developed and upgrade as needed. Similarly, the classification of errors and warnings is also easily adapted. The tool can be downloaded from `http://dmi.uib.es/~cllado/pmif/validation/`.

Our future work includes extending the PMIF to add elements and attributes typically found in QNM that may not be solved analytically, but may be simulated. It will be easy to add the semantic validation for these extensions.

In addition, we are considering documenting the semantic conditions in the PMIF meta-model to better define the requirements for a PMIF XML file to be valid for both implementers and users of the PMIF.

References

[1] W3C, "World Wide Web Consortium," http://www.w3c.org/XML, 2003.

[2] "SPEED, Software Performance Modeling Tool," www.perfeng.com, 2006.

[3] D. Potier and M. Veran, "Qnap2: A portable environment for queueing systems modelling," in *I International Conference on Modeling Techniques and Tools for Performance Analysis*, D. Potier, Ed. North Holland, May 1985, pp. 25–63.

[4] C. Smith and C. Llado, "Performance model interchange format (PMIF 2.0): XML definition and implementation," in *Proc. of the First International Conference on the Quantitative Evaluation of Systems*, September 2004, pp. 38–47.

[5] C. Smith and L. Williams, "A performance model inter-change format," *Journal of Systems and Software*, vol. 49, no. 1, 1999.

[6] C. Smith and C. Llado, "Performance model interchange format (PMIF 2.0): XML definition and implementation. tecnical report," www.perfeng.com/paperndx.htm, LS Computer Technology, Inc., Tech. Rep., April 2004.

[7] Simulog, "Modline 2.0 qnap2 9.3: Reference manual," 1996.

[8] D. Garcia, C. Llado, C. Smith, and R. Puigjaner, "Performance model interchange format: Semantic validation," www.uib.es/ cllado/publications, Universitat de les Illes Balears, Tech. Rep., April 2006.

[9] G. Bolch and et al., *Queueing Networks and Markov Chains. Modeling and Performance Evaluation with Computer Science Applications*. Wiley Interscience, 1998.

[10] "Java API for XML processing (JAXP)," http://java.sun.com/webservices/jaxp/index.jsp, 2006.

[11] Apache, "Xerxes2 Java parser 2.7.1," http://xerces.apache.org/ xerces2-j/, 2005.