A PMIF with Petri Net Building Blocks

Catalina M. Lladó
Computing and Maths Department
Universitat de les Illes Balears
07071 Palma de Mallorca, Spain
cllado@uib.cat

Peter G. Harrison
Department of Computing
Imperial College London
London SW7 2AZ, UK
pgh@doc.ic.ac.uk

ABSTRACT

Performance model interchange formats (PMIFs) support the portability of models and sharing of solutions amongst different tools. XML-based interchange formats have been defined for the interchange of queueing network and Petri net models, amongst others, but there is still scope to extend their application to multiple formalisms, in particular beyond queueing networks. We extend an existing PMIF to hybrid models by including a new type of node, called a "building block", defined as a certain class of Petri nets. The synchronisation primitives of these building blocks can be used to specify fork-join systems whilst, under certain conditions, retaining product-form solutions when embedded in queueing (or other) networks possessing this property already. When a product-form does not exist, the whole network is translated into a Petri net and solved either by simulation or direct solution of the underlying Markov chain by an existing analyser. Finally, we apply the extended PMIF to model a computer system with RAID storage.

Categories and Subject Descriptors

B.8.2 [**Hardware**]: Performance Analysis and Design Aids; C.4 [Performance of Systems]: Modeling Techniques.

General Terms

Performance engineering methodology, Markov processes, portable tools, performance interoperability

Keywords

Performance modelling interchange formats, Petri nets, RAID models, Product-form solutions.

1. INTRODUCTION

In general, model interchange aims to allow existing tools to cooperate seamlessly in carrying out different tasks. XML-based interchange formats for models provide a mechanism whereby the models' information may be transferred amongst various modeling and analysis tools.

ICPE'11, March 14–16, 2011, Karlsruhe, Germany.

Performance MIFs (PMIFs) [13], in particular, are XML interchange formats that provide a mechanism for sharing performance models across appropriate software tools. Use of a PMIF does not require a tool to know about the capabilities of other tools, internal data formats or even their 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 been defined for queueing network models (QNM), software performance models (S-PMIF), layered queueing networks (LQN), UML, Petri nets (PNs) and other types of models. Each one of these is specific to one formalism. In this paper, however, we propose to extend the application of the performance interoperability concept to multiple formalisms or hybrid models.

We describe an extension of the Generalised PMIF (GPMIF) [10] to hybrid models by including new types of nodes, called "building blocks" (BBs), defined as a certain class of Stochastic Petri nets (SPNs). In fact the simplest building block, BB-1, consists of a single Petri net place with a single input and single output transition, with no capacity constraint. It is therefore equivalent to an M/M/1 queue. Higher order building blocks BB-n have n places with pairs of input and output transitions that connect to the same subset of k places ($1 \leq k \leq n$). The BBs are defined in the enlarged GPMIF schema and models that use them can be implemented in general by simulation or direct solution of the underlying Markov chain, or via a product-form solution when one exists.

This last possibility is shown to be quite viable in closed networks by application of the Reversed Compound Agent Theorem (RCAT), the conditions of which are satisfied in the extended GPMIF under conditions very similar to those required of queueing networks [7]. As a result, complex systems can be solved efficiently and mechanically at equilibrium; with a wide range of parameterisations in closed systems and under certain conditions on the rates in open systems. In particular, the BB can describe fork-join operations and we solve such a model by two solution methods for the same specification: via the Petri net analyser PIPE and a product-form solution [1, 2]. This results in a realistic application to RAID-like storage systems.

The paper is organised as follows. In the next section we provide background material on the relevant concepts: PMIFs, queues and Petri nets, including the definition of the new BBs. We also define the further extended GPMIF by showing how an XML specification for BBs can be integrated into the current GPMIF schema (already an extension of the original PMIF schema, as noted above). Section 3 describes

two solution methods: product-form solution and simulation. An abstract case study is presented in section 4 which is solved both as a product-form for the equilibrium state probabilities and by simulation via translation into PIPE. Both open and closed, product-form and non-product-form cases are considered. The paper concludes in section 5.

