Performance Analysis Tools in Europe

Rob Pooley,
Senior Lecturer,
Department of Computer Science,
University of Edinburgh

Abstract

This report presents a survey of tools for performance evaluation of computer systems and networks which originate from commercial and academic organisations in Europe. After a brief definition of the term *performance analysis* the papare considers the techniques available to someone wishing to perform such an analysis and the tools which can be used to support such work. Techniques are divided into measurement and workload estimation, queueing network modelling, Petri net modelling, use of process algebras and formal protocol languages, simulation, performability and integrated tools and environments. Under each of these headings a small number of the most representative and widely available tools are presented. Where tools offer similar functionality, a brief comparison is offered. As far as possible, some indication of the reasons for using each tool is offered for potential users.

1 Background

The study of the performance of computers attempts to understand and predict the time dependent behaviour of computer systems, including computer networks. It can be broadly divided into two areas - measurement and modelling. These two apparently disjoint approaches are in fact mutually dependent and are both required in any practical study of the performance of a real or planned system. They can be further divided by objective and by technique.The overall process of estimating or predicting the performance of a computer system is often referred to as performance analysis or performance engineering.Recent work has tried to combine the estimation of likely performance of a system with estimates of its reliability, to give a joint measure known as the system's *performability*. Many of the underlying techniques used in studying performability are derived from those used in performance analysis.

Although these topics have been the subject of extensive research and their importance is widely recognised, many practising software and hardware designers are reluctant to use them, because the are perceived as being difficult and time consuming. This need not be true, however, as a number of powerful and use friendly tools are now available to assist in their use. In recent years a major international conference has emerged dealing specifically with performanc etools and the techniques which they embody. The proceedings of the *International Conferences on Modelling Techniques and Tools for Computer Performance Evaluation* [3,14,28], are a primary source for many of the tools described below. Here only those tools currently available for or likely to have a major influence on exploitation of performance evaluation are presented. Many more specialised tools are not here and readers may wish to consult the proceedings cited for further information on these.

In this survey, tools are classified by the techniques they use. A broad division is made into measurement based tools and modelling based tools. Within these categories, various sub-categories are identified.

2 Measurement based tools

All measurement proceeds by monitoring the behaviour of a system under certain conditions of workload which are of interest. There are two main purposes for such observation. It may be desirable to measure the actual workload under certain conditions or it may be desirable to measure the performance of the system under a known workload. A useful survey is provided by [9]

Measuring the workload on a computer system is useful for two reasons. It can give essential information when tuning a system, by revealing bottlenecks or hot spots, where a device or resource is being used at or near its capacity. This may also reveal under-used resources within a system. It is also needed to capture the pattern of demand for a system (its workload) when using modelling to predict how well alternative systems would cope.

Measurement can be done by hardware or by software. Some hardware has built in facilities for counting events, for instance. This information can then be collected by monitoring software and processed. When dealing with traffic on a bus or network link, special hardware monitors are often available. In micro-coded architectures measurement facilities may be provided at that level, with a status bit to enable or disable the actions. Software monitoring is provided at many levels, recording activity at very low levels, such as disk accesses, at intermediate levels, such as operating system calls or line by line monitoring of programs, and at high levels, typically recording application level activity such as database requests. It is often desirable, but always difficult, to map requests at one level onto requests at another.

Benchmarking involves the monitoring of performance measures on a real system while subjecting it to some defined workload. This workload is referred to as a *benchmark*. Typical measures are response time to an editing command or time to satisfy an SQL query. The main problem in benchmarking is supplying the workload to the system. This requires that events be triggered at the required rates and with the required patterns of occurrence. Benchmarks may be interactive or

batch, synthetic or trace based.

Collection of information can be done by recording summary information cumulatively, such as means and standard deviations for measures of interest, but this loses much detailed information which cannot be regenerated. An alternative, which requires large amounts of storage for its data, is the collection of time-stamped traces of events. These can then be post-processed to extract many different kinds of information. Post-processing includes *filtering* those events of interest and discarding or disregarding others. It is also often desirable to try to identify classes of events, grouping them by some characteristic of interest. Thus, local area network traffic is often analysed to discover the relative frequencies of different sizes of packets. Once such classes are identified it may be possible to map them onto higher level events. For instance, large packets are often assumed to come from file transfers, medium sized packets from user requests and small packets from system commands. Although such an approach needs to be used with care, it offers useful insights.

The statistical technique of *cluster analysis* is used to identify classes within an event trace. This can proceed automatically and identify both the number of significantly grouped classes and their relative frequencies or it can be guided, typically by defining the number of classes required. Rare values at the extremes of the range observed are termed *out-liers* and are usually disregarded. Clustering is particularly useful when measuring a workload to be used by a model with classes of customers, since the classes in the workload can be represented within the model. See the sections on queueing networks and stochastic Petri nets below. Identification of classes can also help in tuning, allowing more effective placement of data on discs, for instance.

It is also common to try to fit statistical distributions to monitored data. Standard statistical techniques are used for this, although it may not always be easy. Pre-analysis of the classes within a workload may allow each to be filtered and fitted separately. The derived stochastic workload model is very useful in supporting modelling. Such a *synthetic workload* can be used to generate statistically independent replications for simulation, for instance.

WAT

Within the ESPRIT II IMSE project [27], the Workload Analyser Tool (WAT) was produced by the University of Pavia in collaboration with the Unversity of Milan. Although intended as part of the overall IMSE environment, described in more detail below, a standalone version of WAT was also produced. This originally ran under the IMSE graphical interface system, using Sun View, but is being ported to X Windows.

WAT provides cluster analysis and other statistical analysis and is driven by a graphical user interface. It accepts traces in a number of standard formats and further formats can be added by modifying the input section.

MEDEA

An advanced tool for these purposes is the MEasurements Description Evaluation and Analysis tool (MEDEA), which was produced for the CNR project “Sistemi Informatici e Calcolo Parallelo” [23], also by the University of Pavia. This provides a tool which is portable across systems supporting UNIX, X Windows and Motif. It is operated through a graphical interface and supports the following stages of trace analysis:

1) A data manipulation module performs a preliminary analysis of the trace data to correlate the events recorded during the execution of an application. This leaves the data ready for further analysis.

