Chapter 2

Conducting a Modelling Study

2.1. Introduction

In this chapter we take a broad look at how, when confronted with a specific computer system analysis problem, to apply the general "methodology" of queueing network modelling. This skill must be developed through experience — it cannot be absorbed passively. Recognizing this, we present a set of case studies selected to illustrate significant aspects of the methodology, sharing with the reader the experience of others.

The success of queueing network modelling is based on the fact that the low-level details of a system are largely irrelevant to its high-level performance characteristics. Queueing network models appear abstract when compared with other approaches to computer system analysis. Queueing network modelling is inherently a *top-down* process. The underlying philosophy is to begin by identifying the principal components of the system and the ways in which they interact, then supply any details that prove to be necessary. This philosophy means that a large number of assumptions will be introduced and assessed in the process of conducting a modelling study. Three principal considerations motivate these assumptions:

- *simplicity*

  There is a strong incentive to identify and eliminate irrelevant details. In fact, we will adopt a rather liberal definition of "irrelevant" in this context by generally including any system characteristic that will not have a *primary* (as opposed to *secondary*) *effect* on the results of the study. Examples include:

  — Although a system may have a large number of identifiable workload components, we may be interested in the performance of only one of them. In this case, we may choose to employ a model with only two classes, one representing the workload component of interest and the other representing the aggregate effect of all other workload components.

- The primary effect of a CPU upgrade will be a decrease in CPU service demands. A change in the average paging and swapping activity per job may also result, but if so, this is a secondary effect.

- *adequacy of measurements*

  The measurement tools available on contemporary computer systems often fail to provide directly the quantities required to parameterize queueing network models. Queueing network models require a small number of carefully selected inputs. Measurement tools, largely for historical reasons, provide a large volume of data, most of which is of limited use for our purposes. Considerable interpretation may be required on the part of the analyst. Examples include:

  - Typically, a significant proportion of CPU activity is not attributed to specific workload components. Since the CPU tends to be a heavily utilized resource, correct attribution of its usage is important to the accuracy of a multiple class model.
  - Surprisingly, even determining the multiprogramming level of a batch workload sometimes is difficult, because some system tasks ("quiescent" or "operator" jobs) may be counted by the measurement tool.

- *ease of evaluation*

  As noted in Chapter 1, we must restrict ourselves to a subset of general networks of queues that can be evaluated efficiently. To stay within this subset, we must make compromises in the representation of certain computer system characteristics. Examples include:

  - Extremely high variability in the service requirement at a particular resource can cause performance to degrade. Direct representation of this characteristic makes queueing network models costly to evaluate, though, and examples where it is a major determinant of performance are rare. It generally is omitted from models.
  - Memory admission policies typically are complex, and the memory requirements of programs differ. The evaluation of a model is considerably eased, though, if we are willing to assume that the memory admission policy is either first-come-first-served or class-based priority, and that programs have similar memory requirements, at least within each class.

Skill in introducing and assessing assumptions is the key to conducting a successful modelling study. In general, it is important to be explicit concerning the assumptions that are made, the motivations for their introduction, and the arguments for their plausibility. This allows the analyst's reasoning to be examined, and facilitates evaluating the sensitivity of the results to the assumptions.

The material in this chapter has a spectrum of interpretations ranging from fairly shallow to fairly subtle. The reader with little experience will find a collection of brief case study descriptions indicating the applicability of queueing network models. The reader with moderate experience will learn something of the ways in which queueing network modelling studies are conducted. The reader with considerable experience will discover insights concerning various aspects of conducting a modelling study that can be used to great advantage. Because of this spectrum of interpretations, we will ask you to review this chapter during Part V of the book.

2.2. The Modelling Cycle

The most common application of queueing network modelling involves projecting the effect on performance of changes to the configuration or workload of an existing system. There are three phases to such a study. In the *validation* phase, a *baseline model* of the existing system is constructed and its sufficiency established. In the *projection* phase, this model is used to forecast the effect on performance of the anticipated modifications. In the *verification* phase, the actual performance of the modified system is compared to the model's projections. Taken together, these three phases are referred to as the *modelling cycle*, illustrated in Figure 2.1.

The validation phase begins with the definition of the model, which includes selection of those system resources and workload components that will be represented, identification of any system characteristics that may require special attention (e.g., priority scheduling, paging), choice of model structure (e.g., separable, hybrid), and procedures for obtaining the necessary parameters from the available measurement data.

Next, the system is measured to obtain *workload measures*, from which model inputs will be calculated, and *performance measures*, which will be compared to model outputs. In some cases these are the same; for instance, device utilizations are workload measures (they are used to calculate service demands) and also performance measures (they are used to assess the accuracy of the model). On the other hand, the multiprogramming level of a batch workload is strictly a workload measure, and system response time is strictly a performance measure.