2. BB COMPONENTS IN GPMIF

This section describes the specification of the Petri net building blocks in GPMIF (Generalised PMIF). This is based on PMIF 2.0, which is a common representation for system performance model data that can be used to move models among modeling tools that use a QNM paradigm [12]. GPMIF allows, in addition, the specification of non-standard queueing networks, such as G-networks, and certain fixpoint solutions [9, 6][1].

A natural, orthogonal direction to extend GPMIF is to find non-queueing building blocks that can be defined as primitive service centres with associated workloads that interact with the existing components. We look to Petri nets for such building blocks because of their ease of expressing features found in real networks and their widespread use. To this end, consider a building block that consists of a set of places $P_1, \ldots P_N$, a set $\mathcal{T}_I$ of input transitions whose input vectors are null (i.e. $\mathbf{0} = (0, \ldots, 0)$), and a set $\mathcal{T}_O$ of output transitions whose output vectors are null[2]. All the arcs have multiplicity 1.

Definition 1 (Building block (BB) [2]). *Given an ordinary (connected) SPN S with set of transitions $\mathcal{T}$ and set of N places $\mathcal{P}$, then S is a* building block *if it satisfies the following conditions:*

1. *For all $T \in \mathcal{T}$ then either $\mathbf{O}(T) = \mathbf{0}$ or $\mathbf{I}(T) = \mathbf{0}$. In the former case we say that $T \in \mathcal{T}_O$ is an* output transition *while in the latter we say that $T \in \mathcal{T}_I$ is an* input transition. *Note that $\mathcal{T} = \mathcal{T}_I \cup \mathcal{T}_O$ and $\mathcal{T}_I \cap \mathcal{T}_O = \emptyset$, where $\mathcal{T}_I$ is the set of input transitions and $\mathcal{T}_O$ is the set of output transitions.*
2. *For each $T \in \mathcal{T}_I$, there exists $T' \in \mathcal{T}_O$ such that $\mathbf{O}(T) = \mathbf{I}(T')$ and vice versa.*
3. *Given two places $P_i, P_j \in \mathcal{P}$, $1 \leq i, j \leq N$, there exists a transition $T \in \mathcal{T}$ such that the components i and j of $\mathbf{I}(T)$ or of $\mathbf{O}(T)$ are non-zero.*

Condition 1 requires that all the transitions are either input or output transitions, while Condition 2 states that if there exists an input transition T_y feeding a subset of places y, then there must be a corresponding output transition T'_y that consumes the tokens from the same subset; i.e. for each input transition T_y there must exist an output transition T'_y whose input vector is equal to the output vector of T_y. Finally, Condition 3 simply requires the SPN to be connected.

Figure 1 illustrates an example of a BB consisting of 3 places $\mathcal{P} = \{P_1, P_2, P_3\}$, 3 input transitions $\mathcal{T}_I = \{T_3, T_{23}, T_{12}\}$ and 3 output transitions $\mathcal{T}_O = \{T'_3, T'_{23}, T'_{12}\}$.

[1]Later, GPMIF was further generalised to accommodate the extended RCAT, whilst maintaining compatibility with its previous versions [8].

[2]The components of an input/output vector specify the number of tokens taken from / added to the corresponding places. Thus a null vector implies tokens arrive/depart externally.

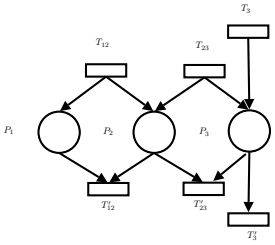

Figure 1: A 3-place building block.

Note that if two or more input (output) transitions have the same output (input) vector, we can fuse them in one transition whose rate is the sum of the rates of the original transitions. Therefore, without loss of generality, we assume that all the input (output) transitions have different output (input) vectors.

To simplify the notation, we use T_y (T'_y) to denote an input (output) transition, where y is the set of place-indices of the non-zero components in the output (input) vector of T_y (T'_y). For instance, transition T_{23} (T'_{23}) in the net of Figure 1 is the transition that produces (consumes) the tokens in P_2 and P_3.

