THE UNIVERSITY *of* EDINBURGH

MODELLING INTERACTIVE WORKLOADS OF TIME-SHARING COMPUTER SYSTEMS

Vera Noethe

Ph.D.
University of Edinburgh
1982

ABSTRACT

One of the major obstacles to performance evaluation studies is the definition of a representative or accurate workload. User scripts, a detailed description of user activities to specify an interactive workload, can be used as input for remote terminal emulators as well as a representation of an interactive workload the users impose on a system. User scripts are often based on the widely accepted user behaviour model 'LIST-MODS-RUN'. A method of analysis is described which suggests that this user behaviour is not realistic. More accurate models are given to describe user behaviour at system command language level. It is also studied how different users groups adapt to different loads. The results are then used to define a workload model. An automatic workload generator is described to generate workload models in form of user scripts ready for execution on ERTE which is a system designed to exert interactive workloads on a multiaccess system. Experimental runs with ERTE demonstrate the accuracy of the model and provide the basis for more realistic performance measures on EMAS and the development of ERTE.

C O N T E N T S

INTRODUCTION

The definition of accurate workload models is a crucial task in system evaluation studies. Because of the interdependence of system performance and workload characteristics the problem of workload characterization arises in almost every system or performance evaluation study.

Although the importance of workload characterization is widely recognized and much effort has been put into the development of this area, reliable techniques to describe and generate accurate workload models for interactive time-sharing systems are still missing. Several reasons can be given:

a) Describing a workload requires knowledge of user/machine interactions. Modern computer systems are very complex and sophisticated and there are innumerable ways of interacting with the computer. As the user community consists of human beings it is difficult to find patterns and describe them in a convenient way.

b) Information on user data is difficult to obtain, because

- monitoring facilities can falsify the obtained results
- technical problems can make it impossible to collect the data you want

- for privacy reasons users may not want to be monitored.

c) As there is no inherent feature set to describe a workload and the number of possible features is very high, the selection of the parameters is subjective and may well be inappropriate.

d) Having chosen a set of parameters the next step is to analyse the data. Problems involving several parameters suggest the application of a multivariate statistical technique. Most of those techniques (clustering techniques or factor analysis) are very controversial and the results have to be carefully judged. We therefore suggest another technique which generates more credible results.

e) Workload models tend to be highly system-dependent and cannot be easily moved to another system.

The aim of this thesis is to provide a method of modelling and generating interactive workload models for interactive time-sharing systems. For this target we have

1.) to study and describe the way the users work with an interactive terminal,

2.) to study behavioural differences between different user groups

3.) to study behavioural differences under different workloads

4.) to design, implement and validate an artificial executable workload model based on real observations at system-command-language level.

It is demonstrated that the models suggested so far are unrealistic and more accurate models are suggested. The workload model is then implemented in the form of user scripts so that they can serve as input for ERTE (EDINBURGH REMOTE TERMINAL EMULATOR). Experimental runs on ERTE are performed to validate the model and give an idea of the usefulness and potential of ERTE.

The contents of the chapters can be summarized as follows:

In chapter I we give a general overview of workload characterization problems to integrate our study.

In chapter II we briefly describe the system and the system command language where the necessary measurements were made to collect data for the model. As our measurements are done sessionwise we describe a method of analysis to split user sessions into different groups with each group having similar characteristics concerning the use of the system-command-language and the way the editor is used. We also study CPU time, pageturn, thinktime and elapsed execution time characteristics per command and number of commands issued per session. Furthermore we show that we find behavioural differences for different user groups and behavioural differences under different

loads. The results are used to describe more accurate user behaviour models for interactive time-sharing systems.

In chapter III we describe an automatic workload generator which takes the values defined in chapter II for the model parameters as input and generates user scripts ready for input to ERTE. In the first place the workload generator is desgined to generate as accurate and realistic user sessions as possible. For experimental runs it may be interesting to study a system's performance under an imaginary or a possible future workload. For this purpose the workload generator is designed in such a way that it is possible to generate such workloads by changing the values for the input parameters appropriately.

In chapter IV finally we run the generated user scripts on ERTE to validate the workload model and demonstrate in which ways ERTE and the automatic workload generator can be used.

In chapter V we summarize the results and draw conclusions for modelling workloads of interactive time-sharing computer systems.

C H A P T E R O N E

I. WORKLOAD CHARACTERIZATIONS

The workload of a system is defined as the set of all inputs a system receives from its environment. This definition requires some further explanation. In every evaluation study the system's boundaries have to be clearly defined. Those parts of the operating system responsible for resource management are usually regarded as part of the system and not of the workload.

A user job, for example, is usually considered as being part of the workload, not of the system. Compilers, assemblers, data base systems can be assigned to the workload or the system, depending on the particular study. If a workload is to be assessed it is essential to specify the time frame for the workload, whether it is an hourly, daily, weekly or yearly workload.

A system's performance is dependent on the workload the users impose on the system [HELL,FER1], and the workload is dependent on the users' behaviour. Users' behaviour is influenced by the facilities a system offers, for example the features of the system command language, the available compilers, the utility routines

etc.. Furthermore, it is assumed that users react quite appreciably to changes in system components or performance. But so little is known about these factors that in many evaluation studies it is assumed that a system's workload is constant over a certain period of time.

Obviously, there is a strong interrelationship between the workload and the performance of a system. Therefore the definition of an accurate and representative workload is a crucial and important task and arises in many computer system evaluation studies. However, in system evaluation studies, much consideration is given to correctness problems, reliability and user friendliness. Performance aspects re only considered at a late stage of a system's evelopment.

Initially, performance aspects may be thought of as being less important. For instance, if performance proves to be poor, one may expect e.g. a faster CPU or more memory to bring the system's performance to the expected level. Unfortunately this strategy does not always turn out to be true. The result are computer systems with a poor performance and unhappy users or customers. This is why the importance of performance aspects seems now to find wider acceptance and system evaluators try to include performance considerations as early as possible in system development.

If a new system is to be designed manufacturers would like to be able to accurately predict the performance for specific user applications and the requirements of the potential users. An accurate prediction of the workload imposed on the system later is essential, as the manufacturers have to consider satisfactory combinations and organizations of hardware and software, i.e. system configuration [ROEH].

In later stages of a system's lifetime an accurate workload model has to be presented to a target system to assist in performance improvements. Although the importance of workload characterization is now widely recognized, no reliable and accurate models have been suggested so far. In this chapter we describe the main problems and the approaches to workload characterization to integrate our study into this research area.

1. PERFORMANCE MEASURES

As the term performance will be used frequently throughout this thesis, we clarify our usage. Measures of performance can be classified into user-oriented and system-oriented measures [KOBA,FER1].

User-oriented Measures

User-oriented measures are such quantities as the turnaround time for batch queues or the response time or execution times in an interactive system environment.

In the literature we find several different definitions for response times. We may define the response time as the time interval between the arrival of a request and the time of completion. The arrival time may be defined as the time when the return key is pressed or the completion time when the first line is printed. The system reaction is often referred to as the interval of time that elapses from the moment an input arrives until it receives its first time slice or service. Execution time may be defined as the elapsed time a system needs to execute a command. All these terms are random variables. We can therefore describe their distributions, expected values, variances etc..

System-oriented Measures

The main system oriented measures are throughput and utilization. Throughput is defined as the number of jobs a system can process per time unit. The utilization of a resource is that fraction of time the resource is busy. The utilization is measured for the CPU, the I/O-channel, compilers etc.

2. WORKLOAD MODELS

As the workload of a system is not known in advance it cannot be presented to a system being developed. Therefore a natural workload has to be replaced by a workload model. During an evaluation study there are two workload-oriented operations to be performed. These

are:

a) the specification of the type of workload which is supposed to drive a system.

Evaluators may be interested in specifying the performance of a system under the production workload or finding the maximum throughput rate. In such a case a workload has to be selected which saturates the system.

b) The characterization of the workload under study.

This means, it has to be decided whether a natural workload, or a suitable model is to be used for the particular study. In measurement sessions a system may process a sample of the real workload. This sample, too, can be considered a model of a real workload, whereas in analytic or simulation studies workload models have to be defined. We now give a brief survey of the different kind of models an evaluator can choose [FER1].

2.1 Types of workload models

A natural workload model is a sample job stream taken from a production workload and used to drive a system at the moment when the workload is produced by the users. Obviously such models can only be used in measurement studies. This kind of workload is also called a model because it is shorter than the real workload. All models which do not consist of a live sample from a

production workload are called artificial workloads. We distinguish between executable and nonexecutable artificial workloads.

Nonexecutable artificial workload models are used in analytic and simulation studies. The workload models are usually restricted because of the constraints of the model solution techniques. However, the boundaries of solvable queuing systems are expanding and the workload models are getting less restrictive, although the accuracy of such models is still limited. An advantage of analytic workload models is that they are very flexible, at least concerning those aspects of the workload they model. Models used in distribution driven simulators and those used in analytic studies are very much alike. Models used in distribution driven simulators may be more accurate, as any distributions for the workload parameters are acceptable. Furthermore, they are very flexible and reproducible. Because of their higher degree of detail they require more information. This means they are more difficult to construct. Their usage costs are generally higher as they drive a simulator.

Executable artificial models are used in measurement experiments. This kind of model is preferred to natural workload models because they are reproducible and more flexible. On the other hand, they are more expensive to construct. When the measurement takes place, the system

must be exclusively dedicated to this workload model, which is for heavily used systems a major drawback.

It is this kind of workload model we are going to construct and experiment with in this thesis.

Executable artificial workload models

There are several kinds of different executable workload models evaluators can use. The models described below all have their advantages and disadvantages. Again the choice depends on the particular study and the facilities offered [LUCA].

Instruction mixes

A very popular approach is the so-called instruction mix. Instruction mixes are used to represent the workload on the CPU by describing the relative frequency of the instructions such as ADD, MULTIPLY etc.. The average execution time per second can be calculated if the instruction execution time of each instruction is known for a given CPU.

The instruction mix is of valuable help in the design and implementation of a processor, if the design of a CPU adopts microprogramming. The instuction mix may indicate whether to minimize space occupancy of instructions or to emphasize performance. But there are

several problems to be considered. The first is representativeness. It is expensive in machine time to produce instruction mixes and therefore evaluators rely on already existing instruction mixes. If you do want to evaluate a realistic instruction mix, it is difficult to select a set of representative programs.

Another drawback is that the instuction mix does not include any serial dependencies. In the case of a pipelined CPU the execution speed is dependent on the sequence of the instructions and not just on the frequency [STON,TATE]. Furthermore, the instruction mix does not include demands for I/O resources. The applicability of instruction mixes is thus restricted and results should be used carefully.

Obviously, the instruction mix is a nonexecutable workload model. However, it is fairly easy to construct an executable workload model from it by taking a program that is characterized by the same instruction frequency. In this case the workload may consist of a single job and serve as a job model.

Instead of instruction mixes you can take simple statement or command mixes as suggested in [WRIG]. If a command mix is calculated consisting of the relative frequency of system's commands we find the same problems as for instruction mixes: frequencies provide no information on dependencies and correlations among the

commands thus leading to unrealistic performance results.

Kernel programs

Another approach is called 'kernel program method' or 'standardized benchmark' to represent a load on the CPU. Typical kernel programs for scientific installations consist of matrix inversion, square root approximation, polynomial evaluation etc., or for commercial applications a file updating kernel. For interactive systems scripts can be defined which belong to the category of kernels. Scripts consist of a sequence of interactive commands and represent a model of a set of interactive terminal users accessing a system simultaneously.

A critical issue with kernels is their representativeness. The scripts or kernels may seem a good possible model, but in practice the actual workload may differ. Kernel programs are more widely applicable than instruction mixes as they include information on the instruction sequences. For a pipe-lined machine evaluation or modeling I/O operations kernels provide a reasonable way of presenting a relatively accurate workload to the system under study.

Benchmarks

A selection of real jobs considered to be representative of a given class of application programs is called a benchmark. Given a set of mixed benchmarks a system's performance can be estimated under 'realistic' circumstances. In a system selection process benchmark tests are very common: comparable benchmarks are run on all proposed systems; then the total run times of the benchmarks on the different systems are compared thus getting an idea about the relative performances of the systems under study. Although benchmarks have some advantages over instruction mixes or kernel programs, some disadvantages should be pointed out.

Firstly, the selection of representative jobs is not as easy as it seems to be at first sight, especially for complex and sophisticated systems with a great variety of users and applications. Secondly, to get hold of representative programs is another problem which should not be underestimated. Privacy reasons may prevent users from offering their programs for benchmarks.

Another drawback is the inflexibility of benchmarks. Joslin [JOSL] describes some of the difficulties and offers some solutions. Benchmarks are also restricted to existing systems and the jobs running on that particular system. Benchmarks cannot be moved from one

system to another to compare performances, as the benchmark jobs depend on the facilities of a particular system.

Synthetic Jobs

A synthetic job is an artificial, parameterized program having the typical characteristics of a real program. The behaviour of such programs can be controlled by changing the parameters. The parameters of these synthetic jobs are the CPU time demands, storage space requirements etc.. The big advantage of synthetic jobs is their flexibility as they can simulate a wide range of programs [BUCH,WOOD,BERN,KERN].

However, again we face the problem of representativeness. It is difficult to assess whether a particular choice of parameters provides a representative program. Whether to use synthetic programs depends on the performance measure. If the system throughput is to be examined, a set of synthetic jobs may well be suitable, whereas if the response time is to be measured, the synthetic job may well be unrepresentative of the real workload and misleading results are likely to be obtained.

Executable Traces

Traces are obtained by tracing the production workload; e.g. for batch processes the orders of card decks are recorded or for interactive services the commands submitted by the user community, the CPU time used for each command etc.. It is also necessary to trace and store the files accessed by the jobs/users. Traces have quite a high degree of representativeness compared with instruction mixes or kernel programs. Another positive aspect of traces is that they are not too expensive to construct.

The disadvantages of traces are their inflexiblity and lack of system independence. As traces also include special file calls it is not possible to have play backs on different systems. Another disadvantage is the lack of compactness. An example of a trace driven model is presented in [MINE].

2.2 Characteristics of workload models

There are, obviously, good and poor quality models. This means the quality of a model must be judged against some criteria. In our examples we have already mentioned several criteria such as flexibility or representativeness. We continue with a more detailed description of the most common criteria against which to judge workload models.

a) Representativeness or Accuracy

In the literature we often find the term representativeness when discussing workload models. However, this term is very vague. Thus Ferrari [FER2] suggests replacing it by the term accuracy. There are several definitions of accuracy:

- A workload is accurate if it consumes the same physical resources at the same rates as the workload it tries to represent.

This definition is unsatisfactory as it is heavily machine-dependent and secondly the choice of the resources whose consumptions are to be considered and of the rate definitions influence the performance of the system in an unknown way. The set of resources chosen may be incomplete, or if it is not, it is unknown what consumption rates are needed to determine the performance indices of interest.

Machine dependence is irrelevant in studies where a system is not modified, the second drawback mentioned above is present in all studies.

- Workload Wb models workload Wa if it performs the same functions as Wa.

This function-oriented definition is applicable if the workload to be modeled performs very few well-known

tasks. However, the relative proportions of the various tasks in workload Wb or the criteria for choosing the input data cannot be determined by applying this definition.

- Workload Wb is an accurate model of workload Wa for System S and performance index P if the values of P produced by Wb when running on S equals those produced by Wa running on the same system, where P can be a set of global performance indices.

Obviously, this definition is also machine-dependent. If we assume that both the workload and the model can be transported to another system S' it is difficult to say whether they are still equivalent if S is replaced by S'. There are, of course, techniques, to enhance model validity such as using real components instead of synthetic ones. The problem of model validity will be discussed later in more detail.

Which definition an evaluator wants to work with depends very much on the particular study; in selection studies the functional definition may be more attractive, but in general the performance oriented definition should be employed whenever possible, with a workload model as functionally similar to the real workload as possible.

b) Flexibility

A flexible model should be easy to modify to reflect changes in the real workload. This is especially the case if a system is to be tested under different workloads.

c) Simplicity of construction

Another important point is that a workload model should be as simple to construct as possible. In many cases it is time- and cost-consuming to gather the data for the workload model; measurement is a complex task, very often special hardware tools are required (hardware monitors) which may be very expensive.

d) Compactness

Compactness gives information about the degree of detail and accuracy. Compact models are in general not very detailed but easier to construct.

e) Usage costs

There is obviously an interrelationship between compactness and usage costs. Compact models are relatively crude, less detailed and hence easier to use.

f) System independence

To design a system-independent workload may be a very desirable target although this is impossible to achieve. The workload the users impose on a system depends very

much on the facilities of the particular system, i.e. the command language, the utility routines, the hardware configuration and the actual user population. Complete system independent workload characterizations are unrealistic but a characterization as independent as possible should be a major target.

g) Compatibility

Workload models and the system or system models must be compatible, i.e. a model which is supposed to drive a real system must be composed of programs executable by that system.

3. DESIGN OF AN EXECUTABLE ARTIFICIAL WORKLOAD MODEL

If we are going to design an executable artificial workload model, we first have to define the job parameters to be included in the system. As already mentioned above, there is no inherent feature set describing a workload [AGR1,AGR2,AGR4,FER1]. Therefore the selection of these job parameters is very subjective and care should be taken to choose parameters relevant to an accurate workload description. The selection of the parameters is dependent on the level of characterization.

Levels of characterization

Workloads may be described at 3 different levels [FER2].

1) at the physical level in terms of the consumption and/or consumption rates of the resources (CPU time,main memory space etc.).

2) at the virtual level in terms of the amounts and/or rates of resources (higher-level language statements,virtual memory space), or

3) at the user level in terms of commands the users type in, command sequences or number of commands per session.

The higher the level of characterization the less is the dependence of the characterization on the system. It is easier to to define workloads at a lower level as more is known about those parameters. From this point of view characterizations should be designed at the lowest possible level. Characterizing workload at the physical level is a very popular approach [FOXL,AGR1,ARTI,ESPO,MEAD]. Depending on the context it may not be sufficient to define a workload at the physical level. In an improvement study, for instance, a level 1 model could be dependent on system aspects which could be modified as a result of the study. In this case a level 2 model is necessary as both the modified and unmodified system must be exercised under

the same workload. If the modifications affect the characterization on which the model is based it is hard to verify that you do have the same workload.

Another point to mention is workload prediction. If you are interested in predicting the future workload of a system, it is more suitable to define a workload at the application level and observe changes in behaviour and application tasks. As already mentioned above, not much is known about these factors which makes it very difficult to define a workload at this level.

Defining the values for the job parameters

For the moment we assume we have a set of job parameter p1,.....,pn. There are several ways of constructing a workload model from them [FER1]:

a) The probability distributions for these parameters are derived from sample jobs. This method does not take into account any correlations or dependencies among the parameters, the parameters are derived independently from each other. The disadvantages described for instruction mixes apply for this kind of model.

b) Another possibility is to collect a certain number of jobs and take a sample. The parameters of the sample jobs are used to define jobs in the workload model. In this case correlations and dependencies among parameters

are automatically modeled. However, it is necessary to collect quite a large number of jobs to get representative workload models. This method bears some resemblance to executable traces.

c) If we describe a job by a set of n parameters we have an n-dimensional space of job parameters. The jobs are now classified or grouped by applying multidimensional techniques such as cluster analysis. The idea is to find 'natural groupings' in the underlying data and take from every group some representatives to construct a workload model.

d) A similar starting point is suggested by Sreenivasan [SREE]. Instead of clustering the jobs, the characteristics of the drive workload are determined by defining the joint probability density of the real workload.

This is achieved in the following way: if Xi,....,Xs are the s variables to characterize a workload a job is regarded as a single point in the s-dimensional space, whose axes may represent resource demands. A workload is consequently regarded as a set of points. Another possibility is to divide the region into a number of discrete cells, by dividing each axis into L parts thus getting L**s different cells. For each cell the percentage of the total number of jobs are then computed - which is the joint probability density of the

workload. Obviously, depending on the number of variables and division chosen we get an unduly fine and practically unmanageable number of probabilities or we have a crude division and get high variances for each cell. A practical example for this solution is thus only presented for a two-dimensional space.

4. IMPLEMENTING AN EXECUTABLE MODEL

The next step towards an executable artificial workload model is to find a convenient way of implementing the workload model. This depends of course to some degree on how the model has been designed.

Benchmarks. Obviously, if the job parameters have been derived from real jobs, benchmarking is the most realistic way to implement the model. For all other modelling techniques benchmarks can also be used [BUZE].

Kernel programs. Kernel programs or standardized benchmarks can be used to implement any of the workload models. However, to mimic a complex workload may be difficult with kernel programs [SALT].

Synthetic Jobs. It is also possible to implement the workload models with synthetic jobs. But this again is only advisable if the underlying model is fairly simple. An example of this approach can be found in [BUCH].

