On the Foundations of Artificial Workload Design

Domenico Ferrari

Computer Science Division, EECS Department
and the Electronics Research Laboratory
University of California, Berkeley

ABSTRACT

The principles on which artificial workload model design is currently based are reviewed, and found wanting for three main reasons: their resource orientation, with the selection of resources often unrelated to the performance impact of resource demands; their avoiding to define an accuracy criterion for the resulting workload model; and their ignoring the dynamics of the workload to be modeled. An approach to establishing conceptual foundations for the design of interactive artificial workloads is described. The approach tries to take the problems found in current design methods into proper account, and to delimit the domains of applicability of these methods. In doing so, it also provides guidance for some of the decisions to be made in workload model design.

CR Categories and Subject Descriptors: D.4.8 [**Operating Systems**]: Performance - *Measurements, modeling and prediction*; C.4 [**Computer Systems Organization**]: Performance of Systems - *measurement techniques, modeling techniques*.

General Term: Performance.

Additional Key Words and Phrases: artificial workloads, benchmark design, clustering, interactive workloads, performance-oriented workload models, representativeness, sampling, workload characterization, workload modeling.

1. Introduction

Workload modeling is one of the most fundamental aspects of performance evaluation. No system evaluation study can avoid confronting the problem of modeling or at least selecting one or more workloads [FERR72]. This is easily proved by observing that the performance indices to be evaluated in such a study are critically dependent on the workload processed by the system [FERR78]. All performance analysis techniques, i.e., those techniques that can provide us with the values of a system's performance indices, require one or more workload models to be built. This is not only the case of analytic and simulation modeling techniques, but also of both types of measurement approaches: the one in which the system to be measured is driven by an artificial workload, that is supposed to represent a real (current or future) workload, as well as the

This research has been supported in part by the National Science Foundation under Grants MCS78-24818 and MCS80-12900.

one which uses a sample of the system's natural workload. To realize that this sample constitutes a workload model, it is sufficient to consider the criteria according to which it is usually chosen: for instance, the natural workload processed on a particular working day between 11 a.m. and 12 noon may be selected to represent the heavy loads the system to be measured is processing late in the morning of most working days.

In this paper, we restrict our attention to those workload models that are to be used as system drivers in measurement experiments. In other terms, we consider only workload models that are *artificial* and *executable*. This category of models includes *benchmarks* and *scripts* [BENW75] [FERR78], and makes experiments better reproducible than when they are driven by natural workload models. Thus, given reproducibility requirements are satisfied by an artificial model in a much shorter time than by its natural counterpart. With respect to natural models, artificial models are certainly more expensive to build, potentially less accurate, and more disruptive of a system's operation, as they are to be run on a dedicated system. However, the much greater control of an experiment they provide the experimenter with, their ability to represent non-existent (e.g., future) workloads, and their much easier portability have made them indispensable in several types of measurement studies, for example, in competitive procurement, system sizing, capacity planning, and performance comparisons for marketing purposes.

The subject of artificial workload design has not received as much attention in the technical literature as its importance deserves. Artificial workloads have been built for many years, and are being built now, in large numbers, following in most cases intuition and common sense rather than systematic, scientifically sound procedures. A relatively small number of researchers has ventured into this risky and difficult field, and proposed design techniques whose use has substantially increased during the last decade. Section 2 briefly reviews the philosophical bases of these techniques and presents three fundamental questions that may be asked about them. Answers to these questions are proposed and discussed in Sections 3 and 4. The latter section presents new conceptual bases for the current techniques, focusing on the design of artificial models for interactive systems. The treatment in Section 4 allows one to delimit the domain of validity and applicability of the design techniques that have been proposed, and to obtain guidance in making the various decisions one is confronted with when designing a workload model. These results are summarized in Section 5.

2. Current Design Methods

Most of the systematic design methods that have been proposed follow the procedure outlined below when they are to construct an executable model of a real, measurable workload.

Step 1. The type of the *basic components* of the workload (and of the model) is identified. Common component types are the job, the command or interaction, the transaction, and the job step.

Step 2. The parameters to be used to characterize each component are chosen. Usually, these parameters represent physical resource demands, or amounts of resources consumed, by each workload component, but in some cases they correspond to logical or even functional demands. Examples of physical resources are CPU's, main memory, I/O channels and devices, communication networks, terminals, and other peripherals. Logical resources are language processors and other software subsystems (e.g., editors, debuggers, file systems,

database systems). Examples of functional resources are command classes (compiling, editing, file manipulation, and so on), transaction classes, and job classes.

Step 3. The values of the parameters selected in Step 2 are measured for each component of the workload to be modeled. Each component will be represented by a tuple of values, not all necessarily numerical.

Step 4. Statistical techniques are applied to the tuples of parameter values measured in Step 3 with the objective of identifying a small representative subset of the set of tuples. These techniques include sampling and clustering methods. In some cases, sampling is applied to the frequency distributions of the numerical parameter values measured in Step 3 [SCHW72]. In other cases, to the population of workload components in the workload [SHOP70] [WOOD71] or to the joint frequency distribution of their parameters [SREE74]. Clustering [AGRA76] [ARTI78] groups together workload components whose locations in the hyperspace of the characterizing parameters are relatively close according to a suitable definition of distance; a small number of tuples is then selected from each cluster to represent it in the model. Principal components analysis [SERA81] and feature selection techniques [MAMR77] have also been applied. In all cases, the model has a much smaller number of tuples, hence of components, than the workload to be modeled.

Step 5. Each tuple selected in Step 4 is replaced by a workload component that is (at least approximately) characterized by the tuple. The set of all these components constitutes the workload model. The components of the model do not have to be chosen from among those of the workload to be modeled: they may be synthetic programs with argument values selected so as to make their characterizing parameters equal (or close) to those in the tuple they are intended to represent.