Keeping the model interchange format compatible with PMIF 2.0 [13] and GPMIF [10], we added the following new elements to represent BBs and their services:

- *PNBuildingBlock* with the attribute *Name* and and the sub-elements:
  - *TransitionPairsList* represents the list of transition pairs of the BB, as described in the previous section. Syntactically, it is defined as a list of *TransitionPair*s. Each of these is specified by a *TransitionID* and an *OutputServiceTime*, corresponding to the output transition of the BB. The rate of the input transition is undefined at this stage; it will be determined by the parameters of rest of the model.
  - *PlacesList* is the list of places of the BB. Each element *Place* has an attribute *PlaceID* and it also comprises a list of *TransitionPairRelated* elements, specifying to which transition pairs this place is output/input.
- *BBInputProbability*, a new type of *ServiceRequest* (since it might depend on the workload), specifies the routing probabilities to the different transitions comprising the BB and has the attributes:
  - *BBID*, to identify the BB.
  - *WorkloadName*, specifying the workload that it refers to.
  - *TransitionID*, to identify the specific transition of the BB.
  - *InputProbability*, giving the routing probability value.

2.1 Specification of a central server system with RAID

The following excerpt shows the BBs specification based on the above schema for the RAID case study of section 4.1.

Specification of Nodes and Workloads is as specified in PMIF, see [13].

```
<PNBuildingBloc Name=''RAID''>
 <TransitionPairsList>
  <TransitionPair TransitionID=''T1''
                  OutputServiceTime=''0.2''/>
  <TransitionPair TransitionID=''T12''
                  OutputServiceTime=''0.083333''/>
  <TransitionPair TransitionID=''T2''
                  OutputServiceTime=''0.2''/>
 </TransitionPairsList>
 <PlacesList>
  <Place PlaceID=''P0''>
   <TransitionPairRelated TransitionID=''T1''/>
   <TransitionPairRelated TransitionID=''T12''/>
  </Place>
  ...
</PNBuildingBloc>
...
<ServiceRequest>
  ...
 <BBInputProbability BBID=''RAID''
                     WorkloadName=''C1''
                     TransitionID=''T1''
                     InputProbability=''0.25''/>
 <BBInputProbability BBID=''RAID''
                     WorkloadName=''C1''
                     TransitionID=''T12''
                     InputProbability=''0.5''/>
</ServiceRequest>
```

The full example and the schema is at `dmi.uib.es/~cllado/mifs/`.

3. EQUILIBRIUM SOLUTIONS OF GPMIF SPECIFICATIONS

The steady state probabilities of GPMIF models (when equilibrium exists) may be obtained by direct solution of the underlying Markov process by a standard numerical method, finding a product-form solution if such exists, and by simulation. We have implemented the second and third of these; the first could easily be done using the same specification as that provided to the simulator and software such as Dnamaca [11].

3.1 Product-forms by RCAT

The most general form of the RCAT theorem applies to pairwise synchronisations amongst any finite number of Markov processes [8]. Consider an isolated building block, which is open by definition. Unless there are population constraints at any of the BB's places – e.g. finite capacity – all the inputs to every place are outgoing from every state; this is because in building blocks they come from input transitions which are always enabled. Similarly, there is no restriction on the state from which a BB generates an output; hence every output causes a transition into every state of a building block. Thus RCAT can be applied, and a product-form therefore constructed in a network of BBs, provided the reversed rates of each output transition is the same at every one of its instances. This is not always the case, but the property does hold when the BB is a reversible Markov process. This is determined by the following result.

PROPOSITION 1. *Consider a BB-n, S with n places, and let $\mathcal{N} \subseteq 2^{\{1,\ldots,n\}} \setminus \emptyset$. Let $\rho_y = \lambda_y/\mu_y$ for $T_y, T'_y \in \mathcal{T}, y \in \mathcal{N}, |y| \geq 1$. If the following system of equations has a unique solution ρ_i, $(1 \leq i \leq N)$:*

$$\begin{cases} \rho_y = \prod_{i \in y} \rho_i & \forall y: \ T_y, T'_y \in \mathcal{T} \wedge |y| > 1 \\ \rho_i = \frac{\lambda_i}{\mu_i} & \forall i: \ T_i, T'_i \in \mathcal{T}, 1 \leq i \leq n \end{cases} \quad (1)$$