5. MODEL CALIBRATION AND VALIDATION

A model's usefulness is considerably reduced unless it is calibrated and validated [KOBA,FER1], although this aspect is often neglected. The objective is to demonstrate that a model is reasonably accurate. We have already considered different definitions of accuracy; if we take the performance oriented definitions we will have to demonstrate that the model W produces the same performance figures as the natural workload does. If it does not, the underlying model is obviously faulty. Inaccuracy can be due to several reasons [FER1]:

- formulation inaccuracy, if the model is incomplete or lacks the necessary detail,
- solution inaccuracy, if the solution model is incorrect or a simplified method of solution was applied,
- parameter inaccuracy, if the values for the parameters are not chosen properly.

To remove formulation inaccuracies, the model has to be corrected and tested. This stage is called model calibration. However, it is obvious that a model can only be calibrated under a limited number of conditions. Thus, a model is valid for some input conditions if it satisfies the requirements when its input variables correspond to that condition.

For our study we are going to develop and design a workload model following the technique of grouping jobs according to some parameters. To implement this model we are going to write user scripts to simulate interactive users at the terminal. Those scripts are then presented to a target system via a remote terminal emulator.

Our first step towards this target is to construct something like a trace for interactive terminal users. From this trace we define a set of job parameters and try to find typical terminal users by applying special grouping techniques described in the next chapter. We then define user scripts which can be viewed as kernel programs.

But before we start to explain our model, we discuss a popular technique for designing workload models, called cluster analysis, used to find groupings in a multidimensional space. The reason why we describe this technique in detail is that it enjoys quite a high popularity for the problem under study. However, as we shall see, not enough attention is drawn to the weaknesses of this approach.

6. CLUSTER ANALYSIS TECHNIQUES

The use of cluster analysis techniques starts by examining the system's workload. Although this approach

is often applied to design workload models [AGR1,AGR2,AGR3,ARTI,ESPO,FANG], inherent problems and weaknesses of this technique make it difficult or even impossible to verify the results obtained. This fact is often neglected and cluster analysis techniques are applied without the necessary care. The following general considerations about cluster analysis techniques will outline the problems of this approach.

The application of cluster techniques is a very subjective approach. This is already reflected in the very vague definitions of what a cluster technique is. A cluster technique is usually defined as a technique which tries to group points in a multidimensional space which are "similar" to each other. What is similar is of course difficult to define and the very many clustering techniques found in the literature allow the users to influence their results almost "as they please". But because of this subjective nature the users should know about these problems and judge the results obtained from a clustering technique as carefully and critically as possible.

If cluster analysis techniques are to be applied the user is faced with the following problems:

a) The selection of the feature set

In a workload characterization study the selection of the feature set is quite a difficult problem. As there are so many parameters that may give relevant information of a system's workload the user has to cut them down to a smaller number. As there are no a priori feature sets for this purpose the selection of the features may well be inappropriate.

b) Before applying clustering techniques to the data of the selected feature set the user has to test whether the data are random or not. Clustering techniques have the unfortunate feature of producing clusters from every kind of data whether they are random or not. Every serious cluster analysis study should therefore start with a test of the randomness of the raw data. A survey of how to test randomness of data can be found in [DUB2]. It should be pointed out, that such tests cannot "prove" the nonrandomness of data because the interpretation of such results is again subjective but they can well be employed to help to decide whether a clustering technique is reasonable at all.

Nonrandomness of the data does not mean that cluster techniques are reasonable to apply, it only suggests that they could yield reasonable results.

c) Cluster analysis involves the defining of similarities or distances between individuals. Similarity coefficients which take values in the range

of 0 and 1 measure the relationship between two individuals, given the values of a set of p variates common to both. Examples of similarity coefficients for binary data are (a+b)/d,a/(a+b+c), (2a)/(2a*b*c). An unfortunate feature of similarity coefficients is that they take different values and the selection heavily depends on the underlying data. This selection is apparently also subjective and a good knowledge of the data base is needed to select and justify the selection of the similarity coefficient.

Another possibility is to define distances between individuals. For this purpose we define a metric. If d(x,y) is a numerical function of pairs of points of a set E, then d(x,Y) is a metric for E if

1) d(x,y)=0 if x=y; 2) d(x,y)=d(y,x) 3) d(x,z)+d(y,z)>=d(x,y)

The most common distance measure is the Euclidian distance between points i and j denoted by Dij :

$$Dij=\left\{\sum_{k=1}^{p}(X_{ik}-Y_{jk})^2\right\}^{1/2}$$

where Xik is the value of the k th variable for the i-th entity. A refinement can be achieved by giving weights to different features, for example whether differences in matches are important or unimportant for different features. [AGR1]. There are many more different distance measures to be found in the literature and surveys can be found in [EVER,ANDE].

To apply Euclidian distance it is often necessary to scale the data as differences in scale may change the results and the relative distance between individuals. A common scaling technique is defined by : Zik=(Xik/Dk) where Dk is the standard deviation of the k-th variable. Agrawala,Mohr [AGR1] scale their variables in such a way that about 95% of the observed data lie in side-10 hypercube to avoid the feature space being distorted by the presence of outliers.

e) The next problem is to select a clustering technique. As already mentioned above no matter which cluster techniques is selected it will almost certainly produce clusters.

However, different clustering algorithms produce different clusters and the user is faced with the problem of finding out which clusters reflect the "natural groupings". This seems to be the most crucial point in cluster analysis techniques, as there do not exist reliable techniques to show that the produced clusters do really reflect the natural groupings of the underlying data.

There are suggestions in the literature to define error estimates and the stability of the clusters [EVER,AGR1,ANDE,ARTI]; Dubes and Kain [DUB1] give a survey on approaches and difficulties of cluster validity problems and explain why it is so difficult to

demonstrate the validity of clusters. They regard cluster analysis as a tool for discovery rather than an end in itself and advise users interested in validity aspects to try different techniques and look for common clusters.

In [DUB2] some techniques are described to compare cluster techniques, but also here no reliable techniques can be provided. This does not mean that cluster techniques should be ignored. Users should be aware of the difficulties and as statistics does not provide the necessary techniques to validate cluster techniques, users who work with cluster techniques have to apply general knowledge and experience of the subject matter to interpret cluster results. This, of course, is a tricky task and detailed knowledge of the data base and abilitity to interpret and judge the credibility of the results is needed.

EXAMPLES

To demonstrate these difficulties, two clustering techniques are presented and applied to the data on a VMS 11/780-Accounting file. This accounting file was collected from the department of computer science's VAX 11/780. The feature set consists of

- the consumed CPU-time (10 msec units)
- the number of pagefaults
- the peak working set size

- the number of direct I/O requests
- the number of buffered I/O requests
- the elapsed time of session in seconds

These features are recorded for each logged on session. The following algorithms are applied:

1.) Modified version of ISODATA [ANDE]

A set of cluster centres has to be given. The algorithm calculates for every pattern the closest cluster centre and assigns the pattern to this cluster. If a pattern is sufficiently removed from all the cluster centres a new cluster is created and this pattern assigned to this cluster. When all patterns are assigned, the cluster centres are recomputed and a new data pass is initiated. This algorithm has a scaling option. The distance measure employed is the Euclidean Distance.

2.) Clustering using a similarity measure based on shared neighbours [JARV]

This nonparametric clustering technique defines similarity between patterns in terms of shared near neighbours. For each pattern the n nearest neighbours are computed (using Euclidian Distance). Each pattern can be regarded as its own zeroth neighbour. An integer label table of length n has to be set up,with each entry initially set to the first entry of the corresponding neighbourhood row. The algorithm works as follows: All possible pairs of neighbourhood rows are tested and both label entries are replaced by the smaller of the existing one if both zeroth neighbours (the points being tested) are found in both neighbourhood rows and at least t neighbour matches exist between the two rows. The clusters under the k,t selection (k nearest neigbours, t matches) are now identified by identical labeling of the points belonging to the clusters. Figure I.1.

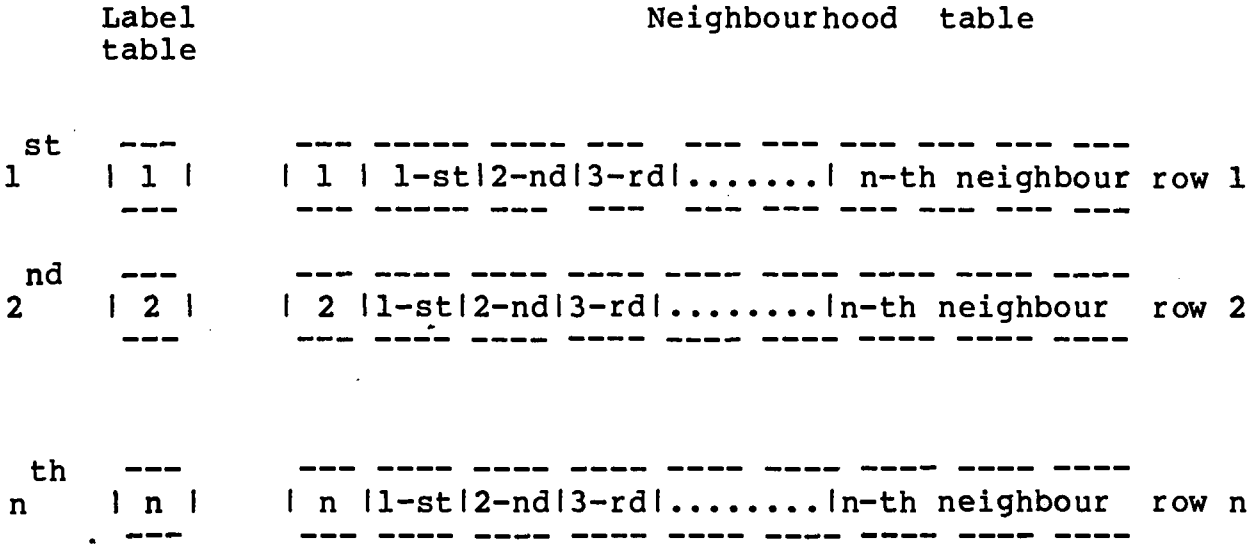

Figure I.1.

T E S T E X A M P L E

Table 1 represents the results obtained from the modified version of ISDODATA for 300 patterns or interactive sessions.

No	Members	CPU	Pageflts	WSpeak	dir I/O	buff I/O	Login
1	159	79	158	107	50	13	1
		49	86	23	28	6	0
2	46	563	823	202	411	44	10
		271	193	41	296	10	10
3	21	856	1579	260	335	70	8
		314	457	56	80	22	5
4	18	2359	3604	284	2072	159	23
		367	729	45	857	37	10
5	11	2272	1915	280	906	108	16
		1122	223	54	270	23	7
6	6	4658	4177	339	1026	317	19
		809	861	27	274	59	4
7	4	3036	6109	359	1829	193	30
		309	681	51	436	76	15

Table I.1.

This table shows that 6 clusters could be found with more than 4 members. For each cluster the mean value and standard deviation for each feature are shown. It suggests that a significantly large number of sessions are very short and have a low resource consumption rate (Clusterno 1,2). Although sessions with higher resource

consumption rates and longer login-times can be found in clusters 3 and 4, clusters with really demanding sessions are missing.

Table 2 lists some of the non-clustered processes.

No.	Members	CPU	Pageflts	wspeak	dir I/O	buff I/O	login
11	1	23276	16306	361	15969	1429	109
12	1	40343	35602	361	10939	1814	137
13	1	11808	8544	361	4129	681	46

Table II.2.

86% of the 300 interactive sessions could be clustered, but as table 2 already indicates, the 14% jobs which could not be clustered contribute to a very high degree to the overall resource consumption rate and login time. Table 3 gives more accurate details:

```
JOBS:
clustered:      261   UNCLUSTERED:  40  TOTAL:    301
CLUSTERED ARE:       86.7%

CPU:
CLUSTERED:   152023   UNCLUSTERED:  378409 TOTAL: 530432
CLUSTERED ARE       28.66%

PAGEFAULTS:
CLUSTERED:   207253   UNCLUSTERED:  249641 TOTAL: 456894
CLUSTERED ARE       45.36%

WSPEAK:
CLUSTERED:    42069   UNCLUSTERED:  11751  TOTAL:  53820
CLUSTERED ARE       78.17%

BUFF I/O:
CLUSTERED:    87517   UNCLUSTERED:  155660 TOTAL: 243177
```

CLUSTERED ARE 35.99%

DIR I/O:
CLUSTERED: 11618 UNCLUSTERED: 32080 TOTAL: 43698
CLUSTERED ARE 26.59%

LOGIN-TIME:
CLUSTERED: 1420 UNCLUSTERED: 2457 TOTAL: 3877
CLUSTERED ARE 36.63%

Table I.3.

About 14 per cent of the jobs consume about 50 per cent of the system resources. It is obvious that in case of a workload characterization study these 14% of jobs cannot be classified as "outliers" and neglected. To classify these large processes cluster analysis techniques fail, or at least this algorithm does. If these 14% demanding sessions are to be classified, relationships between the features can be examined leading to another yet unsolved problem.

The second algorithm produces the following results:

No	Members	CPU	Pageftls	Wspeak	dir I/O	buff I/O	login
1	114	53	107	88	33	10	1
		19	30	8	7	3	0
2	40	306	627	153	207	33	3
		134	334	19	134	9	4
3	39	5081	4850	308	1830	250	25
		4547	1731	61	1423	163	15
4	27	506	683	238	212	45	3
		433	235	18	109	18	3
5	24	1540	1766	293	653	88	12
		1195	402	48	415	37	8
6	18	1539	1249	206	1562	76	70
		1437	648	35	1666	39	57
7	8	95	168	105	47	15	0
		5	14	4	2	1	0
8	5	24064	16400	330	14223	1306	103
		11242	10444	30	2233	633	20

Table I.3.

The percentages of the resources clustered are higher for this algorithm. With 91 per cent of clustered jobs 80 per cent of the total CPU time could be clustered. The values for the other parameters lay in a similar range. However, these results are due to the fact that the nearest neighbours are grouped, no matter how far away they are from each other. Sessions may be grouped which should not be considered to be similar. For example, this algorithm groups resource intensive jobs which cannot be grouped by the first algorithm because the Eucledian Distance is too big. Those jobs are 'outliers' for the first algorithm.

Both algorithms suggest that there are quite a lot of sessions which are very short. The CPU time used is around 50 Msecs, the number of I/O operations is very small and the login times are less than a minute. These sessions are in group 1 of algorithm 1 and groups 1 and 7 of algorithm 2. Another fairly common group generated by both algorithms is in groups 2 and 3 of algorithm 1 and groups 2 and 4 of algorithm 2, although the actual values differ.

Group number 6 of algorithm 1 and group number 3 of algorithm 2 could also be considered 'similar', but the values of these groups differ already significantly. The same applies to groups 4 and 5 of algorithm 1 and groups 5 and 6 of algorithm 2. Group number 8 of algorithm 2 does not have a corresponding group among the groups generated by algorithm 1.

This example shows some of the inherent difficulties of cluster analysis techniques. As there are no techniques to verify whether the clusters do reflect the 'natural grouping' of the underlying data we have to rely on intuitive interpretations of the results.

The next step is to define a workload model based on the clusters and validate it. However, validation and calibration of workload models is very tricky and is often neglected. Due to difficulties in finding a suitable remote terminal emulator Artis [ARTI] has only

reported benchmarking for batch processing. Neither Esposeto and Fangmeyer nor Agrawala and Mohr have provided experimental data on model validation and calibration. They have not shown their workload model to be an accurate model for the real workload.

This does not mean that cluster analysis should not be applied for the workload characterization problem. These considerations are to demonstrate some problems and difficulties of cluster analysis. But even after a serious and detailed cluster analysis study, the problem of validation is still decisive, because a user might well find natural groupings. But this does not necessarily mean that these groupings and the parameters chosen are the relevant ones for simulating accurately a workload. The right choice of parameters with 'approximate' classifications may well be more important than discussing and trying many different techniques to find 'natural groupings' using less important parameters for the workload characterization problems. To find out which parameters are needed to define a workload model much more attention should be paid to actual model validation and calibration.

Cluster analysis techniques are still serious contenders as a basis for model definition. One reason for this is that alternative techniques such as factor analysis or principal component analysis generate the same controversy as cluster analysis techniques [CHAT].

For our model definition we choose a different approach. We describe the workload model basically at system-command-language level. For this purpose we divide the commands on EMAS into five different groups and calculate the percentage of each command group for each different session separately. Since sessions are then grouped according to their relative percentage of different command groups we avoid scaling problems. Instead of applying one of the standard cluster analysis algorithms we programme an algorithm (described in the next chapter) which is somewhat unconventional for this purpose but still straightforward. And above all, it is demonstrated that the choice of parameters and this algorithm are suitable to define accurate and credible interactive workload models. The design of the artificial executable workload is described in the following chapter.

C H A P T E R T W O

II. DESIGNING THE WORKLOAD MODEL

In this chapter we describe how we design the workload model. For this purpose we apply an algorithm to structure user sessions into different groups. The groups generated by the algorithm are the basis for the workload models. Behavioural differences between different user groups and changes of behaviour under different workloads are studied. An attempt is made to compare some of the results from our experiments with results from other systems to find out which aspects of user behaviour are system dependent or system independent.

1. THE MEASUREMENT ENVIRONMENT

We first describe the environment in which the measurements were made and the system-command-language in order to understand to which kind of system and user population these results can be applied.

1.1. The system configuration

We start by describing the environment in which the

measurements were made and the system-command-language. The data for this analysis were collected on a dual processor ICL 2972 under EMAS (EDINBURGH MULTI ACCESS SYSTEM) [MILL,WIGH,WHIT,REES]

EMAS was originally developed and designed for ICL 4-75 computers and later re-implemented on an ICL 2980 and ICL 2972 [STEP]. EMAS is a multi-purpose time-sharing system written in the high-level language IMP [STEP]. Foreground and background users run their programs in independent virtual memories and share the available resources. Virtual memory support, basic device driving and scheduling is provided by supervisor, which is non-paged; basic file system, communication service and some supervisory functions are performed by director. Director occupies part of the virtual memory belonging to each process. Explicit I/O requests for interactive devices are handled by Director whereas slow peripherals, such as card readers or line printers are handled by a system process called demons.

When a user logs on to the system or runs background jobs a process containing only director is created. Director then extracts from the user file index the identifier of the basefile. This program package contains the principal components of the subsystem (Picture II.1.2). The subsystem is loaded and control transferred in the case of non-privileged mode to the basic command interpreter (BCI).

BCI interprets the commands typed in by the users at interactive terminals as calls to external routines. The FDP (File Directory Package) uses the primitive file facilities of Director to provide the basic set of file facilities of Director to the rest of the subsystem and manages the virtual memory. The EDCP (Environment Definition and Control Package) provides an interface for the typical file handling requirements of programs and compilers. The Contingency handler uses the typical facilities in Director to control the action in the subsystem.

The conceptual organisation of EMAS processes is shown in Figure II.1.

