# Construction and Use of Multiclass Workload Models[1]

*Maria Calzarossa*
Dipartimento di Informatica e Sistemistica
Università di Pavia
I-27100 Pavia, Italy
E-mail: mcc@gilda.unipv.it

*Giuseppe Serazzi*
Dipartimento di Elettronica e Informazione
Politecnico di Milano
I-20133 Milano, Italy
E-mail: serazzi@ipmel2.elet.polimi.it

**ABSTRACT**

Although the performance of a system is determined by the characteristics of the load being processed, much more care is usually devoted to the construction of a detailed system model than to an accurate definition of the workload model. A precise description of actual workloads is obtained with multiclass workload models. Major difficulties on their implementation are related to the identification of the classes and to the reproduction of the dynamic behavior of the load. Small relative errors in estimating the average resource demands of a class lead to larger errors in the performance indices of that class. The proposed multi-step methodology allows the construction of multiclass workload models which preserve, when used as system model parameters, both the static and dynamic characteristics of the original load.

## 1 Introduction

A critical problem in the performance evaluation of multiprogrammed systems is the development of accurate models of their workload. Actual workloads consist of several types (e.g., transaction, batch, and time-sharing) with widely varying resource demands, not only between different types but also within each type.

As a consequence, single class workload models, i.e., models where all the components (e.g., jobs, tasks, interactive commands) are assumed to be statistically identical, have a very limited representativeness. The corresponding system models provide only a single estimate of each performance index which represents the average taken over all the workload components. This approach has its major drawback in providing statistics only at the level of the global workload.

More representative models, i.e., multiclass workload models, are usually obtained by describing the load by a few classes of components which allow performance statistics on a per class basis. The accurate parameterization of the classes and the reproduction of the dynamic

[1]This work was supported in part by the the Italian Research Council (C.N.R.) "Progetto Finalizzato Sistemi Informatici e Calcolo Parallelo" under Grants N. 91.00933.PF69 and N. 92.01571.PF69 and by the Italian M.U.R.S.T. under the 40% and 60% Projects.

characteristics of the load are critical tasks.
The number of classes and their parameters are influenced by several factors (e.g., classification algorithm, and types of parameters used to describe the components). Also, the non-monotonicity of performance indices in multiclass environments makes the errors in the average parameters of the classes and the induced errors in the performance indices no longer proportional to each other [GD80, TS85, TD86]. Furthermore, in order to make the workload and system models tractable both with respect to time and space requirements, a small set of classes must be used. Hence, there is trade-off between the precision of the model, which is directly related to the number of classes, and the classes identified.
Accurate multiclass workload models must also be able to reproduce the dynamic behavior of the system. The mixes, that is, the various workload components of each class, e.g., interactive command types, that are simultaneously into the system at the different instants of time, are to be considered. Indeed, the identification of the mixes is a crucial problem in that they determine the bottleneck/s of a system which in turn are responsible for its performance.
In the sequel, the terms "workload component" and "interactive command" will be used interchangeably.

The approaches commonly followed for the construction of workload models (see e.g., [AMB76, Sera81, FSZ83, HL84]) do not take into account some of the aspects above described. The classes are identified by applying statistical techniques to all the workload components executed in the observation interval and regarded as if no temporal relationships existed among them. Hence, the domain of applicability of these results is mainly limited to studies which only require a static characterization of the workload. Indeed, any correspondence between the classes identified and the mixes of components is lost.

In this paper we describe an integrated modeling approach for the construction of accurate multiclass workload models. The workload analyzed in our multi-step methodology consists of the set of the commands submitted by all the users of an interactive system in the observation interval $T$. The parameters describing each command are: type, arrival and departure times from the system, user identifier and resource demands.
An initial functional characterization of the workload components on a per user basis allows us to take into account the dynamic aspects of the load and to easily identify and reproduce the mixes. Groups of commands with homogeneous characteristics are then obtained for each user and the description of his behavior is also provided. Through the application of well-known ag-

gregation/disaggregation techniques in the framework of workload models, performance indices on a per command basis are easily computed.

The multiclass model obtained is not affected by the uncertainties related to the classification algorithm and to the types of parameter used to describe workload components. It is also independent of the technique adopted to characterize the dynamic properties of the load because the components are analyzed on a per user basis. The input load of the corresponding system model is described at the user level and the performance indices are provided at this level as well as for each individual command.