2) A format manipulation module allows the definition of a format as a subset of performance parameters which are associated with the current workload component. User defined formats can be stored in an internal library, simplifying subsequent analysis by reducing user interactions.

3) A cluster analysis module allows the identification of classes of events with respect to certain parameters. MEDEA uses the *k-means* algorithm for this purpose.

4) A fitting module allows compact analytic descriptions of a workload, which represent the variation of workload parameters with respect to independent variables, such as time. This is important in determining how workload varies during the course of a working day, for example.

5) A functional description module allows a logical, rather than a physical, description of the workload. This supports the view of the workload in terms of the membership of components to a specific cluster, rather than in terms of overall resource utilisations, such as processing time. The proportion of a cluster contributed by a particular operation or application can be determined.

6) A data visualisation module allows interactive examination of the workload models produced. This is often a powerful aid to understanding such models.

MEDEA is a general purpose workload analysis tool. A good case study of its use in analysing the behaviour of a parallel system is contained in [23].

SP

A slightly different approach to deriving workload parameters for models is taken by the SP Tool [25], produced by BNR Europe within the IMSE project. This uses a hierarchical structuring of systems into components and modules, allowing workloads at different levels to be mapped onto each other. The definition of how much work, in terms of processor usage, memory and communication capacities, at one level corresponds to units of work at another, is called a *complexity function.* As well as allowing measurements to be mapped onto required input parameters of performance models, SP can also be used for certain simple direct modelling, such as capacity management decisions.

3 Performance modelling

Modelling can be sub-divided into analytic, numerical and simulation techniques. When we solve a model we can obtain an estimate for a set of values of interest within the system being modelled, for a given set of conditions which we set for that execution. These conditions may be fixed permanently in the model or left as *free variables* or *parameters* of the model, and set at runtime. Each set of m input parameters constitutes a single point in m-dimensional input space. Each solution of the model produces one set of observations. Such a set of n values constitutes a single point in the corresponding n-dimensional observation space. By varying the input conditions we hope to explore how the outputs vary with changes to the inputs.

Clearly this is very similar to the kinds of experiments we might wish to conduct with measurements of a real system, e.g. benchmarking of a computer. The advantages of modelling are several:

- the system may not exist yet, especially when we are using simulation as a design aid;
- the system may be too expensive to buy simply to monitor, for instance to choose which system to purchase;
- the workloads we want to use may not be easy to generate in a reproducible manner or even at all on the real system;
- the monitoring facilities on the real system may not be adequate for our purposes;
- we may want to test the system under conditions which would cause damage to the real system.

Simulation is the most general and versatile means of modelling systems for performance estimation. It has many uses, but, since it is essentially using statistical sampling of a series of observations, its results are usually only approximations to the exact answer and the price of increased accuracy is much longer execution times.

Analytical techniques provide general models which may be solved symbolically for the steady

atate measures of that system and so used to efficiently explore ranges of parameters. Unfortunately only a very restricted set of models have such solutions. Even fewer have exact solutions, leading to the necessity of finding approximation techniques. Analytical techniques are usually applied to queueing networks, where the structure of the network allows good rules for finding appropriate models, notably those in the class known as BCMP networks [2].

Somewhere between analytical and simulation models it is possible to use numerical techniques, where the steady stae behaviour of a system is found without detailed simulation, but only in terms of a given set of parameters. Numerical techniques vary in their efficiency and their accuracy. They are still only applicable to a restricted class of models (though not as restricted as analytic approaches). Many approaches increase rapidly in their memory and time requirements as the size of the model increases.

4 Queueing networks

The search for analytically and numerically tractable models has found useful results from networks of servers and queues.These can be divided into two main groups, known as *product form* and non-product form. Product form networks have the property that they can be regarded as independently operating queues, where steady state can be expressed as both a set of global balance equations on customer flow in the whole network and a set of local balance equations on each queue. Local flow balance says that the mean number of customers entering any queue from all others must equal the number leaving it to go to all others, including customers which leave and rejoin the same queue immediately.

It is possible to consider networks with different classes of customers, which are treated differently by servers according to their class. Such networks can be broken into a set of *routing chains*, defining the probability that a job in one node and with one class will next move to another particular node and take on another particular class. This leads to the more general formulation of product form below.

There is no guarantee that a particular problem will yield a product form solution. We may therefore need to find some other approach. This usually means some numerical technique. It may, however, be possible to model a good enough approximation analytically.

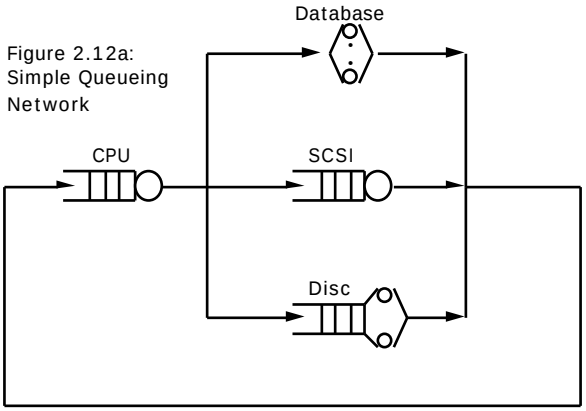
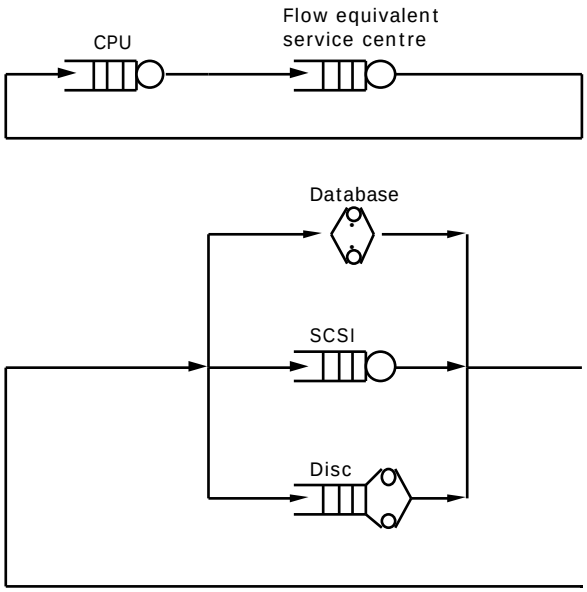


Figure 2.12a: Simple Queueing Network

Figure 2.12b: Decomposed queueing network



In some cases the approach is to represent the underlying stochastic model, known as a Markov chain, directly. This involves generating all the states of the system and then manipulating the resulting matrices. It can cause major space and time complexity problems. There are many numerical methods for dealing with queueing networks. These nearly all work on the transition rate matrix of the underlying Markov process.