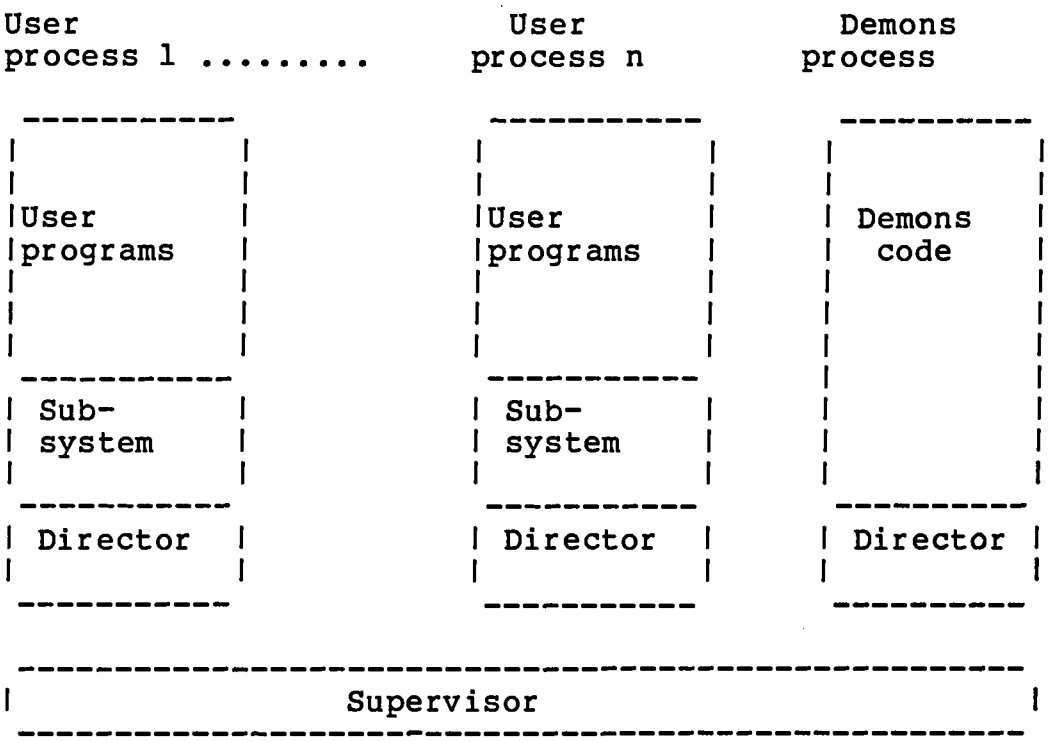

Figure II.1

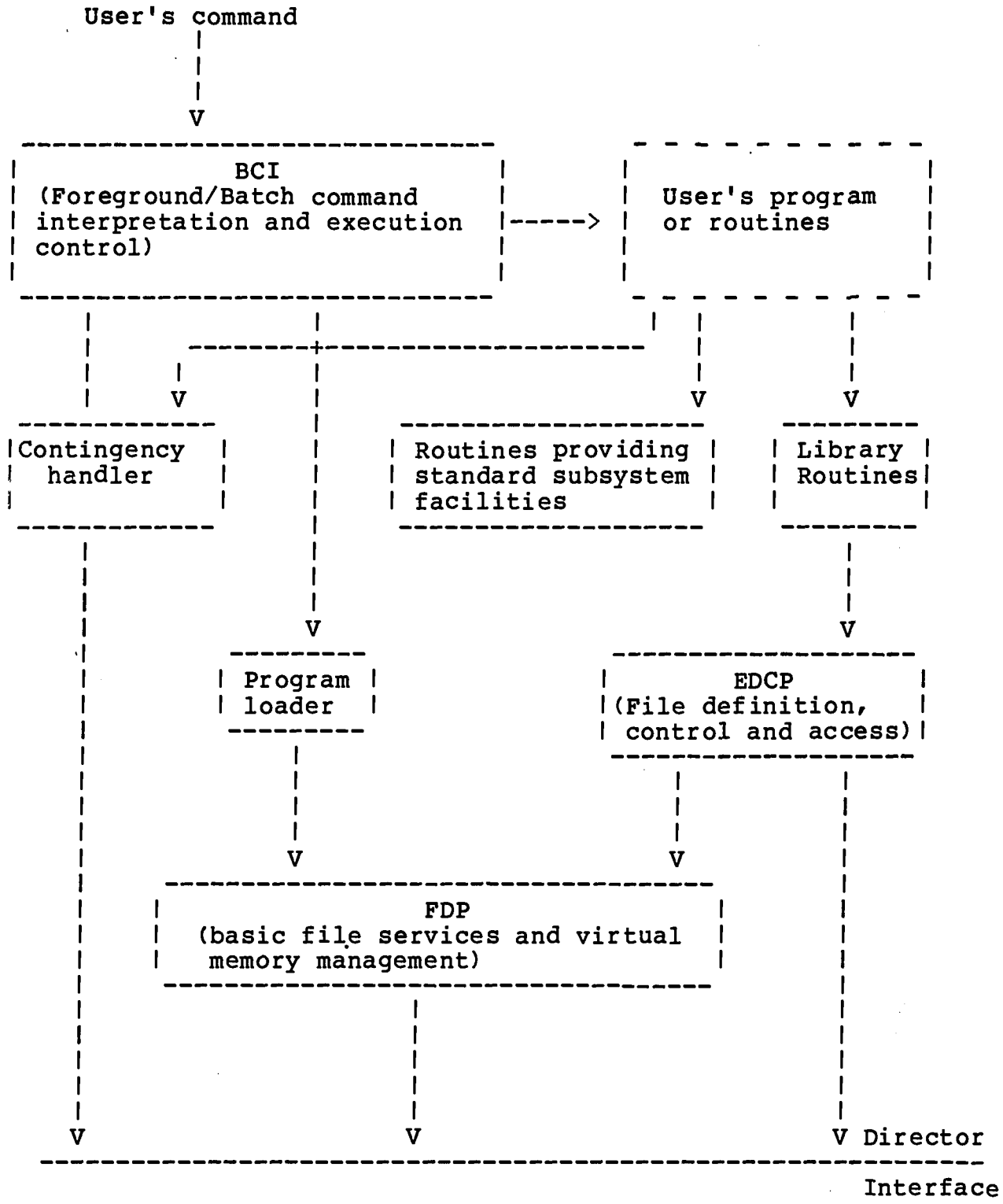

Figure II.2 Principal components of the subsystem

The EMAS Hardware configuration

2 2972 Order Code Processors (OCP).

These processing units for the system contain units such as the arithmetic unit, address generation unit, address translation unit. I/O operations are left to peripheral controllers.

2 Storage Multiple Access Controllers (SMAC).

Main store is composed of several autonomous modules each composed of a store multiple access control and a number of 128 K store blocks. All accesses to store take place over the SMAC.

2 Store Access Controllers (SAC)

These are autonomous units which provide 4 or 8 Trunk links each to which a similar number of peripheral controllers can be attached. They also control the concurrent passage of data between the peripheral controllers and the OCP.

3 Disk File Controllers (DFC)

These units control the EDS Disk drives. One DFC can control up to 8 EDS.

14 Exchangeable Disk Storages (EDS)

These free standing disk drives have a capacity of 100 mbyte per pack.

3 General Peripheral Controllers (GPC)

These free standing autonomous units control the basic peripherals (e.g. printer, card reader, operating stations).

Figure II.3. displays the current configuration.

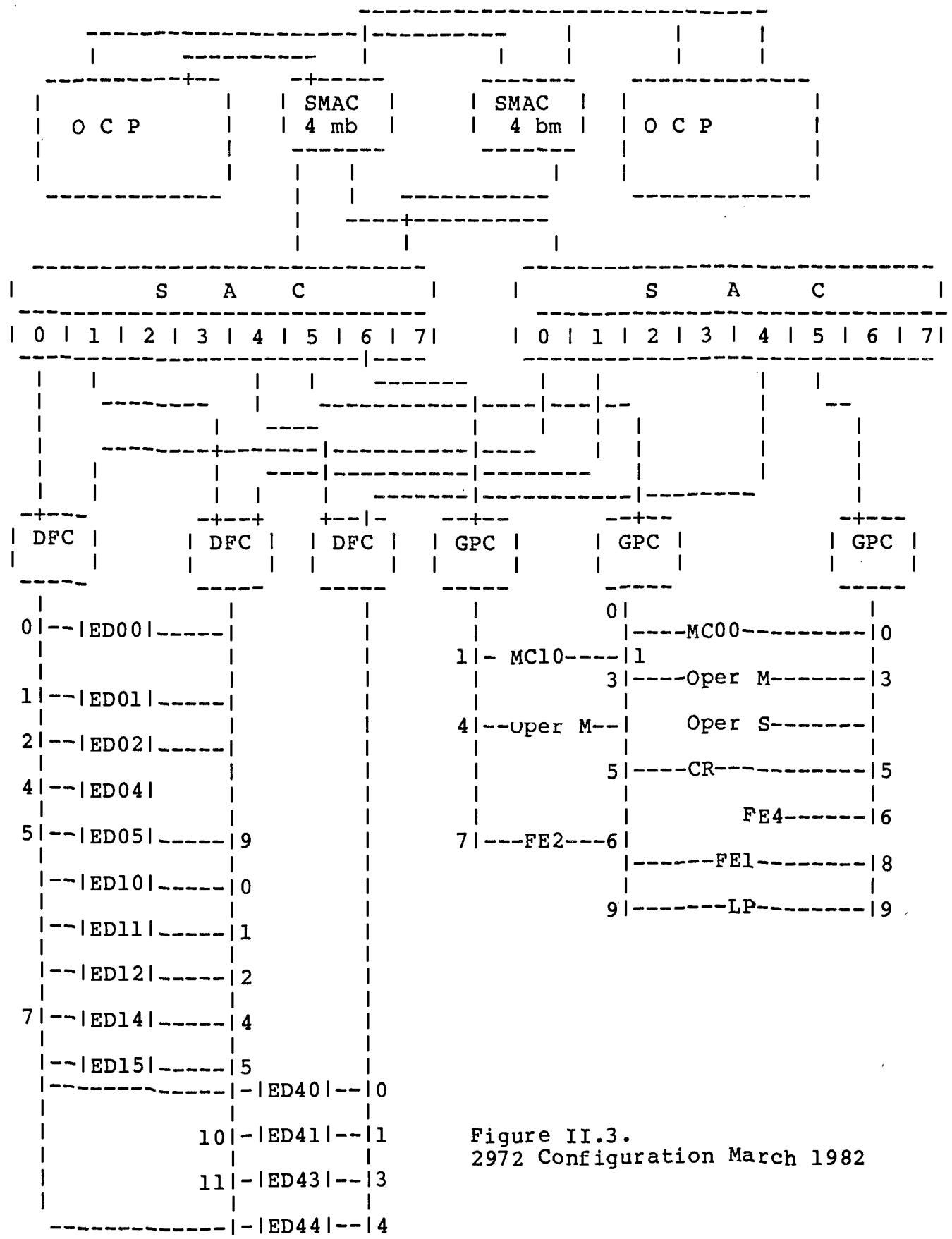

Figure II.3.
2972 Configuration March 1982

1.2 The system command language

The commands of the EMAS system command language can be grouped as follows:

General File Utility Commands. These commands operate on files as units and operate on any type. This group includes commands to perform general file tasks such as listing the contents of a user's directory, destroying a file or permitting a file to other users.

Type Specific File Commands. These commands are used to carry out functions on files such as sending, copying or listing of specific files. This group also includes commands to send files over a network to other host computers.

Edit Commands. This group contains calls to all available Editors on EMAS (ECCE, EDIT, SCREED, VECCE etc.).

Compile Commands. This group includes calls to the compilers available on EMAS (IMP, PASCAL, FORTRAN, ALGOL etc.).

Run Commands. In this group we include all commands to run user written programs.

Other Commands. The commands in this group are not further classified and include sending messages to other users, enquiries of system statistics, defining special I/O channels and start and stop commands to terminate a session.

A detailed description of the subsystem and the command language can be found in [MILL].

2. TYPES OF USERS

Since EMAS is a multi-purpose operating system, we find many different kinds of users working on EMAS, from the complete novice user to experienced research staff from different departments and to system programmers. Thus, the kind of tasks these users perform on EMAS varies widely. To find out what kind of load is imposed by the users, we decided to divide the user community into groups and study their behaviour separately. In particular, we are interested in whether novice users, medium users and expert users differ and if they differ what are the differences and consequences? Therefore we classified the user community into three groups [NOE1]:

- Novice users
- Medium users
- Expert users

In our case the selected novice users are Computer Science students at the beginning of their first year, medium users are Computer Science students at the end of

their first year, whereas the expert users are research staff or research students from different departments (chemistry,physics, computing centre, computer science). Those three user groups can be expected to differ in the following points. The novice and medium users are supposed to program practical assignments for their tutorials. Novice users are mainly engaged in learning how to interact with the computer to get a given task done. Medium users should have gained some experience in working with the computer. Expert users know how to interact with the computer and regard the computer as a means of solving their problems.

As our target is to define realistic user scripts for ERTE, which is a system to present interactive workloads on a multi-access system in such a way that it appears to be from real users [ADA1], we study these user groups separately to see, in what respect novice,medium and expert users differ. Those differences may be important if you have to define a special mix of different user groups on the system. If novice users develop a different behaviour after some time, the workload changes and has to be redefined. Therefore we should consider the possibility that at the end of a term the workload may be different because of the progress of the novice users and provide the necessary data to generate different user scripts.

Users of EMAS have the possibility of setting an option called 'Permrecall'. If they do, a circular file is generated containing the most recent interactive I/O operations (64 kbytes stored) until this option is explicitly switched off. This kind of information is collected automatically and no measurements had to be done for this.

As the information in this file was not sufficient to define realistic user scripts for ERTE, we had to put in some more monitoring to get data on the parameters such as think time, pagefaults and CPU time consumed per command and information on the sizes of the files the users access.

The fact that EMAS is written in the high level language IMP made it possible to put in this extra monitoring to do the necessary measurements. All monitoring was done at subsystem level by putting in some extra software. Another very helpful characteristic of EMAS is that all the users have their individual subsystem. So only a limited number of users were set on the modified subsystem while the majority of the user population runs on the standard version of EMAS.

As the data we obtain from this kind of monitoring can be considered strictly private users had to be asked individually to set this option, to work with the

modified subsystem and give permission to analyse their data. Obviously, many users do not like the idea of being monitored. This is a major reason why it was not possible to monitor the whole user community. Another reason is the fact that users tend to use alias commands (i.e. defining individual names for commands). The meaning of alias commands was not always obvious and the users had to be asked individually to record the commands properly. Because of this necessary interaction with the users it was advisable to keep the number of monitored users low. We monitored 100 novice, 50 medium and 20 expert users over a period of about 1 month.

3. STRUCTURING OF USERS SESSIONS

To structure users' sessions we have to apply a technique to split the users' sessions into different groups. In chapter I we explained in detail why cluster analysis techniques are controversial in the present state of the art. We therefore apply another technique to structure the sessions. But first we have to decide which parameters to select to describe user sessions.

3.1 Parameter Selection

As already mentioned above, there is no inherent feature set to describe user behaviour. Therefore the selection of the feature set may be very subjective and inappropriate. In this study we consider user behaviour

under the following aspects: First we look at the way users work at system command language level. This can be described by the commands and command sequences they use and the think times they exhibit.

However, this information alone would not be sufficient to describe user behaviour and define realistic user scripts for ERTE. We also have to look at resource consumptions, such as the CPU time used and number of pageturns per command , as ERTE experiments [SHAW] show that the number of pageturns for the same commands is fairly load independent on EMAS. Another important aspect to study is whether those parameters change under different loads and how different user groups react to different loads. To define credible user scripts this information is necessary if we are going to redefine scripts for different purposes.

We start to split users' sessions according to their percentages of different command groups. This has the advantage that we do not have any scaling problems.

3.2 The analysis algorithm

We now describe the algorithm to split these users' sessions into different groups. The first step is to characterize user behaviour at system command language level. For this purpose we classified the commands

available on EMAS into the six groups General File, Specific File, Edit, Compile, Run, and Others.

We first determined for each session how often commands of each of the six command groups were issued and thus associated a six component vector with each session. As we were not interested in the absolute values of these vectors we replaced them by their respective percentages.

Then we grouped sessions with a similar percentage of Edit commands. Each of the resulting groups was again subdivided into groups with a similar percentage of Compile commands. At the next levels we grouped according to the percentages of Run, General File, Specific File and Other commands.

At all levels grouping was done in such a way that after our last grouping we had generated 17 groups for expert, 20 for medium and 20 for novice users.

Groups with less than 16 members were considered to be 'outliers' and too 'individual' and were not considered in our subsequent analysis. In chapter I we described the problems with outliers if cluster techniques are applied to data of resource consumption rates: although the outliers account only for a small number of jobs, in our case 15%, they consume about 80% of the resources. However, we do not have a similar

problem in this study, those 10% to 20% classified to be too 'individual' do only account for 10% to 20% of the resources used. Those 10% to 20% of outliers include both resource intensive and resource light sessions.

Since we always grouped in a one-dimensional space it was easy to check whether the groupings were reasonable. Weighting of features can be achieved by the number of groups which are generated for each feature.

This is important to keep the number of groups generated as small as possible. In our example we started with edit, compile and run commands. After having generated groups for these command groups we did not allow many further splittings for file and other commands. This has been done as edit, compile and run commands consume more CPU time than file and other comands.

4. RESULTS

To find out whether there are differences between different user groups we apply the same algorithm to different user groups and study and compare the results generated by the algorithm. We start with the results for the expert users.

4.1 Expert users

Table II.1 shows the groups generated for 600 expert users sessions. For each group the number of members, the average percentage of each command group used in those sessions and the standard deviations are indicated. Besides the different use of command groups we also consider the CPU time used, the login time and the total number of commands issued per session. These numbers do not include login and logoff commands (Figures II.1-II.9).

Table II.1 shows that the expert user sessions can be split into different groups with each group having different characteristics.

Group 1 contains about a quarter of all users' sessions. These users use the computer as a communication system. They either just log on and off, without issuing further commands, to look for possible

messages or they read new messages and/or send messages to other users.

Group 2 and 3 are relatively similar to each other and to group 1. Users in group 2 perform mainly short operations on their files such as looking at their directory, renaming or destroying files or permitting access of their files to other users. Users in group 3 perform mainly specific file operations such as analysing, copying, listing a file or sending files over a network to other host computers. In addition both user groups may send messages to other users or make general enquiries about the system. CPU time and login time characteristics are again very similar.

Users in group 4-8 can be classified to be edit oriented as they use a higher proportion of Edit than of Compile and Run commands. These users are obviously developing and debugging programs and rarely call a compiler or run programs.

Users in group 9 issue about the same number of Edit, Compile and Run commands.

Users sessions in group 10 can be classified to be run oriented. These users mainly work with their debugged programs.

The rest of the sessions are too 'individual' and are not separately listed as they account only for about 10% of all sessions.

4.2 Medium users

The same algorithm as described above applied to data of 600 medium users sessions yields similar results (Table II.2):

Users in group 1 also use the system mainly as a communication system and make some enquiries about their files or list a file. They very often make more enquiries about the system than expert users in group 1 such as calling 'users' to find out how many users are logged on to the system.

Users in group 2 mainly issue list commands and make some general enquiries about their files and the system. This group is comparable to groups 2 and 3 of the expert users.

Users in group 3,4 and 5 can be considered edit oriented as they issue significantly more Edit than Compile or Run Commands. They obviously develop and debug programs. The corresponding expert user groups are the groups 4-8.

Users in group 6 issue about the same number of Edit, Compile and Run statments. The corresponding expert users are in group 9.

Users in group 7 and 8 issue more Run than Edit commands and are therefore classified to be run oriented.

Group 9 contains strictly run oriented sessions, i.e sessions without any Edit and Compile statements. This group is comparable to group 10 of the expert users.

About 20% of the medium users sessions are too individual and cannot be reasonably grouped with other sessions.

4.3 Novice users

Table II.3 shows the results for novice users sessions. For this kind of user group we find the following pattern:

In group 1 we find the communication oriented sessions. About 12% of the novice users belong to this group.

Group 2 and 3 are called file-oriented although communication oriented commands are also frequently

issued. These file oriented sessions account for only 10% of all sessions.

Groups 4 and 5 are edit oriented. This is more than 25% of all sessions.

Sessions in group 6 are edit-compile oriented.

Groups 7,8 are edit-compile-run oriented, which acounts for 20% of the sessions.

Group 9 to 12 are run oriented which accounts for 12% of all sessions.

NUMBER OF COMMANDS, CPU TIMES AND CONNECTION TIME PER SESSION

So far we have only studied the command groups user issue. The next step is to relate these sessions to the number of commands issued, CPU times and connection time per session. This is possible to some degree, but in many cases variations are high.

In communication and file oriented sessions only very small numbers of commands are issued, in communication oriented sessions the most common (70%) number of commands issued is 1, in file oriented (40%) 5.

Connection times are short for communication and file oriented sessions, on average 1 minute for communication and 4 minutes for file oriented sessions.

CPU time used for communication oriented sessions is 1 second on average, for file oriented sessions 2 seconds.

For the remaining kinds of sessions, i.e. edit,edit-compile,edit-compile-run and run oriented sessions it is more difficult to relate them to connection times, CPU time and number of commands issued per session. We found that the distributions for these parameters were similar to the overall distributions for these sessions. Thus, from the number of commands issued, the logon time and the CPU time used we cannot conclude on the kind of session. Figures II.1 - II.9 show the distributions for edit, edit-com,edit-com-run and run oriented sessions.

Table II.1. Groupings found for expert users

Group	Members	Edit Commands %	Com Commands %	Run Commands %	Gen.File Commands %	Spec.File Commands %	Other Commands %
1	154	0	0	0	3	1	97
		0	0	0	5	2	3
2	100	0	0	0	49	9	41
		0	0	0	3	12	15
3	50	0	0	0	7	47	46
		0	0	0	5	12	12
4	37	46	9	2	7	6	31
		6	12	3	10	8	15
5	58	29	1	1	12	10	48
		3	2	2	12	12	15
6	52	15	7	6	6	21	44
		4	7	8	7	12	13
7	39	15	7	6	37	17	18
		4	7	8	10	7	7
8	28	29	28	12	4	7	20
		3	20	11	5	6	6
9	20	15	24	13	8	8	29
		4	9	11	9	10	5
10	16	0	0	69	5	7	18
		0	0	19	7	10	13

Table II.1.