This paper is organized as follows. In Sections 2 and 3 the procedure for obtaining the representative commands of each user of an interactive system and for modelling the user behavior through probabilistic graphs are described. The identification of the composite classes, one for each user, of the workload model (aggregation process) and the disaggregation of the performance indices produced by the system model for single commands or groups of commands are presented in Section 4. An application example of the proposed methodology is given in Section 5. Finally, Section 6 concludes the paper with some future research directions in this area.

## 2 The Modeling Approach

The main steps of our methodology are (see Figure 1):

step 1 - *subdivision* of the global workload $G$ of commands submitted by all the users into several sequences $G_i$ ($i = 1, 2, ....$) on a per-user basis; $G_i$ contains the commands submitted by user $i$;

step 2 - *identification*, in each $G_i$, of groups of commands with similar characteristics and *selection* of a few representative commands corresponding to the centroids of the groups;

step 3 - *description* of the behavior of each user $i$ through a graph-based technique able to reproduce the dynamic aspects of the load, i.e., the sequence of command executions;

step 4 - *aggregation* of all the commands or groups of commands of each user graph into a single composite class, construction of the multiclass workload model and *definition* of the system model parameters;

step 5 - *solution* of the system model and *disaggregation* of the global performance indices obtained for each user for deriving the per-command ones.

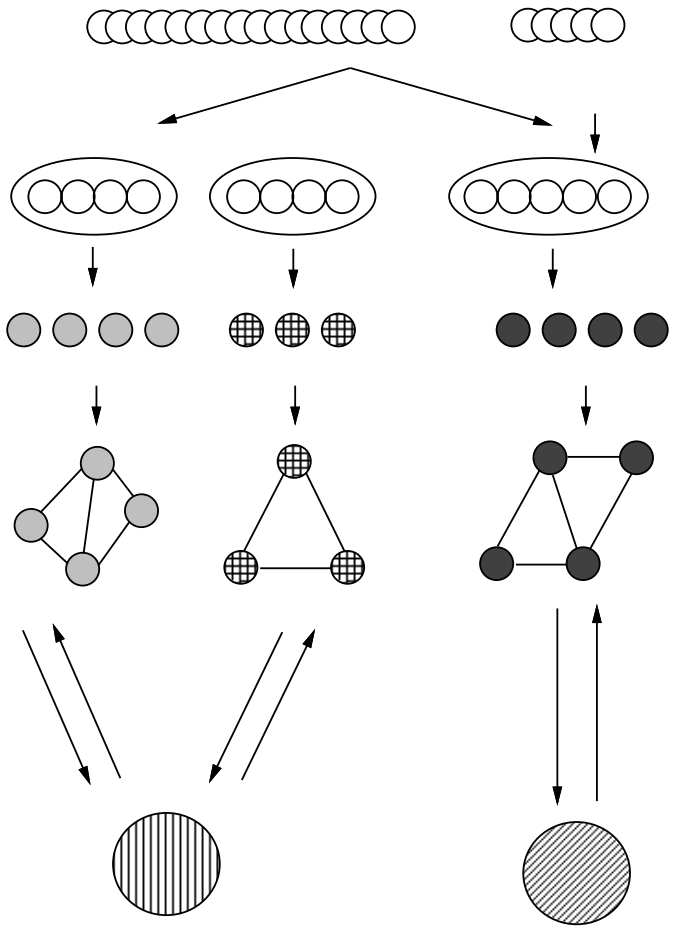

Figure 1: Main steps of the proposed methodology.

The operations performed in step 1 represent the basis of our methodology. The initial functional characterization of the workload based on users allows us to represent into the model both the static and dynamic properties of the load itself. Indeed, in any interactive system with $N$ users, a mix is represented as the set of the $N$ commands, one per user, which are in the central subsystem or in the think state at the terminal station, in a particular instant of time.

Hence, this initial functional characterization yields the subdivision of the global workload $G$ into several sequences $G_i$, ($i$ = 1, 2, ..., $N$), one per user. Each $G_i$ consists of the commands submitted by user $i$, ordered according to their arrival time at the system.