The workload measures then are used to parameterize the model, a step that may require various transformations. The model is evaluated, yielding outputs. These are compared to the system's performance measures. Discrepancies indicate flaws in the process, such as system characteristics that were ignored or represented inappropriately, or model inputs whose values were established incorrectly. Unfortunately, the absence of

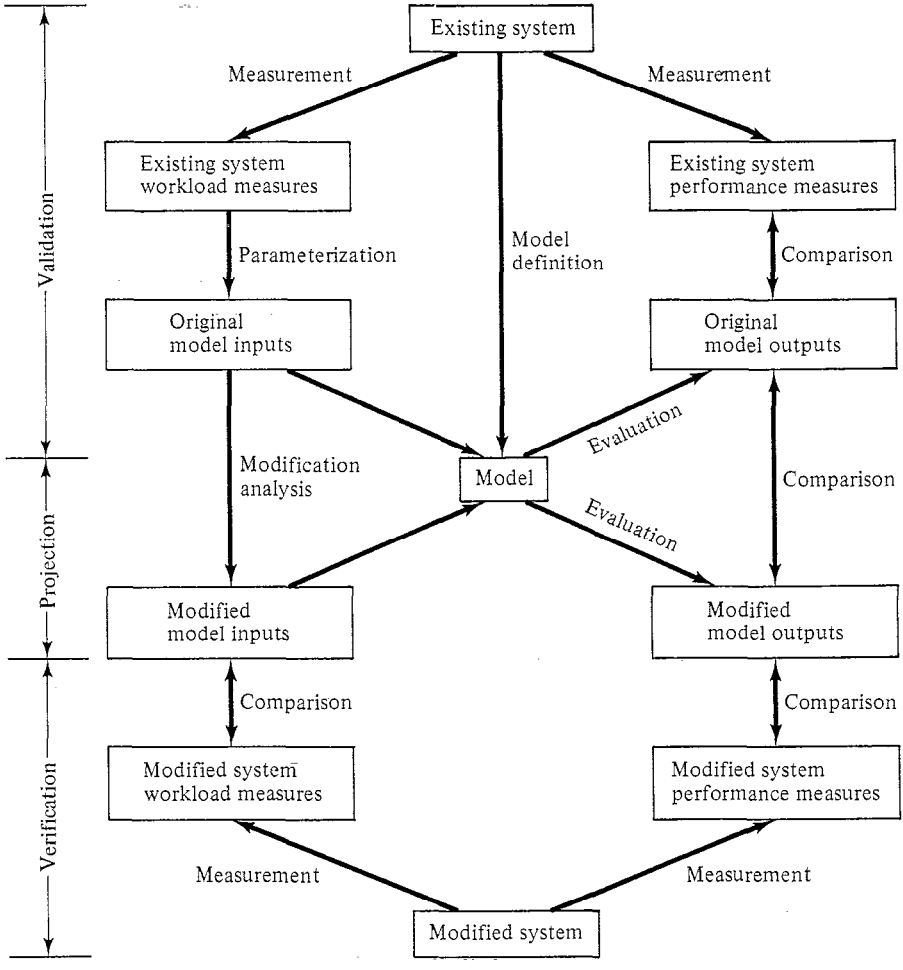

Figure 2.1 – The Modelling Cycle

such discrepancies does not guarantee that the model will project properly the effect of system or workload modifications. Confidence in a model's predictive abilities may come from two sources. The first is repetitive validation over a number of measurement intervals, perhaps involving selected modifications. For example, if the objective of a modelling study is to assess the benefits of additional memory, it may be possible to repeat the validation phase while various amounts of existing memory are disabled. The second is completion of the verification phase, discussed below.

In the projection phase, model inputs are modified to reflect the anticipated changes to the system or workload. This is a complex process, to which we will devote considerable attention later in the book (Chapter 13). The model then is evaluated. The difference between the modified model outputs and the original model outputs is the projected effect of the modification.

Finally, in the verification phase, the modified system is measured and two comparisons are made. First, its performance measures are compared to the model outputs. Second, its workload measures are compared to the model inputs. Discrepancies between the projections of the model and the performance of the system can arise from two sources: the omission or mis-representation of (retrospectively) significant system characteristics, and the evolution of the system in a way that differs from that which was anticipated. Understanding and evaluating these sources of discrepancy is crucial to gaining confidence in queueing network modelling as a computer system analysis technique. The accuracy of a model's performance projections can be no greater than the accuracy of the workload projections furnished as input.

To illustrate the modelling cycle we describe two case studies undertaken at a computing complex consisting of a number of IBM 370/168s, 370/168-APs (dual processors), and 3033s running the MVS operating system along with applications such as TSO (interactive processing), IMS (database management), JES (spooling), and TCAM/VTAM (terminal management). The objective of each study was to determine the impact of a significant workload modification.

In the first study, the question under consideration was: "Can the workloads presently running on two separate 370/168 uniprocessors be combined on a single 3033?" (A 3033 is considered to have 1.6 to 1.8 times the processing power of a 168.) On each of the original systems, the principal application was an IMS workload. In addition, one of the systems had a background batch workload, and each had various system tasks.

In the validation phase, each of the original systems was measured and modelled. IMS response time was the performance measure of greatest interest, since response time degradation was the anticipated effect of the modification.