Table II.2. Groupings found for medium (end of first year) users

Group	Member	Edit Commands %	Com Commands %	Run Commands %	Gen.File Commands %	Spec.File Commands %	Other Commands %
1	135	0	0	0	11	9	78
		0	0	0	12	12	23
2	71	0	0	0	9	60	31
		0	0	0	12	1	17
3	36	51	13	2	5	5	23
		2	12	3	8	7	18
4	67	29	16	7	7	17	24
		3	10	8	10	12	12
5	32	40	29	17	1	5	9
		3	6	9	2	4	6
6	28	15	12	15	8	17	31
		3	8	7	7	6	1
7	40	15	12	36	6	10	19
		3	8	6	7	8	7
8	52	29	16	36	3	7	8
		3	10	8	4	6	6
9	28	0	0	69	1	4	27
		0	0	20	1	6	15

Table II.2.

Table II.3. Novice users

Group	Member	Edit Commands %	Comp. Commands %	Run Commands %	Gen.file Commands %	Spec.File Commands %	Other Command %
1	90	0	0	0	3	1	98
		0	0	0	2	2	10
2	43	2	1	1	1	38	53
		3	1	1	2	1	14
3	20	5	0	2	36	17	45
		8	0	4	17	5	15
4	98	46	16	10	5	8	16
		5	9	9	6	7	14
5	64	24	11	10	6	24	26
		5	6	8	6	12	14
6	46	28	25	6	4	22	14
		5	4	6	6	9	9
7	67	26	25	29	1	12	7
		4	4	6	2	7	5
8	45	22	13	37	3	15	12
		6	4	9	4	8	5
9	23	6	16	49	4	12	14
		5	5	12	6	7	10
10	19	2	3	42	3	17	34
		3	4	11	6	15	18
11	12	0	1	85	0	1	14
		0	2	12	0	2	11
12	23	8	18	13	7	28	26
		5	5	10	10	10	14

Table II.3

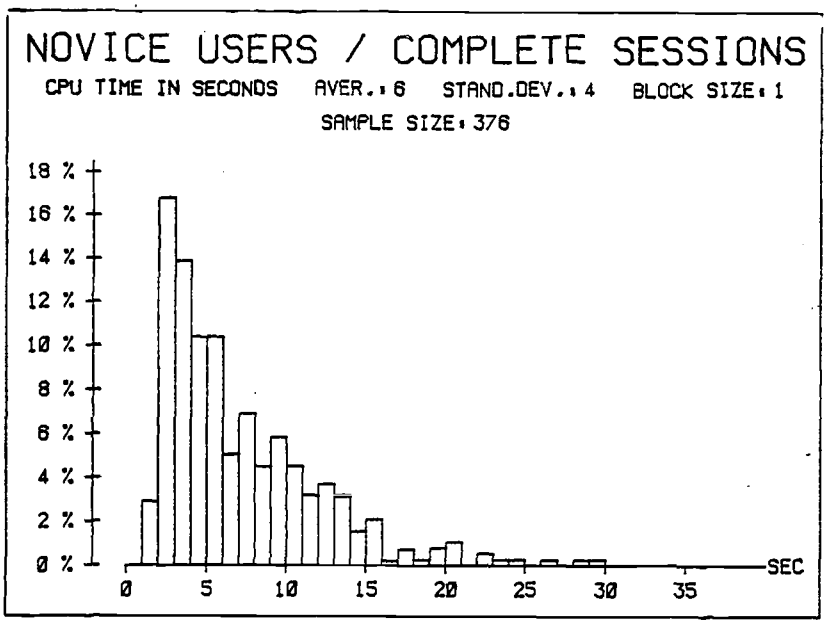

Figure II.1

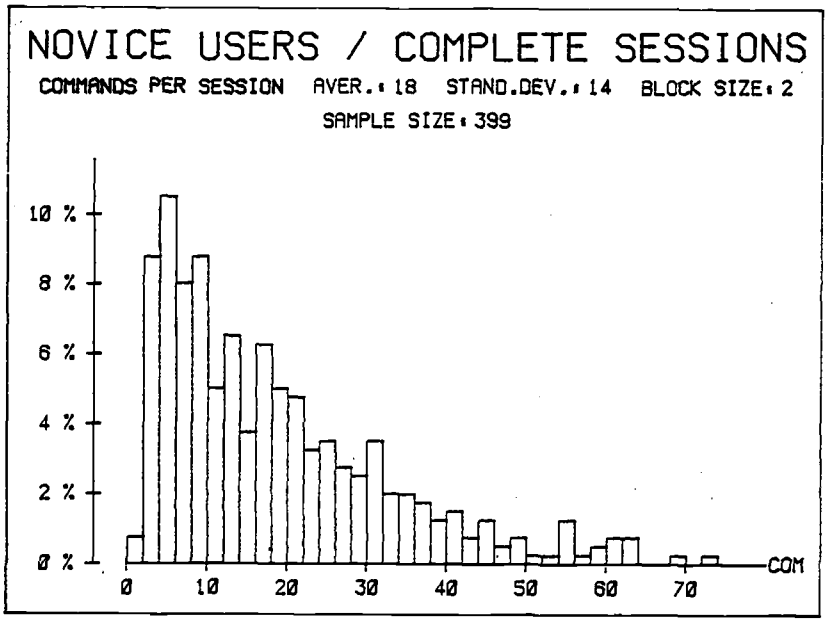

Figure II.2

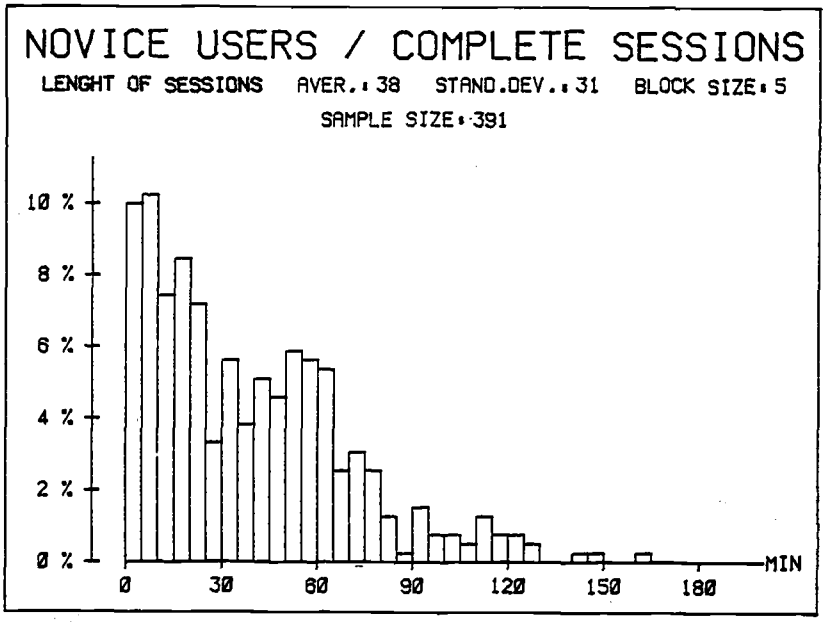

Figure II.3

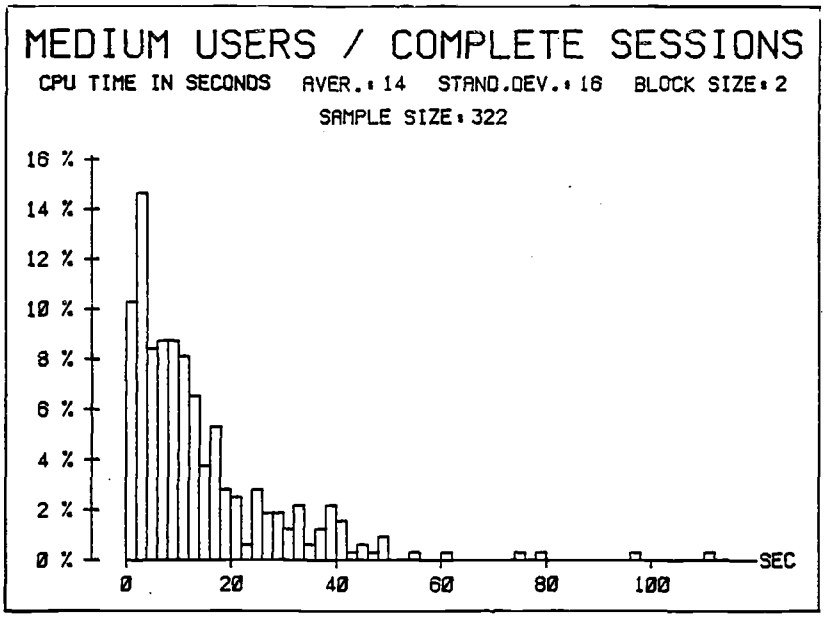

Figure II.4

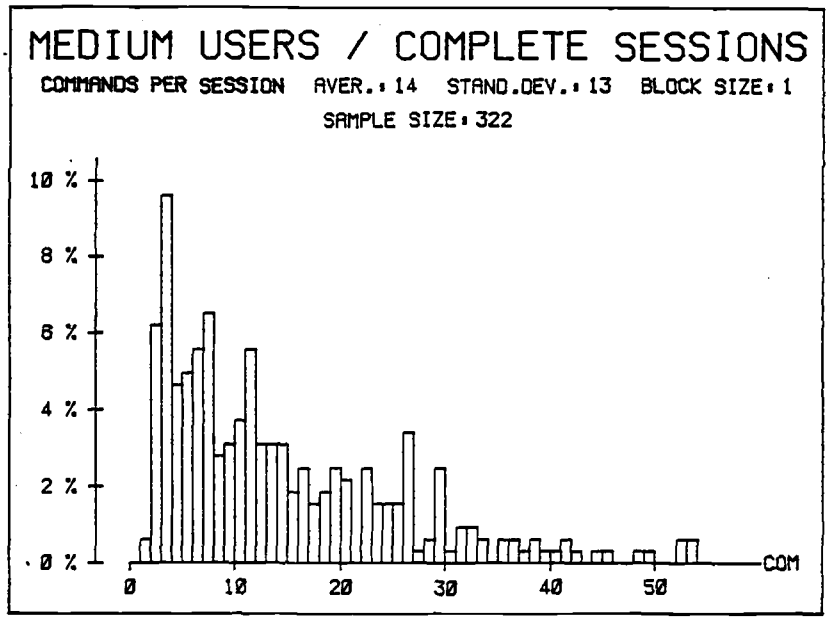

Figure II.5

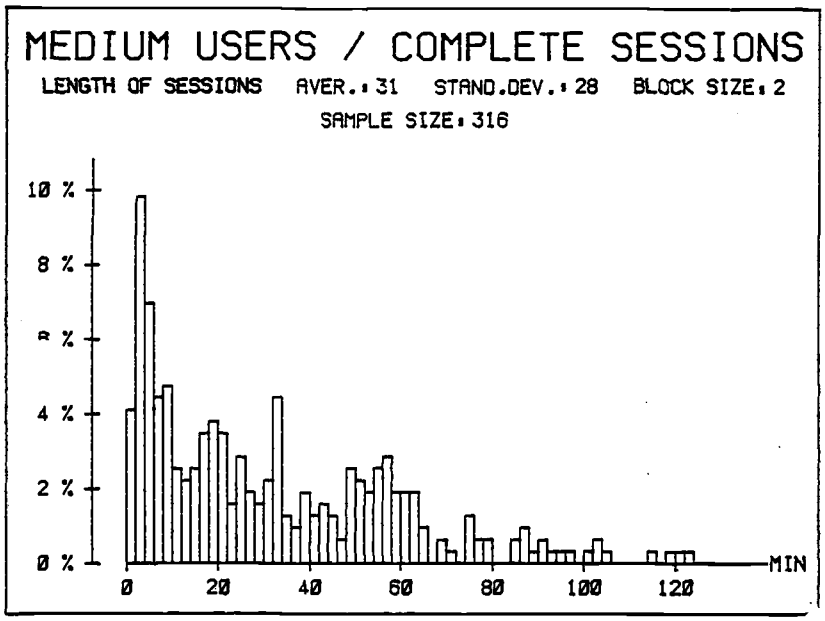

Figure II.6

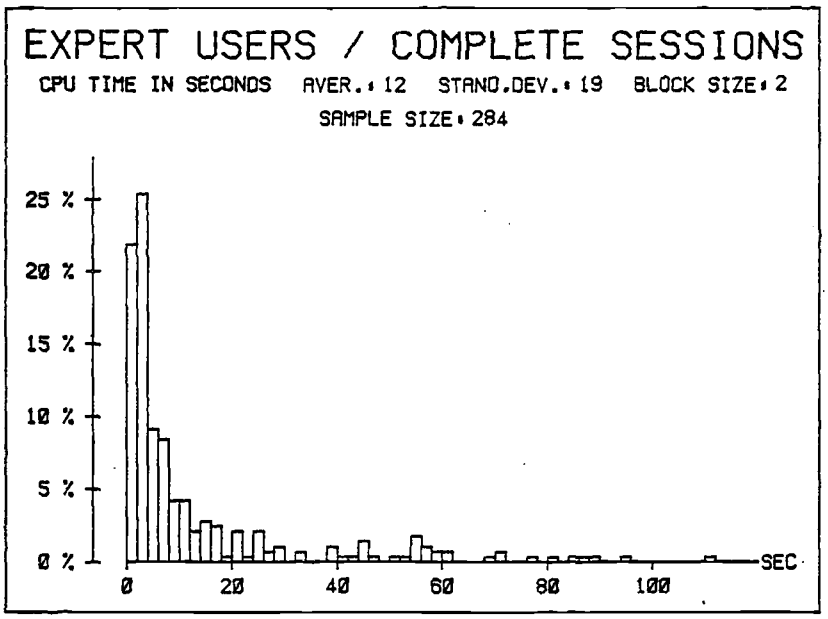

Figure II.7

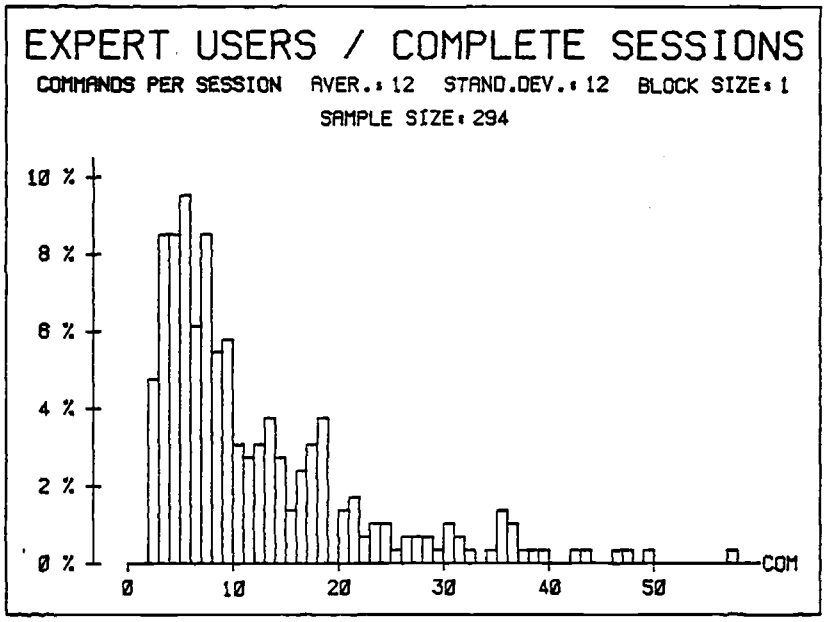

Figure II.8

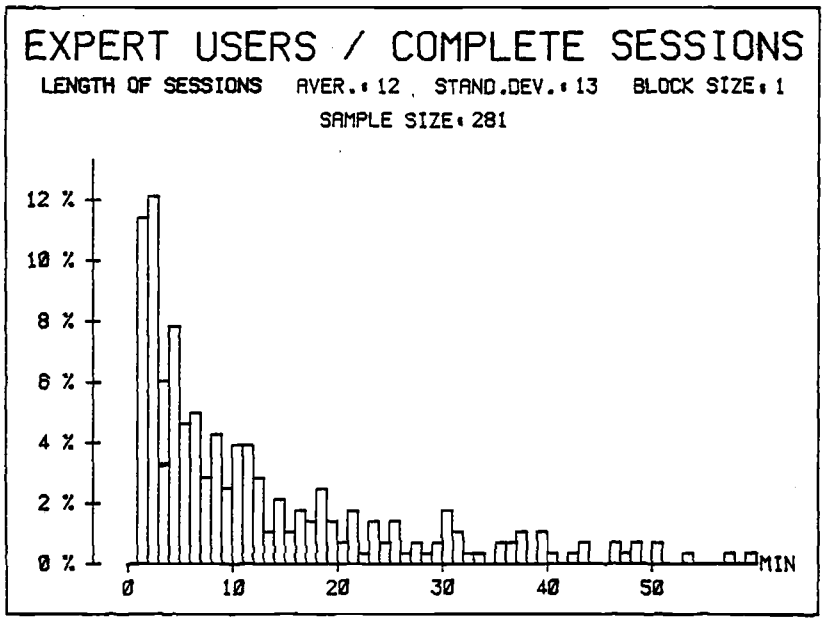

Figure II.9

5. USER BEHAVIOUR MODELS

To develop user behaviour models we first characterize several types of sessions generated by the algorithm described above and describe typical command sequences.

For all user groups we find similar types of sessions, but different user groups tend to prefer different kinds of sessions:

Communication oriented sessions. As mentioned above about a quarter of all expert users sessions belongs to this group. The proportion of medium users sessions which belong to this group is slightly lower, but only 12% of novice user sessions belong to this group.

File oriented sessions. In file oriented sessions users tidy up their directories, list, rename, copy or send files over a network to other host computers. About 25% of the expert user but only 12% of the medium users sessions and 7% of the novice users sessions can be classified to be file oriented.

Edit oriented sessions. In edit oriented sessions it can be assumed that the users mainly develop and debug programs. More than 25% of all expert and novice users sessions and about 20% of all medium users sessions are

edit oriented.

Edit Compile Run oriented sessions. In these sessions there is about an equal proportion of Edit, Compile and Run statements. The purpose of such sessions should be to remove run time errors from programs; only 5% of expert, medium and novice users sessions are of this type.

Run oriented sessions. In purely run oriented sessions users work with their programs. The proportion of such sessions (no Edit or Compile commands are issued) is rather low for both expert and novice users (below 10%). About 16% of the medium and novice users sessions can be classified to be run oriented although Edit and Compile commands are still issued.

Often workload models are just based on simple distributions and frequencies of instructions or in our case commands. We can distinguish between static and dynamic frequencies of commands etc.. Static frequencies of commands indicate the distributions of all commands together. The dynamic frequency refers to the distribution of the commands within a fixed time interval. Models based simply on distributions do not contain information about correlation and other dependencies which may be present in the workload being modelled.

If simple mixes are used it is implicitly assumed that the behaviour of a system is only affected by the instructions or statements executed and not by the command sequences. This assumption is not always valid. If we take a non-pipelined CPU with a dedicated memory and want to determine the CPU time required to process a special workload, this assuption will be valid. But having a pipelined CPU, the instruction mix does not account for the interactions between the execution of consecutive commands. However, the performance of a pipelined machine is influenced by this. The same is valid for CPU and hardware resources. This means, the real processing time, response time or throughput rate cannot be determined without realistic command sequences.

Therefore we calculated a transition probability matrix which indicates the transition probabilities from command group to command group. The widely accepted user behaviour model 'LIST-MODS-RUN' is only applicable to some user groups.

Communication oriented sessions. The most 'typical' users tend to issue commands of the following type: 'Start-Users-Tell-Stop' or 'Start-Documents-Mail-Stop' or just 'Start-Stop'.

File oriented sessions. File oriented sessions may look like 'Start-Files-Files-Copy-Destroy-Stop' or

'Start-Files-Analyse-List-Destroy-Tell-Stop'. Users of this type are in groups 2 and 3 of the expert and group 2 of the novice users.

Edit oriented sessions. Users who mainly develop and debug programs issue commands in the form:
'Start-Edit-Edit-Com-Edit-Edit-List-Com-Run-Users-Stop'.
Sessions of this type are in groups 4-8 of the expert and 3-5 of the novice users.

Edit Compile Run oriented sessions. Users who issue an equal share of Edit Com Run statements are in group 7 of the novice users. A session looks like:
'Edit-Com-Run-Edit- List-Com-Run-Users-Stop'.

Run oriented sessions. A more run oriented user is described by the model:
'Start-Run-Run-Edit- Com-Run-Run-Tell-Stop'.
Such users are in group 7 and 8 of the novice users.

