SOFTWARE TOOLS TO SUPPORT A PERFORMANCE MODELLING ENVIRONMENT

A F Pais, BSc, MBCS

Head of Software Standards & Tools Group
Performance Engineering Division
British Telecom Research Laboratories
Martlesham Heath
Ipswich, Suffolk, U.K. IP5 7RE

SUBJECT AREA

3 Basic Theories and Models:
3.1 General Teletraffic Models
3.2 Queueing Theory
3.7 Statistical Methods
3.8 Simulation Methods

A performance modelling software tool set under development in British Telecom's Performance Engineering Division is described.

The main features of the tool set are the provision of well established analytical algorithms and techniques, along with powerful facilities for simulation including automatic code generation from model descriptions. The software may be ported to a variety of hardware ranging from small workstations to powerful main-frame computers. Associated with the tool set is an expanding database whereby validated models may be stored for reuse or for the derivation of new models by synthesis or decomposition.

1. INTRODUCTION

The cost of performance modelling is small compared with total system costs, whilst the penalty for adequate system performance analysis can be very great. However, in spite of this situation, system designers are often compelled to "go it alone" because the performance analysts are unable to react fast enough to design proposals. Performance engineering tools are required to speed up the production of the quantitative performance predictions essential in the design of cost effective systems.

The pressing need for enhanced performance modelling facilities has been cited by a number of researchers around the world [3,4]. In the UK this led to a collaborative BT/ICL ALVEY funded project called SIMMER [1,2], an acronym for SIMulation MEthodology Research. This collaboration, which lasted about a year, proved very valuable and set the scene for further work in each company. The follow-on work within BT has led to the development of the prototype performance engineering tool set described in this paper.

The initial objective of the development is to provide powerful, portable, and extendible software tools to facilitate performance engineering consultancy work. In the longer term, the software will be developed into a performance engineering expert system, associated with a knowledge-base of validated models and will attract users outside the realm of performance specialists. The tool set, besides stimulating and facilitating the use of performance modelling in system design, will provide a valuable performance engineering training capability.
The main features of the tool set are the provision of well established analytical algorithms complemented by comprehensive simulation facilities, which include a capability to generate code from a model description. Facilities for modelling the queueing networks used for computer sizing and capacity planning are also available. The software can be implemented on a variety of hardware ranging from small workstations to powerful main-frames, with access via a simple VT100 terminal using a menu interface.

2. REQUIREMENTS FOR A PERFORMANCE ENGINEERING TOOL SET

The requirements for a performance engineering tool set were specified after extensive surveys and literature searches [3,4] to establish what tools and techniques were already available and how these could best be brought together and expanded upon. These surveys were supplemented by discussions within the Performance Engineering Division. The considerable breadth and depth of experience accumulated within the Division provided a unique opportunity to obtain informed opinions on the tools required to improve the effectiveness of a wide range of teletraffic and performance engineering activities.

The key requirement was identified as the need to provide the tools necessary to enable a fast and flexible approach to performance modelling to be applied throughout the design and development process. In particular, it was noted that the application of different techniques at different stages of the process is essential. For example, at the feasibility stage approximate analytical methods may suffice, while at later stages simulation is often unavoidable. In short, a combination of techniques with an ability to switch rapidly between them is indispensable. Provision of this integrated facility is the essence of the project.

Particular requirements were identified as:

- Quick access to a library of analytical solutions, from the simple Erlang B, through single stage queueing systems of varying complexity, to more application-specific algorithms addressing, say, slotted ring performance.

- Solutions to classical queueing network problems.

- Simulation languages and support facilities to provide for the traditional hand-coding of simulation models, eg in SIMULA.

- Direct model-to-code generation facilities to enable the automatic generation of simulation programs from high level model descriptions.

- An environment to enable the design and operation of structured "What If?" experiments.

- A database of validated models for reuse or for the derivation of new models by synthesis or decomposition.

- Portability - that is the ability to run the tools on a wide range of computing facilities.

- The ability to download models to a more powerful computing environment.

- Ease of use with extensive "Help" facilities.

It was agreed that the tools should initially cater for expert users, ie Performance Engineers, and only then be extended for use by non experts.
3. THE TOOL SET

The tool set comprises four main components:

- Performance Engineering Algorithmic Techniques (PEAT)
- Network Study Tools (NETS)
- Simulation Tool (SIT)
- Experimenter Tool (EXT)

PEAT, NETS and SIT broadly cater for problems of increasing complexity. PEAT is generally associated with single stage of service models, while NETS caters for multi-stage of service models and SIT addresses those models which cannot be solved analytically. EXT, though not a performance modelling tool in its own right, is at the centre of the system providing a facility for the tools to interact with one another, and to pass the output of one tool as the input to another.

The following sub-sections describe each of the four components in more detail.