In the projection phase, a single model was defined in which each of the original workloads (IMS-1, IMS-2, and batch) was individually represented, with CPU service demand adjusted to account for the speed differences of the CPUs. It was assumed that the I/O subsystem of the 3033 would be the combination of the I/O subsystems of the 168s, so I/O subsystem parameters were not changed in any way.

performance measure	workload component	model output	measurement data
CPU utilization	IMS-1	43%	40%
	IMS-2	31%	32%
	batch	3%	3%
	total	77%	75%
response time	IMS-1	0.84 secs.	1.3 secs.
	IMS-2	0.79 secs.	0.89 secs.
throughput	batch	2 jobs/hr.	1.7 jobs/hr.

Table 2.1 – The Modelling Cycle: Case Study 1

In the verification phase, the workloads were combined on the 3033. Performance measures were compared to the model outputs. Table 2.1 displays the results, which are typical of those that can be expected in a study such as this: the projections of the model are sufficiently accurate to be of great utility in planning, and the discrepancy in utilizations is less than the discrepancy in response times.

The second study involved the five loosely-coupled systems described below:

system	CPU type	workload
1	3033	JES for all systems
2	370/168-AP	interactive graphics, batch
3	370/168-AP	batch
4	3033	TSO, IMS, batch
5	3033	batch

The question under consideration was "Can the workload of System 5 be distributed among the four other systems without significant adverse effects on performance, allowing System 5 to be released for cost reduction?"

In the validation phase, Systems 2 through 5 were measured and modelled. (System 1 was excluded from the study.)

In the projection phase, the batch multiprogramming level in the models of Systems 2, 3, and 4 was increased to correspond to the addition of 27% of the workload of System 5. (Management hoped to place 19% of System 5's workload on System 1 and 27% on each of Systems 2, 3, and 4.) This simple approach was possible because of the similarity of the batch workloads on the various systems.

In the verification phase, System 5's workload was distributed among the remaining systems. For each system individually, performance measures were compared to the model outputs. In each case, the anticipated

effect of the modification was an increase in the resource consumption of the batch workload (its multiprogramming level had increased), and a degradation in the performance of the other workload components. Tables 2.2, 2.3, and 2.4 display the results for Systems 2, 3, and 4 respectively. Once again, the results are typical of those that can be expected. When used in studies involving system modification, queueing network models may project relative performance with greater accuracy than absolute performance. Consider the response time of the interactive graphics workload in Table 2.2. The original model yielded 4.8 seconds, where 5.2 seconds was measured. The modified model yielded 5.0 seconds. It makes sense to interpret this as a projected response time degradation of 4% ($\frac{5.0-4.8}{4.8}$). In fact, the measured response time degradation was 7.5%.

perf. measure	workload component	original system	original model	modified model	modified system
CPU util.	graphics	76%	74%	74%	72%
	batch	11%	10%	13%	13%
	total	87%	84%	87%	85%
resp. time	graphics	5.2 secs.	4.8 secs.	5.0 secs.	5.6 secs.
t'put.	batch	28/hr.	27/hr.	35/hr.	30/hr.

Table 2.2 – The Modelling Cycle: Case Study 2, System 2

perf. measure	workload component	original system	original model	modified model	modified system
CPU util.	batch	63%	64%	76%	73%
t'put.	batch	101/hr.	104/hr.	130/hr.	120/hr.

Table 2.3 – The Modelling Cycle: Case Study 2, System 3

perf. measure	workload component	original system	original model	modified model	modified system
CPU util.	TSO	65%	67%	65%	63%
	IMS	3%	2%	2%	2%
	batch	15%	15%	21%	20%
	total	83%	84%	88%	85%
resp. time	TSO	4.3 secs.	4.4 secs.	5.0 secs.	5.9 secs.

Table 2.4 – The Modelling Cycle: Case Study 2, System 4

Although we have presented the modelling cycle in an orderly fashion, conducting a modelling study is by no means a strictly sequential process. There are strong dependencies among the various components of the validation and projection phases. Compatibility must be achieved between the definition of the model, the measurements used to parameterize the model, and the techniques used to evaluate the model. Achieving this compatibility, and reconciling it with the objectives of a particular modelling study, is inherently iterative in nature.

2.3. Understanding the Objectives of a Study

It is obvious that the validation phase of a modelling study requires a thorough understanding of the computer system under consideration. Perhaps it is less obvious that a thorough understanding of the objectives of the study is of equal importance. In fact, though, this latter understanding is a key component of the top-down philosophy of queueing network modelling. Many system characteristics that would need to be represented in a fully general model may be irrelevant in a particular study. Identifying these characteristics leads to a simpler model and a simpler modelling study.

A typical example of this phenomenon involved a computer manufacturer about to announce a new CPU in a minicomputer architectural family. During the design of this CPU, extensive low-level performance studies had been carried out, yielding measures such as the average execution rate for various instruction mixes. Prospective customers, however, would be interested in higher-level characterizations such as "In a specific configuration, how does it compare to existing CPUs in the architectural family in terms of the number of users it can support?"