then the BB's balance equations – and hence equilibrium probabilities when they exist – have product-form solution:

$$\pi(m_1, \ldots, m_n) \propto \prod_{i=1}^{n} \rho_i^{m_i}. \quad (2)$$

PROOF. We show that the BB-n is reversible when at equilibrium. By the symmetry between the input and output transitions in the definition of a BB, there is certainly either no transition between any given pair of states or a transition in each direction. Let the equilibrium probabilities be $\pi(\mathbf{m}) \equiv \pi(m_1, \ldots, m_n)$. Then the detailed balance equations for transitions between states $\pi(\mathbf{m})$ and $\pi(\mathbf{m}')$, where $m'_k = m_k + 1$ for $k \in y$ and $m'_k = m_k$ for $k \notin y$ are:

$$\pi(\mathbf{m})\lambda_y = \pi(\mathbf{m}')\mu_y \quad \text{or} \quad \pi(\mathbf{m})\rho_y = \pi(\mathbf{m}')$$

This equation is satisfied for all $y \in \mathcal{N}$, $T_y \in \mathcal{T}$ and hence for all transition-pairs. The BB is therefore reversible with the equilibrium probabilities stated in the proposition. □

It is well known that M/M/1 queues satisfy the conditions of RCAT when departures are interpreted as active synchronising actions and arrivals as passive actions. Since the M/M/1 queue is reversible, the reversed rate of a departure is simply the arrival rate of its reversed transition, a constant. For the BB, as noted already, the first two conditions of RCAT are satisfied because of the nature of input and output transitions. Similarly the reversed rates are constant (assuming the arrival rates are state-independent) provided that the conditions of Proposition 1 are satisfied, in this case, $\rho_{12} = \rho_1\rho_2$ or $\lambda_1\lambda_2\mu_{12} = \mu_1\mu_2\lambda_{12}$.

3.2 Translation into a SPN

Performance Model interoperability between Queueing Networks and Petri nets is desirable since it can be very useful to compare performance results coming from tools that use different formalisms and the distinct benefits of the tools can be shared and combined. In our case, we can take advantage of the efficiency of product-form solutions on the one hand and be able to solve general models, without the constraints that hamper product-forms, on the other. In [5] a tool that allows for the transformation of a QN specified using PMIF into a Stochastic Petri Net (SPN) is presented. The resulting SPN can be read and solved by PIPE2 (Platform Independent Petri net Editor 2) [4] and TimeNet [14]. The QN → PN tool uses the ATL transformation language to translate from the PMIF schema to a SPN schema (*eD-SPN.xsd*) used by TimeNet and that can also be inported/-exported to/from PIPE2 . This way, we can use both tools dependent on which one is more convenient. We enhance this transformation tool so that it can also transform PMIF models with BBs (we call it eQN->PN, for "enhanced QN->PN tool"). Since the BBs are actually SPNs anyway, the transformation enhancement is mainly syntactic. We can then use the eQN->PN tool to tranform *any* PMIF model with BBs (product-form or not, open or closed) into a SPN model that can be solved by PIPE. If the PN is bounded, we can use a GSPN solver, such as PIPE's, to obtain equilibrium probabilities when they exist [4].

4. MODELLING RAID SYSTEMS

Redundant Arrays of Inexpensive Disks (RAIDs) have been used for cost-efficient storage, increased performance through parallelism and fault tolerance for many years [3].

A RAID subsystem is problematic to represent in a queueing model since it involves a fork-join operation, whereby an arriving task, representing an access request, *forks* into a number of subtasks that each go to a different disk. This is because data is "striped", i.e. divided up into a number of segments that are allocated to different disks in the array, and/or mirrored, i.e. copied to another disk. The subtasks are run asynchronously, perhaps queueing with subtasks from other accesses, and then recombined after all have been served, i.e. are *joined*. Stripes can be of any size (number of segments, or disks used) up to the number of disks in the array, n say. Large accesses will use a number of full stripes of n segments and a partial stripe of size less than n. Smaller requests will just comprise a partial stripe. We assume the partial stripes use a sequence of adjacent disks, with wraparound, starting at any disk with equal probability. Thus, in a BB-n, we need to specify one workload for each possible combination of disks used for each stripe size. There is only one for a full stripe – the compete set of n disks. For partial stripes of size $k < n$, there are n combinations of disks that can be selected, corresponding to the choice of disk for the first segment in the stripe. Hence there are $1 + n(n-1)$ workloads in the PMIF, corresponding to each of these combinations.

