A Performance Model Interchange Format

Connie U. Smith† and Lloyd G. Williams§

†Performance Engineering Services,
PO Box 2640, Santa Fe, New Mexico, 87504-2640 USA
Telephone (505) 988-3811

§Software Engineering Research, Boulder, Colorado, USA

February, 1995

This material is based upon work supported by the National Science Foundation under award number DMI-9361824. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

1.0 Introduction

Software Performance Engineering (SPE) techniques have the potential to reduce cost and improve reliability of systems. The SPE techniques use performance models to provide data for the quantitative assessment of the performance characteristics of software systems as they are developed. With SPE, developers build performance into systems rather than (try to) add it later. SPE has evolved over the last 15 years and has been demonstrated to be effective during the development of many large systems [BELL88; FOX89; PATE91; SMIT90a; SMI94d]. Despite SPE's documented successes it still faces technical barriers that hinder its widespread use. The principal problem is the gap between software developers who need the techniques and the performance specialists who have the skill to conduct comprehensive performance engineering studies with most of today's modeling tools. Thus, extra time and effort is required to coordinate the design formulation and the design analysis. This limits the ability of developers to explore the design alternatives.

This work is part of a project to develop SPE tools to enable developers to conduct performance assessments of design alternatives. The ideal SPE tool will provide support for many SPE tasks in addition to the obvious requirement for performance modeling [SMIT91; SMIT94a]. This paper addresses the ability to use the *performance modeling* tool best suited to the software/hardware architecture issues and the life cycle stage of the assessment. The performance model solution process should be transparent to the user of the SPE tool. To accomplish this, we need a standard Performance Model Interchange Format (PMIF) to exchange information between the SPE tool and various performance modeling tools. The purpose of the PMIF in this project is to support SPE; however, the PMIF is useful to the general performance community for exchanging models among modeling tools for a variety of reasons, such as model validation, solution technique comparisons, etc.

To conduct an SPE performance analysis, software design information is used to create a *software model* for each *performance scenario* to be modeled. The software model is solved to yield an initial assessment of the performance scenario results, and to produce parameters for a *system model.* The system model quantifies response time, utilization and other metrics due to device contention among the performance scenarios that execute on the target hardware. Thus, two transformations must occur between the specification of the software design information and the solution of the performance model:

- the software design information must be transformed into a software model
- the software design information must be transformed into a system model and combined with the *solution* of the software model.

The full PMIF for the SPE tool must include mechanisms for representing both software execution models and system execution models. It must also contain mechanisms for interfacing with tools that use a variety of solution techniques, such as analytical, simulation, and parallel simulation. It must also contain a specification for a graphical representation. It must accommodate a variety of model paradigms such as execution

graphs, queueing network models, Petri net models, and (perhaps) Markov chains. It must specify not only input but performance metrics that result.

The full PMIF is beyond the scope of this initial investigation. This paper presents an initial proposal for the PMIF. It addresses a specific type of performance model: Queueing Network Models (QNM) that may be solved using exact analytic solution algorithms [JAIN90; MENA94; MOLL89]. It demonstrates the feasibility of developing a standard QNM PMIF by defining a PMIF for a manageable subset of performance models, identifies the best format for the standard representation, and identifies key issues that must be resolved such as standard terminology, how to exchange information between tools that have different modeling capabilities, and how to incorporate future extensions. We propose this initial version of the PMIF to the performance community and solicit feedback. We hope to reach consensus on the approach that will lead to a future *standard* for performance model interchange that will be supported by a variety of performance modeling tools.

There are two key issues in the development of a PMIF. The first is the choice of the appropriate representation technique to express the interchange format, and the second is the content of the PMIF. The next section reviews related research, then section 3 summarizes the representation issues and describes the approach used for the PMIF: the development of a formal queueing network model (QNM) meta-model that leads to a transfer format that serves as the PMIF. Section 4 examines the PMIF content issues. Section 5 presents an overview of the queueing network model (QNM) meta-model and the derived transfer format. Section 6 presents a case study to illustrate the feasibility of the PMIF, and Section 7 presents a summary and conclusions. The details of the PMIF are in [SMI94b].

2 Related Work

The overall mission of this project is to enhance tool support for SPE. There has been considerable related work in this area. One type of related research is directed to better SPE modeling support. Examples include [BEIL88; GOET90; GRUM91; ROLI92; SMIT91; SMI94d; TURN92]. Other researchers have integrated performance analysis capabilities into CASE tools [BALD89; BUHR 89; LOR91; SHEN90]. Several researchers have demonstrated a connection between extended software specifications and performance models. Examples include [GÖTZ93; OPDA92b; VALD92].

The specific subject of this paper is a standard model interchange format for queueing network models. Related research in this area is limited. Beilner advocates hierarchy and modular descriptions of models and shows how multiple solution techniques can be used with these model descriptions. The HIT environment demonstrates the feasibility of this approach [BEIL90]. Likewise, the Espirit Integrated Modelling Support Environment, IMSE, integrates several individual modeling tools and demonstrates the potential of interchanging tools for performance studies. The integration is accomplished through the Object Management System [HILL92].

Similar standards and interchange formats have been developed in other related fields. Work in this area is reviewed in Section 3.2 where we use these other domains as models for this work.

3 PMIF Representation Issues

First we review the requirements for a suitable representation technique and examine some viable candidates. Then we describe the approach selected for this project.

3.1 PMIF Requirements

The following are key requirements for an appropriate representation technique for a PMIF:

- Expressive power - the format must be capable of expressing a wide range of models:
  - from a small number of servers to very large numbers of servers
  - from one to many workloads
  - both open and closed models
  - solved using either analytic or simulation solution techniques.
- Extendibility - we wish to initially define a format for a subset of QNMs that may be solved using efficient, exact analytic techniques, then add extensions to cover additional facets of QNMs.
- Compatibility with existing tools and theory - it must be easy for a tool to support the format regardless of whether the tool supports a stochastic modeling approach or an operational analysis approach.
- Visual QNM representation - in addition the QNM details required to solve the model, we wish to exchange a picture of the model among tools that have a graphical user interface or a visual representation of the model.
- Model results - we wish to exchange both a model description and the results derived from the modeling tool.
- Ease of translation - it must be easy to generate the format for a model, and easy to translate the PMIF into an internal representation of a model.
- Tool support - we prefer a format that lends itself to a standard lexical analyzer and parser that could be used by all tools that wish to support the PMIF.

3.2 Formats Considered

Several different techniques for representing the PMIF were considered. These include: the Electronic Design Interchange Format, the EIA/CDIF approach, a BNF grammar, and other approaches. The relative merits of these are discussed briefly below.