Statistical techniques are then applied in step 2 to the commands of user $i$ in order to identify groups of components with homogeneous characteristics. Indeed, a typical workload consists of many commands described by a large number of parameters. Hence, it is necessary to derive a compact and manageable model.

It is important to notice that the commands of the same user can be grouped in different ways. For example, from a classification based on hardware resource demands, such as CPU time consumed and number of I/O's to disks, the groups are characterized by different resource requirements. When the classification is based on functional parameters, such as command type, we obtain groups of commands performing the same functions. Depending upon the objective of the study, one type of parameter is more suitable than another. In predictive studies (e.g., capacity planning and procurement) function-oriented variables are used. Hardware and software parameters are typical of workload models constructed for tuning and design purposes.

Another source of variability of the characteristics of the groups is related to the intrinsic properties of the statistical techniques adopted, e.g., clustering algorithms. Different algorithms yield different partitions in the data because of the metrics implemented and their different behavior with respect to the outliers. The same applies to the various scaling techniques used in order to make the data comparable. As we will see in Section 4, our methodology overcomes these problems and leads to a stable modelling approach.

# 3 User Behavior Characterization

Until now, we have ignored any temporal relationship among the commands within either a single user sequence or between different user sequences. The identification of the representative commands (step 2) is a fundamental step in order to characterize the time-varying user behavior (step 3) and to provide for each user the values of the performance indices at the command level (step 5). These objectives are met by identifying the stochastic process that approximates each $G_i$. The behavior of each user is described by the so called user behavior graph (ubg) introduced in [Ferr84] and applied in [CF86, CMT90].

From $G_i$, we derive another sequence in which the commands, listed in chronological order according to their arrival time at the system, are replaced by the identifiers of the group they belong to. This sequence is modeled as a Markov chain whose state is represented by the command which currently enters into the system. Hence, the finite state-space of this process

is $E_i = (1, ..., c_i)$, i.e., the set of the $c_i$ command groups for user $i$. A method of statistical inference based on the $\chi^2$ test of goodness of fit is used to test the order of the Markov chain and the nature (i.e., homogeneity) of the transitions between states [AG57, Hari83]. Note that the $\chi^2$ test works particularly well in this type of analysis because, as already pointed out, the sample sizes are usually quite large.

When the commands of a sequence $G_i$ are grouped together on the basis of a predefined criterion and a first order Markov chain is assumed, the ubgs are employed for modelling the dynamic behavior of the users.

Indeed, probabilistic graphs lead to a synthetic representation that is natural, easy to understand, and can be immediately applied as input to system models. As already mentioned, depending on the criterion adopted to group the commands and on the objective of the study, the resulting graphs can be closer to the applications performed by the user or to the resources consumed by his commands.

As we will see, the modelling approach adopted in our methodology is stable in that the type of characterization adopted does not affect the parameter values of the classes that will be used as input for the system model.

# 4 The Multiclass Model

The application of the first three steps of our methodology yields the construction of workload models which include both the static and dynamic properties of the load itself. We then use such a workload model for defining the parameters of a system model represented by a closed product-form queueing network [BCMP75] with $N$ users. The application of standard aggregation techniques (Sect. 4.1) to the commands of user $i$ ($i = 1, 2, ..., N$), subdivided into $c_i$ groups (i.e., classes), leads to the definition of a single "composite class". Hence, in the system model with $N$ users there are $N$ composite classes, one for each user. The performance indices, provided for the global workload as well as on a per user basis, can also be derived on a per command group basis (Sect. 4.2). The parameters of the composite classes are not affected by the classification process adopted and by the number and the characteristics of the classes identified. Furthermore, the aggregation process makes different behavior graphs obtained for a user equivalent from the modeling viewpoint. However, as we will clarify later, this does not mean that workload models based on ubgs are no longer necessary. Indeed, they are needed

when the performance indices on a per command group basis are required.

Note that the aggregation and disaggregation techniques are well-known methods (see e.g., [BB80], [Lave83]). Up to now they have mainly been applied as means of reducing the complexity of the queueing network system models. Their application in the framework of workload models has never been emphasized. Our contribution is aimed at embedding these techniques in an integrated methodology able to solve the problems related to the construction of accurate multiclass workload models.

The aggregation and disaggregation methods will be briefly surveyed in the next two sections.