The main constraints on product form networks concern independence (i.e. the Markovian property of the overall network). This makes it impossible to deal with the need for individual customers leaving one server to have to arrive in the same order at another one. Thus, many communication protocols are not able to be represented as product form networks. Queueing theory also gives only limited insight into transient behaviour, as it is usually used under steady state assumptions.

4.1 Tools for solving queueing networks

The most important commercial European queueing network based tools come from Simulog AS, a French company formed originally as a cooperation between the research insitute INRIA and the commercial company Bull. Their products, QNAP and MODLINE, are built on the basis that a single method of describing a queuieng network can be used to drive a variety of backend solvers, exploiting the various solution techniques suitable for that model. This question of suitability of a model depends on the structure and size of the network. The smaller and more restricted the model, the greater the choice of solvers. Another interesting tool based on queueing networks is MACOM,

which was developed at Universität Dortmund for the analysis of communication networks.

QNAP

QNAP II [32] was the first major package from Simulog. It uses a high level textual language for description of models. This language is compiled into a form solvable by the range of solvers offered. These include, in order of decreasing restrictiveness, exact solvers, for BCMP networks, a number of efficient numerical solvers, for less restricted models, a Markovian solver, for reasonable sized models preserving Markovian assumptions, and a simulation solver for any models describable in the QNAP language. Simulation is frequently the only method usable for realistic models, but quick, first cut answers may often be found from simpler models. Recent enhancements to QNAp have included approximate methods for solving fork/join models and other models with specific dependencies. Only simulation can deal with non-Markovian distributions for random variables in models.

The QNAP language is structured around entities called service centres. These may be simple server nodes, as in classical queueing networks, or may have more complex behaviours, described in an algorithmic language. Such a structuring into objects has proved very useful in developing more user friendly tools.

The QNAP language also has a control part, which allows the use of the model to be described. An obvious option here is to request that a particular solver be tried. If QNAP is unable to solve a model with the specified method it may either automatically resort to a more general approach or request resolution with a different choice. Options for tracing and debugging simulations are also supported in current versions of QNAP.

MODLINE

Although QNAP is still available on its own, it is now part of Simulog's MODLINE modelling tool. This incorporates several features developed during the ESPRIT II IMSE project [27]. Most obvious is the addition of a graphical user interface, allowing models to be built from a menu of symbols as a queueing network diagram. Given the structure of the QNAP language, it is straightforward to see how such a representation provides similar descriptive power. Nodes can be parameterised to define a complete model and service centre nodes can have QNAP code associated to provide general descriptions of behaviour.

A second novel feature, developed from work done at the University of Edinburgh during IMSE, is an experiment description facility. This allows modellers to describe repeated runs of a model, with varying parameters, and collects outputs from each in a systematic manner. This greatly extends the control features of QNAP.

MODLINE also has features for animation of simulation execution, for selective instrumentation of models to avoid generation of unwanted results and for display of outputs in various ways.

MACOM

MACOM [21] was developed to support the Markovian Analysis of COMmunication systems. Its model view consists of sources, sinks and service and control elements. It is particularly specialised towards the class of wait-loss-systems, which are commonly applied to communications modelling. There are many other types of system which can be described by this form of model and it is not restricted to those systems for which it was developed.

In an analogous manner to the MODLINE experimentation facilities, MACOM allows desired measures and derived statistics and input parameters (including experiment series) to be defined as *evaluation descriptions*. Evaluation descriptions, models and experiments (sets of runs and their results) are maintained by the *environment manager*, which also launches runs of models and ensures consistency of related elements.

Models and evaluation descriptions are constructed using a graphical user interface, again analogous to that of MODLINE. The environment manager uses a window based interface, which presents menus of elements under headings, Models, Evaluations and Experiments. Buttons control creation and delation of elements and execution of runs.

MACOM solves models by efficient numerical evaluation of the underlying Markov chain. Systems with up to half a million states have been solved in this way, using current generation SUN workstations. Its graphical interface is supported under both Sun View and X Windows.

5 Stochastic Petri Nets (SPNs)

As an alternative to queueing networks, the second half of the 1980s saw growing interest in using Petri nets to model performance. This passed through timed and stochastic variants of Carl Petri's original nets. Their former purpose had been to express and analyse the behavioural properties of systems, particularly those with concurrent events.

A Petri net is essentially an extension of a finite state automaton, to allow several concurrent threads of activity to be described in one representation, by means of tokens. It is essentially a graphical description, being a directed graph with its edges defining paths for the evolution of a system's behaviour and its nodes or vertices being of two sorts, *places* and *transitions.* All incoming edges to a place must come from a transition and vice versa. *Tokens* are held in places and when all the input places to a transition are *marked,* i.e. have at lest one token, that transition is *enabled* and *fires,* depositing a token in each of its output places.

There are a number of extensions to these simple *place/transition nets,* mostly to increase the ease of describing complex systems. The most widely used is to define *multiplicities* for the edges, which define how many tokens flow down an arc simultaneously. This is a shorthand for an equivalent number of edges linking the same pair of vertices.