The general procedure just described has been applied in a variety of ways, including some that would seem too different from the others to be amenable to the same conceptual framework. An example of such a design method is the one which consists of sampling the real job stream of the system to extract the components of the model [SHOP70]. Even this method may be seen as following the above procedure, though in this case Step 2 does not require any work, as each job is characterized by itself and not by its parameters, Steps 3 and 4 are performed simultaneously, and Step 5 may also be performed at the same time as 3 and 4 but not alway is [WOOD71]. In some cases, two types of components are successively used: for example, Haring [HARI82] clusters terminal sessions according to the percentages of the various software resources they request, and then builds a probabilistic model of each class of sessions taking the requests for each class of software resources as the states of a session. Another example is provided by Artis [ARTI78], who clusters jobs based on their physical resource consumptions, then defines mixes in terms of the clusters the jobs in each mix belong to, and clusters mixes according to their compositions. These more sophisticated methods clearly follow procedures that are more complex than the one described above; however, our framework can be used to describe these methods without changing the basic steps, simply by allowing different paths through the steps (e.g., loops) to be followed.

In all cases, the above procedure for designing artificial workloads raises three major problems that will now be discussed.

Problem 1 The parameters selected in Step 2 are resource demands at various levels of abstraction (at the physical level in most cases, but sometimes at the logical or even at the functional level). What are the resources whose demands are to be included in the characterization? The usual answer is: those whose consumptions have a significant impact on performance. No effort, however, is usually made to determine the identities of these resources. Thus, each artificial workload that is built is based on an *ad-hoc* list of resources, whose necessity and sufficiency is not known, and which often differs for unexplained reasons from the lists used in similar installations. For example, some lists include the number of characters output by each command, some others do not. Is this number important or not? When is it important?

Problem 2 There is no provision for guaranteeing that the model resulting from the procedure will be acceptably accurate. In fact, since the accuracy (or *representativeness*) of a workload model is not even quantitatively defined, there is no criterion by which it can be evaluated. And yet, accuracy is the primary virtue of any model. An inaccurate model is obviously useless.

Problem 3 The statistical techniques used in Step 4 are normally applied to a population of tuples from which any temporal information has disappeared. With only a few exceptions [ARTI78] [FERR81] [CALZ82], in model design methods component arrival times, component sequences, and mix compositions are usually lost. Reducing workloads from time series to populations of components is equivalent to neglecting their dynamics, or stating that their dynamic behavior is unimportant. Our empirical observations show that this need not be the case. Is an artificial workload produced by one of the current design methods a good model in spite of these observations? Should we not discard these methods and try to devise new ones, which would take the dynamics of the workload to be modeled into proper account?

3. The Performance-Oriented Criterion

Before presenting our approach to the problems outlined in Section 2, we summarize in this section a definition of workload model accuracy that has been proposed in the past [FERR72] [FERR78] [FERR79] [FERR81]. This definition will be useful in providing a solution not only to Problem 2 above, but, because of the strict interrelationships among the three problems, especially between Problems 1 and 2, to the others as well.

All definitions of accuracy based on the similarity between the modeled workload and the workload model are questionable because of our lack of knowledge about workloads and their interactions with system organizations. Nobody questions the validity of experiments performed on models of automobiles in a wind tunnel, even though these models are much smaller in size than real automobiles, are built of a different material, and cannot run: as shown by theoretical and experimental results, aerodynamic resistance is not influenced by the material or by the presence of a running engine, but only by the shape of the surface, and the actual resistance of an object can be computed from the resistance of a smaller object with the same shape. No such theoretical or experimental results are usually invoked to justify the selection of the features in terms of which a workload is characterized and reduced to a model (see the discussion of Problem 1 in Section 2).

To arrive at a sensible definition of accuracy or representativeness, one has to begin by asking what is the purpose of building a workload model. If the objective is the evaluation of a system's performance, then the definition must be performance-oriented.

Definition. Given a system S and a set of performance indices P, workload W'' is a (perfectly accurate) model of workload W' with respect to S and P if

$$P'' = P',$$

where P' and P'' are the performances with which S processes W' and W'', respectively.

This definition is schematically depicted in Figure 1. The corresponding definition of accuracy only requires, to be specified, that a metric for the space of performance indices be defined. Note that set P may include variables with different physical dimensions, distributions, time series, and other types of indices that may make defining such a metric a non-straightforward task.

The above definition is not always applicable in practice, but can always serve as the basis for conceptual experiments. Even more importantly, it provides an explicit criterion for evaluating selections of workload features by replacing the vague notion of similarity between the workload and the model with the quantitatively verifiable closeness of the values of their performance indices.

Except for certain cases [FERR81], this definition of accuracy does not directly suggest a method for designing an artificial workload. However, it provides guidance in performing Step 2 of the procedure outlined in Section 2. All those, and only those, workload parameters that have a noticeable influence on P at the desired level of abstraction should be selected. This influence, however, cannot be guessed, but must be determined by experimentation or by sensitivity analyses based on a good model of the system.

4. On the Design of an Artificial Interactive Workload

We begin our discussion of the foundations of interactive workload model design by making three assumptions.

Assumption 1. The system we deal with is an interactive system whose performance is satisfactorily analyzable by a product-form closed queueing network model of the type represented in Figure 2. In other words, the values of the performance indices we are interested in can be calculated with acceptable accuracy by solving the queueing network in Figure 2, that satisfies the conditions of the so-called BCMP theorem [BASK75]. Note that the central subsystem in the figure is an open network of (N-1) stations, and that it may include load-dependent servers, some of which may have resulted from applying flow-equivalent approximation techniques to a more complex, non-product-form queueing model of the system.

Assumption 2. The behavior of each of the m interactive users of the system can be suitably represented by a probabilistic graph like the one depicted in Figure 3. Each node in the graph represents an interactive command type, with the exception of node 0, which is the "dormant node". Users who are not using the system reside in node 0. When a terminal session starts, the state of the user becomes 1 (the "login node"). During the session, other commands are executed (i.e., other nodes are visited) following the arcs of the graph. At the beginning of each terminal time period, a user chooses the next command based on the probabilities appearing in the *user behavior graph* (e.g., the one in Figure 3). It is clear from this description that the R nodes in the graph, that is, the command types, are to be modeled as R different classes of customers in the

network in Figure 2, and that a user may change class only when moving from a station in the central subsystem to station 1, i.e., the terminals station. Also, if all the sequences of commands constituting user sessions are realizations of a single graph, the queueing network has a single chain. Note that node 0 can be eliminated by adding the times spent in it to the terminal times of node 1.

