HOME SUBSCRIBE SEARCH FAQ SITEMAP CONTACT US

KEYWORD SEARCH
GO!
Search Tips
Advanced Search
PUBLICATION LOOKUP
GO!
Title
JUMP TO TOPIC

Art of Computer Systems Performance Analysis Techniques For Experimental Design Measurements Simulation And Modeling
by Raj Jain
Wiley Computer Publishing, John Wiley & Sons, Inc.
ISBN: 0471503363 **Pub Date:** 05/01/91

Search this book:
GO!

Previous | Table of Contents | Next

CHAPTER 3
SELECTION OF TECHNIQUES AND METRICS

> **response time** *n*. An unbounded, random variable T_r associated with a given TIMESHARING system and representing the putative time which elapses between T_s, the time of sending a message, and T_e, the time when the resulting error diagnostic is received.
>
> —S. Kelly-Bootle
> *The Devil’s DP Dictionary*

Selecting an evaluation technique and selecting a metric are two key steps in all performance evaluation projects. There are many considerations that are involved in correct selection. These considerations are presented in the first two sections of this chapter. In addition, performance metrics that are commonly used are defined in Section 3.3. Finally, an approach to the problem of specifying the performance requirements is presented in Section 3.5.

3.1 SELECTING AN EVALUATION TECHNIQUE

The three techniques for performance evaluation are analytical modeling, simulation, and measurement. There are a number of considerations that help decide the technique to be used. These considerations are listed in Table 3.1. The list is ordered from most to least important.

The key consideration in deciding the evaluation technique is the life-cycle stage in which the system is. Measurements are possible only if something similar to the proposed system already exists, as when designing an improved version of a product. If it is a new concept, analytical modeling and simulation are the only techniques from which to choose. Analytical modeling and simulation can be used for situations where measurement is not possible, but in general it would be more convincing to others if the analytical modeling or simulation is based on previous measurement.

TABLE 3.1 Criteria for Selecting an Evaluation Technique

Criterion	Analytical Modeling	Simulation	Measurement
1. Stage	Any	Any	Postprototype
2. Time required	Small	Medium	Varies
3. Tools	Analysts	Computer languages	Instrumentation
4. Accuracy[a]	Low	Moderate	Varies
5. Trade-off evaluation	Easy	Moderate	Difficult
6. Cost	Small	Medium	High
7. Saleability	Low	Medium	High

[a] In all cases, result may be misleading or wrong.

The next consideration is the time available for evaluation. In most situations, results are required *yesterday*. If that is really the case, then analytical modeling is probably the only choice. Simulations take a long time. Measurements generally take longer than analytical modeling but shorter than simulations. Murphy's law strikes measurements more often than other techniques. If anything can go wrong, it will. As a result, the time required for measurements is the most variable among the three techniques.

The next consideration is the availability of tools. The tools include modeling skills, simulation languages, and measurement instruments. Many performance analysts are skilled in modeling. They would not touch a real system at any cost. Others are not as proficient in queueing theory and prefer to measure or simulate. Lack of knowledge of the simulation languages and techniques keeps many analysts away from simulations.

Level of accuracy desired is another important consideration. In general, analytical modeling requires so many simplifications and assumptions that if the results turn out to be accurate, even the analysts are surprised. Simulations can incorporate more details and require less assumptions than analytical modeling and, thus, more often are closer to reality. Measurements, although they sound like the real thing, may not give accurate results simply because many of the environmental parameters, such as system configuration, type of workload, and time of the measurement, may be unique to the experiment. Also, the parameters may not represent the range of variables found in the real world. Thus, the accuracy of results can vary from very high to none when using the measurements technique.

It must be pointed out that level of accuracy and correctness of conclusions are not identical. A result that is correct up to the tenth decimal place may be misunderstood or misinterpreted; thus wrong conclusions can be drawn.