3.2.1 Electronic Design Interchange Format (EDIF)

EDIF is a standard interchange format for exchanging information about VLSI designs among tools that support the VLSI design process [CRAW84; EDIF]. It was developed by a committee of experts in the VLSI design field and has been successfully used for over 10 years. It supports a wide range of designs, and uses the concept of *levels* for extendibility - level 0 is supported by all tools that support the standard, each higher level adds features for different facets of the design analysis. The layout is an essential part of VLSI design. Therefore, EDIF has an integral notation convention for a visual representation of designs as well as other design details. It uses a LISP format, so it is easy to generate and interpret using LISP-based tools. A standard scanner and parser

are generally available. The EDIF standard did not address the exchange of analysis results.

The concepts embodied in EDIF are appropriate for a PMIF. They have also been adopted as a standard for a CASE data interchange format (CDIF) for exchange of information about software designs among CASE tools. CDIF is more closely related to performance modeling than VLSI design. Therefore, CDIF is also considered as a basis for PMIF in the next section.

3.2.2 EIA/CDIF

This approach is described in the draft EIA/CDIF (Electronic Industries Association/CASE Data Interchange Format) standard [EIA94]. CDIF is actually a family of standards that describe a mechanism for transferring information between CASE tools. The standards define a transfer format that allows tools that have different internal databases and storage formats to exchange information. An exchange takes place via a file and internal tool information is translated to and from the file's transfer format.

In the CDIF standard, the information to be transferred between two tools is known as a *model*. The contents of a model are defined using a *meta-model*. A meta-model defines the information structure of a small area of CASE (such as data modeling or data-flow diagrams) known as a "Subject Area." Each meta-model is, in turn, defined using a *meta-meta-model*.[1] The meta-meta-model is based on the Entity-Relationship-Attribute (ERA) approach.

The CDIF meta-meta-model can be used to define a QNM meta-model. The QNM meta-model could define a CDIF "Subject Area." A CDIF Transfer Format could then be used to define a standard format for transferring QNM information between tools.

The transfer format is analogous to the EDIF LISP-based interchange format. EDIF started with the definition of the transfer format; CDIF starts with a meta-model that formally defines the information to be transferred, then a straightforward derivation translates the meta-model into the transfer format.

CDIF supports a wide range of designs and design notations through the subject areas. Extendibility is handled by defining new, named meta-models and transferring the meta-model definition along with the model transfer format[2]. The graphical notation used for software design depends on the design method chosen. Therefore, CDIF defines graphical notations in a separate subject area. With the CDIF approach, the interchange of the graphical information for a QNM would use two different meta-

---

1 The terminology surrounding multiple layers of models, such as those used in the CDIF standard, can be confusing to the uninitiated. A *model* contains some information such as a QNM model parameters. A *meta-model* is a model of the information contained in a model; i.e., it models the model. A *meta-meta-model* is a model of the information contained in a meta-model.

2 For example, if one wishes to add passive servers to the QNM meta-model, the transfer format would include an extension section that defines the passive server entity, its attributes, and relationships to other entities, then include the data for passive servers in the model that is described with the transfer format.

models one for the model parameters and another for the model picture. The CDIF transfer format uses a LISP format, so it is easy to generate and interpret using LISP-based tools. There is currently no standard scanner and parser that is generally available. CDIF does not address the exchange of analysis results.

3.2.3 Other Representation Formats

Other options for describing an interchange format include:

- BNF grammar - a formal notation for defining a QNM language
- Tool-specific language - select a representative performance modeling tool, adopt its specification language as the standard, then add extensions as necessary
- Information Processing Graphs (IPG) - a graphical notation used to define the information in a queueing network model [BROW85; SMIT90a].

The first two options are language-based formats, the third is a graphical-based format. All of these options could support a wide range of models; however, extendibility is a serious limitation. For example, how does one add the definition of a server icon to a BNF grammar or to a language-based specification? The addition of a passive server may require extensive changes to a BNF grammar. Similarly, how does one transfer the icon for an IPG server in a file format? A tool-specific language would require a compromise on its QNM terminology. The code that might be required for generating, scanning, and parsing the tool language could be extensive. What if the original tool changes or extends the language – would the standard automatically change? None of the options explicitly addresses the exchange of analysis results.

3.3 Approach Selected

A related part of this project used the EIA/CDIF approach to define an SPE meta-model that formally defines the software design information required for conducting an SPE study [EIA94]. The reasons for selecting the EIA/CDIF approach are described in [WILL95]. Because of the close relationship among the design information, the software model, and the system model, we use the same approach to define one type of system model: a *queueing network model* (QNM). The approach creates a QNM meta-model defined in the same notation as the SPE meta-model: the EIA/CDIF standard supplemented with a subset of the OMT Object Model Notation [RUMB91] to graphically document the QNM meta-model. The QNM meta-model is then used to define the *transfer format* that enables the exchange of information specified in the meta-model between tools that support the format.

Thus the extended EIA/CDIF standard is an appropriate representation technique to express the interchange format. The next sections address the content of the PMIF.

4 PMIF Contents

The PMIF must be capable of expressing a wide range of models:

- those containing a small number of servers to very large numbers of servers

- from one to many workloads
- both open and closed models
- that may be solved using either analytic or simulation solution techniques.

The PMIF must also be useful with existing tools. It must include modeling features that tools provide, support the modeling paradigms prevelant in today's tools, and use terminology common in tools and modeling research. The next section examines the features supported by a representative set of performance modeling tools. The following section reviews the terminology used by these tools and current textbooks on performance modeling to select an initial set of names that adequately defines the QNM information to be exchanged.

4.1 Representative Performance Modeling Tools

We first created a taxonomy of representative performance modeling tools to ensure that the QNM meta-model adequately describes their performance model information requirements. Table 1 shows the representative tools and features that influence PMIF contents.

Table 1. Performance Modeling Tool Features

Product	Model type	Solution type	Interface	Service request	Time units	Routing	QD	Domain
BEST/1	System	Analytic	Forms	Computed Demand	Device dep.	Demand	Standard Priority	Yes
CSIM	Custom	Simulation.	C	Custom	Implicit	Custom	Many	Custom
MAP	System	Analytic	Language	Demand	Implicit	Demand	Standard Priority	Yes
QASE	Both	Simulation	GUI	Computed Time	Menu choice	Computed	FCFS Priority	No
QNAP2	System	Hybrid	Language	Time	Implicit	Probability	Many	Custom
QSolver/1	System	Analytic	Spread-sheet	Demand	Implicit	Demand	Standard Priority	Yes
SES	System Custom	Simulation	GUI	Time	Implicit	Probability	Many	Custom
SPE•ED	Both	Simulation	GUI	Computed Time	Implicit	Computed	Standard Priority	No

The tools selected for the table are representative of the types of products currently available. The products support system and software models as well as custom models that may model more general systems in addition to computer systems. Thus, they are the types of products that we wish to interchange model information among. The following characteristics are pertinent to PMIF choices:

- The tools use a range of solution types: analytic, simulation, and hybrid. Different tools require different additional information. The analytic tools

support a subset of all models, the simulation tools can use additional model information.