### 4.1 Aggregation Process

A command, issued by a terminal station, cycles through the service centers of the system until it returns to the terminal station. Let $K$ denote the global number of service centers in the system excluding the terminal station. Each command $j$ in the sequence $G_i$ (consisting of $N_i$ commands) is characterized by its requirements to the various service centers of the system. Let $D_{jl;i}$ denote the global demand of the command $j$ of user $i$ to the service center $l$ $(l = 1, 2, ..., K)$, $(j = 1, 2, ..., N_i)$, $(i = 1, 2, ..., N)$.

Several ubgs can be associated to the same user. Let us consider, for example, the ubg $k$ of user $i$ consisting of $c_{ki}$ command groups. We first compute the average demand $\overline{D}_{ml;ki}$ of the $m$-th command group and the transition probability matrix $\mathrm{P}_{ki} = [p_{rs;ki}]$ ($r$,$s$ = 1, 2, ..., $c_{ki}$) associated to the ubg $k$, respectively (see e.g. [Ferr84]). $p_{rs;ki}$ is the probability that user $i$ issues a command of group $s$ after a command of group $r$. The ubg $k$ can be modeled by a discrete time Markov chain characterized by the matrix $\mathrm{P}_{ki}$. The steady-state solution of such a chain yields the frequency of execution $\pi_{m;ki}$ [2] of $m$-th command group; that is, we have a set of positive values given by:

$$\pi_{m;ki} = \sum_{r=1}^{c_{ki}} p_{rm;ki}\pi_{r;ki} \qquad m = 1, ..., c_{ki}, \qquad \sum_{m=1}^{c_{ki}} \pi_{m;ki} = 1.$$

By using the $\pi_{m;ki}$, it is possible to aggregate all the $c_{ki}$ command groups of ubg $k$ for user $i$ into a composite command, i.e., into a composite class, whose global demand at service center $l$ is given by:

[2] Note that the $\pi_{m;ki}$'s can also be interpreted as the relative throughputs of the $m$-th command group in the ubg $k$. Hence, their values can be estimated on real systems and the composite class directly computed from $G_i$ without applying steps 2 and 3. However, the possible use of the workload models suggests not only an initial functional characterization of the load based on users but also the need of having ubg models obtained after the classification process.

$$D_{l;ki} = \sum_{r=1}^{c_{ki}} \pi_{r;ki} \overline{D}_{rl;ki}. \qquad (1)$$

Note that all the possible ubgs lead to the same composite class, that is, $D_{l;ki} = D_{l;i}$, $\forall k$.
In such a way, the solution of the queueing network model, characterized by $N$ distinct classes, one for each user, without class changes, and by a multiprogramming level equal to $N$, is obtained with reduced computation time and space requirement.

### 4.2 Disaggregation Process

The performance indices obtained by solving the queueing network model with $N$ composite classes are on a per user basis. Many performance studies require more detailed results, i.e., on a per command group basis. Such results can be derived by "disaggregating" (step 5) the performance indices obtained with the $N$ composite classes.

Let $X_i$ be the throughput for user $i$; the throughput of $m$-th command group ($m$ = 1, 2, ..., $c_{ki}$) for the ubg $k$ is given by:

$$X_{m;ki} = \pi_{m;ki} X_i$$

Furthermore, after having computed the average queue length at service center $l$, $Q_{l;i}$ for user $i$, the average number of commands $N_{m;ki}$ of the $m$-th group which are in the system can be computed as:

$$N_{m;ki} = \sum_{l=1}^{K} \frac{\pi_{m;ki} \overline{D}_{ml;ki}}{D_{l;i}} Q_{l;i}$$

and then, by applying Little's formula, we have the average response time for the $m$-th command group of user $i$.
Note that the aggregation/disaggregation process can be applied at any level of detail. For example, since the characterizing parameters of the composite class can be directly computed from $G_i$ skipping both steps 2 and 3, it is possible to disaggregate on the basis of the single commands of $G_i$. This is usually not advisable for many reasons. Redundant, and hence useless, information would be obtained and such workload models would not be suitable for making any prediction. In general, steps 2 and 3 are required in order for the workload model to be manageable and practically usable in various performance studies.

# 5 Application Example

In this section an application of our methodology on real workloads is described and a comparison of our approach with the approaches commonly adopted is presented.