The use of Petri nets in performance modelling now centres on the Generalised Stochastic Petri Nets (GSPNs) defined by Ajmone Marsan and others[1]. These incorporate the following extensions:

- *multiplicities* on arcs;
- *timed transitions*, where firing delays (usually restricted to be exponentially distributed) between the enabling of a timed transition and its firing;
- *immediate transitions,* which fire instantly but where a choice of output arcs may be represented, in a similar manner to branching probabilities in queueing networks;
- *deterministic firing rates* are sometimes allowed, restricted so that only one such transition is enabled at one time;
- *inhibiting arcs* which prevent a transition from firing as long as a connected place is marked; these are particularly useful in describing blocking;
- *colouring of places and tokens* in most tools based on this approach; *coloured tokens are* used to represent classes of tokens in a similar way to classes in BCMP queueing networks; *coloured places* allow folding of symmetrical subnets for a more compact representation and, for a subclass known as *well formed Petri nets*, more efficient solution, based on aggregation of states.

GSPNs are useful since they allow many of the structural and behavioural properties of a net to be examined, as well as its performance to be calculated. They are usually solved by numerical techniques or by simulation, but recently Henderson has defined a set of restricted nets with product form solutions. It is not yet clear how useful this will be in practice. Methods which allow much larger Markov models to be solved efficiently also offer the prospect of GSPNs becoming

more useful.

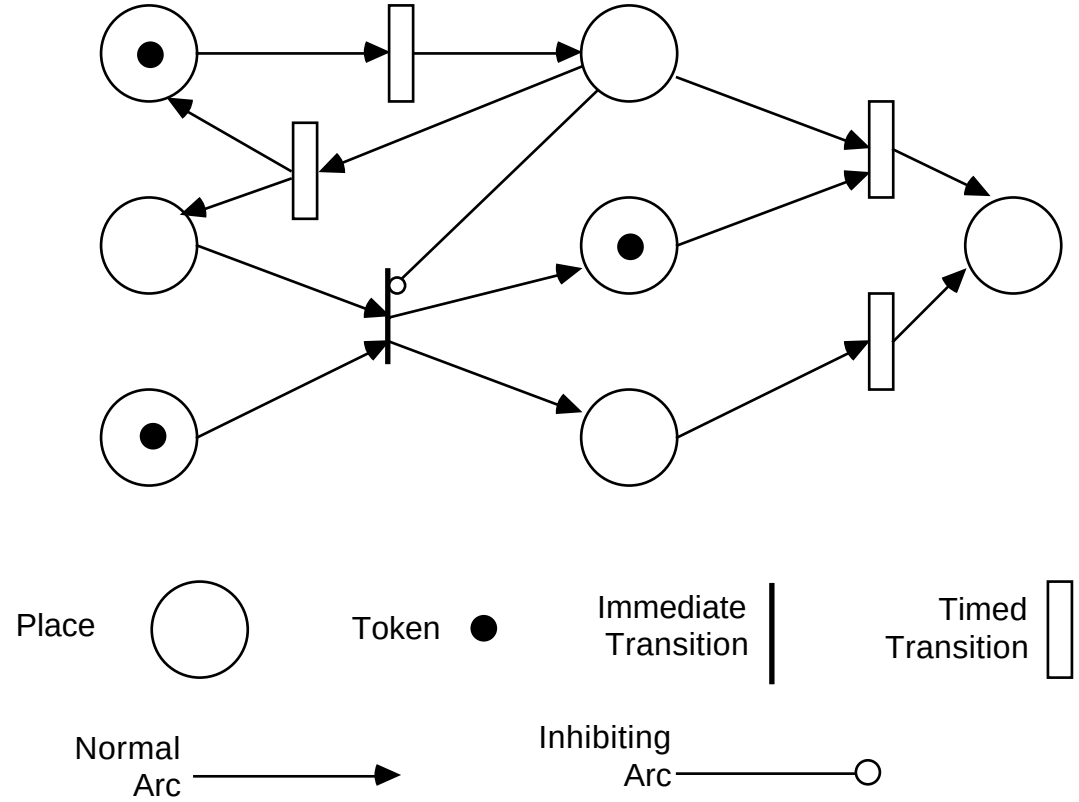


Figure 2: GSPN

5.1 Tools for GSPNs

There are a number of interesting tools based on GSPNs and their variants. Those developed in Europe all offer a graphical interface for construction of the model and a number of solvers for different classes of nets. None offer solvers for product form nets at the moment.

GreatSPN

The earliest of the European tools for Petri nets is GreatSPN, from Università di Torino [11]. This has evolved considerably from a fairly simple tool for graphical construction and numerical solution of GSPNs. Originally built to run under Sun View, it has now been ported to X Windows, using X View. It runs on all UNIX platforms supporting this. It is still being enhanced, but the following is a fairly complete summary of the facilities offered.

Model construction is supported by placing and linking icons from a menu, representing places, transitions and arcs. These must be given suitable values for delays, probabilities etc. by typing into menus which appear when icons are opened by clicking.

The resulting net may be analysed for structural and behavioural properties, such as deadlocks and invariants, by selecting the appropriate analysis from a menu.

The model may also be solved, in many cases, by efficient numerical techniques based on generating the underlying Markov chain. Both steady state and transient analysis is supported. Much current research and development for GreatSPN centres on improving such solution techniques and allowing solution of larger, more complex models. Only models where all transitions have exponential random firing times can be solved in this way in the current version of GreatSPN. Where two transitions are enabled simultaneously, they are assumed to model a race

condition. Recent versions of GreatSPN implement well formed coloured Petri nets as a means of exploiting sub-nets suitable for more efficient solution by folding and aggregation techniques.

A discrete event simulation backend may be used to solve the model or to step through its execution, displaying the movement of tokens at each firing. The latter, interactive execution, often referred to as the *token game*, allows exploration of the detailed behaviour of the model to answer questions about why certain events occur. Where simulation is used to solve a model, timings are not restricted to being exponentially distributed.

DSPN Express

DSPN Express [22] is based directly on the user interface and some of the solution techniques of GreatSPN. It comes from the Technical University of Berlin. Unlike GreatSPN, it was able to use X Windows from the start and will run on any UNIX system with X Windows.