- The modeling tools originally had language-oriented user interfaces. Now, many support graphical user interfaces (GUI), spreadsheet-type interfaces, and forms-based interfaces.
- There are three basic approaches to specifying service requests for servers in the model:
  - Time: specifies the service time per visit to a server
  - Demand: specifies the total service time for all visits to a server
  - Computed: calculates either time or demand from other data specified in the model.
- Most tools model time units implicitly: all specifications must be in the same relative time units (e.g. sec. or hours) the modeler can choose the time unit most convenient for the problem. QASE lets the user choose a time unit for each specification. BEST/1 uses a time unit appropriate for the device or specification, such as ms. for disk devices and transactions per hour for an arrival rate.
- Routing specifications vary considerably. SES and QNAP2 specify routing among servers by specifying the probability that a job goes from device A to device B. Analytic tools that specify demand service requests do not need routing because they specify the total demand for all visits. CSIM specifies routing through logic in C code, and QASE and *SPE•ED* compute routing from other specifications.
- Queue-scheduling disciplines (QD) also vary among tools. Simulation based tools generally support many more QDs than analytic tools. "Standard" QD for analytic tools are those that have efficient, exact analytic solutions (FCFS, PS, and IS). Many support priority scheduling with analytic approximations.
- "Domain" is IBM/MVS terminology for a feature that limits the number of jobs in the system by workload. Some analytic tools support this with an approximate analytic solution. Others have no support for "domains". Simulation tools can support the concept through primitives that let the modeler simulate the holding of jobs when the number in the system exceeds the limit.

This initial version of the PMIF demonstrates the feasibility of the approach by addressing a subset of the full PMIF. It seems best to have the initial version of the PMIF support a minimum content set and use "meta-model extensions" to add feature sets supported by a subset of the tools. This prototype PMIF will represent the subset of data needed by all the analytic tools. The graphical view of the model must be handled by the PMIF; however the visual representation of models is not a central issue in establishing PMIF feasibility. Therefore, this feasibility study omits graphics. The additional extensions for visual representation are a topic for future research. Features supported by some tools, such as priority scheduling and domains that can be solved with approximate solution algorithms need further research. There is currently no consensus on the service time versus demand and transition probability versus server visits. PMIF will support either demand or time service requests. It specifies routing through visits (or demand) because most analytic tools use this approach. The other tools can convert visits to probabilities when they import the PMIF model. It does not include results. Further research is needed to integrate results.

4.2 QNM Terminology

QNM terminology varies considerably with modeling tools and textbooks on performance modeling. The PMIF must compromise among the options and must contain the information that can be translated into the other options. This section examines the key terms that have the greatest variability. Table 2 shows the choices supported by the set of tools from the previous section, and from four recent performance modeling texts. Terminology choice often depends on the paradigm that the authors or developers select. Most terminology comes from either an operational analysis or a stochastic modeling paradigm. Both QASE and CSIM support a generic paradigm that focuses on the models themselves and uses terms that describe the system being modeled.

The terms used to describe workloads vary tremendously. Most use either class or workload class. The types of workloads are either described using the data processing terms of the workloads being modeled (e.g., timesharing, transaction, batch) or the queueing terms open and closed. Some avoid using the terms and specify either an arrival rate (for open) or a population and think time (for closed). Similarly there are many terms used to refer to queues.

There is not yet a consensus in the performance community on any of these QNM terms. Note that there is more agreement among the textbooks than the tools. This is probably because tool developers select terms that should be meaningful to the user

Table 2. QNM Terminology

Tool / Text	Paradigm	Workload Term	Workload Types	Queue Term
BEST/1	Operational	Workload	Transaction Timesharing Batch	Device & Model name
CSIM	Custom	Process	Custom	Facility
MAP	Operational	Class	Transaction, Batch, Terminal	Center
QASE	Custom	Workload	Arrival rate or Number jobs	Device types
QNAP2	Stochastic	Class	Generation rate, Initial population, or custom	Stations & type
QSolver/1	Operational	Workload class	Transaction, Interactive, Batch	Device
SES	Stochastic	Category	Arrival rate or Number users	Server
SPE•ED	Operational	Scenario	Arrival rate or Number users	Device
[JAIN90]	Stochastic	Class	Open, closed	Service center
[MENA94]	Operational	Workload Class	Transaction, Interactive, Batch	Server
[MOLL89]	Stochastic	Class	Open, closed	Queue
[SMIT90a]	Operational	Category	Open, closed	Node

who is focused on modeling a particular system rather than theoretical terms. Tool developers are usually familiar with QNM theory and thus recognize that *class* and *workload* usually refer to the same thing; and *server* and *queue* are usually the same. Therefore, the QNM meta-model described in the next section selects terms that should be meaningful to the developers of performance modeling tools. Tool users should not need to use the PMIF directly so the terminology does not need to be familiar to them. Specific choices for terminology are described in the next section.

5 QNM Meta-Model

This model is known as the QNM meta-model because it is a model of the information that goes into constructing a QNM. This meta-model serves two purposes. The first is to provide a rigorous definition for the information required for a QNM that may be solved using exact analytical techniques. This is valuable to performance tool vendors because it defines the information that may be exported and imported between tools that support this initial version of a PMIF. It is also an initial step to creating a standard for QNM terminology and the functions that should be supported by performance modeling tools to support SPE.

The second purpose of the meta-model is to generate a prototype version of a formal PMIF using the CDIF transfer format derived from the meta-model. With the PMIF, performance modeling tools can exchange information, and SPE tools can use a performance modeling tool best suited to the software / hardware architecture issues and the life cycle stage of the assessment.

This section begins with a textual description of the QNM meta-model. It is followed by a discussion of the issues discovered in the creation of the meta-model and how they were resolved in this initial version.