6. THINKTIMES

Thinktimes are a further important characteristic of user behaviour. The times presented now are only approximate values. Thinktime starts as soon as the system finishes with the previous command. Therefore the thinktime we record may include time to print some output from the previous command plus the actual thinktime plus typing time. All our measurement was done at subsystem level and it was not possible to get all information of interest. Thus we could not get information on type ahead. Typing ahead means users type in the next command before the system prompts for it. If there is no "delay" between system prompt and the time when the system accepts the command, we can conclude that the users typed ahead.

NOVICE USERS	MEDIUM USERS	EXPERT USERS
34 secs	22 secs	15 secs
36 secs	34 secs	14 secs

Table II.4. Average thinktimes

The following three figures show thinktimes for novice, medium and expert users under different load and the standard deviation.

NOVICE USERS

No of users	10	41	73	103
	5	36	22	19
Average	26	32	33	35
Thinktime	23	29	29	30

Table II.5. Thinktimes for Novice users

MEDIUM USERS

No of users	30	59	77
	8	13	6
Average	24	21	19
thinktime	23	20	17

Table II.6. Thinktimes for Medium users

EXPERT USERS

No of users	16	37	76
	5	13	23
Average	15	15	9
thinktime	14	14	8

Table II.7. Thinktimes for Expert users

The average thinktime for novice users is more than three times as high as for expert users. Medium users lie in between novice and expert users. One reason for such high differences in thinktimes may be the fact that novice users do not type well-known commands automatically and have to think before typing every

command. Medium users have already developed some routine which shortens their thinktimes. Figures II.10-II.12 give average values and distributions for average thinktimes behaviour under all loads, but 'outlier' thinktimes, i.e. thinktimes of more than 200 secs are not included.

Novice users do not tend to type ahead, 15% of the commands issued by medium users are typed ahead (0 secs thinktime), and 27% of the commands typed in by expert users are typed ahead.

However, these differences may be caused by the different kinds of sessions we find. For example, expert users do have quite a lot of communication and file oriented sessions. Short thinktimes may be found predominantly for those kinds of commands. Similar thinktime behaviour may be found for edit, compile or run commands. The next table shows the average thinktimes for edit commands with standard deviations for these three user groups.

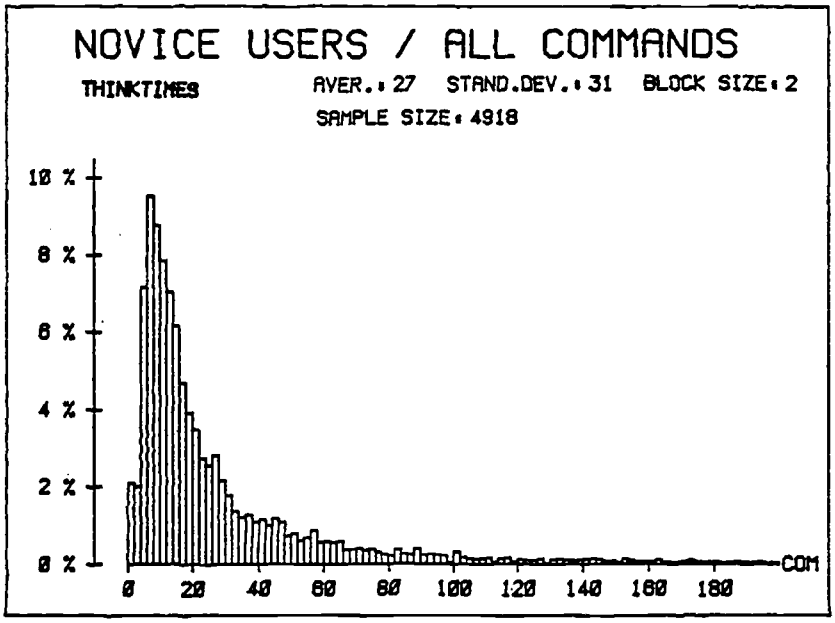

Figure II.10

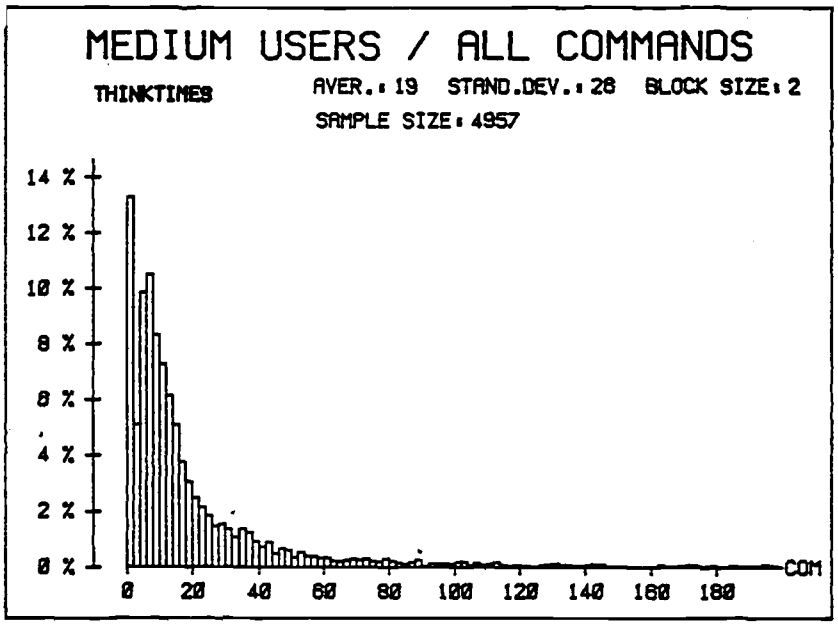

Figure II.11

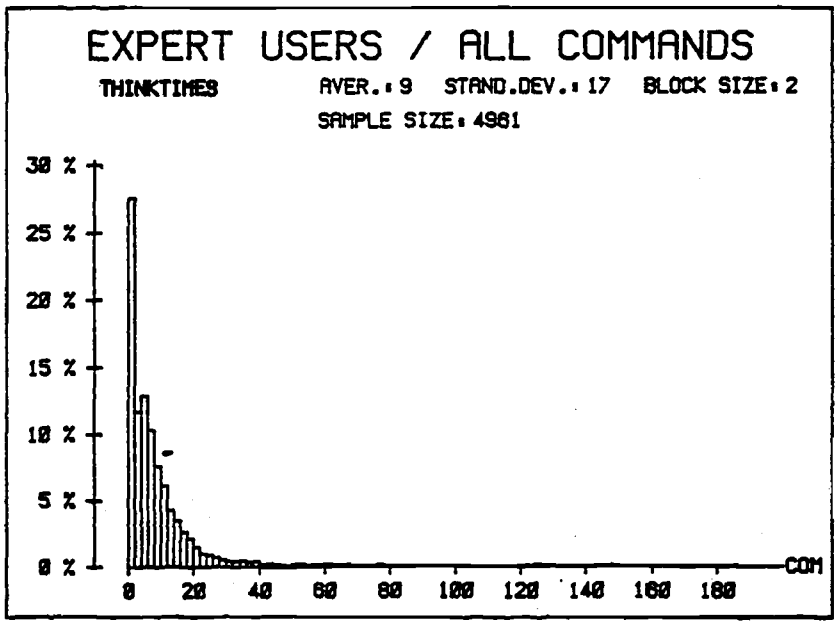

Figure II.12

Command	NOVICE USERS	MEDIUM USERS	EXPERT USERS
Edit	42 secs	27 secs	24 secs
	39 secs	29 secs	29 secs
Compile	18 secs	10 secs	5 secs
	13 secs	10 secs	5 secs
Run	33 secs	22 secs	12 secs
	26 secs	17 secs	9 secs
File	33 secs	23 secs	5 secs
	25 secs	19 secs	18 secs
Others	23 secs	17 secs	4 secs
	21 secs	16 secs	7 secs

Table II.8

Table II.8. Average thinktimes for different command groups

This table shows that thinktimes for novice and medium users are higher than for expert users for all kinds of commands. One common pattern for all groups is that they need the longest thinktimes to issue an edit command. Compile commands are entered after the shortest thinktimes, as well as for expert users who have similarly short thinktimes to enter 'other' groups ,such as tell, mail, view, help. For issuing a run command the three user groups need thinktimes which correspond to their overall average think time. For file oriented sessions the differences in thinktimes differ the most. Whereas for expert users the thinktimes are below their general average values, we find for novice and medium users average values to type in a file command. Thinktimes for novice users are 7 times as high and for medium users about 5 times as high as for expert users. For the other commands we only

find a factor of 3 or 4 for run or compile commands or just the factor 2 for edit commands.

Thus, differences in thinktimes vary for different command groups. The differences are the highest for file and 'other' commands, commands which are more often used the more experienced the users become.

7. BEHAVIOURAL DIFFERENCES UNDER DIFFERENT LOADS

The user behaviour characteristics described so far, characterize the sample we took from the user population as a whole. We structured user sessions without looking at time intervals or the number of users who were actually working on the system. However, the performance of a specific system depends on the session mixes, the think times and the resource consumption caused by the various commands the users issue. And these parameters may well be different depending on the number of users using the system. To define accurate workload models the mix of the users sessions at different time intervals has to be determined. This can be done by looking at what kind of user sessions we find if we study users who were logged on at specific time intervals.

A workload imposing a heavy load on the system is obviously the most interesting one. Heavy workloads can be expected before and after lunchtime and in the early

afternoon. During this time we find around 100 users logged on to the system. In the morning and late afternoon a moderate workload is usually found, 40 to 70 users are working on EMAS, whereas at night or very early in the morning systems tend to have even smaller numbers of users.

On EMAS the number of users give a rough indication of the system's overall performance, because people are automatically logged off if they have not shown any activity within the last 12 minutes. How does a different performance influence user behaviour, if at all?

The question whether and to what degree user behaviour is influenced by a system's performance is difficult to answer. It can be argued that during periods of heavy loads the users tend to do a different kind of work than during periods when the system is underloaded.

For instance, users who have CPU-time intensive work to do, may prefer doing their work in the evening or at night in order to get faster response times and get their work done in a reasonable period of time. Work which does not consume much CPU time can well be done at times of heavy loads. The users may well get satisfactory response times.

On the other hand, most staff and students are leaving in the late afternoon and by then their work has to be done. Such users may not care how long the response times are and keep on going. In particular students are often under pressure because they have to hand in their programs at fixed times and just cannot wait until the heavy load is off.

7.1 Commands used under different loads

But do different loads change the users' behaviour at the system command language level? To answer this question we compare which commands users predominatly prefer under different loads. To see whether there are certain tendencies for or against some commands, we compare the percentages of commands used under different loads. The following tables give the average values under different loads for novice, medium and expert users.

NOVICE USERS

No of users	17	48	72	99
	7	6	6	6
Edit	25	22	21	22
	18	16	17	16
Compile	14	12	12	13
	10	10	10	10
Run	18	16	15	15
	17	16	16	13
Gen file	4	7	4	5
	7	9	7	8
Spec File	15	14	16	16
	13	13	14	14
Others	23	29	33	30
	21	25	28	25

Table II.9.

MEDIUM USERS

No of users	22	47	67	87
	7	5	5	7
Edit	19	18	18	26
	7	18	17	21
Compile	8	10	8	8
	9	10	10	10
Run	15	16	15	14
	16	17	15	17
Gen file	8	7	5	8
	7	10	8	10
Spec File	16	16	16	17
	15	16	16	15
Others	34	30	35	29
	29	23	27	26

Table II.10.

EXPERT USERS

No of users	15	32	50	82
	5	4	5	8
Edit	10	14	11	15
	12	14	12	13
Compile	5	5	6	8
	8	8	8	12
Run	7	6	4	3
	11	9	6	4
Gen file	14	15	18	15
	16	16	21	21
Spec File	17	17	13	15
	12	13	10	15
Others	44	40	44	44
	30	27	30	25

Table II.11.

These figures indicate that users do not change significantly their behaviour at the system command language level. The kinds of commands they use remain relatively constant, only some slight trends can be detected.

NOVICE USERS. Novice users tend to drop slightly their percentages of edit, compile and run commands. The percentages of file commands stays constant, but they increase their percentages of other commands.

MEDIUM USERS. Medium users tend to increase the percentages of edit commands from 19% to 26%, compile, run and file commands stay constant, the percentage of 'other' commands drops.

EXPERT USERS. Expert users behave similarly to medium users concerning edit: they also increase the percentages of edit sessions. But expert users do

decrease their percentages of run commands and increase slightly the percentages of file commands.

The changes are not dramatic, but they indicate that different user groups have different tendencies concerning the use of the system command langugage under different loads. Medium users increase their percentages of edit commands. Editors are usually quite quick and even under heavy loads you may get a reasonable amount of work done. If you do editing you tend to call compilers, to check whether your corrections are successful, so the percentage of compile commands increases slightly. 'other' commands are less frequently used. This may be due to the fact that medium users do not waste their time with issuing commands which are not useful for their work.

Expert users also increase their percentages of edit and compile commands, but in contrast to medium users they decrease their run commands. As we have already mentioned, editors tend to remain responsive under heavy load and expert users, too, make use of this. Run commands tend to be left for less busy periods.

7.2. Resource Consumptions and Execution Times

However, the commands and the command sequences alone do not describe a load in a sufficiently complete way. We also should look at the resource consumption used by the commands, for example the CPU time used and the number of pagefaults issued. Let us first study the average values needed for edit, compile and run commands and then what the average values are under different loads. Again we study these values for the different user groups separately. We also study how execution times differ under different loads.

	NOVICE USERS	MEDIUM USERS	EXPERT USERS
Average CPU	380 msecs	771 msecs	681 msecs
	6862	1635	3346
Average PT	748	494	504
	1235	923	1116
Average Thinktime	32 secs	22 secs	15 secs
	48	40	41
Average Executiontime	26	64	26
	42	107	59

Table II.12.

From this table we see that the average CPU time used per command is very much the same for medium and expert users. Novice users tend to be lower because they submit a lot of small jobs. For the number of pageturns we find that the novice users use slightly more pageturns than the expert and medium users, but this can be explained by the fact that novice users used a screen editor which did produce a lot of pageturns.

Average think times drop gradually the more experienced the users become. For the average execution times we find that novice and expert users have the same averages, the medium users however have significantly longer execution times.

These values suggest that on average there is not much difference concerning CPU time used and pageturns between novice, expert and medium users. Concerning the think times we find that expert users are considerably quicker than beginners. However, those values are averaged values from a sample of the user population. This means, the values presented above are from observed sessions with both few and very many users logged on to the system. If the system is heavily loaded, execution times tend to increase, whereas during times when the system is lightly loaded execution times tend to be shorter. There may be more expert user sessions studied when the system was lightly loaded and thus the data presented above needs more detailed examination. Therefore we split those sessions and study the average CPU time, the average number of pageturns and execution time for those three user groups under different loads.

Under heavy load users may submit smaller jobs in order to keep execution times reasonably short. But it is also possible that the majority of the users submit the same kind of jobs and differences in average values are caused by outliers. We have already seen that think

time behaviour changes under different load, novice users tend to increase their thinktime, but we find shorter think times for medium and expert users under heavy load.

RESOURCE CONSUMPTIONS AND EXECUTION TIMES UNDER DIFFERENT LOADS FOR DIFFERENT COMMAND GROUPS

As average values do not provide enough information to decide whether there are differences between these three user groups and where these differences are, we study average values and distributions for CPU times, pageturns and execution times for the different command groups separately.(Figures II.13-99).

Those figures suggest that users do not change significantly their behaviour under different load. For edit sessions, compile and run commands we find similar distributions for CPU-time and pageturns under any load. This means that the majority of the users submit the same kind of jobs no matter what response they get. At least for novice and medium users this is true.

For expert users we find a slightly different pattern. Expert users tend to avoid busy periods. This is indicated by the small sample sizes we have for expert users under heavy load. Under light load they tend to submit CPU time and pageturn intensive run jobs,

but we hardly find those jobs under heavy load. Expert users do make more use of detaching jobs than medium and novice users and thus avoid long waiting times. However, although we have only a small sample size for compile commands, we do find some quite resource intensive compile jobs under heavy load.

A conclusion we can draw from these histograms is that the majority of the jobs submitted under different loads are very much the same and thus novice and medium users do not adapt significantly to long execution times. Differences can be found for big outlier jobs which are more frequently submitted under heavy load. The percentage of big outliers is in general very small and accounts only for 1% to 5%. As the users we monitored did not change we can conclude that these users have special 'preferences' when to logon to a system. Novice users may still be too inexperienced to avoid long execution times and thus logon at any time. For medium and even more so expert users we observe that they do not logon at any time frequently. But if they logon they apparently do the same work independent of the execution times.

NOVICE USERS. One common feature for all command groups for novice users is that they obviously increase their think times under heavy load. The majority of jobs submitted are not resource intensive, and these

kinds of jobs are very much the same under any load. We find only a few 'outlier' jobs for novice users, which are submitted under any load. Execution times increase for all command groups. For execution times we find the most obvious changes in distributions.

MEDIUM USERS. Medium users behave quite differently with respect to their think times. Whereas novice users tend to increase their think times for all kinds of commands under heavy load, medium users shorten their think times. Concerning CPU times and pageturns for compile/run commands we have the same as for medium users: peaks and distributions do not change significantly. We do find more 'outlier' jobs for medium users than for novice users, which are more often submitted under lighter load.

EXPERT USERS. For expert users it is more difficult to draw conclusions because the sample size we have for heavy loads is very small. But as already pointed out, this in itself is an interesting observation. Those users we monitored obviously run programs when the system is lightly loaded. Another reason why we have so few run commands is the fact that expert users more often make use of the detach command, i.e. they submit their jobs to be run in background mode. But even from the sample size we see that heavy compile jobs are also run when the system is heavily loaded. This means that also expert users do not change their work regarding the

jobs they submit for compilation. But as pointed out above there is a change in the percentage of edit sessions expert users have: as editors remain responsive expert users show a slightly higher preference to editing.

8. SYSTEM INDEPENDENCY

It is difficult to draw many conclusions on system dependencies or independencies of these results, because there are no comparable studies to be found in the literature. However, there are indications that these results may well be transferred to other machines. Recent studies on the departmental VAX 11/780 showed that the average thinktime for VAX users is 12 seconds, with the most common found to be 2 [MCGI]. As we find predominantly expert users on VAX this may well match with the thinktimes behaviour found on EMAS for Expert users. Another study on IBM/168 under SVS [HONA] showed an average thinktime of 28 secs with the most common found to be 6. No information was given on the users mix which is apparently very important to model interactive workloads, but the study was done at a univerity's computer centre and it can be assumed that a lot of students were using the computer. If this is the case, the results do match with a mix of novice, medium and expert users on EMAS. Admittedly these are speculations, but as there are no thorough studies available, this is information worth mentioning.

We found many very short sessions on EMAS, in particular for expert users. In chapter I we applied a clustering algorithm to the data of the accounting file of the departmental VAX 11/780. Both algorithms applied did generate big clusters with login times of less than a minute (these were not login failures!). This too matches with our observations on EMAS for expert users.

SUMMARY

To model interactive workloads for time-sharing computer systems we first structured users' sessions for three users groups separately to study and describe their way how they use the system-command-language.

For expert users we find that a quarter of all sessions are communication oriented, a third are file oriented and the remaining sessions are edit, edit-compile, edit-compile-run and run oriented.

For medium users we have less communication and file oriented session, about a fifth and a sixth each. The remaining sessions are also edit, edit-compile oriented etc..

For novice users we even find fewer communication oriented session, less than a sixth of all sessions and less than a tenth are file oriented. Novice users are

very much engaged in programme development which is reflected by the high number of edit, edit-compile etc.. sessions.

Although CPU time usage and the number of pageturns used is very much the same for all the three users groups on average we do find differences for compile and run commands. Novice users tend to submit very small jobs, i.e. jobs which use little CPU time and produce relatively few pageturns. Jobs medium users submit do consume more CPU time and produce more pageturns. Expert users submit a lot of small jobs but tend also to submit big 'outlier' jobs. Concerning adaptation to different loads and execution times we have to say that all users groups tend to submit the same kind of jobs under any load! However, more outlier jobs can be found under a light load. Adaptation to long execution times is indicated by the fact that the more experienced the users become the more they avoid busy periods.

For modelling interactive workload these results mean that we do not have to simulate different behaviours under different loads, besides the fact that think times tend to be shorter under heavier load and outliers are more frequently found under light load.

In the next chapter we are going to describe an automatic workload generator which generates user scripts ready as input for ERTE. The input data to

generate these scripts are based on the real observations we have described in this chapter.