The goal of every performance study is either to compare different alternatives or to find the optimal parameter value. Analytical models generally provide the best insight into the effects of various parameters and their interactions. With simulations, it may be possible to search the space of parameter values for the optimal combination, but often it is not clear what the trade-off is among different parameters. Measurement is the least desirable technique in this respect. It is not easy to tell if the improved performance is a result of some random change in environment or due to the particular parameter setting.

Cost allocated for the project is also important. Measurement requires real equipment, instruments, and time. It is the most costly of the three techniques. Cost, along with the ease of being able to change configurations, is often the reason for developing simulations for expensive systems. Analytical modeling requires only paper and pencils (in addition to the analyst's time). Analytical modeling is therefore the cheapest alternative.

Previous | Table of Contents | Next

Products | Contact Us | About Us | Privacy | Ad Info | Home

Art of Computer Systems Performance Analysis Techniques For Experimental Design Measurements Simulation And Modeling
by Raj Jain
Wiley Computer Publishing, John Wiley & Sons, Inc.
ISBN: 0471503363 **Pub Date:** 05/01/91

Previous | Table of Contents | Next

Saleability of results is probably the key justification when considering the expense and the labor of measurements. It is much easier to convince others if it is a real measurement. Most people are skeptical of analytical results simply because they do not understand the technique or the final result. In fact, people who develop new analytical modeling techniques often validate them by using simulations or actual measurements.

Sometimes it is helpful to use two or more techniques simultaneously. For example, you may use simulation and analytical modeling together to verify and validate the results of each one. Until proven guilty, every person should be presumed innocent. The performance counterpart of this statement is *until validated, all evaluation results are suspect*. This leads us to the following three rules of validation:

- Do not trust the results of a simulation model until they have been validated by analytical modeling or measurements.
- Do not trust the results of an analytical model until they have been validated by a simulation model or measurements.
- Do not trust the results of a measurement until they have been validated by simulation or analytical modeling.

In particular, the need for the third rule regarding validation of measurement results should be emphasized. This is the most commonly ignored of the three rules. Measurements are as susceptible to experimental errors and bugs as the other two techniques. The only requirement for validation is that the results should not be counterintuitive. This method of validation, called expert’s intuition, is commonly used for simulation models. This and other validation methods can be used for measurement and analytical results and are discussed in Section 25.2.

Two or more techniques can also be used sequentially. For example, in one case, a simple analytical model was used to find the appropriate range for system parameters and a simulation was used later to study the performance in that range. This reduced the number of simulation runs considerably and resulted in a more productive use of resources.

3.2 SELECTING PERFORMANCE METRICS

For each performance study, a set of performance criteria or metrics must be chosen. One way to prepare this set is to list the services offered by the system. For each service request made to the system, there are several possible outcomes. Generally, these outcomes can be classified into three categories, as shown in Figure 3.1. The system may perform the service correctly, incorrectly, or refuse to perform the service. For example, a gateway in a computer network offers the service of forwarding packets to the specified destinations on heterogeneous networks. When presented with a packet, it may forward the packet correctly, it may forward it to the wrong destination, or it may be down, in which case it will not forward it at all. Similarly, a database offers the service of responding to queries. When presented with a query, it may answer correctly, it may answer incorrectly, or it may be down and not answer it at all.

If the system performs the service correctly, its performance is measured by the time taken to perform the service, the rate at which the service is performed, and the resources consumed while performing the service. These three metrics related to **time-rate-resource** for successful performance are also called **responsiveness**, **productivity**, and **utilization** metrics, respectively. For example, the responsiveness of a network gateway is measured by its response time—the time interval between arrival of a packet and its successful delivery. The gateway's productivity is measured by its throughput—the number of packets forwarded per unit of time. The utilization gives an indication of the percentage of time the resources of the gateway are busy for the given load level. The resource with the highest utilization is called the **bottleneck**. Performance optimizations at this resource offer the highest payoff. Finding the utilization of various resources inside the system is thus an important part of performance evaluation.

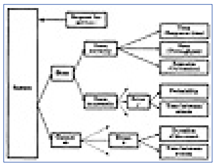

FIGURE 3.1 Three possible outcomes of a service request.