A RAID system of n disks is modelled here by a BB-n. This can represent the forking of arrivals into up to n subtasks, which pass to places at which they are served with processor-sharing discipline as in standard Petri nets. Moreover, the building block can also represent the corresponding combining operations. However, it does not faithfully model joins because the subtasks output in parallel are selected randomly from each input place and do not in general correspond to the same task that forked previously. Nevertheless, it is a good approximation at low utilisations – in fact exact in the limit that the occupancy of the places never exceeds 1. Moreover, at equilibrium, on average the number of forks over a long period will be equal to the number of corresponding joins. Obviously a further limitation is that individual disk service times are assumed to be exponential random variables, but this is common to the other servers in a larger system-model; and indeed to many analytical models prevalent in performance engineering. We anticipate reasonable accuracy in the prediction of system measures such as mean place occupancy levels, device utilisations and throughput, but probably poor on user-oriented measures like response time variance and probability distribution.

4.1 Case study

We construct a model of a RAID incorporated into a typical interactive, multi-access, data storage system. As discussed above, the RAID is modelled by a BB-n node and the other devices and service centres are modelled by conventional queues. The model is depicted in Figure 2 and implemented first, for simplicity of explanation, with a BB-2 node. The method is easy to mechanise and it is straightforward to introduce the additional workloads required for $n > 2$.

We have assumed that there are two other dedicated disks, A and B, a CPU and a "think" node, with infinite server discipline, that represents user interaction in a multi-access system (historically this would be a "terminal system"). We solve this model, when equilibrium exists, first by product-form solution and then by translation into a standard Petri net by PIPE; see Figure 3. Notice that the Petri net is unbounded, with infinite state space. In the general case, with a BB-n representation of the RAID, when the total token-population is N, suppose both the input to T_{12} and the output from T'_{12} are suspended. The remaining operational transitions are now equivalent to an irreducible queueing network and so any configuration of the tokens amongst the six places is possible, subject to the total population of N. Moreover, if the input to T_{12} operates once, the population will go up to $N+1$ and repeating the preceding argument shows that any configuration is possible with total population $N+1$. Conversely, if T'_{12} fires once, the population reduces by one. Hence we see inductively that any state $(n_1, \ldots, n_6),\ n_i \geq 0, 1 \leq i \leq 6$ is reachable. This is a complication for simulation or direct solution, which must truncates the state space appropriately, but a great simplification for any product-form solution which does not need to find a normalising constant (or, rather, can determine it easily as the product of the normalising constants of the individual nodes).

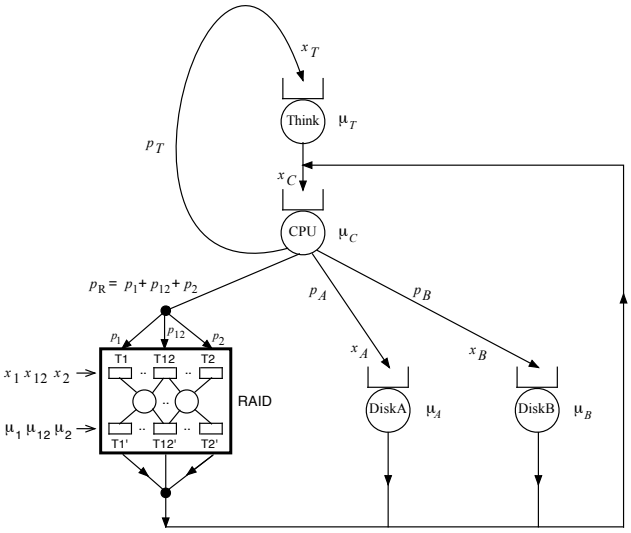

Figure 2: RAID network model.