Assumption 3. The interactive workload to be modeled and the workload model to be constructed are stationary. This means that the desired workload model is not intended to reproduce any particular dynamic variation in workload characteristics, but is rather to exhibit about the same time-invariant distributions of characteristics as the original workload. Thus, our treatment addresses only some of the issues related to workload dynamics (see Problem 3 in Section 2), and does not claim to be applicable to cases in which some specific dynamic aspects of a given workload are to be reproduced by the model.

The workload of the interactive system being considered may be described as a set of at least partially overlapping sequences of commands issued by terminal users. The basic component of such a workload is the command or interaction. A current model design method consists of reducing the numbers of command types that appear in the workload proportionally, so as to preserve their relative frequencies. Doing so means ignoring the sequential links existing among command types: for instance, in Figure 3, **1** is never followed directly by **8**, and **7** is always preceded by **5**. Under the assumptions made above, however, the method may be valid if the conditions of the following theorem are satisfied.

Theorem 1. The equilibrium state probabilities of the queueing network in Figure 2 are invariant with respect to any change in the user behavior graph which does not modify the visit ratios of the command types.

Proof. The BCMP theorem [BASK75] shows that the equilibrium probability of state $(y_1, y_2, \ldots y_N)$, where $y_i = (n_{i1}, n_{i2}, \ldots n_{iR})$ and n_{ir} is the number of users of class r in station i, is given by $C\, g_1(y_1) g_2(y_2) \ldots g_i(y_N)$, where C is a normalization constant and $g_i(y_i)$ is a function of the n_{ir}'s, the μ_{ir}'s (μ_{ir} is the mean service time in station i for users of class r), and the e_{ir}'s, that are the solutions (to within a multiplicative constant) of the set of equations

$$e_{js} = \sum_{\substack{i=1,N \\ r=1,R}} e_{ir}\, p_{i,r;j,s}, \quad (j = 1,\ldots N;\ s = 1,..R). \tag{1}$$

Note that e_{ir} can be interpreted as the relative arrival rate to user state (i, r), and that the symbol $p_{i,r;j,s}$ represents the probability that a user of class r exiting from station i will require service at station j and move to class s. Under the assumptions made above, and recalling that number 1 has been assigned to the terminals station, we have

$$p_{i,r;j,s} = \begin{cases} b_{ij,r}\, p_{rs} & \text{for } j = 1,\ i \neq 1 \\ b_{ij,r} & \text{for } j \neq 1,\ r = s \\ 0 & \text{for all other cases,} \end{cases} \tag{2}$$

where $b_{ij,r}$ is the probability that a user being in class r and at station i will go to station j, and p_{rs} is the probability of visiting class s coming from class r.

Substituting (2) into equations (1), we obtain

$$e_{1s} = \sum_{r=1}^{R} \left(p_{rs} \sum_{i=2}^{N} e_{ir}\, b_{i1,r}\right),$$

$$e_{js} = \sum_{i=1}^{N} e_{ir}\, b_{ij,rs}, \quad (j = 2,\ldots N), (s = 1,\ldots R). \tag{3}$$

Since changes of class may only occur when entering station 1, the relative departure rate from user state $(1, r)$ can be written as $\sum_{i=2}^{N} e_{ir}\, b_{i1,rr}$, and equals the relative arrival rate e_{1r}. Hence, from (3) we have

$$e_{1s} = \sum_{r=1}^{R} p_{rs}\, e_{1r}, \quad (s = 1,\ldots R). \tag{4}$$

From (1), we have that the ratio $\frac{e_{ir}}{e_{js}}$ equals the ratio between the number of visits to user state (i, r) and the number of visits to user state (j, s) during a given time interval at the equilibrium. In the particular type of closed network considered here, the value of e_{1s} is proportional to the number of visits (during a given time interval) to class s (i.e., to node s in the user behavior graph). Equations (4) show that the ratio $\frac{e_{1r}}{e_{1s}}$ equals the ratio between the number of visits to class r and the number of visits to class s during a given time interval. Any transformation of the interclass transition probability matrix $[p_{rs}]$ that does not modify the class visit ratios will not change any of the $\frac{e_{ir}}{e_{js}}$ ratios, since the second set in (3), when the ratios $\frac{e_{1r}}{e_{1s}}$ have been determined through set (4), contains $R(N-1)$ equations and $R(N-1)$ unknowns, and their coefficients are independent of the p_{rs}'s. Since modifying the p_{rs}'s in this way does not have any impact on the n_{ir}'s or the μ_{ir}'s either, the equilibrium state probabilities, hence all of the performance indices that can be computed from them, remain unchanged.

■

A consequence of Theorem 1 is that the performance indices a system model like the one depicted in Figure 2 allows us to compute remain the same when the graph in Figure 3 is replaced, for instance, by the one in Figure 4, where the p^*_{rs}'s have been calculated from equations (4), with the values of the visit ratios derived from those of the p_{rs}'s and the same equations. There are 8 independent nonzero probabilities in the graph in Figure 4, but only 7 of the 8 equations (4) are independent (the values that remain unchanged are those of the 7 independent visit ratios, say $\frac{e_{1r}}{e_{11}}$); thus, the value of one of the unknown probabilities p^*_{rs} is to be arbitrarily chosen. If the graph in Figure 4 had more arcs, more than one unknown would have to be assigned an arbitrary value.

An important remark is that Theorem 1 does not allow a workload model built by this method to be implemented in an arbitrary way: the theorem states that the model will be performance-wise accurate if it simulates the same number of users as in the workload to be modeled, each behaving as described by the graph in Figure 4, or by any other graph that has the same visit ratios to be various types of commands as the original workload. Note that the actual user behavior graph (e.g., the one in Figure 3) does not have to be known, since the visit ratios can be computed from the relative frequencies of command types, which are easily measurable. The crucial, hard-to-answer question is whether the behaviors of all real users can be satisfactorily described by a single graph or require several different graphs. In any case, the model is in

general no longer accurate if we change the number of users, or subdivide the command types among the simulated users (e.g., users 1 and 2 will continue to execute command type 1, user 3 commands 2 and 4, and so on). Thus, graphs like those in Figure 5 cannot be performance-wise equivalent to that in Figure 3.