The manufacturer had a set of fifteen benchmarks that had been used in the past for this sort of characterization. Each of the benchmarks had four workload components: editing, file creation, file modification, and a compile-link-execute sequence. The benchmarks differed in the number of "users" in each workload component. These "users" were generated by means of *remote terminal emulation (RTE)*, a technique in which the system of interest is coupled to a second system which simulates interactive users and gathers performance data.

Unfortunately, it was impossible to configure the prototype of the new CPU with the I/O subsystem of interest for the purpose of conducting RTE experiments. Instead, the following strategy was devised:

- Configure an existing, faster CPU in the architectural family with the I/O subsystem of interest.
- Conduct RTE experiments on this configuration for each of the fifteen benchmarks.
- Use a queueing network model to project the performance of each of these benchmarks when the new, slower CPU is substituted. Establish the CPU service demand in the model by taking into account the ratio of the instruction execution rates of the two CPUs.

Given this strategy, the obvious approach would be to define a rather general model of the system. The inputs to this model would include the workload intensities and service demands of each of the four workload components. The model would be capable of reflecting the different characteristics of the fifteen benchmarks by suitable adjustments to the inputs. After this model had been validated, the CPU service demand for each workload component would be scaled appropriately, and the model then would be used to project the performance of the benchmarks on the new system, again by suitable adjustments to the model inputs.

This approach has a significant hidden complexity. The system under consideration includes a sophisticated memory management policy that employs both paging and swapping. The amount of service demanded by each user at the paging and swapping devices is not intrinsic; rather, it depends upon the particular mix of workload components in each benchmark. Thus, the different characteristics of the fifteen benchmarks cannot be reflected in the model simply by adjusting the workload intensities. Instead, a general queueing network model of the system would need to include, as part of its definition, a procedure for estimating variations in the paging and swapping service demands as functions of the mix of workload components.

Devising such a procedure certainly is feasible, but it adds considerably to the complexity of the modelling study, and it provides a level of generality that is not required. Bearing in mind that the objective of this study was restricted to estimating the relative performance of each of the fifteen benchmarks on the two configurations, we can achieve a significant simplification by assuming that the paging and swapping activity of each user, while sensitive to changes in the mix of workload components, are insensitive to changes in CPU speed. This assumption allows the paging and swapping service demands of each workload component to be measured for each of the benchmarks during the RTE experiments, and provided as inputs to the queueing network model, rather than being estimated using a procedure supplied as part of the model definition.

The two approaches to this computer system analysis problem are contrasted in Figure 2.2. The assumption on which the simplified approach

relies is not valid universally, but any inaccuracies that result are strictly secondary, and in fact are probably smaller in magnitude than those that inevitably would arise in attempting to estimate variations in paging and swapping service demands as functions of the mix of workload components. (We will return to this study in Section 2.5, adding further details.)