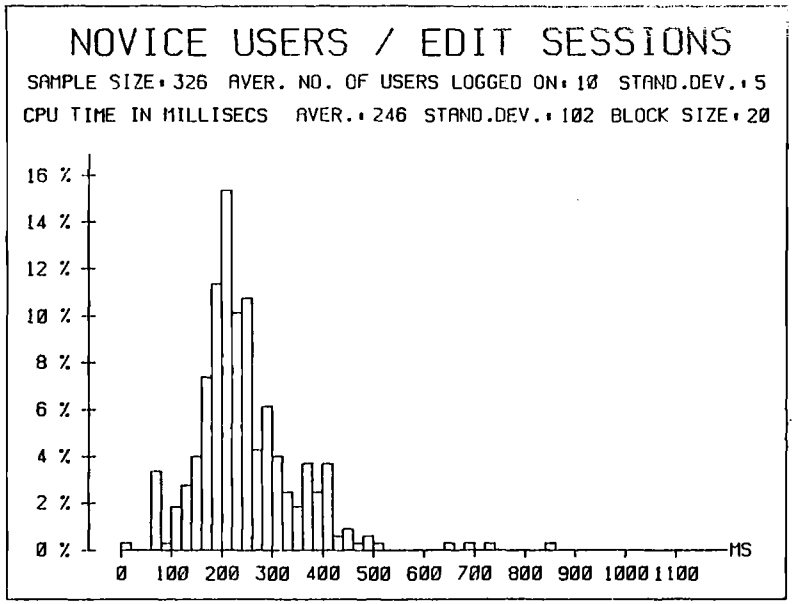

Figure II.13

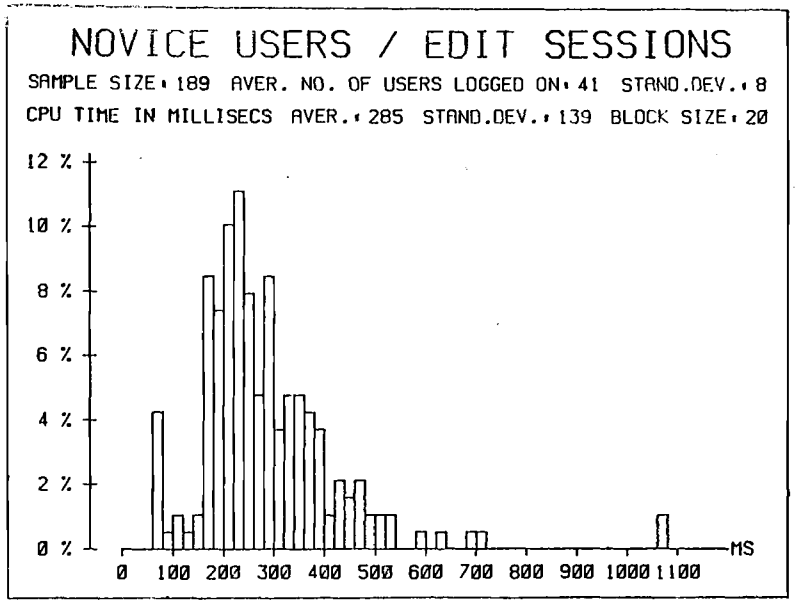

Figure II.14

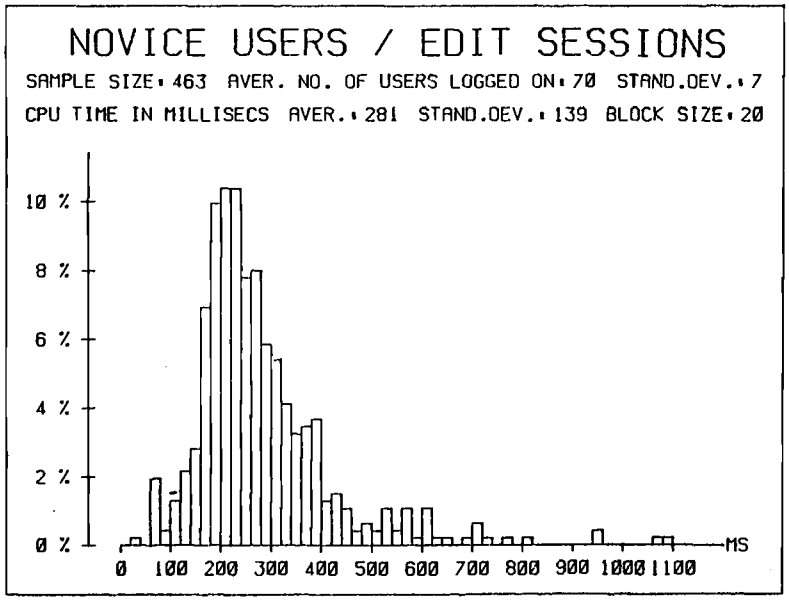

Figure II.15

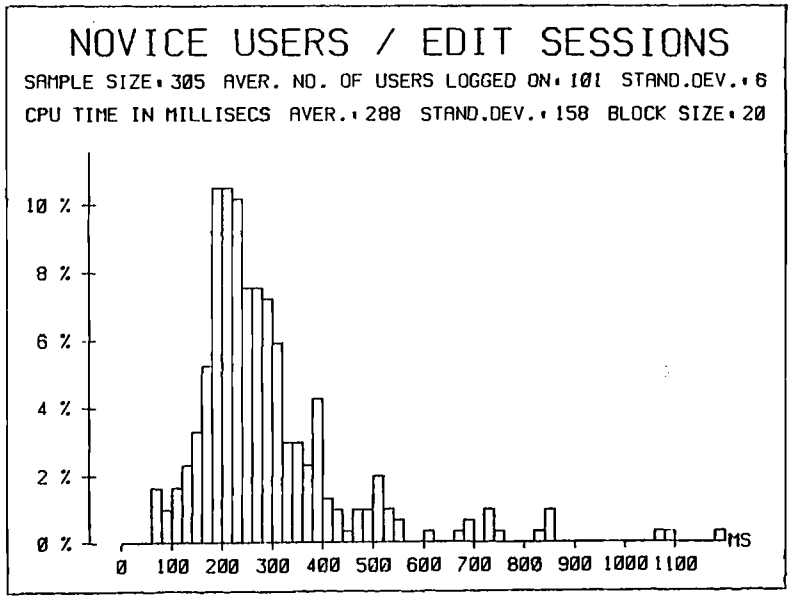

Figure II.16

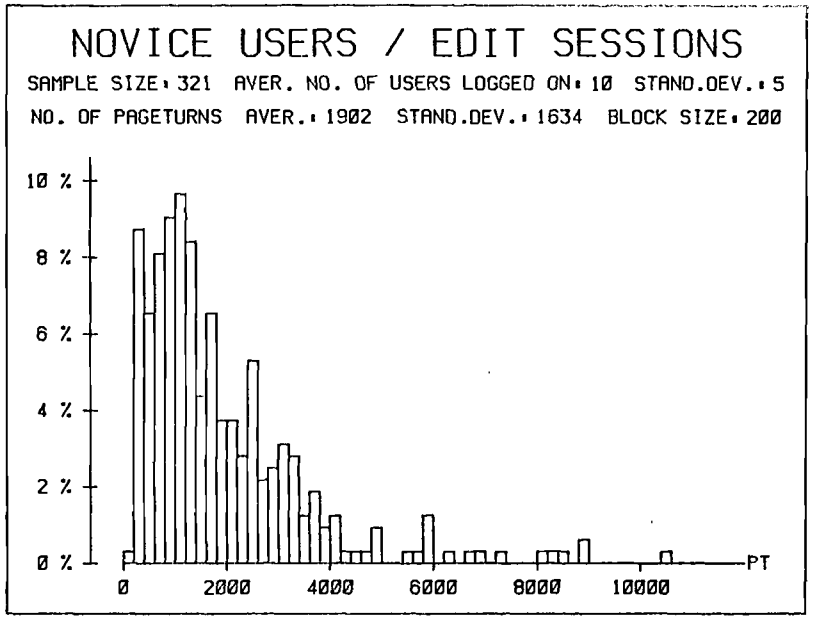

Figure II.17

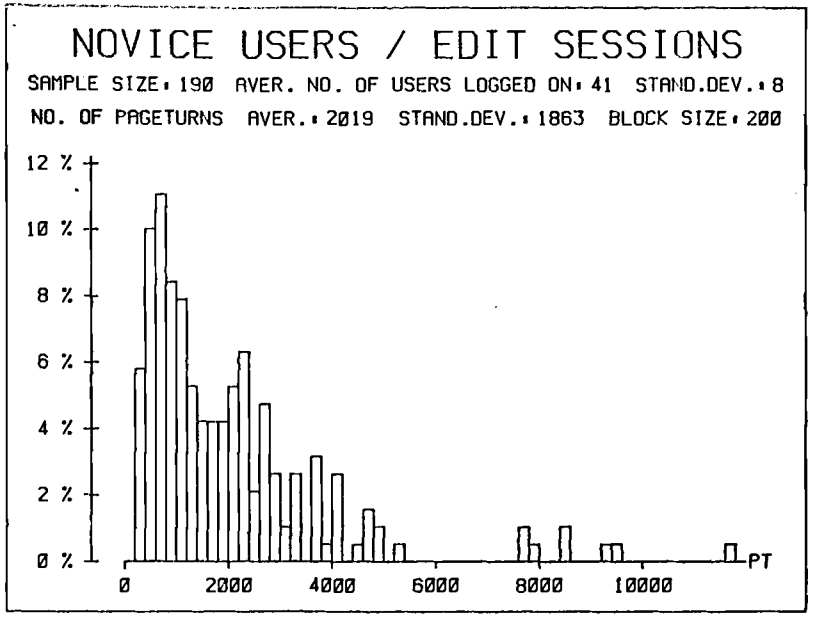

Figure II.18

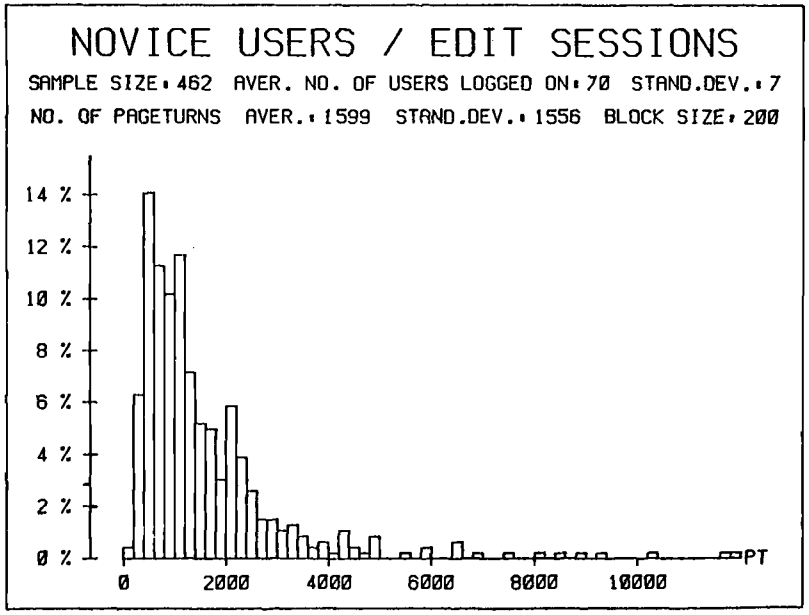

Figure II.19

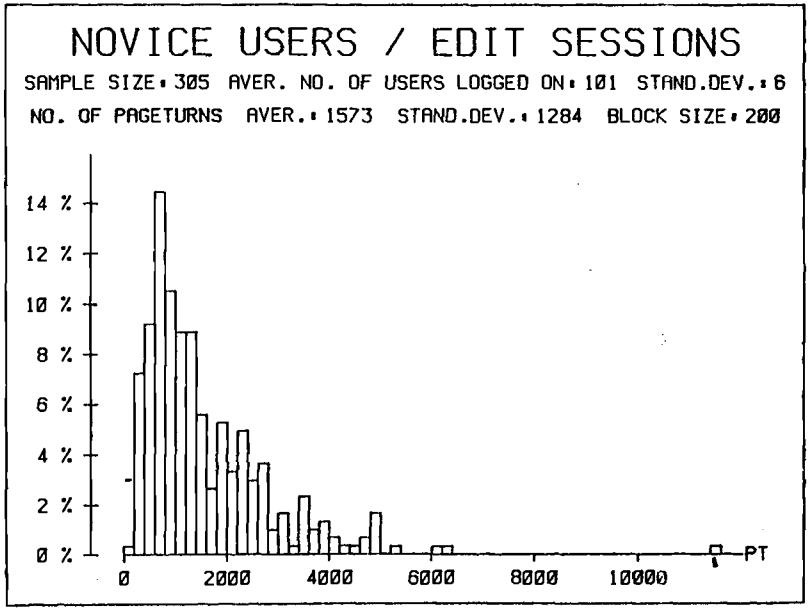

Figure II.20

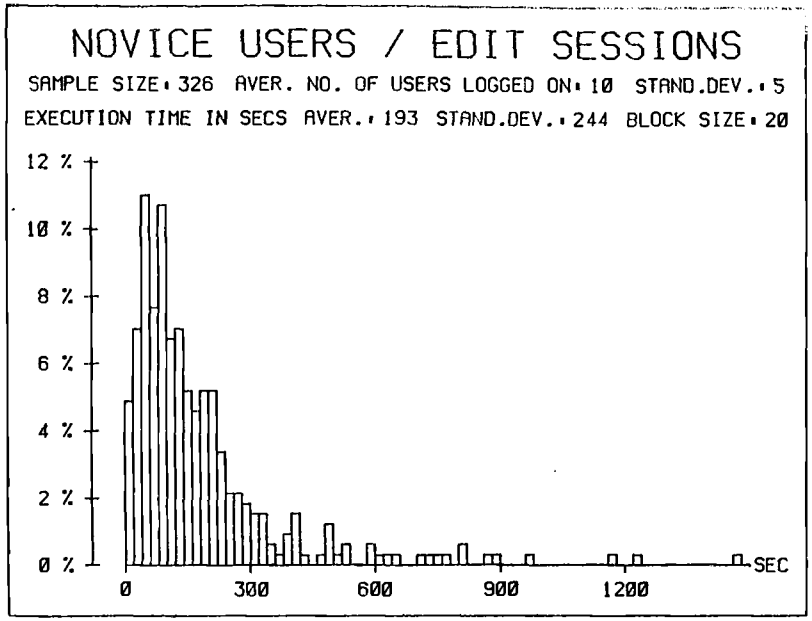

Figure II.21

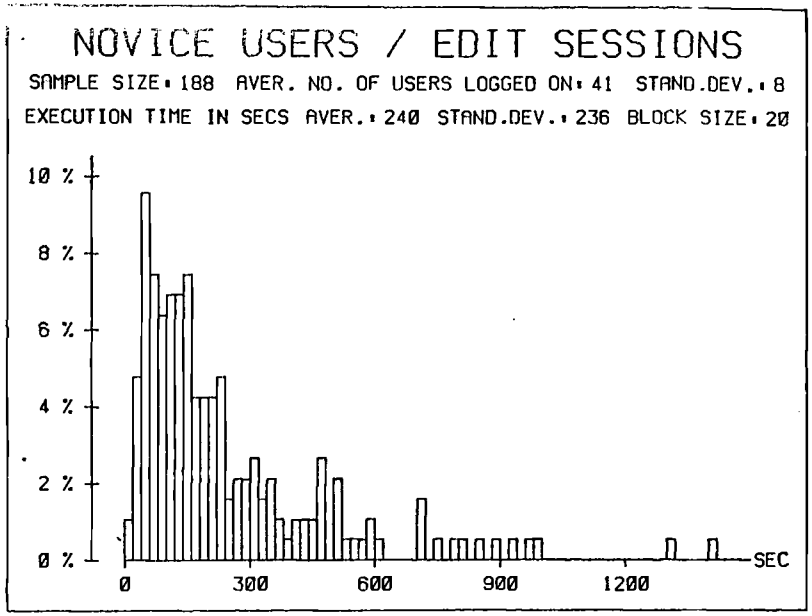

Figure II.22

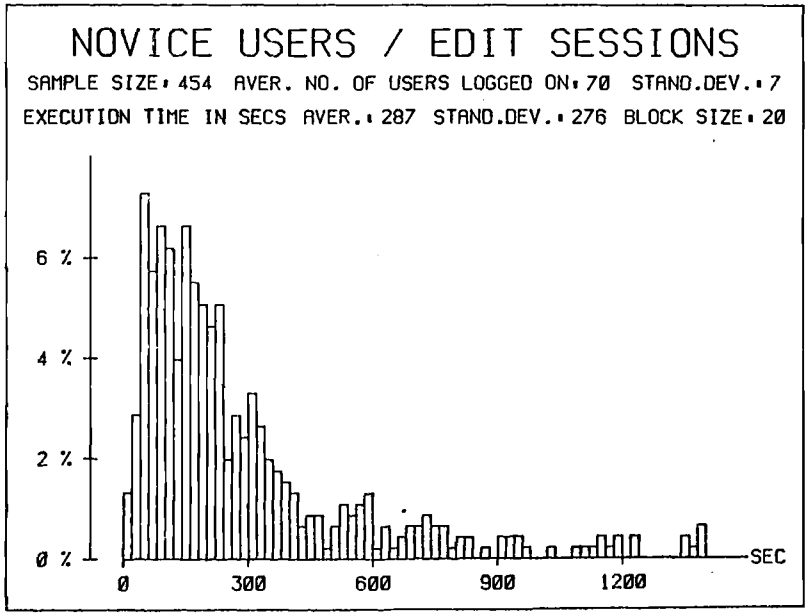

Figure II.23

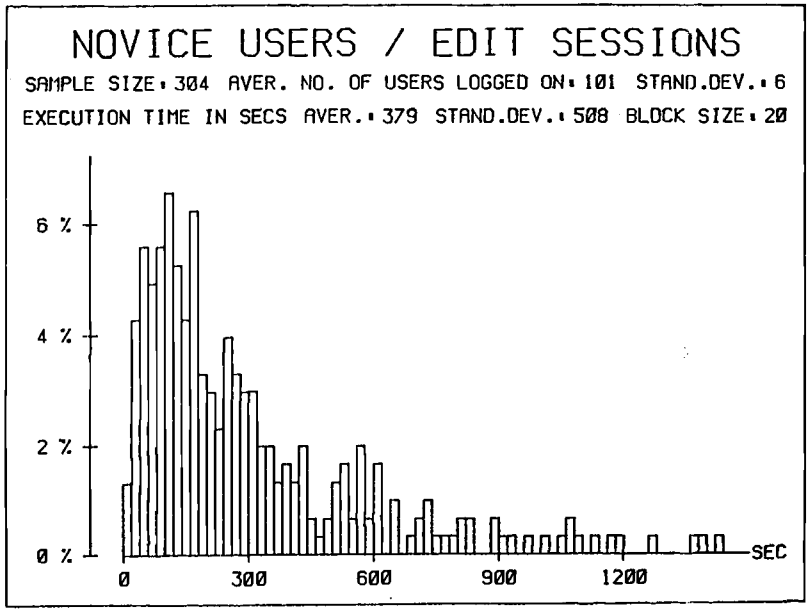

Figure II.24

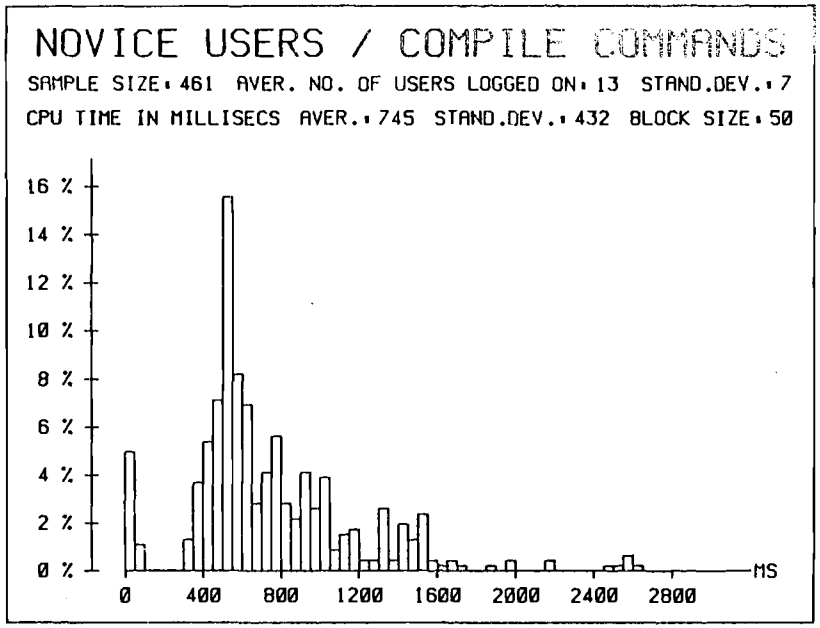