If the system performs the service incorrectly, an **error** is said to have occurred. It is helpful to classify errors and to determine the probabilities of each class of errors. For example, in the case of the gateway, we may want to find the probability of single-bit errors, two-bit errors, and so on. We may also want to find the probability of a packet being partially delivered (fragment).

If the system does not perform the service, it is said to be *down*, *failed*, or *unavailable*. Once again, it is helpful to classify the failure modes and to determine the probabilities of each class. For example, the gateway may be unavailable 0.01% of the time due to processor failure and 0.03% due to software failure.

The metrics associated with the three outcomes, namely successful service, error, and unavailability, are also called **speed, reliability**, and **availability** metrics. It should be obvious that for each service offered by the system, one would have a number of speed metrics, a number of reliability metrics, and a number of availability metrics. Most systems offer more than one service, and thus the number of metrics grows proportionately.

For many metrics, the mean value is all that is important. However, do not overlook the effect of variability. For example, a high mean response time of a timesharing system as well as a high variability of the response time both may degrade the productivity significantly. If this is the case, you need to study both of these metrics.

In computer systems shared by many users, two types of performance metrics need to be considered: individual and global. Individual metrics reflect the utility of each user, while the global metrics reflect the systemwide utility. The resource utilization, reliability, and availability are global metrics, while response time and throughput may be measured for each individual as well as globally for the system. There are cases when the decision that optimizes individual metrics is different from the one that optimizes the system metric. For example, in computer networks, the performance is measured by throughput (packets per second). In a system where the total number of packets allowed in the network is kept constant, increasing the number of packets from one source may lead to increasing its throughput, but it may also decrease someone else's throughput. Thus, both the systemwide throughput and its distribution among individual users must be studied. Using only the system throughput or the individual throughput may lead to unfair situations.

Given a number of metrics, use the following considerations to select a subset: low variability, nonredundancy, and completeness. Low variability helps reduce the number of repetitions required to obtain a given level of statistical confidence. Confidence level is explained in Chapter 12. Metrics that are ratios of two variables generally have a larger variability than either of the two variables and should be avoided if possible.

| Previous | Table of Contents | Next |

HOME SUBSCRIBE SEARCH FAQ SITEMAP CONTACT US

Products | Contact Us | About Us | Privacy | Ad Info | Home

KEYWORD SEARCH

GO!

Search Tips

Advanced Search

PUBLICATION LOOKUP

GO!

Title

JUMP TO TOPIC

Art of Computer Systems Performance Analysis Techniques For Experimental Design Measurements Simulation And Modeling
by Raj Jain
Wiley Computer Publishing, John Wiley & Sons, Inc.
ISBN: 0471503363 **Pub Date:** 05/01/91

Search this book:

GO!

Previous | Table of Contents | Next

If two metrics give essentially the same information, it is less confusing to study only one. This is not always obvious, however. For example, in computer networks, the average waiting time in a queue is equal to the quotient of the average queue length and the arrival rate. Studying the average queue lengths in addition to average waiting time may not provide any additional insights.

Finally, the set of metrics included in the study should be complete. All possible outcomes should be reflected in the set of performance metrics. For example, in a study comparing different protocols on a computer network, one protocol was chosen as the best until it was found that the best protocol led to the highest number of premature circuit disconnections. The *probability of disconnection* was then added to the set of performance metrics.

> **Case Study 3.1** Consider the problem of comparing two different congestion control algorithms for computer networks. A computer network consists of a number of **end systems** interconnected via a number of **intermediate systems**. The end systems send packets to other end systems on the network. The intermediate systems forward the packets along the right path. The problem of congestion occurs when the number of packets waiting at an intermediate system exceeds the system's buffering capacity and some of the packets have to be dropped.
>
> The system in this case consists of the network, and the only service under consideration is that of packet forwarding. When a network user sends a block of packets to another end station called **destination**, there are four possible outcomes:
>
> **1.** Some packets are delivered in order to the correct destination.
>
> **2.** Some packets are delivered out of order to the destination.
>
> **3.** Some packets are delivered more than once to the destination (duplicate packets).
>
> **4.** Some packets are dropped on the way (lost packets).
>
> For packets delivered in order, straightforward application of the time-rate-resource metrics produces the following list:
>
> **1.** Response time: the delay inside the network for individual packets.
>
> **2.** Throughput: the number of packets per unit of time.
>
> **3.** Processor time per packet on the source end system.
>
> **4.** Processor time per packet on the destination end systems.
>
> **5.** Processor time per packet on the intermediate systems.