As well as offering most of the non-simulation features of GreatSPN, DSPN Express allows numerical solution of models incorporating deterministic (constant) time delays in transitions. This is restricted to models where only one such deterministic transition can be enabled at any time, however, which is rather restrictive in many realistic systems. Deterministic delays may also be defined to depend on the current marking. This is potentially very useful in reducing the complexity of a model's description.

DSPN Express does not support simulation. It uses different algorithms to those in Great SPN for certain kinds of solution, which may lead to faster solution of some models.

QPN Tool

An interesting variation on Petri net modelling comes in the implementation of Queueing Petri Nets (QPNs) at Universität Dortmund [5]. The resulting QPN Tool is again similar in its general appearance to GreatSPN. It differs in supporting *timed places*, as well as timed transitions. A timed place corresponds to a service station of a queueing network, for which a Petri net equivalent is known. Thus, in solving a QPN such places can be expanded into the equivalent Petri sub-net with the overall model. In practice, it is actually the corresponding Markov chain that needs to be generated.

Solution of the underlying Markov chain is performed with Usenum, a package for efficient solution of large Markov chains, also developed at Universität Dortmund.

It is argued that QPNs provide a more compact and intuitively understood means of describing complex systems. Whether they will replace GSPNs, of which they are a superset, remains to be seen.

6 Performance Process Algebras and formal protocol languages

Although still the subject of research, process algebras, which evolved to address some of the shortcomings of simple Petri nets for behavioural analysis, are now being extended in a similar way to GSPNs and their use for performance analysis is being tested. Since they are inherently compositional, unlike Petri nets, there are some grounds for believing that they may become a very useful tool. In a similar manner, formal protocol languages and system description languages, which have often been heavily influenced by process algebras, are being investigated as a measn for providing performance models directly from system descriptions and specifications. No generally available tools exist yet for these purposes, but some early experiments are worth reporting. Examples of process algebras for performance modelling include TIPP, from Universität Erlangen-Nürnberg, and PEPA, from the University of Edinburgh.

TIPP

Just as GSPNs evolved from place transition Petri nets, so stochastic extensions to process

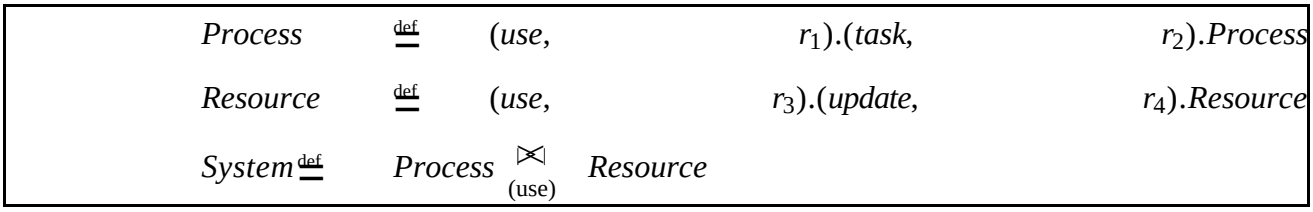


Figure 3: PEPA Model

algebras and new process algebras with stochastic and timing features are being developed. The first serious attempt at a performance modelling extension to a process algebra was TIPP, from Universität Erlangen-Nürnberg [13]. TIPP is similar in its algebraic notation to Milner's Calculus of Communicating Systems (CCS). It has demonstrated the power of such a language in expressing models outside those easily dealt with by previous performance modelling formalisms and also the potential for solving such models by numerical techniques.

PEPA Workbench

Performance Estimation Process Algebra (PEPA) [12] is also similar to CCS in its algebraic structure. Like CCS it uses the notion of bisimulation proofs of observational equivalence to understand whether systems are behaving identically. It adds to these behavioural analysis capabilities, the ability to generate directly a Markov chain from the state transition model underlying the algebraic description, using exponential rates associated with activities.

Unlike Petri nets, such models contain a built in notion of decomposition into sub-models, which is directly expressed in the algebra. Indeed models are built by writing sub-models as *agents* which are then combined by to well defined algebraic operators. Replicated instances of sub-models as components are expressed directly. This can be used to reduce the state space of a model and to find sub-models suitable for use as aggregable sub-models in efficient solution of the Markov chain.

The PEPA workbench allows models written in PEPA to be entered and their underlying Markov chain to be generated in a form suitable for processing by a backend written using the Maple computer algebra package.

6.1 Generating performance models from formal specifications

A number of groups are experimenting with the automatic generation of performance models from annotated formal specifications. Although these are still at the research stage, they have shown some promise for usable tools in the near future.

LOTOS is the CCITT recommended protocol specification language. It is based on the principles of process algebras, combining features of CCS and Hoare's Communicating Sequential Processes (CSP). One approach to using this for performance modelling is Stochastic LOTOS [30], which modifies the semantics of LOTOS to allow performance information to be represented and performance results computed. Although useful results have been demonstrated, the formal power of the specification language is lost. Within the ESPRIT project COMPLEMENT, work has been done in generating QNAP models from annotated LOTOS specifications [31]. These start from standard LOTOS specifications, where the formal semantics can be used to prove behavioural properties and add teh performance annotations before generating the QNAP code. Rules for this derivation are proposed and results shown for simulation of generated QNAP models.

In a similar approach, researchers at CSELT have extended the PROMELA specification language with performance annotations, giving PROMELA+ [10]. A simulation tool for this extrended language has been constructed and tested for some examples. It is claimed that PROMELA+ will allow automated reasoning about behaviour and accurate simulation for performance measures.

SDL is another widely used specification language. At Universität Erlangen-Nürnberg queueing network analysis of SDL models was shown to be feasible. Work at BNR Europe [20], showed that it was possible to generate simulation models from SDL specifications, in a form suitable for solution by the commercial simulation package, SES Workbench®, a product of Scientific and Engineering Software, Inc. This i sproposed as the basis for a toolset similar in purpose to those described above.

Work on the use of SDL as a basis for integrating behavioural and performance modelling is also underway at Universität Dortmund [4,17,18].