4.2 Product-form solution

In this model, every active action represents a service completion at a queue or the firing of an output transition in the BB node. In each case, the reversed rate is the arrival rate of the individual queue or the rate of the corresponding input transition. We can therefore write down the rate equations of RCAT as follows, for the case of a BB-2 RAID-node. First note that the total arrival rate to the CPU queue, x_C in Figure 2, is the sum of the passive arrival rates from the RAID BB and the passive arrival rates from DiskA, DiskB and the Think queues.

$$\begin{aligned} x_T &= p_T x_C \\ x_1 &= p_1 x_C \\ x_{12} &= p_{12} x_C \\ x_2 &= p_2 x_C \\ x_A &= p_A x_C \\ x_B &= p_B x_C \\ x_C &= x_T + x_1 + x_{12} + x_2 + x_A + x_B \end{aligned}$$

where the terms $p_\bullet$ are the routing probabilities indicated in Figure 2. This is a homogeneous set of linear equations with a unique solution up to a constant multiplicative factor $\mathbf{x} = (p_T, p_1, p_{12}, p_2, p_A, p_B, 1)x_C$. There is therefore one degree of freedom, since the normalising constant is the product of the individual node-normalising constants. The degree of freedom is used up by the single constraint of the BB-2 node:

$$x_1 x_2 \mu_{12} = \mu_1 \mu_2 x_{12}$$

which implies that

$$x_C = \frac{\mu_1 \mu_2 p_{12}}{\mu_{12} p_1 p_2}$$

The unconditional product-form (when there is equilibrium) is therefore:

$$\pi(n_1, \ldots, n_6)) = e^{-z_1} \prod_{i=2}^{6} (1 - z_i) \frac{1}{n_1!} \prod_{i=1}^{6} z_i^{n_i} \qquad (3)$$

where we rename the variables as $z_1 = p_T x_C/\mu_T$, $z_2 = p_1 x_C/\mu_1$, $z_3 = p_2 x_C/\mu_2$, $z_4 = p_A x_C/\mu_A$, $z_5 = p_B x_C/\mu_B$, $z_6 = x_C/\mu_C$. Finally, the condition for equilibrium to exist is $z_i < 1$ for $2 \leq i \leq 6$, i.e.

$$x_C < \min(\mu_1/p_1, \mu_2/p_2, \mu_A/p_A, \mu_B/p_B, \mu_C)$$

4.2.1 Numerical parameterisation

The parameter values chosen for the model depicted in Figure 2 are as shown in the specification in section 2.1 for the BB-2 node. The complete parameterisation is, for the service rates: $\mu_T = 1/60, \mu_1 = 5, \mu_{12} = 12, \mu_2 = 5, \mu_A = 50, \mu_B = 20, \mu_C = 100$; and for the routing probabilities $p_T = 1/16, p_1 = 1/16, p_{12} = 1/8, p_2 = 1/16, p_A = 7/16, p_B = 1/4$. From these we calculate the mean queue lengths at all the nodes, $m_i, i = 1, \ldots, 6$, the throughput τ and mean response time R for a user. These follow directly from the equilibrium probabilities 3 and Little's result as:

$$m_1 = z_1; m_i = z_i/(1 - z_i), \quad i = 2, \ldots, 6$$

and

$$\tau = z_1 \mu_T = x_C p_T; R = (m_1 + \ldots + m_6)/\tau$$

Numerical results for the performance predicted by this model are reported in section 4.4, where we also look at the changes we get by increasing the rate μ_{12} of the output transition T'_{12} (representing the access time for stripes of size two) from 12 to 16.

4.2.2 Constraints for product-forms with BB-n

Similar RAID systems modelled with BB-n nodes are no more difficult to analyse but there are more RAID inputs – actually $1 + n(n-1)$ of them – to define, as discussed above. Let us assume that the output rates are fixed and try to choose the input probabilities such that the conditions for product-form are satisfied. Since there are $1 + n(n-1)$ inputs, there are $n(n-1)$ independent probabilities and 1 degree of freedom because the network is closed with homogeneous equations for the rates $x_\bullet$. The number of constraints in Proposition 1 is the number of inputs less the number of places, i.e. $1 + n(n-1) - n = (n-1)^2$ in the RAID model, the number of inputs that fork. Subtracting the one degree of freedom, we have $n(n-2)$ further constraints to satisfy, which is n less than the number of independent input probabilities to assign, $n(n-1)$. Therefore, a product-form can be guaranteed by picking the input probabilities appropriately. This observation allows file allocations to be organised so as to achieve product-forms, facilitating simple quantitative analysis.