The response time determines the time that a packet has to be kept at the source end station using up its memory resources. Lower response time is considered better. The throughput is the performance as seen by the user. Larger throughput is considered better.

The variability of the response time is also important since a highly variant response results in unnecessary retransmissions. Thus, the variance of the response time became the sixth metric.

Out-of-order packets are undesirable since they cannot generally be delivered to the user immediately. In many systems, the out-of-order packets are discarded at the destination end systems. In others, they are stored in system buffers awaiting arrival of intervening packets. In either case, out-of-order arrivals cause additional overhead. Thus, the probability of out-of-order arrivals was the seventh metric.

Duplicate packets consume the network resources without any use. The probability of duplicate packets was therefore the eighth metric.

Lost packets are undesirable for obvious reasons. The probability of lost packets is the ninth metric. Excessive losses result in excessive retransmissions and could cause some user connections to be broken prematurely; thus the probability of disconnect was added as the tenth metric.

The network is a multiuser system. It is necessary that all users be treated fairly. Therefore, fairness was added as the eleventh metric. It is defined as a function of variability of throughput across users. For any given set of user throughputs (x_1, x_2,..., x_n), the following function can be used to assign a fairness index to the set:

$$f(x_1, x_2, \ldots, x_n) = \frac{\left(\sum_{i=1}^{n} x_i\right)^2}{n \sum_{i=1}^{n} x_i^2}$$

For all nonnegative values of x_i’s, the fairness index always lies between 0 and 1. If all users receive equal throughput, the fairness index is 1. If only *k* of the *n* users receive equal throughput and the remaining *n* - *k* users receive zero throughput, the fairness index is *k/n*. For other distributions also, the metric gives intuitive fairness values.

After a few experiments, it was clear that throughput and delay were really redundant metrics. All schemes that resulted in higher throughput also resulted in higher delay. Therefore, the two metrics were removed from the list and instead a combined metric called **power**, which is defined as the ratio of throughput to response time, was used. A higher power meant either a higher throughput or a lower delay; in either case it was considered better than a lower power.

The variance in response time was also dropped since it was redundant with the probability of duplication and the probability of disconnection. A higher variance resulted in a higher probability of duplication and a higher probability of premature disconnection.

Thus, in this study a set of nine metrics were used to compare different congestion control algorithms.

3.3 COMMONLY USED PERFORMANCE METRICS

This section defines and explains some of the commonly used performance metrics. In each case, the definition proposed is only one of many possibilities. Some definitions will need to be changed to suit certain applications.

Response time is defined as the interval between a user’s request and the system response, as shown in Figure 3.2a. This definition, however, is simplistic since the requests as well as the responses are not instantaneous. The users spend time typing the request and the system takes time outputting the response, as shown in Figure 3.2b. There are two possible definitions of the response time in this case. It can be defined as either the interval between the end of a request submission and the beginning of the corresponding response from the system or as the interval between the end of a request submission and the end of the corresponding response from the system. Both definitions are acceptable as long as they are clearly specified. The second definition is preferable if the time between the beginning and the end of the response is long. Following this definition, the response time for interactive users in a timesharing system would be the interval between striking the last return (or enter) key and the receipt of the *last* character of the system’s response.

FIGURE 3.2 Response time definition.

For a batch stream, responsiveness is measured by **turnaround time**, which is the time between the submission of a batch job and the completion of its output. Notice that the time to read the input is included in the turnaround time.