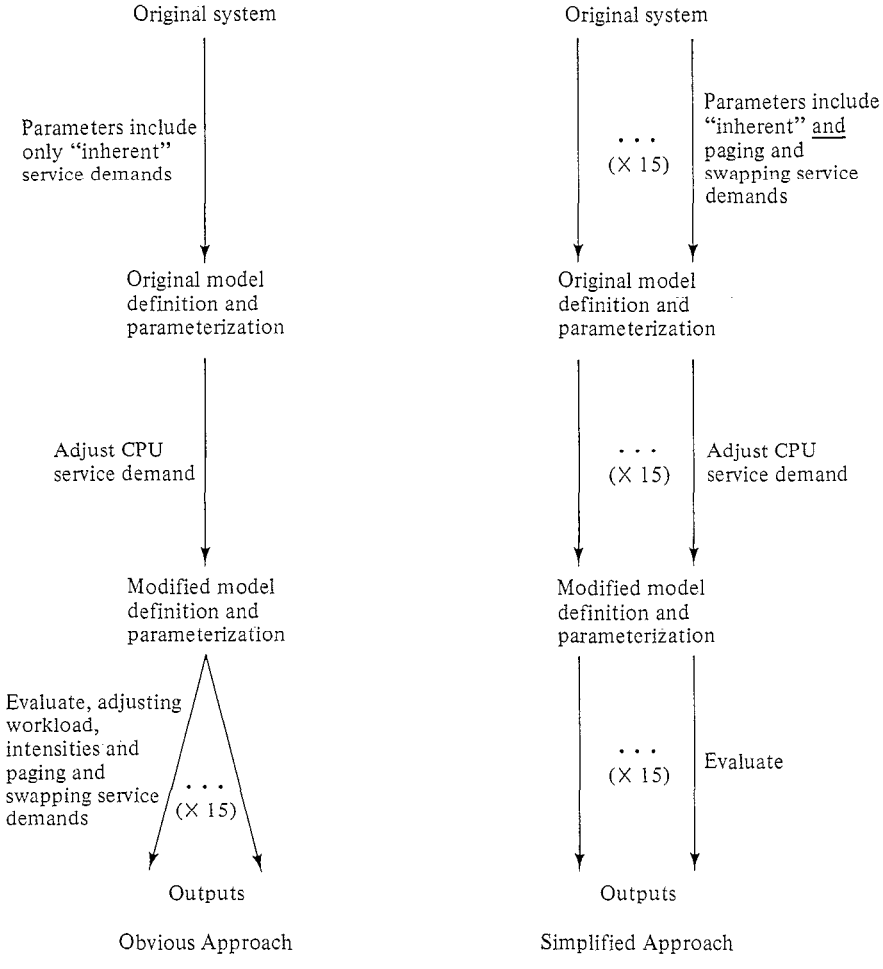

Figure 2.2 – Two Approaches to Modelling a CPU Replacement

2.4. Workload Characterization

In discussing the validation phase of the modelling cycle, we identified *measurement* as the process of obtaining workload measures for the computer system of interest, and *parameterization* as the process of transforming those workload measures into the inputs of a queueing network model. These activities, while not necessarily straightforward, are often considerably less difficult than *workload characterization*: the process of selecting the workload or workloads on which to base the performance study.

Difficult questions arise even in considering an existing computing environment: What constitutes a "typical" workload? How should a measurement interval be selected? Should data from several measurement intervals be averaged? These uncertainties are compounded in considering an environment that cannot be measured directly (e.g., in contemplating the movement of an existing workload to a new system, or the introduction of a new workload to an existing system).

Every approach to computer system analysis — intuition and trend extrapolation, experimental evaluation of alternatives, or modelling — requires workload characterization. Strangely, the imprecision inherent in workload characterization argues for the use of queueing network models. In principle, greater accuracy might be obtained (at significantly greater cost) through experimentation or through simulation modelling. In practice, however, the dominant source of error is apt to lie with the workload characterization, even when queueing network models are employed.

The following case study serves three purposes. The first is to illustrate the use of queueing network modelling in a situation where benchmarking is the traditional approach. The second is to demonstrate *hierarchical workload characterization* as a way to achieve flexibility. By this, we mean progressing in an orderly fashion from a high-level characterization (identification of workload components) through an intermediate level (machine-independent characterizations of each of the components) to a low level (service demands). The third is to show that useful insights can be obtained despite serious imprecision in the workload characterization.

In 1979, a university began a program to acquire medium-scale interactive computer systems for instructional use. In response to a *request for proposals (RFP)*, roughly twenty bids were received, most involving multiple systems. The relative performance of candidate systems was to be a major factor in the acquisition decision. Two approaches to evaluating this relative performance were considered. The first was to construct a multi-user benchmark characteristic of the anticipated workload, then use a remote terminal emulator to run that benchmark on each

candidate system. The second was to perform limited measurements on the candidate systems, then use queueing network modelling to compare performance. The latter approach was appropriate because of the limited time and manpower available for the study, the large number of candidate systems, and the high degree of uncertainty that existed concerning the anticipated workload.

The first step in the study was to characterize the anticipated workload in high-level terms: What were the identifiable workload components? What was the relative volume of each component? What were the significant characteristics of a typical transaction belonging to each component?

Instructional computing previously had been handled in batch mode. The migration of this function to interactive facilities, and its subsequent expansion, was to be a multi-year process involving multiple acquisitions. It was assumed that the initial interactive workload would be similar in composition to the existing instructional batch workload, with the addition of an editing component.

Measurements indicated that the existing workload had only two significant components. Nearly 80% of all transactions were Fortran compilations. Nearly 20% of all transactions were the execution of precompiled library routines to process student-created datasets. A simple characterization of the compilations was the average number of lines of source code: roughly 100. A simple characterization of the executions was their average service demand on the existing system: 4.55 seconds of CPU service and 5.35 seconds of disk service. (The average size of the student-created datasets processed by these transactions was 100 lines.)

It was assumed that an editing session would precede each compilation or execution, so that the overall mix of workload components would be 40% compilations, 10% executions, and 50% editing sessions. Since most editing would be performed by inexperienced typists using line-oriented editors to make a small number of changes to a file, it was assumed that the dominant resource demands would occur in accessing and saving files. The average size of the file being edited, 100 lines, thus was a simple characterization of the editing sessions.

The second step in the study was to translate this high-level workload characterization into parameters for models of each of the candidate systems. Determining workload intensities was not an issue. Each of the three workload components was treated as a transaction workload with an arrival rate equal to the established proportion of the total arrival rate. Model outputs were tabulated for a range of total arrival rates. Determining service demands for each workload component on each system (i.e., the average service required at each device by a transaction belonging to each workload component) involved running three extremely simple

experiments on each system. For compilations, a 100-line program was compiled on an otherwise idle system and CPU and disk busy times were measured. This experiment captured the effects of hardware speed, compiler efficiency, and overhead in initiating and terminating compilations. For executions, the CPU and disk service requirements that had been measured on the existing batch system were scaled. The scaling factor for CPU service was obtained by running a single computational benchmark on the existing system and on each candidate system. The scaling factor for disk service was obtained using a single Fortran I/O benchmark. For editing sessions, the default editor available on each candidate system was used on an otherwise idle system to access a 100-line file, modify a single line, and save the file. CPU and disk busy times were measured.