5.1 QNM Meta-Model Description

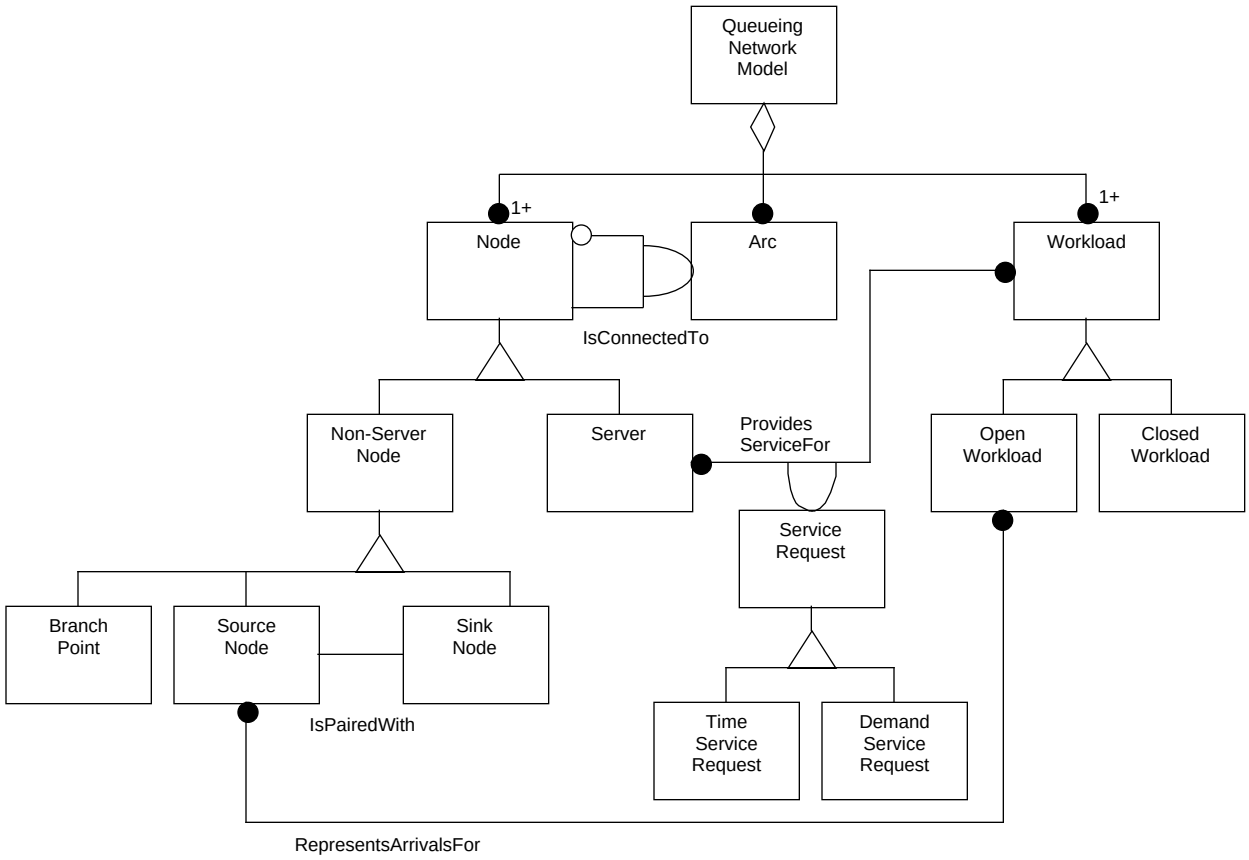
The meta-model Entity-Relationship-Attribute (ERA) diagram is shown in Figure 1. The ERA notation with the OMT extensions is described in [WILL95] and summarized in Appendix A. The following paragraphs describe the entities and their relationships.

A *QueueingNetworkModel* is composed of one or more *Nodes*,zero or more *Arcs*, and one or more *Workloads*. An *Arc* connects one *Node* to another *Node*. Several types of *Nodes* may be used in constructing a *QueueingNetworkModel*:

- *Server*: represents a component of the execution environment that provides some processing service.
- *Non-ServerNode* represents nodes that show topology of the model, but do not provide service. There are three types of *Non-ServerNodes*

    - *SourceNode*: represents the origin of an *OpenWorkload*.
    - *SinkNode* represents the exit point of an *OpenWorkload*.
    - *BranchPoint* is a convenient way to represent the origin or destination of multiple arcs.

Figure 1. Queueing Network Meta-Model



The following shows the entities in the above diagram and their attributes. Note that EIA/CDIF supports inheritance. For example, DemandServiceRequest inherits the attributes from ServiceRequest, so in addition to ServiceDemand it also has the inherited attributes WorkloadName, ServerID, and TimeUnits.

Arc
- Description
- FromNode
- ToNode

BranchPoint

ClosedWorkload
- NumberOfJobs
- ThinkTime
- TimeUnits

DemandServiceRequest
- ServiceDemand

Node
- Name
- ID

Non-ServerNode
- NodeType

OpenWorkload
- ArrivalRate
- TimeUnits

QueueingNetworkModel
- Name
- Description

Server
- Quantity
- SchedulingPolicy

ServiceRequest
- WorkloadName
- ServerID
- TimeUnits

SinkNode

SourceNode

TimeServiceRequest
- ServiceTime
- NumberOfVisits

Workload
- WorkloadName

A *Server* provides service for one or more *Workloads.* A *Workload* represents a collection of transactions or jobs that make similar *ServiceRequests* from *Servers* in the *QueueingNetworkModel*. There are two types of *Workloads:*

- *OpenWorkload:* represents a workload with a potentially infinite population where transactions or jobs arrive from the outside world, receive service, and exit. The population of the *OpenWorkload* at any one time is variable.
- *ClosedWorkload:* represents a workload with a fixed population that circulates among the *Servers*.

A service request associates the *Workloads* with *Servers*. A *ServiceRequest* specifies either the average *TimeService* or *DemandService* provided for each *Workload* that visits the *Server*. A *TimeServiceRequest* specifies the average service time and number of visits provided for each *Workload* that visits the *Server*. A *DemandServiceRequest* specifies the average service demand (service time multiplied by number of visits) provided for each *Workload* that visits the *Server*. The formal EIA/CDIF model definition is in [SMI94b]. It formally defines the entities, each of the attributes, and the relationships.

The EIA/CDIF transfer format is derived from the meta-model. It is a Lisp-style notation. Each of the entities and its attributes is represented as follows:

```
(EntityName EntityID
     (Attribute1 Attribute1Value)
     …
     (AttributeN AttributeNValue)
)
```

For example, an arc entity in a queueing model may be defined as:

```
(Arc QNM001.1
     (FromNode #d4)
     (ToNode #d3)
)
```

Arc is the EntityName and the EntityID is the Meta-entityID followed by an InstanceNumber. [3] The arc has no (optional) description, and specifies the FromNode is nodeID 4 and ToNode is nodeID 3. Section 6 demonstrates the transfer format for a case study.