3.1 Performance Engineering Algorithmic Techniques (PEAT)

This tool addresses a wide range of performance analysis activities, particularly those encountered in telecommunications systems design. Most of the algorithms solve loss and delay problems and have been developed and coded in the Performance Engineering Division. In contrast to the NETS Tool described in section 3.2, the majority of these models pertain to single stage of service in an open system (ie a system in which demands arrive, are served, and then depart). To date, software has been incorporated for Erlang & Engset loss systems; for overflow and alternative routing models; and for exponential, constant and general service time distribution (ie variants on M/M/R, M/D/R, and M/G/R). Some software is also available for simple networking models. The addition and reconfiguration of algorithms is an ongoing activity and is implemented by a system manager menu which enables routines to be installed from a set of options.

At present the tool is interactive and all input and output is via the terminal, though full integration with the other components of the tool set is planned. The output comprises single values or tables of values. The analytical solutions associated with PEAT are usually exact for a class of models in which arrival and service time distributions and priority disciplines are very restricted. The performance measures provided by PEAT are comprehensive and generally include, where appropriate, both mean values and distributions. This is in direct contrast to the queueing network tool, described below, where the results are usually in the form of mean values only. Thus, while many models can be solved by either PEAT or NETS, the choice of solver will depend on the detail of the results that are required.

In order to extend the range of PEAT to deal with a wider range of situations where, for example, arrival and service time distributions are arbitrary or servers are organised in a grading, a simulation fall back is available using the model-to-code facilities available in SIT. The system will recognise that the model cannot be solved analytically and automatically set up a simulation.

3.2 Network Study Tools (NETS)

In its present form this tool is restricted to the study of classical queueing networks. It provides a consistent and user friendly interface from which the user can generate models in a form suitable for solution by the proprietary queueing network packages that are now available. The current, development version, of NETS accesses the QNAP2 package [5,6]. However, the NETS interface, with only minor modifications, can be extended to access other proprietary queueing networks packages such as PANACEA [9] or STEP1[8].

3.2B.2.3
The typical models handled by this tool are concerned with such problems as the sizing and capacity planning of computers, trunk reservation in circuit switched networks, and signalling protocols. Performance is usually described in terms of mean values. Delay distributions are not available but marginal probabilities of queue lengths are sometimes calculable. Simulation fall back, which is provided by some of the proprietary queueing network packages, extends the problem area very considerably and of course provides output which can be very detailed.

The tool, as it now stands, has solution processes which are distinct from those used in general telecommunications network studies and is best thought of as a queueing network tool. However, in the longer term, NETS is aimed at generating and dimensioning networks of a more general nature, with queueing networks being just one of the many applications. As the tool essentially provides facilities for the generation of arbitrarily complex nodes and their inter-connections, it is inherently adaptable to a much wider class of networks. Algorithms for the derivation of performance measures will need to be integrated with the NETS framework.

3.3 The Simulation Tool (SIT)

Simulation is in many ways our most powerful modelling tool. It is usually required, or at least highly desirable, at all but the most preliminary stages of the system design and dimensioning process. However the significant overheads in simulation, namely program development time and computing costs, often compel modellers to sacrifice detail and accuracy and use an alternative solver. The simulation tool attempts to overcome these difficulties and make simulation an attractive problem solver and a viable alternative to some of the prohibitively complex analytical and numerical methods.

SIT includes a well proven set of statistical and analysis routines and random number generators. The facilities of the tool allow simulation to be conducted in the conventional hand-coded style, i.e. by the use of simulation languages such as SIMULA, or by direct model-to-code generation which is described below. Simulation coding may be carried out in a process, event or activity orientated mode. This allows a system to be viewed in different ways and caters for simulationists with a variety of backgrounds.

The hand coding of simulations, however, must be regarded as an interim measure, with the direct model-to-code generation method becoming the norm. This method entails translation from a high level "English like" description of the model via an intermediate language, into executable simulation source code. The source code produced in the current version of the tools is in SIMC, a simulation package developed in-house. However, the approach is flexible and code for SIMULA or DEMOS [7] could equally well have been produced. Direct model-to-code generation, besides relieving the modeller of the tedium of coding, ensures that the program is validated. The technique is based on the synthesis of the model from software components which may be classified as primary or secondary. Primary components are generally basic elements while secondary components could well be complete models. There are currently four primary components, namely generator, server, node and call which are interconnected in different ways to produce a model. New components, which will mainly be secondary, will be added to the continually expanding knowledge-base to be available for synthesis or decomposition of further models.

The model-to-code generation facility provided in SIT is based on a dialogue between the user and the system with the emphasis on the modification and reuse of existing components from the system database. Model creation is in two stages:

1. Model Configuration - allowing the user to create a configuration, or skeleton, of the model. Here the components of the model and their connectivity are specified. Hopefully it will be possible to specify and re-use, perhaps with modification, existing components from the database.
2 Model Generation - allowing the user to convert the skeleton into an actual model by undertaking two further operations, namely attribute modification and functional modification. Attribute modification can be thought of as specifying or modifying variables associated with a component, and functional modification as specifying or modifying the actions of a component.