Table 2.5 shows the results of these experiments for three candidate systems: a VAX-11/780, a Prime 750, and a Prime 550. Note the dramatically different efficiencies observed for the two Fortran compilers available on the Primes. Note also the relative inefficiency of the interface between the editor and the file system on the VAX.

system	workload component	service demand, secs.	
		CPU	disk
Digital VAX-11/780	compilation	2.0	1.0
	execution	11.9	10.7
	editing session	0.5	0.8
Prime 750	compilation		
	compiler A	0.8	0.2
	compiler B	7.0	1.0
	execution	13.7	7.1
	editing session	0.15	0.05
Prime 550	compilation		
	compiler A	1.3	0.75
	compiler B	11.3	3.75
	execution	27.9	21.4
	editing session	0.3	0.1

Table 2.5 – Service Demands for Three Systems

Based on these values, queueing network models of the candidate systems were parameterized and evaluated. (Representing multiple disks involved distributing the calculated disk service demand among several service centers. Parameterization was simplified by the fact that it was not necessary to consider overhead due to memory contention, which typically grows with workload intensity. It was a stipulation of the RFP that systems be overconfigured with respect to memory.) Figures 2.3 and 2.4 show typical results of the study: average response time versus total

transaction arrival rate for compilations and executions, respectively, for the VAX-11/780, the Prime 750 with compiler A, and the Prime 750 with compiler B. Note that the performance of the Prime depends critically on the choice of compiler, and that this choice affects all users, not just those doing compilations. (A reminder: these results have significance only for the specific configurations and workloads under consideration.)

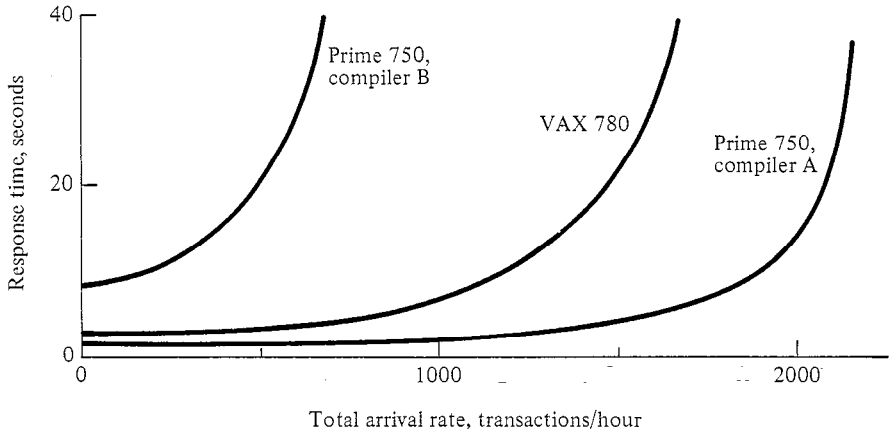

Figure 2.3 – Compilation Response Time Versus Total Arrival Rate

Variations can be investigated with ease. The effect of a disk load imbalance can be explored by shifting the proportion of the service demand allocated to each service center. The sensitivity of the results to the workload characterization can be studied; e.g., the relative arrival rates of the three workloads could be altered.

2.5. Sensitivity Analysis

Every computer system analyst encounters situations in which questionable assumptions must be introduced. *Sensitivity analysis* can be used to determine the extent to which such assumptions cast doubt on the conclusions of the study. A sensitivity analysis can take many forms. Two of the most common are:

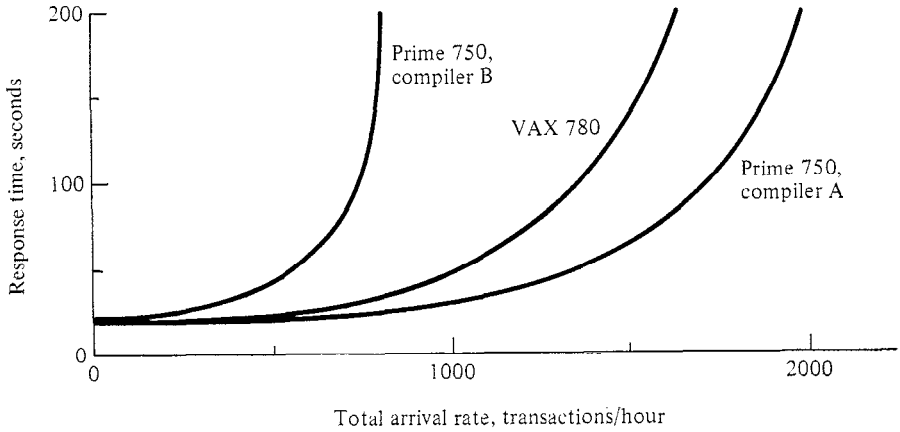

Figure 2.4 – Execution Response Time Versus Total Arrival Rate

- The analyst may test the robustness of the results to the assumption in question. Doing so involves evaluating the model a number of times for variations in the assumption, and comparing the results.
- The analyst may obtain bounds on the expected performance, by evaluating the model for extreme values of the assumption.