The analyzed workload has been measured on an interactive system running in a scientific environment under the Unix operating system. The data collected are the commands submitted by 12 users during an observation interval of about 4 hours. The characterizing parameters of each command are: type, beginning time, CPU time consumed (expressed in seconds) and number of disk I/O blocks transferred.

From the global workload $G$, 12 sequences $G_i$, one for each user, have been derived (step 1) and the classification process has been applied to each of them (step 2). As previously mentioned, different types of characterization can be chosen. If we adopt a functional criterion, the command groups are obtained by partitioning the commands according to their types. On the other hand, a resource oriented approach is more complex because it implies the use of classification algorithms.

The tool used in step 2, i.e., the Workload Analyzer tool [HP89, Pool92], employes a clustering algorithm which produces a number of partitions ranging from 1 up to a maximum predefined value. The identification of the best partition is based on some indices related to the intra- and inter-partition variances of the parameters used. Note that the criteria adopted by the classification algorithms do not yield a unique "best" partition because of the complexity related to the clustering process which has to recognize the differences existing among commands with similar parameter values. Hence, a few "sub-optimal" partitions are identified and the selection of the best one according to the objective of the study has to be performed.

For the sake of simplicity, we will present the results of the classification process for one of the 12 users, namely, user 1. The functional characterization applied to the sequence $G_1$ (consisting of 159 commands) yields 17 groups which correspond to the 17 different command types submitted by the user. On the other hand, if the clustering algorithm, based on a resource oriented characterization, is applied to the 159 commands of $G_1$, then it produces three "sub-optimal" partitions $k = 1, 2, 3$, namely, into $c_{11} = 2$, $c_{21} = 4$ and $c_{31} = 6$ groups, respectively. As an example, the parameter values corresponding to the centroids of each of the groups of these resource-oriented partitions are presented in Table 1.

| Partition $k$ | Group $m$ | Number of components | CPU Time [s] $\overline{D}_{mCPU;ki}$ | Number of I/Os | $\pi_{m;k1}$ |
|---|---|---|---|---|---|
| 1 | 1 | 117 | 4.63 | 27.77 | 0.734 |
| | 2 | 42 | 15.2 | 151.64 | 0.266 |
| 2 | 1 | 108 | 4.26 | 24.19 | 0.677 |
| | 2 | 24 | 6.03 | 182.12 | 0.152 |
| | 3 | 21 | 19.71 | 34.43 | 0.133 |
| | 4 | 6 | 26.965 | 318.5 | 0.038 |
| 3 | 1 | 66 | 2.565 | 11.6 | 0.417 |
| | 2 | 24 | 4.38 | 108.6 | 0.152 |
| | 3 | 39 | 9.79 | 33.9 | 0.241 |
| | 4 | 6 | 26.965 | 318.5 | 0.038 |
| | 5 | 12 | 24.185 | 22.5 | 0.076 |
| | 6 | 12 | 6. | 228. | 0.076 |

Table 1: Characterizing parameters and relative frequencies of the various groups constituting the three resource oriented sub-optimal partitions identified for user 1.

The stochastic properties of each sequence $G_i$ have also been investigated. Several sequences, corresponding to the various partitions, have been derived where each command has been replaced by the identifier of the group it belongs to. The order and the nature of the transitions of the Markov chain have been tested for each sequence with a 5% level of significance for the $\chi^2$ tests. The results have shown that a first-order model characterized by homogeneous transitions is an appropriate representation for all these sequences.

These conclusions, together with the results of the test applied for verifying the non-randomness of the sequences, support a model based on the ubgs to describe the dynamic properties of the workload (step 3).

Figure 2 shows the user behavior graphs for the partitions $k = 1$ and $k = 2$ (see Table 1) of the commands of user 1 in $c_{11} = 2$ and $c_{21} = 4$ groups, respectively. Each node of the graphs represents one of the command groups identified by the clustering algorithm. $m_{c_{k1}}$ is the identifier of the $m$-th command group in the partition into $c_{k1}$ groups. The characterizing parameters and the steady-state probabilities $\pi_{m;k1}$ of each node $m_{c_{k1}}$ are also reported in Table 1. Note that $\overline{D}_{mI/O;ki}$ is obtained by multiplying the number of I/O operations by the disk service time, that is, 36 ms.