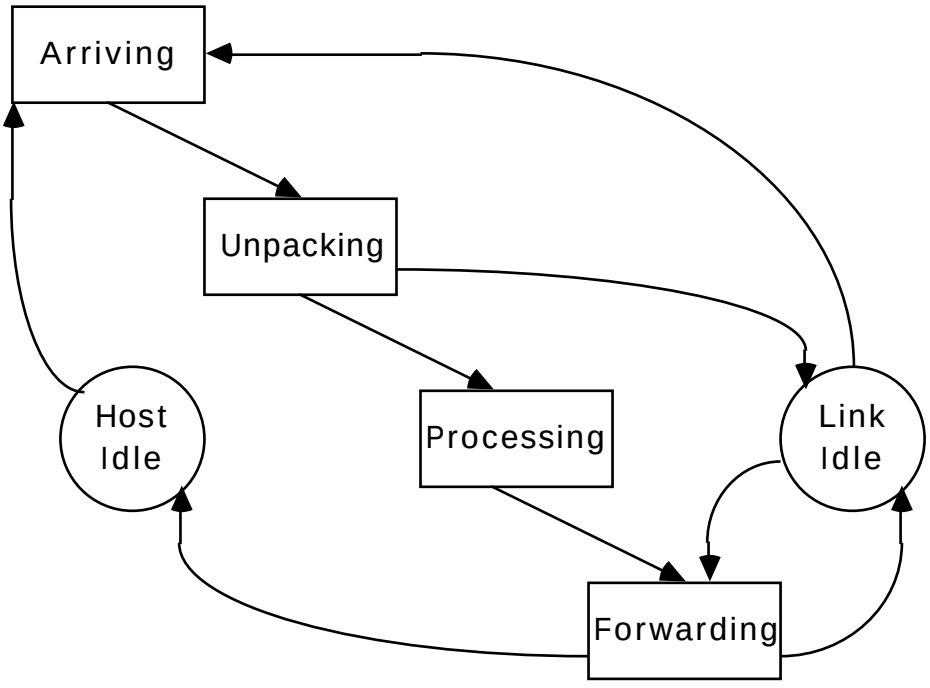
7 Simulation

As the most general usable, but most computationally expensive performance modelling technique, simulation is sometimes rather taken for granted and little very new software appears in some periods. It was noted above that simulation was one of the solution techniques used in many of the tools and toolsets described. It is particularly important in widening the classes of queueing networks and Petri nets which can be solved, particularly removing the need for exponential delay restrictions. Here we note a few of the more interesting simulation only packages which are current in Europe. There are many others which differ little from those described.

HOCUS

There are a number of different views which are commonly used in discrete event simulation modelling. *Activity scanning* centres on the definition of the activities in a model. Entities are assumed to flow through the model waiting for other entities before engaging in activities in a certain order. Again a number of languages exist which support this approach. Below is an example of a model which is defined for the HOCUS™ package, marketed by PE Consultants.

Figure 4: Hocus diagram of a simple network model

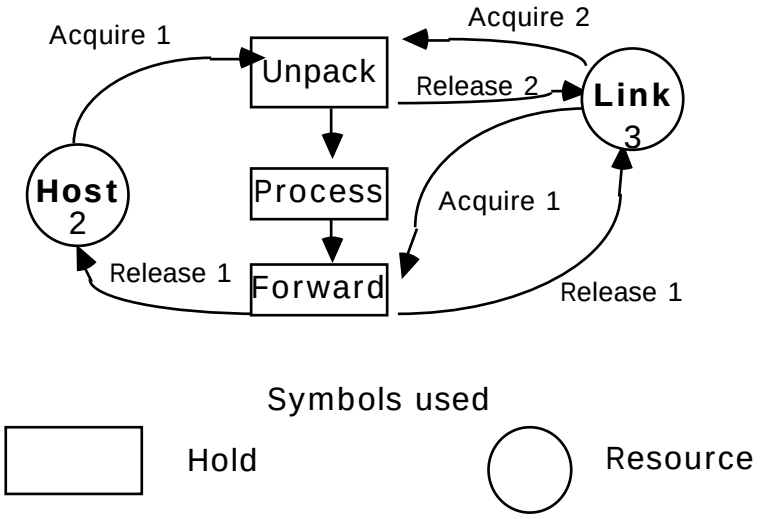


SIMULA and DEMOS

In the process view the model is defined as a set of active components called processes or entities and a set of passive resources which control the activities of the active processes. This is a widely used means of modelling computer systems and networks. The earliest version of this approach

was the transactions view of GPSS, but the classic implementation is Discrete Event Modelling on SIMULA (DEMOS), produced by Graham Birtwistle at the University of Bradford and extended by Henk Sol of Delft University [8].

Figure 5: DEMOS activity diagram of the same simple model



Since DEMOS has a graphical description language, it has been straightforward to add a graphical input interface to it. A number of versions of such tools have been attempted over recent years. Most recently the Demographer front end has been produced at the University of Edinburgh [29]. This has versions under MS/DOS and X Windows/UNIX. Both are written in SIMULA.

As well as conventional DEMOS modelling, Demographer supports hierarchical modelling with sub-processes, based on extended activity diagrams. This allows component based modelling to arbitrary depths. Demographer is also able to generate CCS process algebra models from activity diagrams. These can be used for formal reasoning about the properties of models, such as deadlock freedom.

8 Performability

Performability attempts to quantify not just the behaviour of a system while it is running, but the overall work supplied by a system when its likely down time is estimated. This means that a *reliability* model is first solved, to estimate what percentages of the time various levels of service are likely to be delivered, given the likelihood of breakdown of various parts of the system. For each such level of service, a *performance* estimate is then made, typically by solving a queueing network or simulation model. These performance estimates are then used to weight the percentages of time for which they are expected to hold and the sum of these weighted times is the performability measure. Since the reliability model is typically a markovian model (often a Stochastic Petri Net model) and the performance to be expected from a particular level of service is a form of reward for achieving that service, such models are known as Markov reward models.