Inadequate measurement data frequently is the culprit that prompts a sensitivity analysis. To illustrate the role of sensitivity analysis in coping with this situation we return to the CPU replacement case study introduced in Section 2.3. As illustrated in Figure 2.2, the approach adopted entailed fifteen separate experiments, one per benchmark. Each experiment consisted of three phases: the existing system was measured while executing one of the benchmarks, a queueing network model was constructed and validated, and this model was used to project benchmark performance with the new CPU, by manipulating the CPU service demand parameter of each workload component.

Difficulty was encountered during the validation phase because a significant proportion of the system's I/O activity was not attributed to specific workload components by the available measurement tools. For example, it was possible to determine the total number of swaps during a measurement interval, and also the average disk service demand per swap, but it was not possible to determine which user or workload component was the "victim" of the swap. Had the study been based on a single class model, this would not have been a problem. However, the objective was to assess the impact of the CPU replacement on each of the

four workload components individually, so a multiple class model was required.

Various methods of allocating this measured I/O activity among the four workload components yielded different values for some of the input parameters of the model. Not surprisingly, different response time projections from the model resulted. As an example, for one of the benchmarks the measured response time for file modification transactions was 10 seconds, while for three different but equally reasonable methods of allocating measured I/O activity among the four workload components, the model projected response times of 6, 7, and 11 seconds. (Similarly spurious results were obtained from this model for the response times of the three other workload components.)

Consider the set of inputs for which the model projected a response time of 6 seconds. When the CPU service demand parameter was adjusted to reflect the substitution of the slower CPU, this model projected that response time would be 7.2 seconds. It makes no sense to claim that the response time for file modification transactions on the new system will be 7.2 seconds, because the measured response time on the existing, faster system was 10 seconds. Nor does it make sense to claim that response time will increase by 20% ($\frac{7.2-6.0}{6.0}$), because there is no reason to believe that the projected effect of the CPU substitution is insensitive to the method used to allocate measured I/O activity among the workload components. We can hypothesize such an insensitivity, though, and then test this hypothesis. Table 2.6 displays projected response times for the system with the existing CPU and the new CPU, for the three approaches to I/O activity allocation. Although the absolute response time values differ for the three approaches, the projected percentage changes do not. Thus, we can conclude that the effect of the CPU substitution will be in increase of roughly 20% in the response time of file modification transactions, from 10 seconds (the measured value) to 12 seconds. (Similar results were obtained for the other three workload components.)

2.6. Sources of Insight

A major virtue of queueing network modelling is that the modelling cycle yields many insights about the computer system under study. These insights occur during workload characterization, model definition, system measurement, model parameterization, and modification analysis. It is important to bear in mind that the model outputs obtained during the projection phase of the modelling cycle are only one of many sources of insight. Consider the following case study.

method of allocating I/O activity	workload component	response time, seconds		
		model of original CPU	model of new CPU	projected change
A	editing	...	...	...
	file creation	...	...	...
	file mod.	6	7.2	+ 20%
	compile-link-execute	...	...	...
B	editing	...	...	...
	file creation	...	...	...
	file mod.	7	8.3	+ 18%
	compile-link-execute	...	...	...
C	editing	...	...	...
	file creation	...	...	...
	file mod.	11	13.1	+ 19%
	compile-link-execute	...	...	...

Table 2.6 – Response Times for Three Assumptions

An insurance company decentralized its claims processing by establishing identical minicomputer systems at twenty geographically distributed sites. As the workload grew, these systems ceased to provide adequate response, and a two-step capacity expansion program was begun: an immediate upgrade at every site to one of two software-compatible systems available from the original vendor, followed by a three year process of "unconstrained" system acquisition and software conversion. Queueing network modelling was used to evaluate the alternatives for each step. In this section, we consider the choice of a "transition system" for each site.

Working together, the vendor (IBM) and the insurance company had estimated that performance would "improve by a factor of 1.5 to 2.0" if the existing system (a 3790 in each case) were replaced with the less expensive of the two transition systems (an 8130), and "improve by a factor of 2.0 to 3.5" if it were replaced by the more expensive of the transition systems (an 8140). (Note the considerable ambiguity in these statements.) The charter of the modelling study was to determine at which of the twenty sites the more expensive system would be required in order to achieve acceptable performance during the three-year transition period.

The information provided in support of the study included measurements of the existing 3790 system taken at several sites under "live"

workload, measurements of the 3790 and the more expensive transition system (the 8140) during benchmark trials in which varying numbers of clerks entered transactions from scripts, and information from the vendor comparing the CPU and disk speeds of the three systems. The "live" workload tests revealed that although there were three distinct workload components, one of these, which had been identified in advance as being of primary interest, was responsible for roughly 75% of the transactions and 90% of the resource consumption. A single class model was therefore deemed appropriate. The benchmark tests confirmed the vendor's estimates of relative hardware speeds, although they were too limited (in terms of the range of workload intensities considered) to yield any insight about overall performance. From consideration of all of the available information it was possible to calculate the service demands shown below:

	service demands, seconds	
system	**CPU**	**disk**
3790 (existing)	4.6	4.0
8130	5.1	1.9
8140	3.1	1.9

As indicated, the two transition systems were equipped with identical disks that were roughly twice as fast as the disks on the existing system. The transition systems differed in their CPUs: the 8130 CPU was, in fact, slightly slower than that of the existing 3790, while the 8140 CPU was roughly 50% faster.