As can be seen, the two graphs are very different both with respect to the resources con-

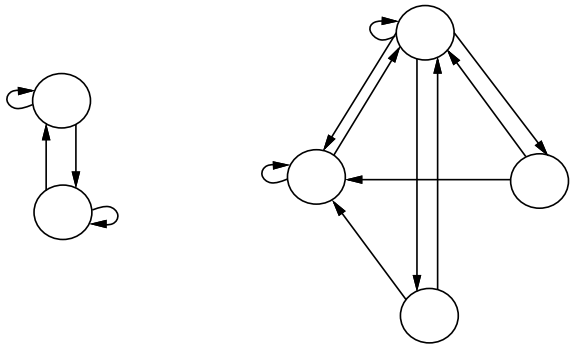

Figure 2: User behavior graphs consisting of 2 nodes (a) and 4 nodes (b) corresponding to two possible partitions (into 2 and 4 groups) for user 1.

sumed and with respect to their transition probabilities. However, these graphs and all the graphs that can be constructed for the same user are equivalent from the system modeling viewpoint (step 4). Indeed, the same composite class, characterized by $D_{CPU;1} = 7.4$ seconds and $D_{I/O;1} = 2.18$ seconds (see Eq. (1)) is derived for all the possible ubgs of user 1.

After the application of this process to all users, the input parameters for the system model are available. We adopt a "central server" queueing network model consisting of a terminal station, a CPU and an I/O device. The multiprogramming level is equal to the number of users, that is, 12 and the average think time at the terminal station is 10 seconds.

By solving the analytic model with 12 composite classes we obtain the global response time $R$ and the corresponding values $R_i$ for each user. For example, $R$ and $R_1$ are equal to 99.77 and 91.59 seconds, respectively.

The disaggregation process (step 5) can be applied to all the ubgs of the various users and yields, for each of them, the performance indices on a per command group basis. Table 2 presents the response times for each node of the two ubgs of Fig. 2.

When we need to derive performance indices on a per command type basis, we can apply the disaggregation process to the functional ubgs. As already mentioned, the functional ubg for user 1 consists of 17 nodes which correspond to the 17 different command types submitted by the user himself. For example, the disaggregation technique applied to such a graph provides a response time of the command *ls* which lists the contents of a directory equal to 43.95 seconds. The value for the same command type is 31.47 seconds for user 10.

The drawbacks of the approaches commonly adopted for the construction of multiclass

| Partition $k$ | Group $m$ | Response Time [s] |
|---|---|---|
| 1 | 1 | 40.479 |
| | 2 | 158.268 |
| 2 | 1 | 36.3739 |
| | 2 | 61.866 |
| | 3 | 201.396 |
| | 4 | 290.584 |

Table 2: Response times obtained by disaggregating the performance indices for the partitions in 2 and 4 groups.

workload models can now be outlined. The classification process is usually applied to the global workload $G$ considered as a whole without any initial functional subdivision of the commands based on users. In this case, the mixes are impossible to reproduce because there is no correspondence between the classes identified and the commands that are simultaneously into the system. Furthermore, commands submitted by different users can belong to the same class and vice versa.

Based on this approach, the number of classes obtained with a functional characterization of $G$ is 53, i.e., 53 different command types have been submitted by the 12 users. This means that the corresponding system model with 53 classes would not be manageable and tractable. Hence, it is necessary to adopt a resource-oriented characterization. The clustering algorithm applied to $G$ provides four sub-optimal partitions, with, 3, 5, 8 and 9 groups (classes), respectively. The same system model was then solved with each of these four workloads as input and the remaining parameters unchanged.

| Number of classes | Global Response Time [s] |
|---|---|
| 3 | 93.52 |
| 5 | 93.70 |
| 8 | 69.49 |
| 9 | 67.54 |

Table 3: Global response times obtained with four workloads.

Table 3 shows the global response times in the models with 3, 5, 8 and 9 classes, respectively. As can be seen, the values are very different either between each other and with respect to the exact value ($R = 99.77$ s). The minimum relative error is less than 8%. However, the maximum relative error, obtained with 9 classes, is about 40%. This is mainly due to the poor representation of some of the classes. Indeed, crucial problems, which are automatically solved by our methodology, are related to the choice of the multiprogramming level and to the representation of the outliers. Hence, the classes with fewer components cannot be explicitly represented in the model since the multiprogramming level is fixed. This is typically the case of outliers. Therefore, the performance indices are affected by large errors, as in the models with 8 and 9 classes, respectively, and the problems related to the uncertainties of the classification process are not overcome.