DyNQNTool

It is clearly possible to use two separate models to achieve an estimate of performability, but the DyNQNTool [16] offers a combined approach. It has embedded within it both a Petri net based reliability modelling tool and a queueing network performance modelling tool. In fact it uses the tools SPNP and Sharpe from Duke University for these purposes.

SURF-2

SURF-2 [24] is a tool for performability analysis from LAAS-CNRS in France which allows either GSPNs or Markov chains to be used to describe a system. Both are entered graphically. GSPN

models can be analysed for structural properties. Models may be aprameterised by using symbolic values, which must be assigned actual values before solution.

SURF-2 uses a database to maintain associated models and parameter assignments. These are grouped into model folders, which also hold GSPN analysis objects, with the underlying marking graph and Markov chain generated for a GSPN model, and result objects. All objects can be displayed or turned into Postscript files for printing.

9 Integrated performance environments

With such a wide diversity of tools and techniques, potential users are often confused as to which best suit their needs. To combat this, as has been shown, some tools are aiming at providing easy routes from non-performance descriptions of systems to performance models of these. others, notably QNAP/MODLINE, offer a range of solvers behind a single modelling interface. One final group of tools try to combine these benefits in *integrated modelling environments*.

HIT

HIT [6,7] is specifically built to support modelling of computer systems in a hierarchical manner, based on a layered machine view of such systems. This allows modularisation of models, corresponding to components within layers of the real system. Such a view gives a form of description which is very natural for the types of systems considered.

The user interface to HIT uses either the textual language, HI-SLANG, or a graphical model construction interface, HITGRAPHIC. In either form, modules at a higher level are mapped onto services from modules at lower levels. At the lowest level simple services are described. Modules, termed COMPONENTS, are described as LOADs applied to MACHINEs.

Written entirely in SIMULA, HIT can generate automatically models for solution by discrete event simulation, exact solution as product form networks and approximate solution for other classes of network, numerical solution of underlying Markov chains and structured decomposition an daggregation of large models for efficient solution.

HIT runs on most platforms supporting a SIMULA compiler, including most UNIX workstations and IBM and Siemens mainframes. HITGRAPHIC is written in C and runs on top of X Windows. It was developed at Universität Dortmund with initial support from Nixdorf Computer AG and BMFT.

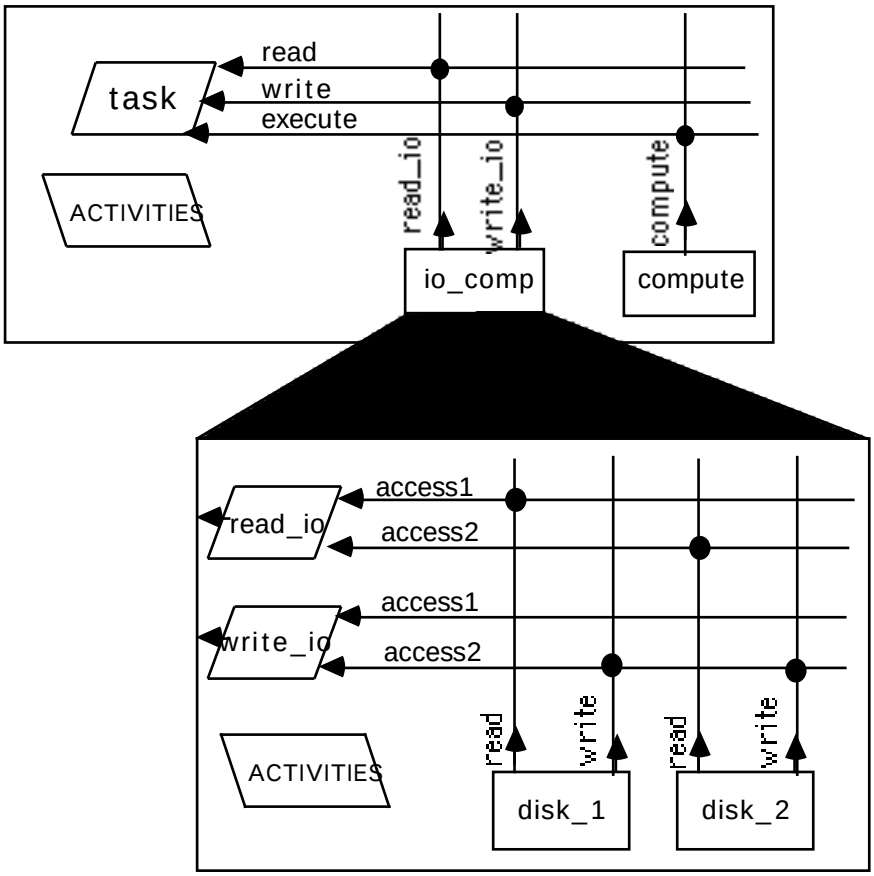


Figure 6: HIT Graphic Model

IMSE

The ESPRIT II project *Integrated Modelling Support Environment* [27] brought together most of the leading performance tool researchers in Europe, together with the companies Simulog, Thomson and BNR Europe (formerly STC). Although the environment produced within IMSE has not itself become a commercial product, its influence on many other tools has been profound. In particular it demonstrated the enormous potential of integration of individual tools, by a *shared graphical user interface*, a *shared object store/browser* and facilities for creating and executing modelling studies as *experiments*.

9 Prospects for the Future

The future of performance modelling tools must lie in increased integration of tools and techniques. No single method or form of description is able to provide all the functionality needed by end users of these tools. Designers are not likely to take the trouble to learn several different tools and their principles, especially as they are also being pressed to learn others for functional analysis and yet others for project management. With wide availability of object oriented databases, such integration is likely to become efficient and pleasant to use. For modellers it will lift the focus of attention from model construction to experimentation, where end users can share in quantitative modelling.

Graphical interfaces are expected for most tools, but they have limitations in expressing complex models. Hierarchical modelling is going to be essential in managing the complexity of models. It may also provide a way of making solution of larger models more tractable.

Finally, performance is unlikely to remain the separate subject it has traditionally been. The merging of performance and functional modelling is already feasible. With better understanding of these techniques, users will be bale to exploit a single formalism when expressing designs and ask

questions about their overall behaviour. Performance modelling will give way to total modelling.