Now we make a key observation. On the existing system, the workload is CPU-bound. Furthermore, since response times are unacceptable, we can assume that the workload intensity is sufficiently high that the CPU is approaching saturation. The faster disks of the 8130 are of little value under these circumstances, while its slower CPU is a significant liability. Without further examination, we can conclude that replacing the 3790 with the 8130 will cause a degradation in response time.

On the basis of this analysis, the insurance company performed benchmark tests on the 8130. These tests confirmed the analysis, with the result that all sites were upgraded to 8140s. (This study will be considered further in Chapter 5.)

2.7. Summary

The most challenging aspect of computer system analysis using queueing network models is not the technical details of defining, parameterizing, and evaluating the models. Rather, it is the process of tailoring the general "methodology" of queueing network modelling to a specific

computer system analysis context. Unfortunately, while the former is easily taught, the latter is best learned through experience. In this chapter we have attempted to share with the reader the experience of others, by presenting a set of case studies selected to illustrate significant aspects of the methodology. Among the points that we have emphasized are:

- Queueing network modelling inherently is a top-down process in which the low-level details of a system are presumed to be irrelevant to its high-level performance characteristics.
- Because queueing network models are abstract, many assumptions are made in conducting a modelling study. These assumptions are motivated by simplicity, adequacy of measurements, and ease of evaluation. It is important to be explicit concerning the assumptions that are made, the motivations for their introduction, and the arguments for their plausibility.
- Conducting a modelling study is an iterative process because of dependencies that exist among the definition of the model, the measurements used to parameterize the model, the techniques used to evaluate the model, and the objectives of a particular modelling study.
- Confidence in a model's predictive abilities can be acquired through repetitive validation over a number of measurement intervals, perhaps involving selected minor modifications.
- This confidence can be reinforced through the verification process: measuring a modified system, then comparing its performance measures to the model outputs and its workload measures to the model inputs.
- When used in studies involving system modification, queueing network models may project relative performance with greater accuracy than absolute performance.
- A clear understanding of the objectives of a modelling study can contribute to simplicity in the model and in the modelling effort.
- Concentrating on representing the primary effects of a system or workload modification also can contribute to simplicity.
- Workload characterization is a challenging, inherently imprecise process. Useful insights can be obtained despite this imprecision. Characterizing a workload hierarchically helps to achieve flexibility.
- Sensitivity analysis can be used to determine the extent to which questionable assumptions cast doubt on the conclusions of a study. Two common forms of sensitivity analysis are testing the robustness of model outputs to variations of assumptions, and obtaining bounds on model outputs for extreme values of assumptions.

- Valuable insights are gained throughout the modelling cycle, not merely during the projection phase.

2.8. References

The identification of simplicity, adequacy of measurements, and ease of evaluation as factors motivating the introduction of assumptions is due to Kienzle and Sevcik [1979], who also suggested the division of the modelling cycle into validation, projection, and verification phases.

The MVS case studies described in Section 2.2 were conducted by Lo [1980]. The CPU performance comparison described in Sections 2.3 and 2.5 was conducted by Myhre [1979]. The system acquisition case study described in Section 2.4 was conducted by Lazowska [1980]. (Figures 2.3 and 2.4 are taken from this paper.) The insurance claims processing case study described in Section 2.6 was conducted by Sevcik, Graham, and Zahorjan [1980].

[Kienzle & Sevcik 1979]
M.G. Kienzle and K.C. Sevcik. A Systematic Approach to the Performance Modelling of Computer Systems. *Proc. IFIP W.G.7.3 International Symposium on Computer Performance Modelling, Measurement and Evaluation* (1979), 3-27.

[Lazowska 1980]
Edward D. Lazowska. The Use of Analytic Modelling in System Selection. *Proc. CMG XI International Conference* (1980), 63-69.

[Lo 1980]
T.L. Lo. Computer Capacity Planning Using Queueing Network Models. *Proc. IFIP W.G.7.3 International Symposium on Computer Performance Modelling, Measurement and Evaluation* (1980), 145-152. Copyright © 1980 by the Association for Computing Machinery.

[Myhre 1979]
Scott A. Myhre. A Queueing Network Solution Package Based on Mean Value Analysis. M.Sc. Thesis, Department of Computer Science, University of Washington, February 1979.

[Sevcik et al. 1980]
K.C. Sevcik, G.S. Graham, and J. Zahorjan. Configuration and Capacity Planning in a Distributed Processing System. *Proc. 16th CPEUG Meeting* (1980), 165-171.