5.2 Meta-Model Issues

This meta-model proposes the following compromises on terminology and features:

- Visits vs. probabilities: We propose using the operational analysis term *visits* rather than the stochastic modeling *probability*. Fewer tools use probabilities and they can calculate probabilities from visits.
- Demand vs. service time: We propose both – a ServiceDemand may be either a TimeServiceRequest or a DemandServiceRequest.
- Queue scheduling disciplines: QD is an attribute of Server. We propose an enumerated type that initially supports the "standard" set (Processor Sharing, First-Come-First-Served, and Infinite Server). Extensions will be straightforward.
- Workload types: We propose OpenWorkloads and ClosedWorkloads. These terms are familiar to tool developers.

---

[3] This does not strictly follow the EIA/CDIF standard. Their specification is unclear, we submitted a request for clarification, and substituted this approach until we can determine how to comply with the standard.

Several entities and relationships are not required for this prototype, but are included for future expansion:

- Arcs are not needed if visits or demand are specified. They will be needed for the graphical representation of the model.
- Non-server nodes are not needed for the efficient, exact solution subset. They will be needed for the graphical representation and for simulation models.
- The relationship between the open workloads and the source node is not needed for this model subset. It will be needed for simulation models.

In the future, we may want a third type of ServiceRequest that specifies a ServiceUnit (such as 50 ms. per I/O) for a Server then has a UnitServiceRequest that specifies the number of units of service requested (such as 11 I/Os). A few tools currently use this feature, and it is desirable for SPE models.

6 Case study

A case study based on a simple automated teller machine (ATM) illustrates the definition of a QNM with the proposed PMIF. This case study is used in a companion paper that defines the information requirements for a Software Performance Engineering study with an SPE meta-model [WILL94]. The same case study is used here to show the connection between the SPE information that is collected and the system model that may be represented with the proposed PMIF. The PMIF format is the transfer format derived from the QNM meta-model using the EIA/CDIF standards [EIA94].

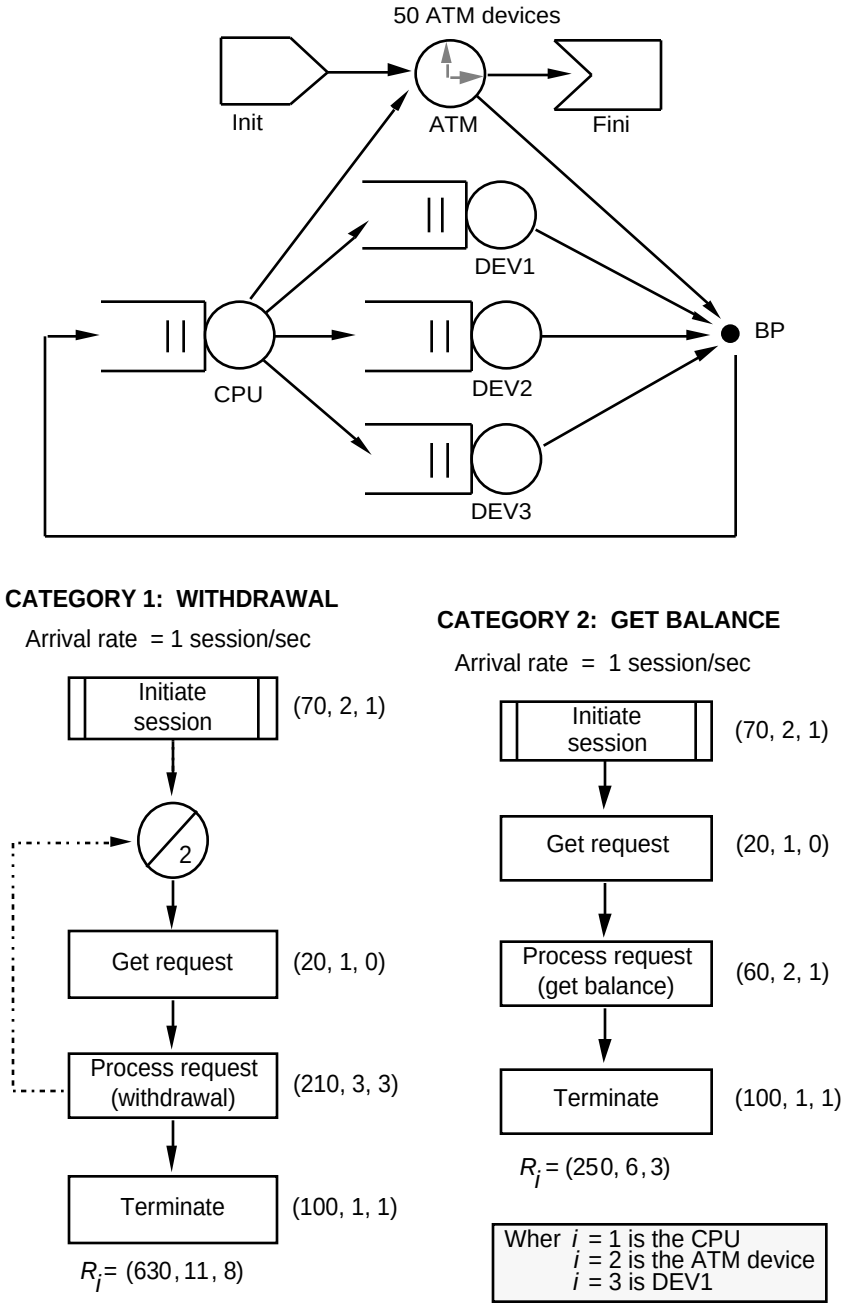
To conduct an SPE study, an analyst first collects the SPE information and creates a *software model* for each performance scenario (workload). The software models are solved using graph analysis algorithms. The model solution yields the resource requirements for the computer devices for each scenario. SPE information about the envisioned computer environment is used to produce the *system model* topology and the service rates for the devices in the model. This is combined with the results of the software model solution to produce the parameters for the system model. The system model may then be solved using any of the tools described in section 4. This section first describes the case study model, then illustrates the PMIF for the QNM system model.

5.1 The ATM Model

The ATM accepts a cash card and requests a personal identification number (PIN) for verification. Customers can perform any of three transactions at the ATM: deposit cash to an account, withdraw cash from an account, or request the available balance in an account. A customer may perform several transactions during a single ATM session. The ATM communicates with a computer at the host bank which verifies the account and processes the transaction.