We shall now discuss the application of clustering techniques to our interactive workload. Each command type in the graph in Figure 3 is characterized by the distributions of its service times at the various stations (except for FCFS stations, where all commands must have the same exponential service time distribution), by the values of its branching probabilities $b_{ij,r}$, and by the distribution of its terminal times. These characterizing parameters, that represent command demands of the system model's resources, define a multidimensional space in which each component (i.e., each command type) is represented by a point. A clustering algorithm will identify clouds of neighboring points, that is, command types whose resource demands are "similar" enough as to be grouped into the same class. Since in our closed network this is a class of classes, it will be called a *superclass*.

Suppose clustering of the classes in Figure 3 results in the four superclasses (1, 4, 6), (2, 7), (3, 8), and (5). The superclass number is reported in parentheses within each node in the figure. Is a workload model containing superclass representatives a valid model? How many representatives should each superclass have? How should the model be implemented? The answers, for the idealized case in which all the classes in a superclass have identical characterizing parameters, are provided by the following theorem, where by *global performance indices* we shall mean those indices which represent the performance of the model under the whole workload, not on a per class basis. Global indices include the mean throughput rate, the mean response time, the utilizations, the mean queue lengths, and the mean waiting times in all stations.

Theorem 2. The values of the global performance indices of the queueing network in Figure 2 are invariant with respect to aggregations of classes with identical demands if each superclass has a visit ratio in the user behavior graph equal to the sum of the visit ratios of its members, and each non-aggregated class retains its previous visit ratio.

Proof. Let the R user classes be aggregated into S superclasses with $S < R$. The generic state of the network, that was $(y_1, y_2 \ldots y_N)$, with $y_i = (n_{i1}, n_{i2}, \ldots n_{iR})$, becomes $(y'_1, y'_2, \ldots y'_N)$, with $y'_i = (n'_{i1}, n'_{i2}, \ldots n'_{iS})$; n'_{ij} is the number of users of superclass j at station i, hence it equals the sum of the numbers of users at station i that belong to the classes in superclass j. The equilibrium probability of state $(y'_1, y'_2, \ldots y'_N)$ is given by

$$P(y'_1, y'_2, \ldots y'_N) = C\, g_1(y'_1) g_2(y'_2) \ldots g_N(y'_N), \tag{5}$$

where, if station i is of type 2 or 4 (a distinct service time distribution with rational Laplace transform for each class, PS or LCFS queueing discipline [BASK75]),

$$g_i(y'_i) = n'_i! \prod_{s=1}^{S} \frac{1}{n'_{is}!} \left[\frac{e'_{is}}{\mu'_{is}}\right]^{n'_{is}}. \tag{6}$$

For the same station i in the original network we have

$$g_i(y_i) = n_i! \prod_{r=1}^{R} \frac{1}{n_{ir}!} \left[\frac{e_{ir}}{\mu_{ir}}\right]^{n_{ir}}. \tag{7}$$

We shall only consider type 2 or 4 stations since expressions (6) and (7) for the other types are very similar and lead to the same conclusions we are about to reach.

In order for the global performance indices to be invariant under class aggregations, we must have

$$P(y'_1, y'_2,\ldots y'_N) = \sum_{s=1,\,S} \sum_{k=v_{s-1}+1,v_s}^{n_{ik}=n'_{is}} P(y_1, y_2,\ldots y_N), \tag{8}$$

where, without loss of generality, we have assumed that superclass 1 contains classes $1, 2,\ldots v_1$, superclass 2 classes $v_1 + 1, \cdots v_2$, and so on, and that $v_0 = 0$, $v_S = R$.

Since

$$\sum_{s=1,\,S} \prod_{i=1}^{N} g_i(y_i) = \prod_{i=1}^{N} \sum_{s=1,\,S} g_i(y_i), \tag{9}$$

$$\sum_{k=v_{s-1}+1,v_s}^{n_{ik}=n'_{is}} \qquad \sum_{k=v_{s-1}+1,v_s}^{n_{ik}=n'_{is}}$$

a sufficient condition for the validity of (8) is that

$$g_i(y'_i) = \sum_{s=1,\,S}^{\sum n_{ik}=n'_{is}} g_i(y_i), \quad (i = 1, 2,\ldots N). \tag{10}$$

From (7), recognizing that $\mu'_{is} = \mu_{ik}$ $(k = v_{s-1} + 1,\ldots v_s)$,

$$\sum g_i(y_i) = n_i! \prod_{s=1}^{S} \frac{1}{\mu'_{is}{}^{n'_{is}}} \sum \prod_{r=1}^{R} \frac{e_{ir}{}^{n_{ir}}}{n_{ir}!}. \tag{11}$$

However, recalling the identity

$$\sum_{\substack{n \\ \sum_{i=1} x_i = x}} \prod_{i=1}^{n} \frac{a_i{}^{x_i}}{x_i!} = \frac{\left[\sum_{i=1}^{n} a_i\right]^x}{x!}, \tag{12}$$

we can also write (11) as

$$\sum g_i(y_i) = n_i! \prod_{s=1}^{S} \frac{1}{\mu'_{is}{}^{n'_{is}}} \frac{\left[\sum_{k=v_{s-1}+1}^{v_s} e_{ik}\right]^{n'_{is}}}{n'_{is}!}. \tag{13}$$

Since $n_i = n'_i$, a sufficient condition for the right-hand side of (13) to coincide with the right-hand side of (6) is that

$$e'_{is} = \sum_{k=v_{s-1}+1}^{v_s} e_{ik}. \tag{14}$$

Thus, if this condition is satisfied, condition (10) is also satisfied, and this conclusion is in particular valid for $i = 1$.

■

Theorem 2 shows that, under the assumptions made in this section, clustering techniques can produce accurate models of interactive workloads provided that the number of users remain unchanged, all simulated users behave according to the same probabilistic graph, and the visit ratios of the clusters (or superclasses) equal the sums of the visit ratios of their members in the original workload. In the model, one representative per cluster is sufficient. If, for some reason, the designer wants to include more than one representative per cluster, the model is still performance-wise accurate if the sum of the visit ratios of these representatives equals the visit ratio of the cluster. A possible behavior graph for the simulated users in our example is presented in Figure 6.