Figure II.25

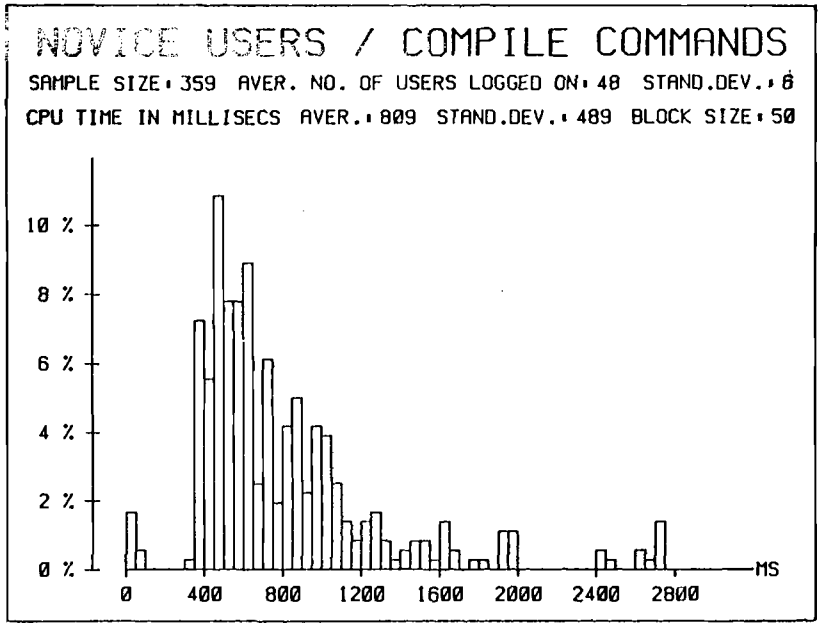

Figure II.26

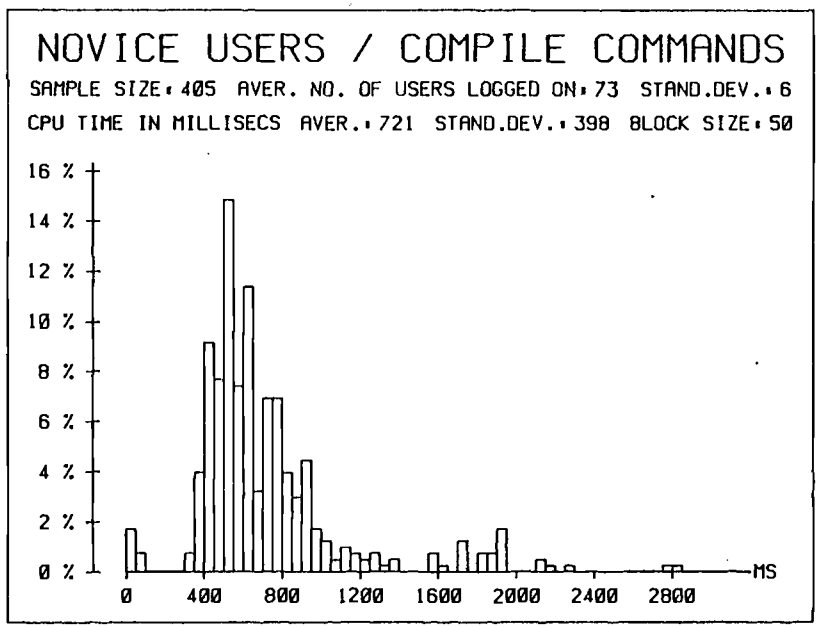

Figure II.27

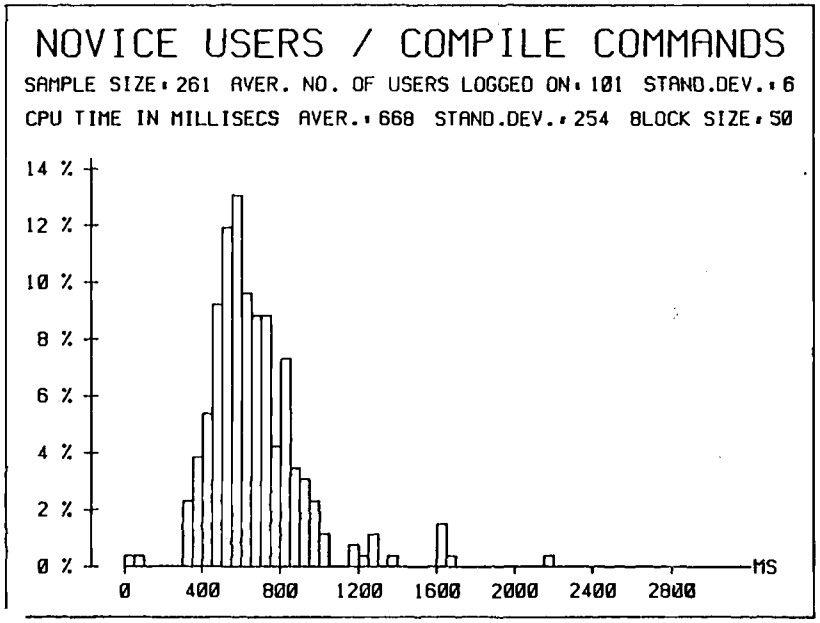

Figure II.28

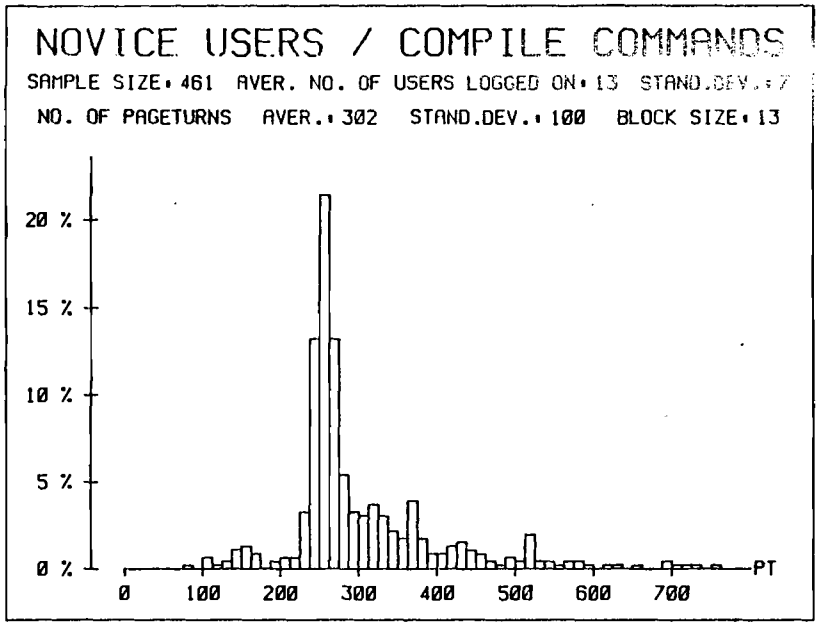

Figure II.29

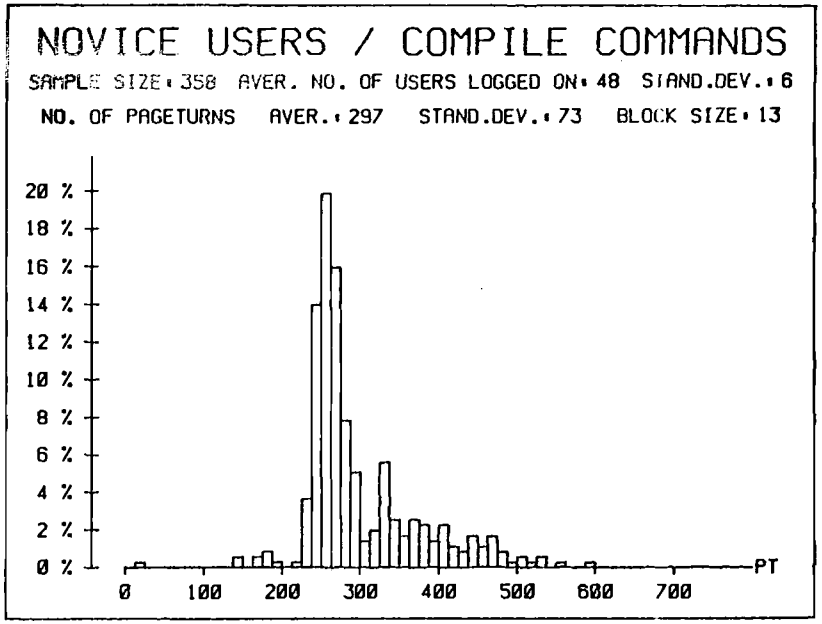

Figure II.30

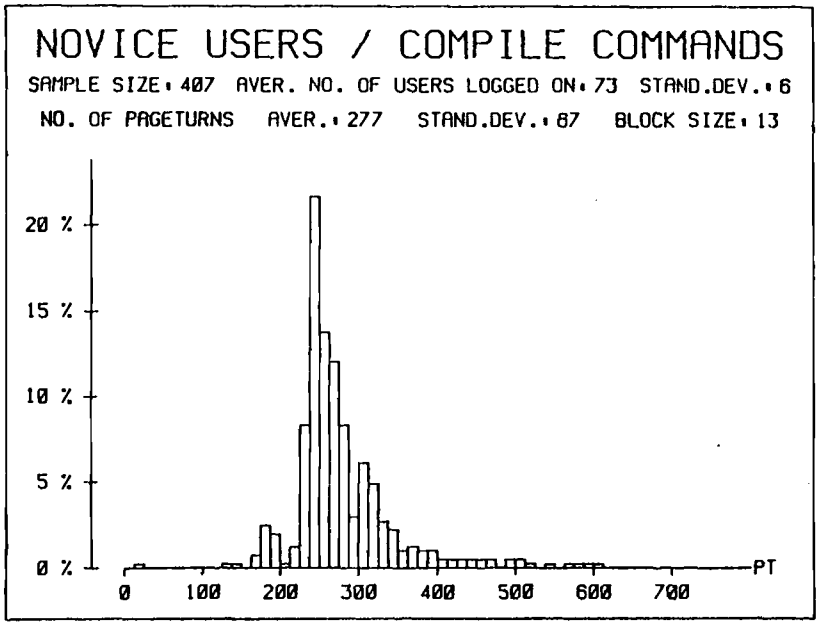

Figure II.31

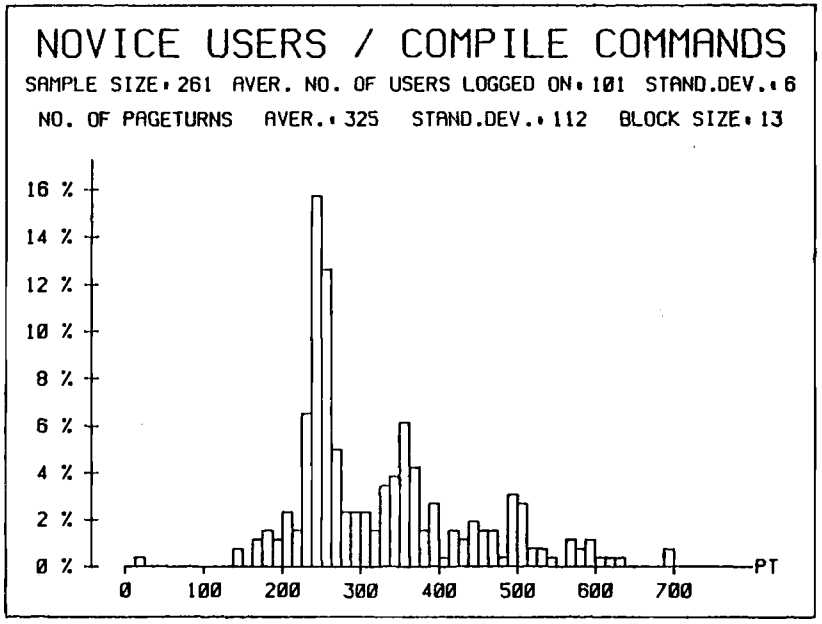

Figure II.32

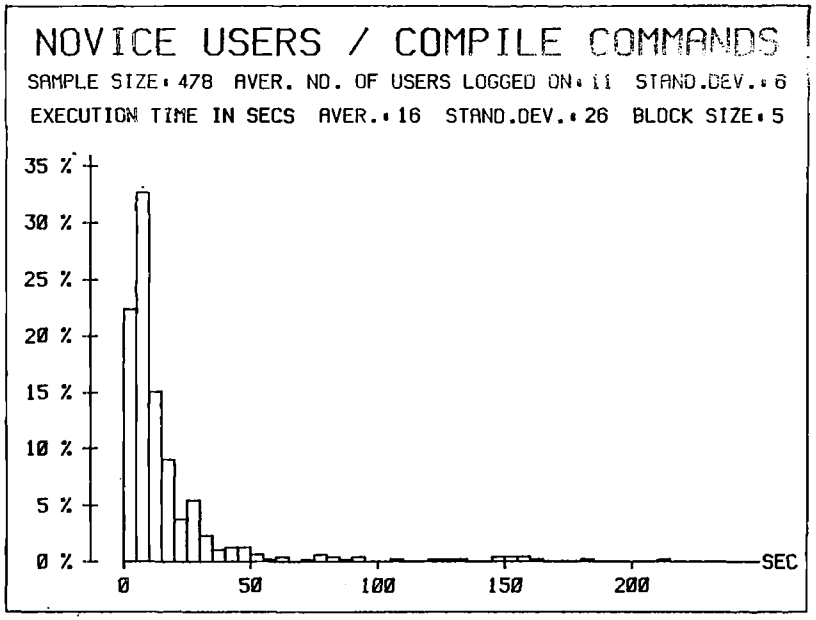

Figure II.33

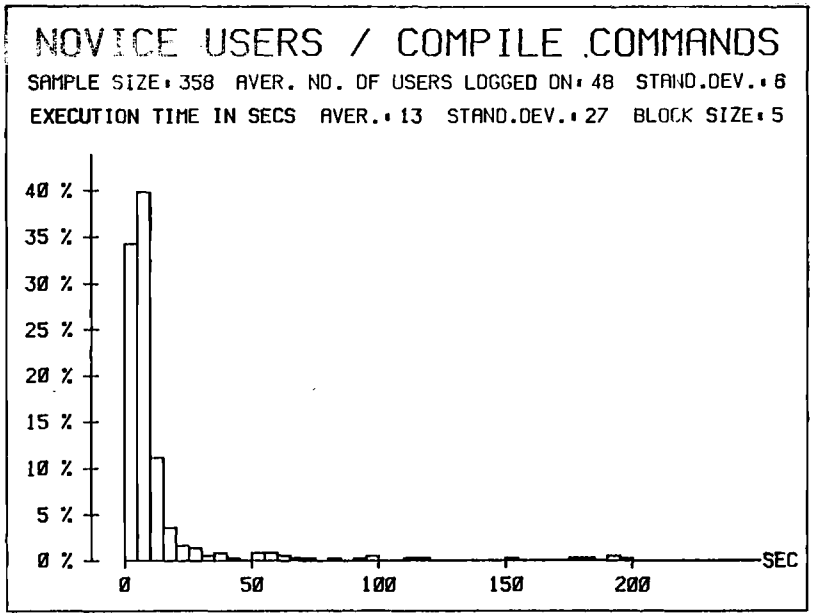

Figure II.34

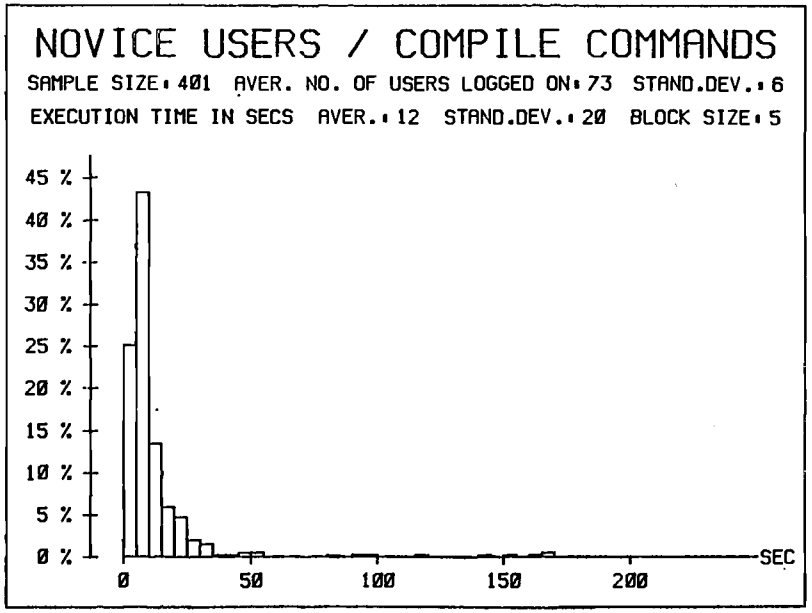

Figure II.35

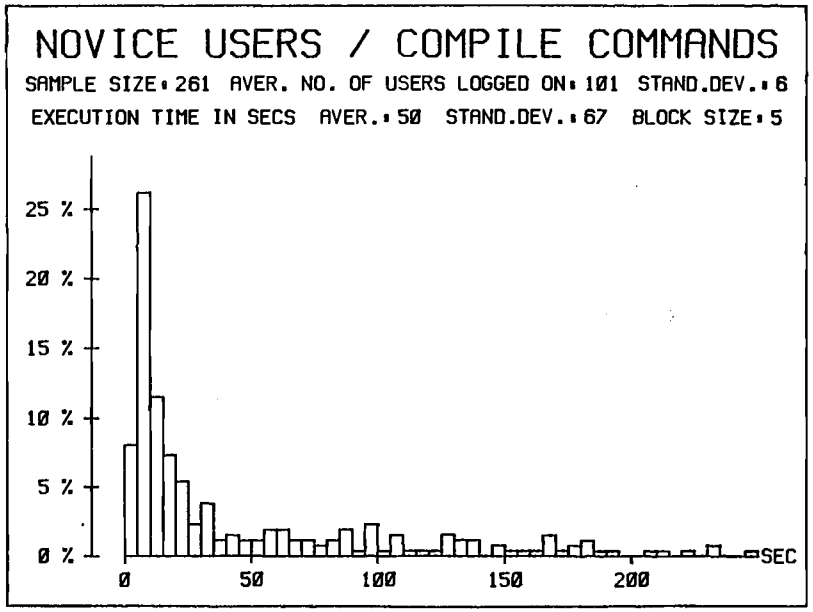

Figure II.36

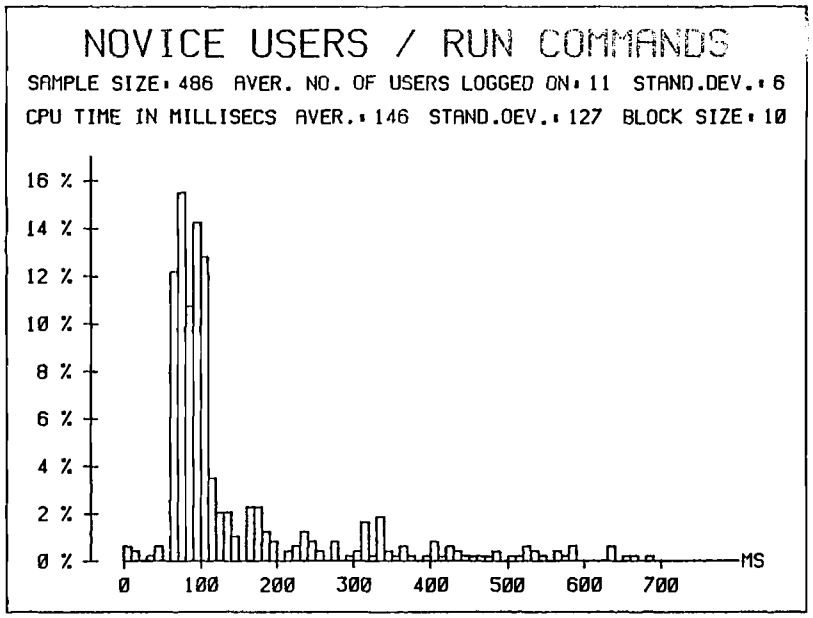

Figure II.37