The case study models two workloads as illustrated in Figure 2. One is an ATM session in which a customer requests a Withdrawal transaction (on the left). The other is an ATM session in which a customer requests a Get balance transaction (on the right). Each of the workloads is depicted with an *execution graph* software model. The model

Figure 2. Case Study Model



creation and solution is described in detail in [SMIT90a]; this section summarizes the modeling process. The software model solution yields the following resource requirements for the servers in the model:

- Withdrawal: (630, 11, 8)
- Get Balance: (250, 6, 3)

for the CPU, ATM, and DEV1 respectively. Note that the case study picture shows three disk devices (DEV1 - DEV3). The software models only use DEV1, so it is the only one illustrated with the PMIF. These resource requirements are translated into the parameters for the QNM that are shown in Table 3.

Table 3. QNM Parameters

Nodes	Visits (Withdraw)	Visits (Get Bal)	Service Time (Withdraw)	Service Time (Get Bal)
Init	1	1		
ATM	11	6	1	1
Fini	1	1		
DEV1	8	3	0.05	0.05
BP	18	8		
CPU [4]	all	all	0.00630	0.00250

5.2 Case Study PMIF

The complete model is in Appendix B. This section describes some key portions of the PMIF.

A Workload in the model is defined as follows:

```
(OpenWorkload QNM007.1
     (WorkloadName “Withdrawal”)
     (ArrivalRate #d1)
     (TimeUnits <Sec>)
)
```

The case study has two open workloads, each has an arrival rate of 1 session per second.

A Server in the model is defined as follows:

```
(Server QNM009.1
     (Name “CPU”)
     (ID #d1)
     (Quantity #d1)
     (SchedulingPolicy <PS>)
)
```

This specification defines one CPU server with a Processor Sharing queue-scheduling discipline. Three servers are defined in the case study: CPU, ATM and DEV1. The case study also has three Non-serverNodes: the Source node “Init”, the Sink node “Fini”, and a BranchPoint “BP”. Their definition is shown in [SMI94b].

A ServiceRequest for the CPU is specified as follows:

```
(DemandServiceRequest QNM004.1
     (WorkloadName “Withdrawal”)
     (ServerID #d1)
     (TimeUnits <Sec>)
     (ServiceDemand #d0.00630)
)
```

This specifies that the Withdrawal workload makes a total demand of 0.00630 seconds at the CPU server.

---

[4] These values are not consistent with the model parameters in Table 5.6 in [SMIT90a]. However this case study uses the total demand for the CPU instead to demonstrate the DemandServiceRequest.

A ServiceRequest for the disk device is specified as follows:

```
(TimeServiceRequest QNM013.1
      (WorkloadName “Withdrawal”)
      (ServerID #d2)
      (TimeUnits <Sec>)
      (ServiceTime #d0.05)
      (NumberOfVisits #d8)
)
```

This specifies that the Withdrawal workload makes 8 visits to DEV1, each visit requires an average of 0.05 sec.

The PMIF in Appendix B also shows the arc specifications and the `IsPairedWith` and `RepresentsArrivalsFor` relationships. The other relationships depicted in the QNM meta-model (eg. IsConnectedTo, ProvidesServiceFor, etc.) are explicitly described with associative entities (e.g. Arcs and ServiceRequests, etc.) therefore their relationship declaration is unnecessary.

6 Summary and Conclusions

This paper has developed a Performance Model Interchange Format (PMIF) for exchanging Queueing Network Models (QNMs) among performance modeling tools. The PMIF is based on the representation technique of the EIA/CDIF standard. It consists of a QNM meta-model depicted with an extended ERA diagram and the formal definition of the entities, relationships and attributes in the model. The transfer format derived from the QNM meta-model serves as the PMIF representation.

The PMIF content represents features from a representative set of today’s modeling tools. The prototype PMIF defined here represents the subset of data needed by all the analytic tools. Features supported by some tools, such as priority scheduling and domains will be added in future versions. Terminology used in the PMIF is based on terms that are meaningful to developers of performance modeling tools rather than terms familiar to tool users.

The PMIF is defined and used in a case study. The paper demonstrates the feasibility of defining a QNM in a standard format that will permit models defined in the format to be solved by all tools that support the standard.

This paper proposes the prototype PMIF representation technique and content to the performance modeling community as the basis for a new standard interchange format. Future modifications and extensions to the PMIF will be based on feedback from researchers and developers of performance modeling tools.

Several extensions are needed: the graphical representation of the model, the model results, and additional model features such as priorities, domains, passive resources, scheduling disciplines, and so on. The QNM model also should be reconciled with the SPE meta-model defined in [WILL95]. The overall goal is to support design decision analysis through the smooth (transparent) transfer of information from software designs to performance model solutions and back to the designer.

References

[BALD89] M. Baldassari, et al., "PROTOB: A Hierarchical Object-Oriented CASE Tool for Distributed Systems," *Proceedings European Software Engineering Conference - 1989*, Coventry, England, Sept. 1989.

[BEIL90] Heinz Beilner, "Strutured Modelling and Tool Support," *Int. Conf. Performance of Computers and Computer Networks*, Johannesburg and Stellenbosch, South Africa, 1990.

[BEIL88] Heinz Beilner, J. Mäter, and N. Weissenburg, "Towards a Performance Modeling Environment: News on HIT," *Proceedings 4th International Conference on Modeling Techniques and Tools for Computer Performance Evaluation*, Plenum Publishing, 1988.

[BELL88] Thomas E. Bell, (Editor), *Special Issue on Software Performance Engineering*, Computer Measurement Group Transactions, 1988.

[BROW85] J.C. Browne, et al., "Graphical Programming for Simulation Models of Computer Systems," *Proceedings 18th AnnualSimulation Symposium*, Tampa, FL, Mar. 1985.

[BUHR 89] R.J. Buhr, et al., "Software CAD: A Revolutionary Approach," *IEEE Transactions on Software Engineering*, vol. 15, no. 3, Mar. 1989, pp. 234-249.

[CRAW84] J. Crawford and R. Smith, "An Electronic Design Interchange Format - EDIF," *Proc. IEEE ICCD 84*, Rye Town Hilton, NY, October 1984, pp. 82-86.

[EDIF] EDIF, "EDIF Users' Group," , Design Automation Department, Texas Instruments, PO Box 225474, MS-3668, Dallas TX 75265,

[EIA94] EIA, "CDIF - CASE Data Interchange Format Overview," No.EIA/IS-106, Engineering Department, Electronics Industries Association, Arlington, VA, January, 1994.

[FOX89] Gregory Fox, "Performance Engineering as a Part of the Development Lifecycle for Large-Scale Software Systems," *Proceedings 11th International Conference on Software Engineering*, Pittsburgh, PA, May 1989, pp. 85-94.

[GOET90] Robert T. Goettge, "An Expert System for Performance Engineering of Time-Critical Software," *Proceedings Computer Measurement Group Conference*, Orlando FL, 1990, pp. 313-320.