Again, no other reduction technique among those we have studied guarantees the accuracy of the method dictated by Theorem 2. For instance, methods leading to graphs like those in Figure 5 (but with fewer nodes in the clustering case) generally lead to models whose accuracy is lower than that of the one described above and cannot be easily predicted.

It is intuitive, and can be formally proved, that, when a workload is described by a collection of disjoint user behavior graphs, each graph is to be dealt with (i.e., transformed or reduced) separately if performance-oriented model accuracy is to be preserved. Each distinct graph corresponds to a chain in the queueing network, and neither users nor classes can generally be moved across chains without modifying the values of the global performance indices.

5. Conclusions

The treatment of artificial workload design methods presented in Section 4 does not claim to provide complete and satisfactory solutions to the three problems discussed in Section 2. It is only a first attempt at establishing scientifically sounder foundations for the design of artificial workloads.

As far as Problem 1 is concerned, we may be accused of not solving but rather transforming it into a queueing model accuracy problem. This is, of course, true, but it is also true that the problem is more precisely defined in its new context, and that knowledge about the accuracy of queueing models is growing. The objection might be raised that building an artificial workload is not necessary if a good model of the system exists. However, our approach does not require such a model to be constructed and tested. It simply suggests that the workload model designer characterize the workload referring to the resources that would be explicitly represented in a queueing model likely to be sufficiently accurate for the purposes of the study. Furthermore, even if a good model of the system were available, in most evaluation studies measurement, when possible, is definitely to be preferred to modeling.

Our approach suggests a performance-oriented solution to Problem 2. If all the assumptions made in Section 4 are satisfied, and if all performance indices of interest are global indices, a workload model resulting from a correct application of the currently used statistical techniques is guaranteed to be perfectly accurate. Techniques like sampling and clustering have been found to be valid under our assumptions provided they are applied according to the rules dictated by Theorems 1 and 2. An analysis of the effects on workload model accuracy of a less than perfect compliance with each assumption remains to be done, and is clearly one of the most important avenues for further research

opened by this investigation.

With respect to Problem 3, our approach shows that there are cases in which one should not worry about reproducing workload dynamics faithfully, as the system performance indices one is usually interested in do not depend on the order of execution of commands. This result is intuitive, but our treatment allows us to understand the assumptions under which it is valid and hence its limitations. Clearly, the order of execution is important when the effects of particular dynamic variations of the workload or of some dynamic control policies (e.g., scheduling algorithms) are to be investigated. In these cases, either the steady-state assumption is violated or the use of a product-form queueing model is not justified. It must be remembered that the theorems in Section 4 on which our conclusions are based apply only to product-form queueing networks. The influence of the various approximations that can be used to solve a non-product-form network on the accuracy of a workload model obtained following the rules of Section 4 certainly deserves to be studied.

Acknowledgements

The author is deeply grateful to Henry Lyne for his suggesting user behavior graphs and visit ratio invariance as a possible means to achieve the invariance of performance. He is also indebted to the Research Division of the IBM Corporation for making the use of RESQ2 possible in this project. Without RESQ2, the author's conjectures would never have been so easily verifiable, and the results of Section 4 would probably have remained conjectures. Finally, the author gratefully acknowledges the wonderful assistance of Ellen Boyle in the preparation of the manuscript.

References

[AGRA76] A.K. Agrawala, J.M. Mohr, and R.M. Bryant, An approach to the workload characterization problem, *Computer 9*, 6, (June 1976), 18-32.

[ARTI78] H.P. Artis, Capacity planning for MVS computer systems, in: D. Ferrari, ed., *Performance of Computer Installations*, North-Holland, Amsterdam, 1978, 25-35.

[BASK75] F. Baskett, K.M. Chandy, R.R. Muntz, and F.G. Palacios, Open, closed, and mixed networks of queues with different classes of customers, *J. ACM 22*, 2 (April 1975), 248-260.

[BENW75] N. Benwell, ed., *Benchmarking - Computer Evaluation and Measurement*, Hemisphere, Washington, D.C., 1975.

[CALZ82] M. Calzarossa, and G. Serazzi, Reproduction of the dynamic aspects of workloads, submitted for publication (1982).

[FERR72] D. Ferrari, Workload characterization and selection in computer performance measurement, *Computer 5*, 4 (July-Aug. 1972), 18-24.

[FERR78] D. Ferrari, *Computer Systems Performance Evaluation*, Prentice-Hall, Englewood Cliffs, 1978, Ch. 5.

[FERR79] D. Ferrari, Characterizing a workload for the comparison of interactive services, *AFIPS Conf. Proc. 48 (NCC 1979)*, 789-796.

[FERR81] D. Ferrari, A performance-oriented procedure for modeling interactive workloads, in: D. Ferrari and M. Spadoni, Eds., *Experimental Computer Performance Evaluation*, North Holland, Amsterdam, 1981, 57-78.

[HARI82] G. Haring, On state-dependent workload characterization by software resources, Proc. 1982 ACM SIGMETRICS Conf. on Measurement and Modeling of Computer Systems, Seattle, WA, Aug. 1982, 51-57.

[MAMR77] S. A. Mamrak, and P.D. Amer, A feature selection tool for workload characterization, *Proc. 1977 SIGMETRICS/CMG VIII Conf. on Computer Performance: Modeling, Measurement, and Management*, Washington, D.C., Nov. 1977, 113-120.

[SCHW72] H.D. Schwetman, and J.C. Browne, An experimental study of computer system performance, *Proc. ACM Nat. Conf.*, 1972, 693-703.

[SERA81] G. Serazzi, A functional and resource-oriented procedure for workload modeling, *Proc. Performance '81, the 8th Int. Symp. on Computer Performance Modeling, Measurement and Evaluation*, North-Holland, Amsterdam, 1981, 345-361.