The time between submission of a request and the beginning of its execution by the system is called the **reaction time**. To measure the reaction time, one has to able to monitor the actions inside a system since the beginning of the execution may not correspond to any externally visible event. For example, in timesharing systems, the interval between a user’s last key stroke and the user’s process receiving the first CPU quantum would be called reaction time.

| Previous | Table of Contents | Next |

HOME SUBSCRIBE SEARCH FAQ SITEMAP CONTACT US

Products | Contact Us | About Us | Privacy | Ad Info | Home

KEYWORD SEARCH

GO!

Search Tips

Advanced Search

PUBLICATION LOOKUP

GO!

Title

JUMP TO TOPIC

Art of Computer Systems Performance Analysis Techniques For Experimental Design Measurements Simulation And Modeling
by Raj Jain
Wiley Computer Publishing, John Wiley & Sons, Inc.
ISBN: 0471503363 **Pub Date:** 05/01/91

Search this book:

GO!

Previous | Table of Contents | Next

The response time of a system generally increases as the load on the system increases. The ratio of response time at a particular load to that at the minimum load is called the **stretch factor**. For a timesharing system, for example, the stretch factor is defined as the ratio of the response time with multiprogramming to that without multiprogramming.

Throughput is defined as the rate (requests per unit of time) at which the requests can be serviced by the system. For batch streams, the throughput is measured in jobs per second. For interactive systems, the throughput is measured in requests per second. For CPUs, the throughput is measured in Millions of Instructions Per Second **(MIPS)**, or Millions of Floating-Point Operations Per Second **(MFLOPS)**. For networks, the throughput is measured in packets per second **(pps)** or bits per second **(bps)**. For transactions processing systems, the throughput is measured in Transactions Per Second **(TPS)**.

The throughput of a system generally increases as the load on the system initially increases. After a certain load, the throughput stops increasing; in most cases, it may even start decreasing, as shown in Figure 3.3. The maximum achievable throughput under ideal workload conditions is called **nominal capacity** of the system. For computer networks, the nominal capacity is called the **bandwidth** and is usually expressed in bits per second. Often the response time at maximum throughput is too high to be acceptable. In such cases, it is more interesting to know the maximum throughput achievable without exceeding a prespecified response time limit. This may be called the **usable capacity** of the system. In many applications, the knee of the throughput or the response-time curve is considered the optimal operating point. As shown in Figure 3.3, this is the point beyond which the response time increases rapidly as a function of the load but the gain in throughput is small. Before the knee, the response time does not increase significantly but the throughput rises as the load increases. The throughput at the knee is called the **knee capacity** of the system. It is also common to measure capacity in terms of load, for example, the number of users rather than the throughput. Once again, it is a good idea to precisely define the metrics and their units before using them in a performance evaluation project.

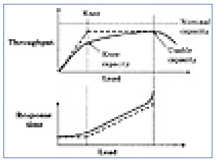

FIGURE 3.3 Capacity of a system.

The ratio of maximum achievable throughput (usable capacity) to nominal capacity is called the **efficiency**.

For example, if the maximum throughput from a 100-Mbps (megabits per second) Local Area Network (LAN) is only 85 Mbps, its efficiency is 85%. The term efficiency is also used for multiprocessor systems. The ratio of the performance of an *n*-processor system to that of a one-processor system is its efficiency, as shown in Figure 3.4. The performance is usually measured in terms of MIPS or MFLOPS.

The **utilization** of a resource is measured as the fraction of time the resource is busy servicing requests. Thus this is the ratio of busy time and total elapsed time over a given period. The period during which a resource is not being used is called the **idle time**. System managers are often interested in balancing the load so that no one resource is utilized more than others. Of course, this is not always possible.

Some resources, such as processors, are always either busy or idle, so their utilization in terms of ratio of busy time to total time makes sense. For other resources, such as memory, only a fraction of the resource may be used at a given time; their utilization is measured as the average fraction used over an interval.