[GÖTZ93] N. Götz, U. Herzog, and M. Rettelbach, "Multiprocessor and Distributed System Design: the Integration of Functional Specification and Preformance Analysis using Stochastic Process Algebras," *Proc. Performance '93*, Rome, September 1993.

[GRUM91] Adam Grummitt, "A Performance Engineer's View of Systems Development and Trials," *Proceedings Computer Measurement Group Conference*, Nashville, TN, 1991, pp. 455-463.

[HILL92] Jane Hillston, "A Tool to Enhance Model Exploitation," *6th Int. Conf. Modelling Techniques and Tools for Computer Performance Evaluation*, R. Pooley

and J. Hillston, ed., Edinburgh, Edinburgh University Press, September 1992, pp. 179-193.

[JAIN90] R. Jain, *Art of Computer Systems Performance Analysis*, New York, NY, John Wiley, 1990.

[LOR91] K. Lor and D.M. Berry, "Automatic Synthesis of SARA Design Models from System Requirements," *IEEE Transactions on Software Engineering*, vol. 17, no. 12, Dec. 1991, pp. 1229-1240.

[MENA94] Daniel A. Menascé, Virgílio A.F. Almeida, and Larry W. Dowdy, *Capacity Planning and Performance Modeling*, Englewood Cliffs, NJ, PTR Prentice Hall, 1994.

[MOLL89] Michael K. Molloy, *Fundamentals of Performance Modeling*, MacMillan, 1989.

[OPDA92b] A. Opdahl and A. Sølvberg, "Conceptual Integration of Information System and Performance Modeling," *Proceedings Working Conference on Information System Concepts: Improving the Understanding*, 1992.

[PATE91] M. Paterok, R. Heller, and H. deMeer, "Performance Evaluation of an SDL Run Time System - A Case Study," *Proceedings 5th International Conference on Modeling Techniques and Tools for Computer Performance Evaluation*, Torino, Italy, Feb. 1991, pp. 86-101.

[ROLI92] J.A. Rolia, "Predicting the Performance of Software Systems," University of Toronto, 1992.

[RUMB91] J. Rumbaugh, et al., *Object-Oriented Modeling and Design*, Englewood Cliffs, NJ, Prentice Hall, 1991.

[SHEN90] V. Shen, et al., "VERDI: A Visual Environment for Designing Distributed Systems," *Journal of Parallel and Distributed Systems*, vol. 18, no. 6, June 1990.

[SMIT90a] Connie U. Smith, *Performance Engineering of Software Systems*, Reading, MA, Addison-Wesley, 1990.

[SMIT91] Connie U. Smith, "Integrating New and 'Used' Modeling Tools for Performance Engineering," *Proceedings 5th International Conference on Modeling Techniques and Tools for Computer Performance Evaluation*, Torino, Italy, Feb. 1991.

[SMI94b] Connie U. Smith, "Definition of A Performance Model Interchange Format," No.PES-1001-94, Performance Engineering Services, October, 1994.

[SMIT94a] Connie U. Smith, "Performance Engineering," in *The Encyclopedia of Software Engineering*, John Wiley and Sons, 1994.

[SMI94d] Connie U. Smith and Bernie Wong, "SPE Evaluation of a Client/Server Application," *Proc. Computer Measurement Group*, Orlando, FL, Dec. 1994.

[TURN92] Michael Turner, Douglas Neuse, and Richard Goldgar, "Simulating Optimizes Move to Client/Server Applications," *Proceedings Computer Measurement Group Conference*, Reno, NV, Dec. 1992, pp. 805-814.

[VALD92] A. Valderruten, et al., "Deriving Queueing Networks Performance Models from Annotated LOTOS Specifications," *Proc. 6th Int. Conf. on Modelling Techniques and Tools for Computer Performance Evaluation*, R. Pooley and J. Hillston, ed., Edinburgh, Edinburgh Press, September 1992, pp. 167-178.

[WILL94] Lloyd G. Williams, "Definition of the Information Requirements for Software Performance Engineering," No.SERM-021-94, Software Engineering Research, October, 1994.

[WILL95] Lloyd G. Williams and Connie U. Smith, "Information Requirements for Software Performance Engineering," *Proceedings 1995 International Conference on Modeling Techniques and Tools for Computer Performance Evaluation*, Heidelberg, Germany, Submitted for publication 1995.

APPENDIX A: OMT OBJECT MODEL NOTATION

The OMT Object Model Notation [RUMB91] is used to document the classes in an application and the relationships among them. This report uses a subset of this notation to graphically document the SPE meta-model. The subset used here was chosen for conformance with the EIA/CDIF proposed standard for CASE data interchange [EIA94]. The graphical symbols used to construct the SPE meta-model are shown in Figure A.1.

A class is denoted by a rectangle labeled with the class name. The attributes of the class may optionally be included in the rectangle. A class models a meta-entity in the CDIF format. For classes with more than a few attribtes, however, this makes the diagram difficult to read. Here, attributes are included only in the textual description of the eta-model.