[SHOP70] W.L. Shope, K.L. Kashmarak, J.W. Ingram, and W.F. Decker, System performance study, *Proc. SHARE 34*, 1 (March 1970), 439-530.

[SREE74] K. Sreenivasan, and A.J. Kleinman, On the construction of a representative synthetic workload, *Comm. ACM 17*, 2 (March 1974), 127-133.

[WOOD71] D.C. Wood, and E.H. Forman, Throughput measurement using a synthetic job stream, *AFIPS Conf. Proc. 39 (FJCC 1971)*, 51-56.

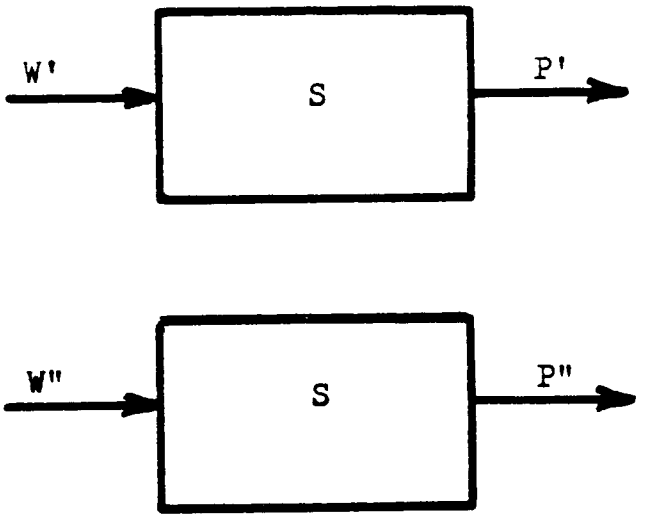

Fig.1. The performance-oriented definition of workload model.

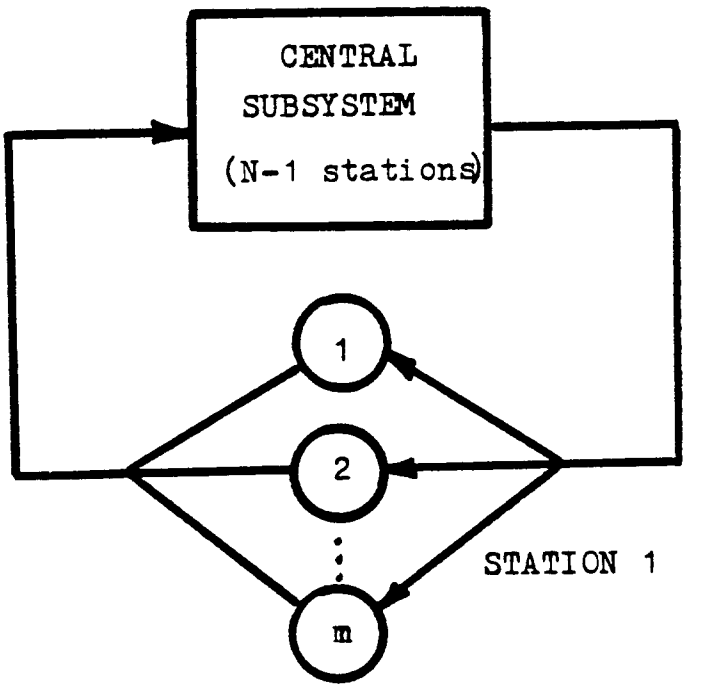

Fig.2. A model of the system.