To surmount the pitfalls inherent in simulation, comprehensive expert system based guidance to the user will be incorporated in this tool. Dialogue between System and modeller will ensure that the model being studied is feasible, and also recommend appropriate simplifications that might be applied.

3.4 Experimenter (EXT)

The Experimenter tool supports the design and execution of experiments using objects produced by the modelling tools, and producing outputs that can be operated on by other tools. EXT, although not a performance modelling tool in its own right, is at the hub of the system, and provides the system with its flexibility. EXT provides the facility for tools to interact with one another and to pass the output of one tool to the input to another. It also provides facilities for conducting "What If?" experiments in a structured manner and, on a more mundane level, facilities for creating, browsing, editing, and saving models.

Whilst the EXT applies an experimentation environment to all the tools, its functions are most closely related to simulation which, being inherently a statistical experiment, needs to be carefully monitored and controlled. EXT will be closely associated with an expert system philosophy which will, amongst other things, control the length of simulation runs by predetermined confidence limits, manipulate and curve-fit simulation results and oversee the optimisation of system parameters.

4 SOFTWARE ASPECTS

In order to ensure a high degree of portability, the C programming language has been selected for the bulk of the system support software. However, performance algorithms in PASCAL and FORTRAN are also present and provision has been made for integrating algorithms in any of the languages supported by the operating system.

The tools are usable over a large range of hardware and, though designed in a UNIX System V environment, they can be ported to other operating systems with only minor changes. They have already been ported onto a DEC VAX running under VMS, and no problem is envisaged in porting to a VAX running ULTRIX (Berkeley UNIX System 4.2). The software is best looked upon as a large C program. System calls and other system dependent features have been kept to a minimum, and the corresponding code has been kept together to simplify modifications that may be needed to suit a particular target system.

5 USER INTERFACE

The interface is menu driven with additional forms facilities available with some of the tools. There is an extensive "Help" program to assist the user around the system and also to provide guidance on the performance engineering implications of the tools. A comprehensive user guide is available but the system is largely self explanatory.

The system is hierarchical with the main menu leading into the top menus of each of the tools. Figure 1 outlines the overall menu structure.

Typical menu and form screens are shown in Figures 2 and 3 respectively.
6 CURRENT STATUS AND FUTURE DEVELOPMENTS

The prototype tools are currently installed on a 68010 workstation running UNIX System V.2, and on a DEC VAX 11/750 under VMS version 4.5. They are accessed from VT100 type terminals connected via a local area network.

The general framework of the tools is well advanced and the basis for an integrated system has been firmly laid. The prototype version, providing a subset of the ultimate facilities, is available to the Performance Engineering Division for use and experimentation. This is providing valuable feedback to the development team.

The next phase of the project will concentrate heavily on expanding the expertise captured by the tools by the addition of further algorithms and models developed in the course of users' projects. The introduction of an expert system philosophy to automate the performance evaluation process will receive considerable attention. Exploratory work using a PROLOG expert system shell has produced very promising results and is being pursued further. The underlying principle of this work entails directing solutions to problems defined in terms of a model description language which may be textual or graphical. This, we expect, will make a considerable advance towards the goal of producing a fully integrated performance engineering tool set which is truly usable by both specialists and non-specialists.

7 ACKNOWLEDGEMENTS

The views expressed here are the author's personal views and do not represent British Telecom policy statements. Acknowledgement is made to the Director, Research and Technology for permission to make use of the information contained in this paper.

The author wishes to express gratitude to his staff for their invaluable contribution to the project, and thanks all members of the Performance Engineering Division for their support in this work.

REFERENCES

1 BT & ICL Collaborators 1985, ALVEY SOFTWARE ENGINEERING PROPOSAL, Improved Methods for Performance Engineering.


7 Birtwistle, GM, DEMOS: Discrete Event Modelling on Simula, University of Bradford (Macmillan).
FIG 1 - OVERVIEW OF MENU STRUCTURE
FIG 2 A TYPICAL MENU SCREEN

Main Menu

Project p1
Model m1

1 PEAT
2 Queueing Network Tool
3 Simulation Tool
4 Experimenter Tool

Select Option and press RETURN

'GOLD' H for Help 'GOLD' E to Exit System 'GOLD' $ for a shell

FIG 3 A TYPICAL FORM SCREEN

NAME

TYPE SERVER SOURCE RESOURCE SEMAPHORE INFINITE

MULTIPLE 1

SERVICE EXP CST HEXP ERLANG UNIFORM RINT COX

INIT 0

SERVICE EXPR

SCHEDULING FIFO LIFO

SCHED OPTS

DELETE NO YES

'GOLD' H for Help 'GOLD' F to save & Exit 'GOLD' Q to quit