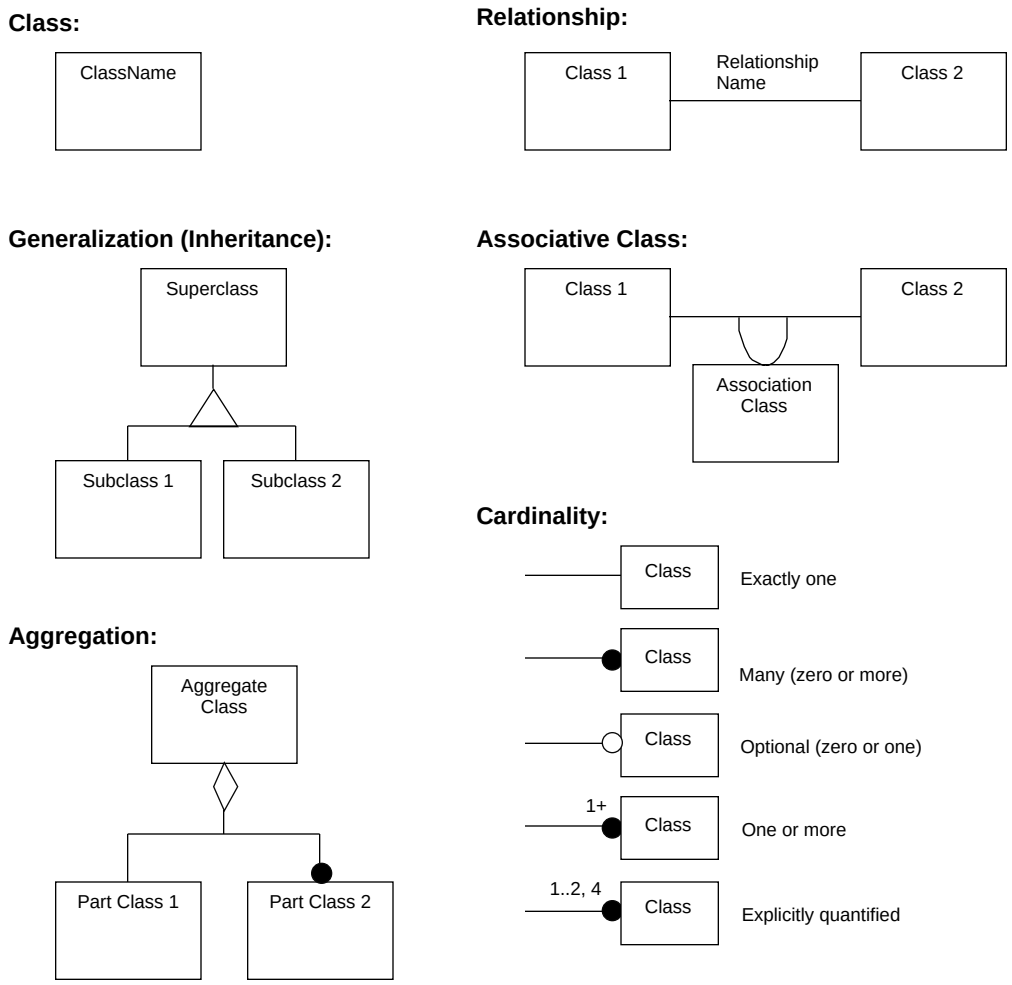


Figure A.1: OMT Object Model Notation Symbols

A relationship between two classes is indicated by a line connecting the two classes.† The line is labeled with the name of the relationship. Cardinality constraints on relationships are indicated by decorations on the line next to the class whose participation they constrain. The various decorations are shown in Figure A.1.

Inheritance is modeled by a generalization relationship. With inheritance, properties common to a group of classes are assigned to a *superclass*. Each *subclass* inherits all of the attributes and relationships of its superclass(es). A generalization relationship is indicated by a triangle whose apex points at the superclass.

Associative classes model relationships as classes. This allows adding information and behavior to the relationship. Each instance of the relationship becomes an instance of the associative class. Associative classes correspond to associative meta-entities in the CDIF format. While the proposed CDIF standard included associative meta-entities, the CDIF graphical notation does not include special syntax to indicate these entities. An associative class is indicated by a loop conncecting the associative class to the relationship that it represents.

Aggregation models a whole/part or "is composed of" relationship in which an instance of one class is composed of instances of one or more other component, or part, classes. The proposed CIDF standard does not include a special representation for aggregation. However, aggregation is a commonly used relationship and it is useful to be able to indicate it directly on the graphical model. In the OMT notation, aggregation is indicated by a diamond attached to the aggregate class.

---

† The OMT notation allows ternary relationships. However, to conform with the EIA/CDIF draft standard, we have restricted relationships to be binary only.

APPENDIX B: PMIF TRANSFER FORMAT

```
#| Model Section |#
(:MODEL
    (QNM Model QNM008.1
        (Name “ATM Sample”)
    )
    (OpenWorkload QNM007.1
        (WorkloadName “Withdrawal”)
        (ArrivalRate #d1)
        (TimeUnits <Sec>)
    )
    (OpenWorkload QNM007.2
        (WorkloadName “Get Balance”)
        (ArrivalRate #d1)
        (TimeUnits <Sec>)
    )
    (Server QNM009.1
        (Name “CPU”)
        (ID #d1)
        (Quantity #d1)
        (SchedulingPolicy <PS>)
    )
    (Server QNM009.2
        (Name “DEV1”)
        (ID #d2)
        (Quantity #d1)
        (SchedulingPolicy <FCFS>)
    )
    (Server QNM009.3
        (Name “ATM”)
        (ID #d3)
        (Quantity #d50)
        (SchedulingPolicy <IS>)
    )
    (SourceNode QNM012.1
        (Name “Init”)
        (ID #d4)
        (NodeType <Source>)
    )
    (SinkNode QNM011.1
        (Name “Fini”)
        (ID #d5)
        (NodeType <Sink>)
    )
    (BranchNode QNM011.1
        (Name “BP”)
        (ID #d6)
        (NodeType <Branch>)
    )
    (DemandServiceRequest QNM004.1
        (WorkloadName “Withdrawal”)
        (ServerID #d1)
        (TimeUnits <Sec>)
        (ServiceDemand #d0.00630)
    )
    (TimeServiceRequest QNM013.1
        (WorkloadName “Withdrawal”)
        (ServerID #d2)
        (TimeUnits <Sec>)
        (ServiceTime #d0.05)
        (NumberOfVisits #d8)
    )
    (TimeServiceRequest QNM013.2
        (WorkloadName “Withdrawal”)
        (ServerID #d3)
        (TimeUnits <Sec>)
        (ServiceTime #d1)
        (NumberOfVisits #d11)
    )
    (DemandServiceRequest QNM004.2
        (WorkloadName “Get Balance”)
        (ServerID #d1)
        (TimeUnits <Sec>)
        (ServiceDemand #d0.00250)
    )
    (TimeServiceRequest QNM013.1
        (WorkloadName “Get Balance”)
        (ServerID #d2)
        (TimeUnits <Sec>)
        (ServiceTime #d0.05)
        (NumberOfVisits #d3)
    )
    (TimeServiceRequest QNM013.2
        (WorkloadName “Get Balance”)
        (ServerID #d3)
        (TimeUnits <Sec>)
        (ServiceTime #d1)
        (NumberOfVisits #d6)
    )
    (Arc QNM001.1
        (FromNode #d4)
        (ToNode #d3)
    )
    (Arc QNM001.2
        (FromNode #d3)
        (ToNode #d5)
    )
    (Arc QNM001.3
        (FromNode #d3)
        (ToNode #d6)
    )
```

```
    (Arc QNM001.4                                   (FromNode #d1)
        (FromNode #d6)                              (ToNode #d3)
        (ToNode #d1)                            )
    )                                           (Arc QNM001.7
    (Arc QNM001.5                                   (FromNode #d2)
        (FromNode #d1)                              (ToNode #d6)
        (ToNode #d2)                            )
    )
    (Arc QNM001.6
    (IsPairedWith QNM036.1 QNM006.1 QNM006.2)
    (RepresentsArrivalsFor QNM037.1 QNM006.1  QNM007.1)
    (RepresentsArrivalsFor QNM037.2 QNM006.1  QNM007.2)
)
```