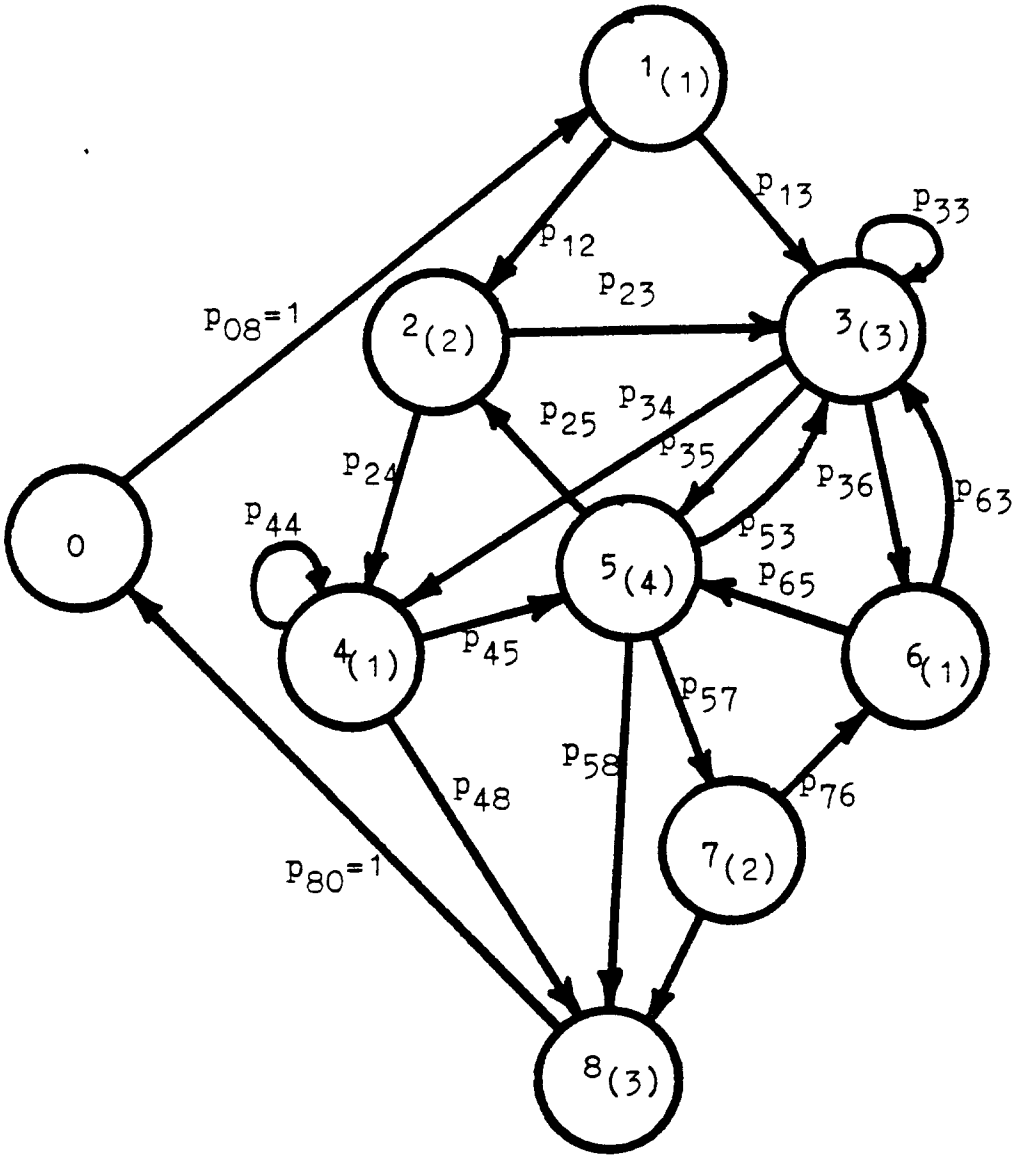

Fig.3. A user behavior graph.

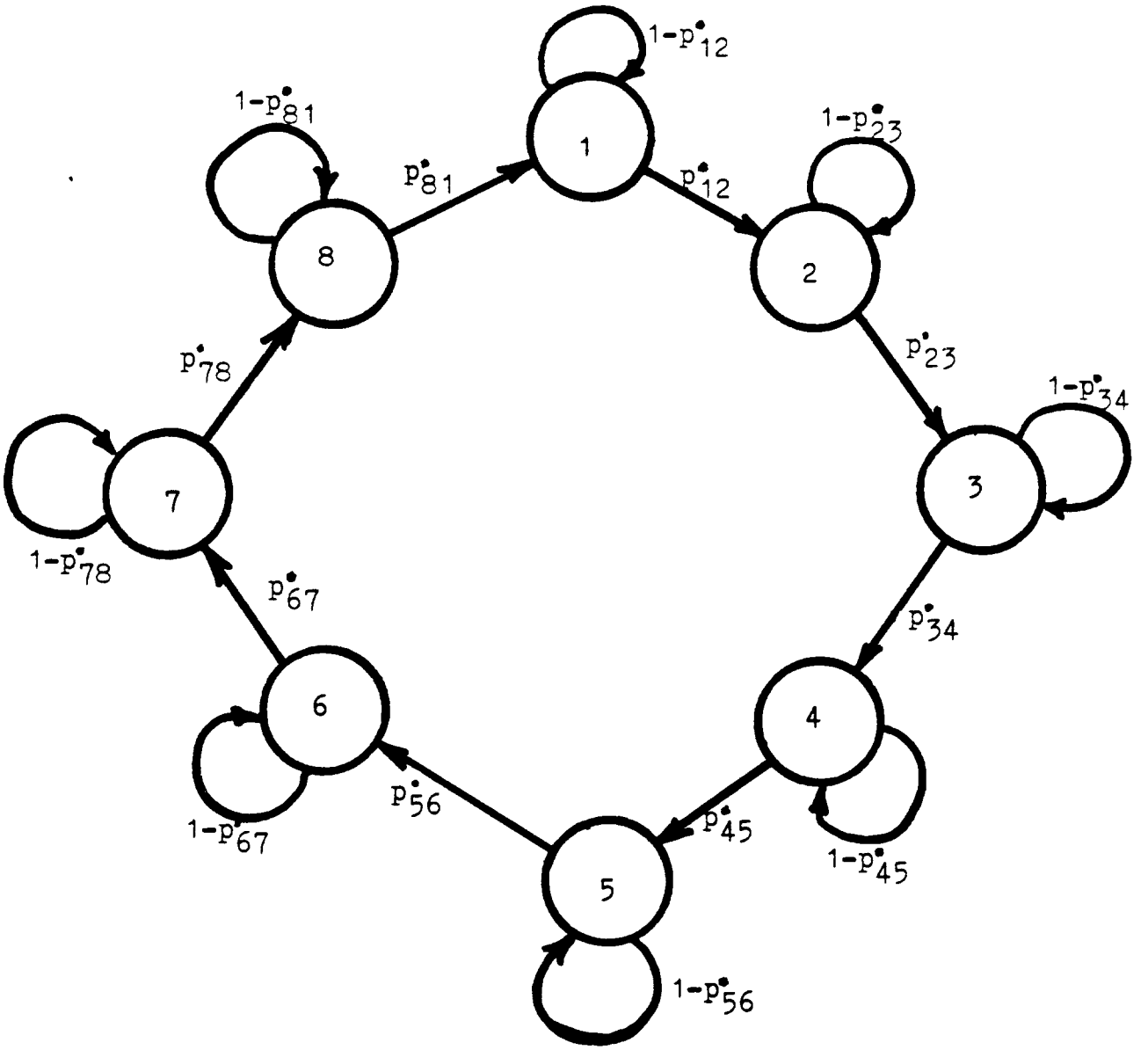

Fig.4. A user behavior graph performance-wise equivalent to the one in Figure 3.