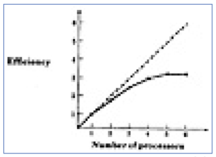

FIGURE 3.4 Efficiency of a multiprocessor system.

The **reliability** of a system is usually measured by the probability of errors or by the mean time between errors. The latter is often specified as **error-free seconds**.

The **availability** of a system is defined as the fraction of the time the system is available to service users' requests. The time during which the system is not available is called **downtime;** the time during which the system is available is called **uptime**. Often the mean uptime, better known as the **Mean Time To Failure (MTTF)**, is a better indicator since a small downtime and a small uptime combination may result in a high-availability measure, but the users may not be able to get any service if the uptime is less than the time required to complete a service.

In system procurement studies, the **cost/performance ratio** is commonly used as a metric for comparing two or more systems. The cost includes the cost of hardware/software licensing, installation, and maintenance over a given number of years. The performance is measured in terms of throughput under a given response time constraint. For example, two transaction processing systems may be compared in terms of dollars per TPS.

3.4 UTILITY CLASSIFICATION OF PERFORMANCE METRICS

Depending upon the utility function of a performance metric, it can be categorized into three classes:

- *Higher is Better* or **HB**. System users and system managers prefer higher values of such metrics. System throughput is an example of an HB metric.
- *Lower is Better* or **LB**. System users and system managers prefer smaller values of such metrics. Response time is an example of an LB metric.
- *Nominal is Best* or **NB**. Both high and low values are undesirable. A particular value in the middle is considered the best. Utilization is an example of an NB characteristic. Very high utilization is considered bad by the users since their response times are high. Very low utilization is considered bad by system managers since the system resources are not being used. Some value in the range of 50 to 75% may be considered best by both users and system managers.

Figure 3.5 shows hypothetical graphs of utility of the three classes of metrics. The utility class of a metric is useful in data presentation, for example, in Kiviat graphs discussed later in Section 10.6.

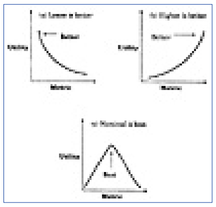

FIGURE 3.5 Types of metrics.

3.5 SETTING PERFORMANCE REQUIREMENTS

One problem performance analysts are faced with repeatedly is that of specifying performance requirements for a system to be acquired or designed. A general method to specify such requirements is presented in this section and is illustrated with a case study.

To begin, consider these typical requirement statements:

The system should be both processing and memory efficient. It should not create excessive overhead.

There should be an extremely low probability that the network will duplicate a packet, deliver a packet to the wrong destination, or change the data in a packet.

Previous | Table of Contents | Next

HOME SUBSCRIBE SEARCH FAQ SITEMAP CONTACT US

Products | Contact Us | About Us | Privacy | Ad Info | Home

KEYWORD SEARCH

GO!

Search Tips

Advanced Search

PUBLICATION LOOKUP

GO!

Title

JUMP TO TOPIC

Art of Computer Systems Performance Analysis Techniques For Experimental Design Measurements Simulation And Modeling
by Raj Jain
Wiley Computer Publishing, John Wiley & Sons, Inc.
ISBN: 0471503363   **Pub Date:** 05/01/91

Search this book:

GO!

Previous | Table of Contents | Next

These requirement statements are unacceptable since they suffer from one or more of the following problems:

1. *Nonspecific:* No clear numbers are specified. Qualitative words such as low, high, rare, and extremely small are used instead.

2. *Nonmeasurable:* There is no way to measure a system and verify that it meets the requirement.

3. *Nonacceptable:* Numerical values of requirements, if specified, are set based upon what can be achieved or what looks good. If an attempt is made to set the requirements realistically, they turn out to be so low that they become unacceptable.

4. *Nonrealizable:* Often, requirements are set high so that they look good. However, such requirements may not be realizable.

5. *Nonthorough:* No attempt is made to specify a possible outcomes.