In particular, since the number of free input probabilities is $n(n-1) - n(n-2) = n$, which is the number of inputs that do not fork, one strategy is to choose the non-forking input probabilities in any way desired, for example proportional to the rates of the corresponding disk drives. The probabilities for the forking tasks are then defined uniquely if the network is to have a product-form. Thus, for $n = 2$ above, there are no further constraints to satisfy and, as we found, the product-form is unconditional; we can choose the two non-forking input probabilities, whereupon the remaining (forking) one is determined. For $n = 3$, the BB-3 has 7 inputs, 4 constraints, 1 degree of freedom and 6 independent input probabilities. Hence a product-form can be found with three of the input probabilities assigned arbitrarily; i.e. we can again choose the non-forking probabilities, after which the other four are determined.

4.3 Direct solution of a Petri net model

The GPMIF xml specification corresponding to the model of Figure 2 (partially written in section 2) is transformed by our eQN->PN tool in another xml file specifying a corresponding Petri net that can be read by PIPE. This net is shown in Figure 3. It is a standard SPN where all timed transitions have single server semantics but the *THINK-Time* transition needs an infinite server semantics in order to model the *Think Device* that corresponds to independently "thinking" users.

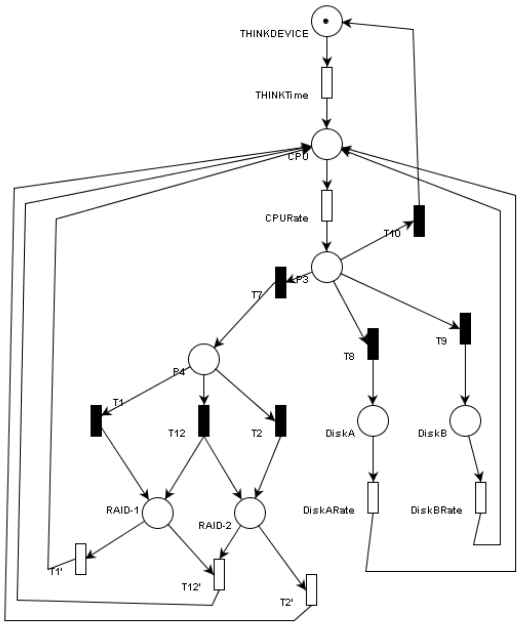

Figure 3: Petri net model produced by PIPE.

4.4 Numerical results

For the model defined in section 4.2.1, we obtained the mean queue lengths, or numbers of tokens at places, shown

in Table 1, as well as an average of 5 tokens in RAID-1 and RAID-2. The non-think node utilisations were 0.833 (RAID-1), 0.833 (RAID-2), 0.583 (DiskA), 0.833 (DiskB), 0.667 (CPU); the throughput was 4.167; the mean network population was 268.4; and the mean response time was 4.416 seconds. As can be seen from the table, these results were confirmed very closely by a discrete event simulation of the PIPE-generated Petri net.

When the service rate of the full stripes (transition T'_{12}) is increased from 12 to 16, the corresponding performance values reduce, as shown in Table 1 – again confirmed closely by the simulation. The mean node occupancies at the RAID-1 and RAID-2 places became 1.667 and the non-think node utilisations were 0.625 (RAID-1), 0.625 (RAID-2), 0.438 (DiskA), 0.625 (DiskB) and 0.5 (CPU). The throughput became 3.125, the mean network population was 194.3 and the mean response time was 2.169 seconds.