References

1. M. Ajmone Marsan, G. Conti and G. Balbo “A Class of Generalised Stochastic Petri Nets for the Performance Evaluation of Multiprocessor Systems”, ACM TOCS, 2(2) May 1984, pp 93-122

2. A.O. Allen *Probability, Statistics and Queueing Theory*, Second Edition, Academic Press, 1993

3. G. Balbo and G. Serazzi Eds *Computer Performance Evaluation - Modelling Techniques and Tools*, 5th International Conference on Modelling Techniques and Tools for Computer Performance Evaluation, Torino, Elsevier, 1992

4. F. Bause and P. Buchholz “Protocol Analysis Using a Timed Version of SDL”, in *Proceedings of the 3rd International Conference on Formal Description Techniques (FORTE '90)*, Madrid, Springer 1991

5. F. Bause and P. Kemper “QPNTool for Qualitative and Quantitative Analysis of Queueing Petri Nets”, in [13] pp 321-334

6. H. Beilner and F.J. Stewing “Concepts and Techniques of the Performance Modelling Tool HIT”, in *Proceedings of the European Simulation Multiconference*, Vienna, SCS Europe, 1987

7. H. Beilner, J. Mäter and C. Wysocki “The Hierarchical Evaluation Tool HIT”, in [14] pp 3-6

8. G. M. Birtwistle, *Discrete Event Modelling on SIMULA*, Macmillan, 1979

9. M. Calzarossa and G. Serazzi, “Workload Characterisation: A Survey”, Proceedings of the IEEE, 81(8), August 1993, pp 1136-1150

10. E. Chiocchetti, R. Manione and P. Renditore “Specification Based Performance Evaluation of Distributed Systems for Telecommunications”, in [14], pp 47-50

11. G. Chiola. “A Graphical Petri Net Tool for Performance Analysis”, in D. Potier Ed. *Proceedings of the International Workshop on Modelling Techniques and Tools for Performance Evaluation*, March 1987, pp 297-307, AFCET, Paris

12. S. Gilmore and J. Hillston “The PEPA Workbench: a Tool to Support a Process Algebra-based Approach to Performance Modelling”, in [13] pp 351-368

13. N. Götz, U. Herzog and M. Rettelbach “Multiprocessor and Distributed System Design: the Integration of Functional Specification and Performance Analysis using Stochastic Process Algebras”, in *Proceedings of Performance ’93*, 1993

14. G. Haring and G. Kotsis Eds. *Computer Performance Evaluation - Modelling Techniques and Tools*, 7th International Conference on Modelling Techniques and Tools for Computer Performance Evaluation, Vienna, LNCS 794, Springer-Verlag, May 1994

15. G. Haring and H. Wabnig Eds. *Short Papers and Tool Descriptions*, 7th International Conference on Modelling Techniques and Tools for Computer Performance Evaluation, Universität Wien, Institut für Angewandte Informatik und Informationssysteme, May 1994

16. B. Haverkort, I.G. Niemegeers and P. Veldhuyzen van Zanten “DyQNTool - a Performability Modelling Tool Based on the Dynamic Queueing Network Concept”, in [3] pp181-196

17. E. Heck, D. Hogrefe and B. Müller-Clostermann “Hierarchical Performance Evaluation Based on Formally Specified Communication Protocols”, IEEE Trans. Comp. Vol. 40 No 4, 1991, pp 500-513

18. E. Heck “SDL-HIT Integration” in Konlof ed. *Method Integration: Concepts and Case Studies*,

Wiley 1992

19. J. Hillston “A Tool to Enhance Model Exploitation” in [25], pp 131-142

20. S.R. Kershaw, N.M. Macintyre and N. Xenios “A Toolset to Support the Performance Evaluation of Systems Described in SDL”, in [14], pp 23-26

21. U.R. Krieger, B. Müller-Clostermann and M. Sczittnick “Modelling and Analysis of Communication Systems Based on Computational Methods for Markov Chains”, IEEE Journal on Selected Areas in Communications, Vol. 8 No 9, pp 456-470, 1990

22. C. Lindemann “DSPNExpress: a Software Package for the Efficient Solution of Deterministic and Stochastic Petri Nets”, in [25] pp 9-20

23. A. Merlo and P. Worley “Analysing PICL Trace Data with MEDEA”, in [13] pp 445-464

24. S. Metge “Modelling and Evaluation of Hardware and Software Systems Dependability Using SURF-2” in [14], pp 19-22

25. C. Minkowitz, V. Vetland and P.H. Hughes “A Modular Approach to System Structure and Specification” in [14], pp 83-86

26. M. Paterok, R. Heller and H. de Meer “Performance Evaluation of an SDL Runtime System - a Case Study”, in [3] pp 89-104

27. R. Pooley “The Integrated Modelling Support Environment”, in [3], pp 1-16.

28. R. Pooley and J. Hillston Eds. *Computer Performance Evaluation - Modelling Techniques and Tools*, 6th International Conference on Modelling Techniques and Tools for Computer Performance Evaluation, Edinburgh, Edits 10, Edinburgh University Press, September 1992

29. R. Pooley “Demographer: A Graphical Tool for Combined Simulation and Functional Modelling”, in R. Pooley and R. Zobel Eds, *UKSS ‘93: Proceedings of the First Conference of the UK Simulation Society*, September 1993, pp 91-95

30. N. Rico and G.v. Bochmann, “Performance Description and Analysis for Distributed Systems using a Variant of LOTOS”, 10th International IFIP Symposium on Protocol Specification, Testing and Validation, July 1990

31. A. Valderruten, O. Hjiej, A. Benzekri and D. Gazal “Deriving Queueing Networks Performance Models from Annotated LOTOS Specifications”, in [25] pp 120-130

32. M. Veran and D. Potier “QNAP 2: a Portable Environment for Queueing System Modelling” in D. Potier Ed. Proceedings of Modelling Techniques and Tools for Computer Performance Evaluation, North Holland, 1985, pp 25-63