On the other hand, if we adopt another common approach consisting of grouping all the command together so as to obtain a single class model, the results are also poor. The average response time is 119.1 seconds with a relative error of about 20%.

# 6 Conclusions

The construction of workload models is usually a neglected problem because of the difficulties related to the large amount of data to be measured and to their accurate and correct interpretation. The multiclass workload models obtained with our methodology together with the application of well-known aggregation/disaggregation techniques, in the framework of workload characterization, allow us to implement accurate models which provide the values of the performance indices for the global workload as well as on a per user and per command basis.

Future research will be devoted to parallel and distributed systems. The parameters typical of the parallel and distributed applications (e.g., average number of processor used, concurrency) will be integrated as a part of this methodology.

# References


[AG57] T.W. Anderson and L.A. Goodman. Statistical inference about Markov chains. *Ann. Math. Stat.*, 28:89–110, 1957.

[AMB76] A.K. Agrawala, J.M. Mohr, and R.M. Bryant. An approach to the workload characterization problem. *Computer*, 9(6):18–32, 1976.

[BB80] S.C. Bruell and G. Balbo. *Computational Algorithms for Closed Queueing Networks*. North-Holland, 1980.

[BCMP75] F. Baskett, K.M. Chandy, R.R. Muntz, and F.G. Palacios. Open, closed and mixed networks of queues with different classes of customers. *Journal of ACM*, 22(2):248–260, 1975.

[CF86] M. Calzarossa and D. Ferrari. A sensitivity study of the clustering approach to workload modelling . *Performance Evaluation*, 6:25–33, 1986.

[CMT90] M. Calzarossa, R. Marie, and K.S. Trivedi. System Performance with User Behavior Graphs. *Performance Evaluation*, 11:155–164, 1990.

[Ferr84] D. Ferrari. On the Foundations of Artificial Workload Design. In *Proc. 1984 ACM SIGMETRICS Conf. on Measurement and Modeling of Computer Systems*, pages 8–14, 1984.

[FSZ83] D. Ferrari, G. Serazzi, and A. Zeigner. *Measurement and Tuning of Computer Systems*. Prentice-Hall, 1983.

[GD80] K.D. Gordon and L.W. Dowdy. The impact of certain parameter estimation errors in queueing network models. In *Proc. 1980 ACM SIGMETRICS Conf. on Measurement and Modeling of Computer Systems*, pages 3–9, 1980.

[Hari83] G.Haring. On Stochastic Models of Interactive Workloads. In A.K. Agrawala and S.K. Tripathi, editors, *PERFORMANCE '83*, pages 133–152. North-Holland, 1983.

[HL84] P. Heidelberger and S.S. Lavenberg. Computer Performance Evaluation Methodology. *IEEE Trans. on Computers*, C-33(12):1195–1220, 1984.

[HP89] P.H. Hughes and D. Potier. The Integrated Modelling Support Environment. Report ESPRIT II - IMSE Project R1.2-3, May 1989.

[Lave83] S.S. Lavenberg, editor. *Computer Performance Modeling Handbook*. Academic Press, 1983.

[Pool92] R.J. Pooley. The Integrated Modelling Support Environment a new generation of performance modelling tools. In G. Balbo and G. Serazzi editors, *Computer Performance Evaluation: Modelling Techniques and Tools*, pages 1–15. North-Holland, 1992.

[Sera81] G. Serazzi. A Functional and Resource-Oriented Procedure for Workload Modeling. In F.J. Kylstra, editor, *PERFORMANCE '81*, pages 345–361. North-Holland, 1981.

[TD86] S.K. Tripathi and L.W. Dowdy. Workload representation and its impact on performance prediction. In G. Serazzi, editor, *Workload Characterization of Computer Systems and Computer Networks*, pages 159–178. North-Holland, 1986.

[TS85] Y.C. Tay and R. Suri. Errors bounds for performance prediction in queueing networks. *ACM Trans. on Comp. Systems*, 3(3):227–254, 1985.