Server	**Method**	T'_{12}**=12**	T'_{12}**=16**
Thinking	Product-form	250.0	187.5
	Simulation	249.5	187.3
DiskA	Product-form	1.4	0.778
	Simulation	1.41	0.773
DiskB	Product-form	5.0	1.667
	Simulation	5.01	1.660
CPU	Product-form	2.0	1.0
	Simulation	2.0	0.995

Table 1: Numerical results comparison

5. CONCLUSION

By facilitating sharing of software model specifications, portability and ease of use, PMIFs are enhancing the performance engineering process and making it accessible to the non-specialist. We have made a significant extension to the GPMIF schema so as to allow fork-join sub-models and a subset of Petri nets to be specified and checked automatically for efficient, product-form solutions. The compositional approach, using queues and Petri net "building blocks" is naturally hierarchical and conducive to application of RCAT, which is what provides product-forms when they exist. Otherwise, solutions for general GPMIF model specifications can be obtained by direct solution of the Petri net form of the model, using either numerical solution of its Markov chain or simulation. We have focused on the latter, using the simulation component of Dnamaca, which can also solve for the equilibrium probabilities (when they exist) of the Markov chain when the state space is truncated suitably. The Petri net derived for the RAID system model is unbounded and so requires truncation, so direct analytic solution has been left for future work. Nevertheless, our product-form solutions show that, under the given approximating assumptions, RAID-type fork and join operations can be incorporated efficiently into standard queueing network models.

In the immediate future, we plan to incorporate general BB-n building blocks for any integer n, including the mechanical checking of the conditions for product-form by Proposition 1. This will make our RAID system models more realistic and suitable for testing against data monitored on real systems. Longer term, we intend to apply our approach to more realistic case studies, to find more general building blocks for fork-join that do not require exponential service times for the sub-tasks (e.g. relax these to Erlang or Coxian) and provide a beta-version implementation.

6. REFERENCES

[1] Platform Independent Petri net Editor 2. `http://pipe2.sourceforge.net/`.

[2] Balsamo, S., Harrison, P., and Marin, A. Systematic construction of product-form stochastic petri-nets. Tech. rep., Universita Caï£¡ Foscari Venezia, August 2009.

[3] board, T. R. A. *The RAIDBOOK : A source Book for RAID Technology*. Lino Lakes MN Publisher, June 1993.

[4] Bonet, P., Llado, C., Puijaner, R., and Knottenbelt., W. PIPE v2.5: A Petri net tool for performance modelling. In *Proc. 23rd Latin American Conference on Informatics* (San Jose, Costa Rica, 9-12 October 2007).

[5] Bonet, P., Llado, C., Smith, C. U., and Puijaner, R. Qn->pn. In *Submitted for publication* (2011).

[6] Gelenbe, E. Random neural networks with positive and negative signals and product form solution. *Neural Computation 1*, 4 (1989), 502–510.

[7] Harrison, P. Turning back time in Markovian process algebra. *Theoretical Computer Science 290*, 3 (2003), 1947–1986.

[8] Harrison, P., and Lee, T. Separable equilibrium state probabilities via time reversal in Markovian process algebra. *Theoretical Computer Science* (2005).

[9] Harrison, P., Lladó, C., and Puigjaner, R. A general performance model interchange format. In *Proc. of the First International Conference on Performance Evaluation Methodologies and Tools (Valuetools)* (2006).

[10] Harrison, P., Lladó, C., and Puigjaner, R. A unified approach to modelling the performance of concurrent systems. *Special Issue on "Advances in System Performance Modelling, Analysis and Enhancement", Simulation Modelling Practice and Theory 17*, 9 (October 2009), 1445–1456.

[11] Knottenbelt, W., Harrison, P., and Kritzinger, P. A probabilistic dynamic technique for the distributed generation of very large state spaces. *Performance Evaluation 39* (2000), 127–148.

[12] Smith, C., and Lladó, C. Performance model interchangeformat (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.

[13] Smith, C. U., Lladó, C. M., and Puigjaner, R. Performance Model Interchange Format (PMIF 2): A comprehensive approach to queueing network model interoperability. *Performance Evaluation 67*, 7 (2010), 548 – 568.

[14] Zimmermann, A., and Knoke, M. TimeNET 4.0. a Software Tool for the Performability Evaluation with Stochastic and Coloured Petri Nets. User Manual. Tech. rep., Technische Universität Berlin. Real-Time Systems and Robotics Group, August 2007.