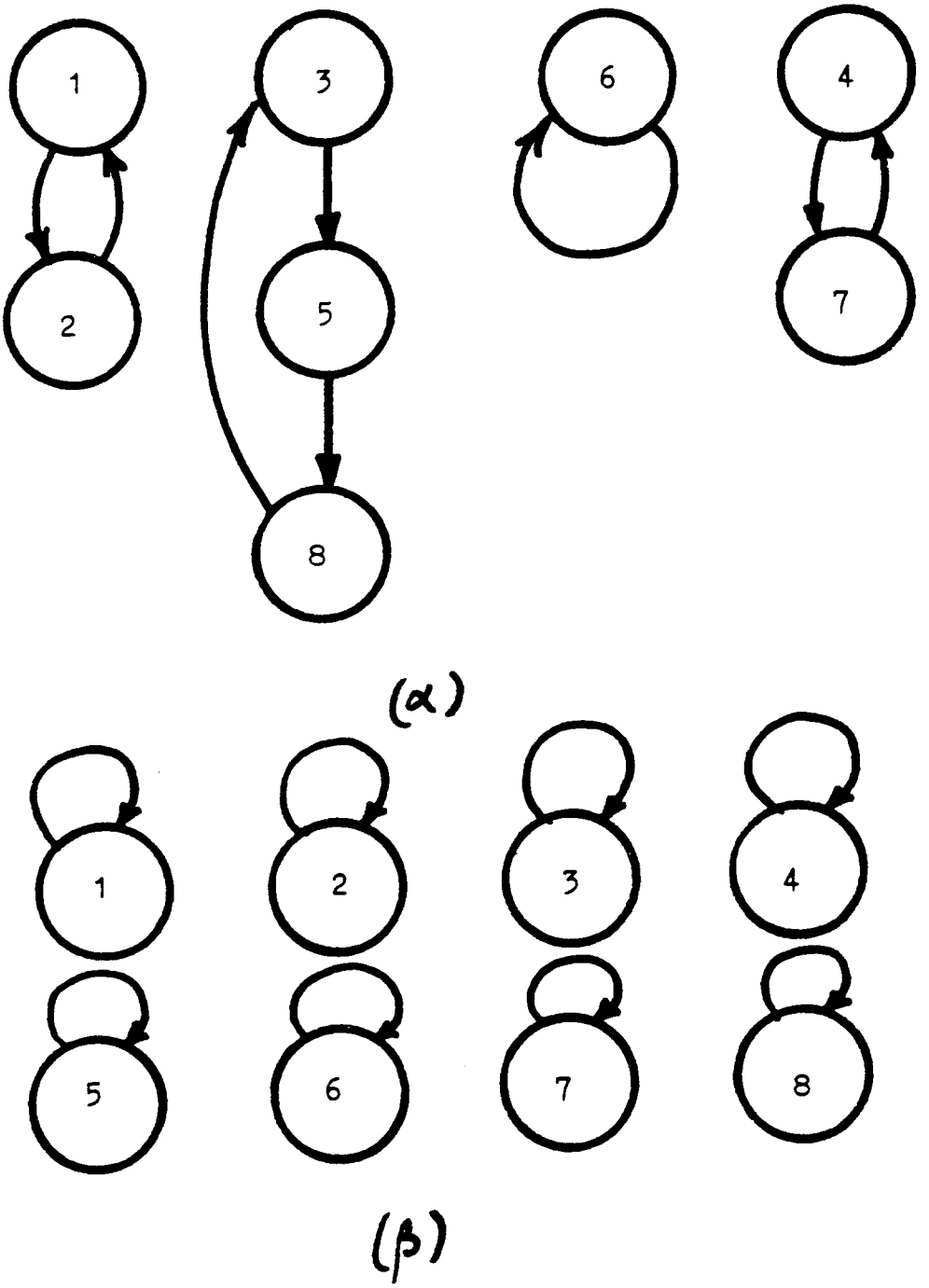

Fig.5. User behavior graphs not performance-wise equivalent to that in Figure 3.

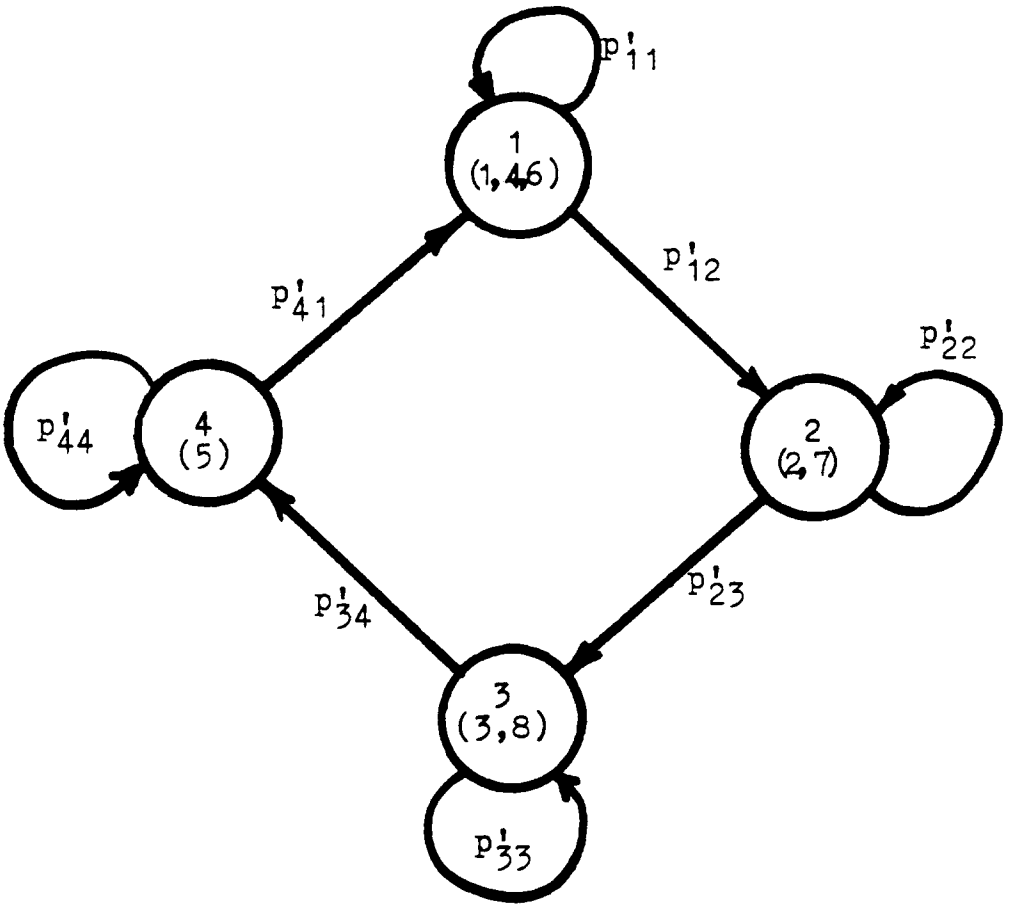

Fig.6. A user behavior graph obtained from the one in Figure 3 through class aggregation.