What all these problems lack can be summarized in one word: **SMART**. That is, the requirements must be **S**pecific, **M**easurable, **A**cceptable, **R**ealizable, and **T**horough. Specificity precludes the use of words like “low probability” and “rare.” Measurability requires verification that a given system meets the requirements. Acceptability and realizability demand new configuration limits or architectural decisions so that the requirements are high enough to be acceptable and low enough to be achievable. Thoroughness includes all possible outcomes and failure modes. As discussed in Section 3.2, every system provides a set of services. For every request for a service, there are three possible outcomes: successful performance, incorrect performance, and nonperformance. Thoroughness dictates that the requirements be set on all possible outcomes.

For the requirements to be meaningful, specify bounds, if any, on the configurations, workloads, and environments.

These ideas are illustrated in the following case study.

Case Study 3.2 Consider the problem of specifying the performance requirements for a high-speed LAN system. A LAN basically provides the service of transporting frames (or packets) to the specified destination station. Given a user request to send a frame to destination station D, there are three categories of outcomes: the frame is correctly delivered to D, incorrectly delivered (delivered to a wrong destination or with an error indication to D), or not delivered at all. The performance requirements for these three categories of outcomes were specified as follows:

1. *Speed:* If the packet is correctly delivered, the time taken to deliver it and the rate at which it is delivered are important. This leads to the following two requirements:

- **(a)** The access delay at any station should be less than 1 second.
- **(b)** Sustained throughput must be at least 80 Mbits/sec.

2. *Reliability:* Five different error modes were considered important. Each of these error modes causes a different amount of damage and, hence, has a different level of acceptability. The probability requirements for each of these error modes and their combined effect are specified as follows:

- **(a)** The probability of any bit being in error must be less than 10^{-7}.
- **(b)** The probability of any frame being in error (with error indication set) must be less than 1%.
- **(c)** The probability of a frame in error being delivered without error indication must be less than 10^{-15}.
- **(d)** The probability of a frame being misdelivered due to an undetected error in the destination address must be less than 10^{-18}.
- **(e)** The probability of a frame being delivered more than once (duplicate) must be less than 10^{-5}.
- **(f)** The probability of losing a frame on the LAN (due to all sorts of errors) must be less than 1%.

3. *Availability:* Two fault modes were considered significant. The first was the time lost due to the network reinitializations, and the second was time lost due to permanent failures requiring field service calls. The requirements for frequency and duration of these fault modes were specified as follows:

- **(a)** The mean time to initialize the LAN must be less than 15 milliseconds.
- **(b)** The mean time between LAN initializations must be at least 1 minute.
- **(c)** The mean time to repair a LAN must be less than 1 hour. (LAN partitions may be operational during this period.)
- **(d)** The mean time between LAN partitioning must be at least half a week.

All of the numerical values specified above were checked for realizability by analytical modeling, which showed that LAN systems satisfying these requirements were feasible.

EXERCISES

3.1 What methodology would you choose?

- **a.** To select a personal computer for yourself
- **b.** To select 1000 workstations for your company
- **c.** To compare two spread sheet packages
- **d.** To compare two data-flow architectures, if the answer was required:
  - **i.** Yesterday
  - **ii.** Next quarter
  - **iii.** Next year

3.2 Make a complete list of metrics to compare

- **a.** Two personal computers
- **b.** Two database systems
- **c.** Two disk drives
- **d.** Two window systems

FURTHER READING FOR PART I

There are a number of books on computer systems performance evaluation. However, most of these books emphasize only one of the three evaluation techniques.

The book by Lazowska et al. (1984) has an excellent treatment of queueing models. Lavenberg (1983) provides a good review of queueing models and simulation. Ferrari (1978), Ferrari, Serazzi, and Zeigner (1983), and Howard (1983) have good discussions of measurement techniques and their applications to a wide variety of performance problems, such as system tuning, workload characterization, program tuning, and others.

Other books on performance analysis and modeling are Gelenbe and Mitrani (1980), Kobayashi (1978), Leung (1987), McKerrow (1987), Molloy (1989), and Sauer and Chandy (1981).