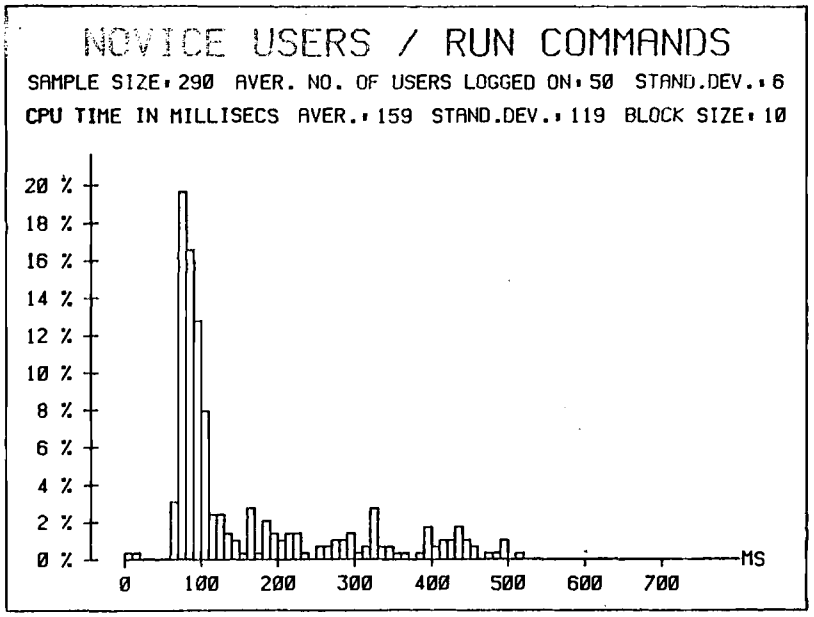

Figure II.38

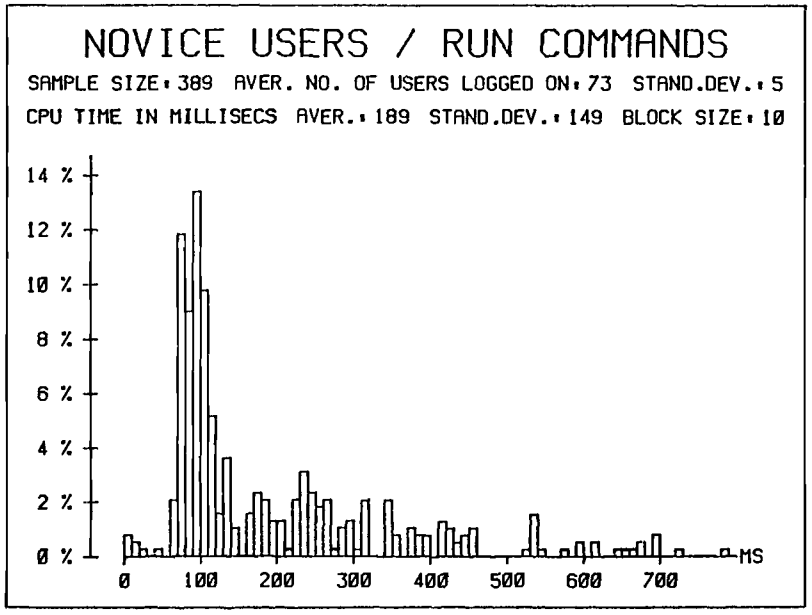

Figure II.39

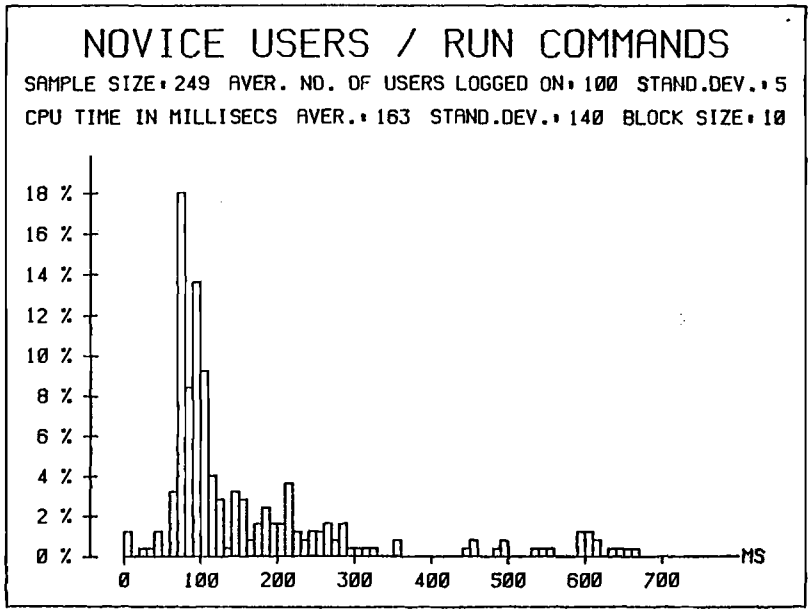

Figure II.40

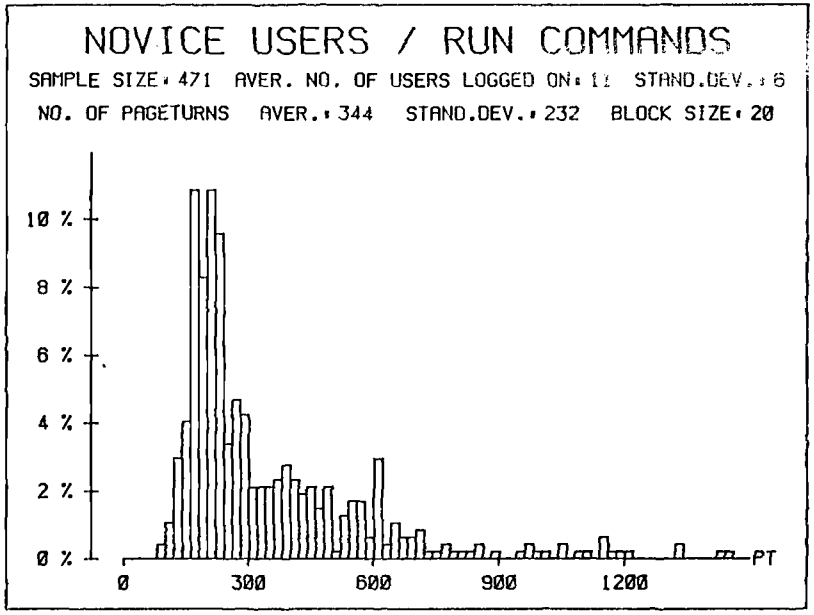

Figure II.41

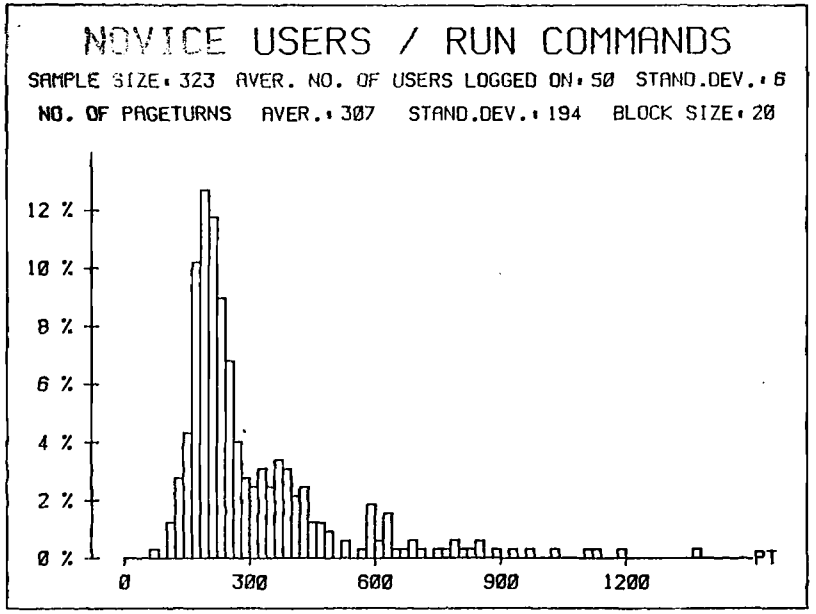

Figure II.42

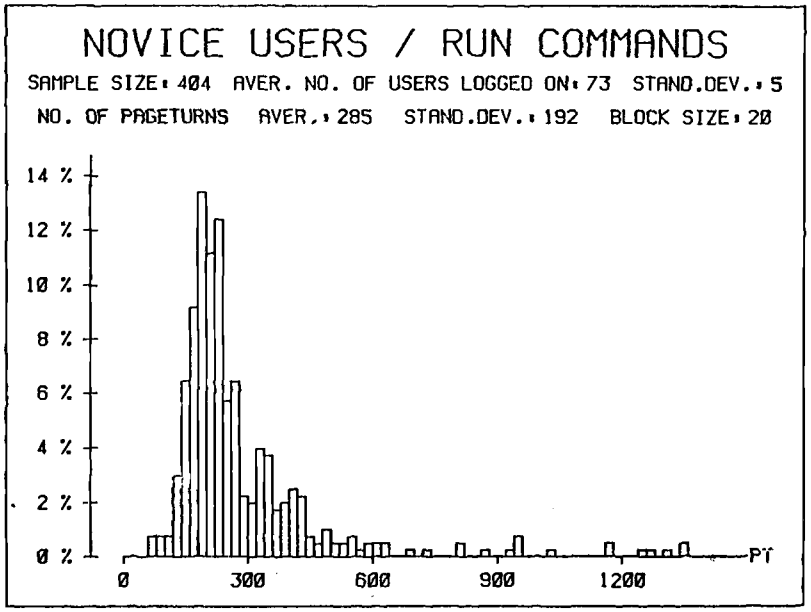

Figure II.43

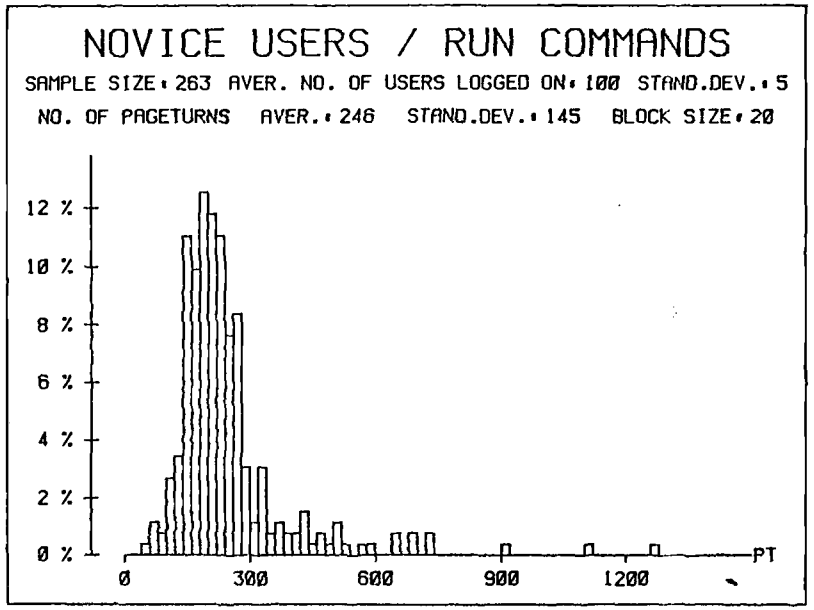

Figure II.44

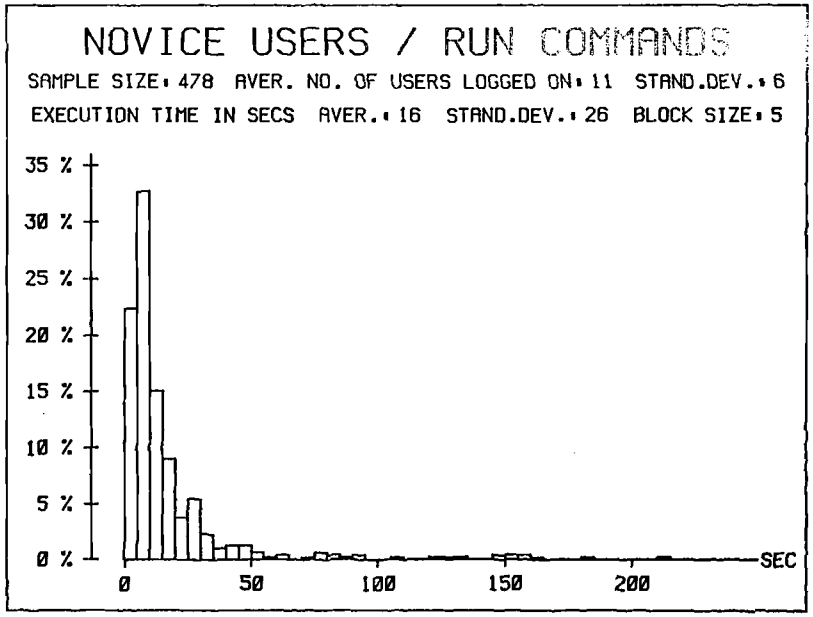

Figure II.45

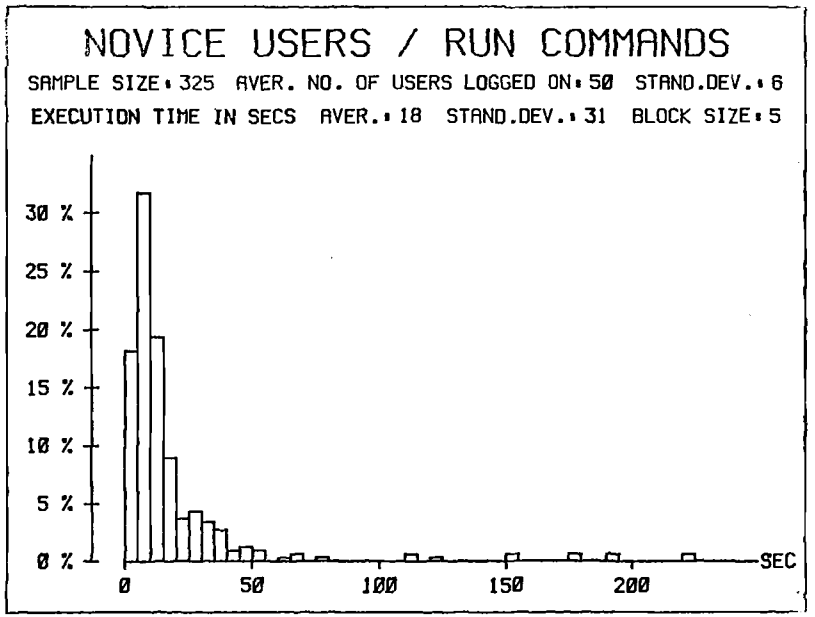

Figure II.46

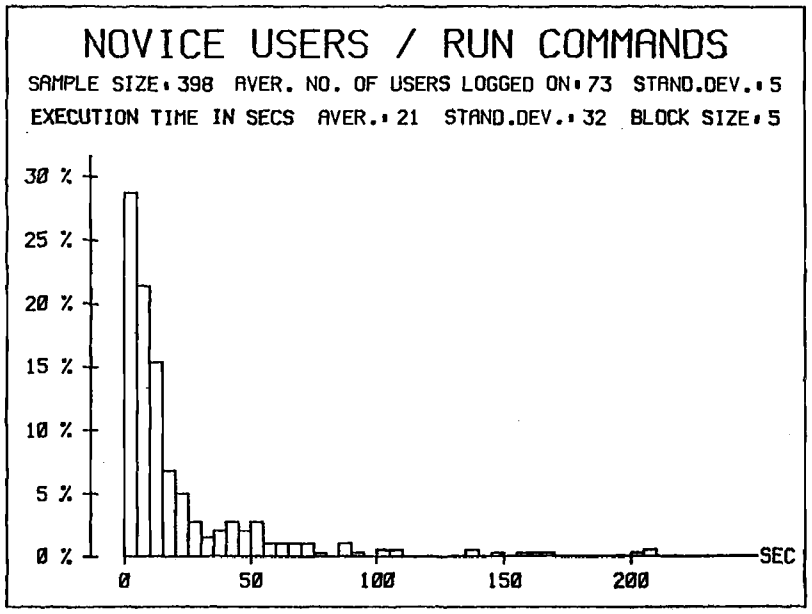

Figure II.47

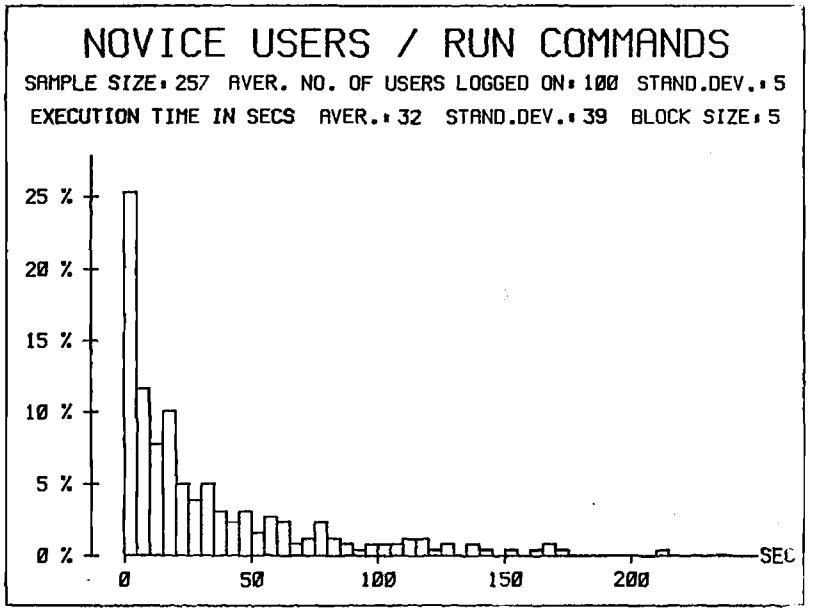

Figure II.48

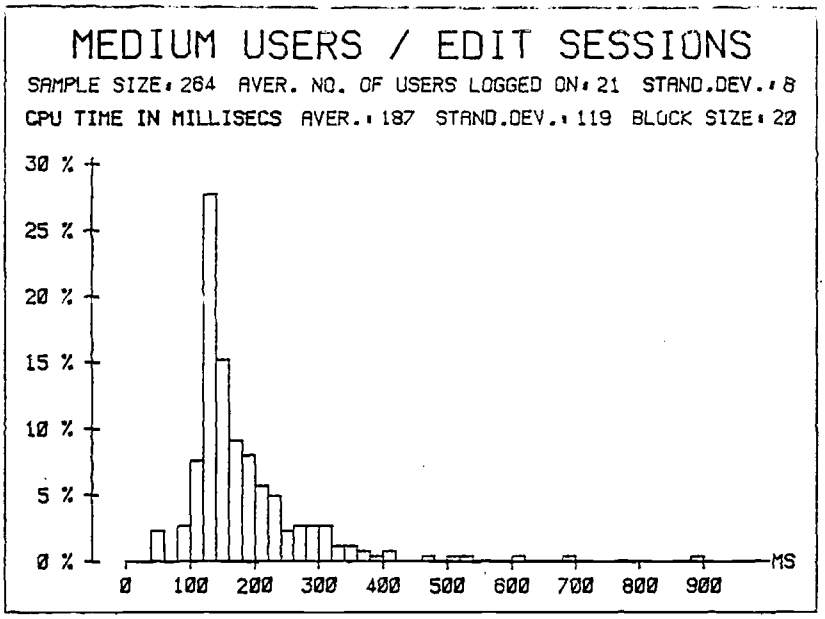

Figure II.49

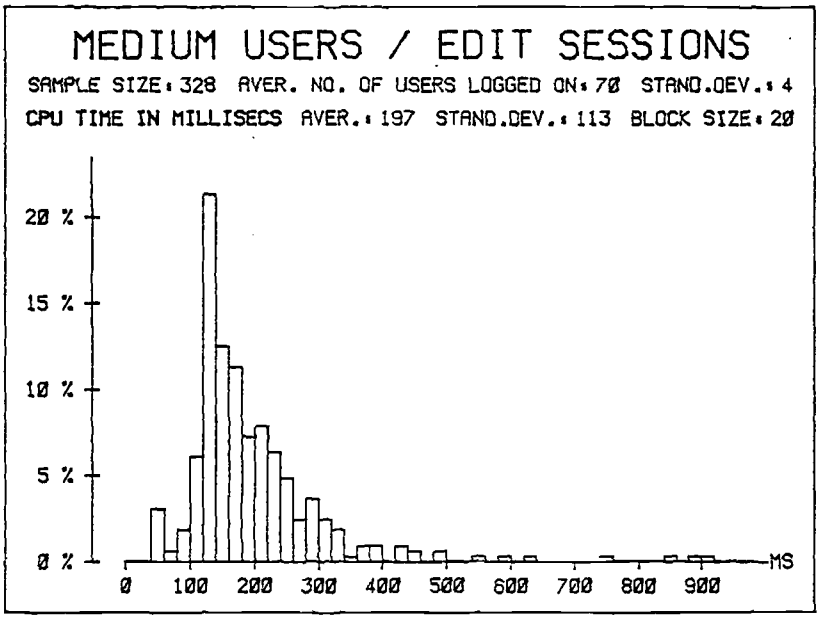

Figure II.50

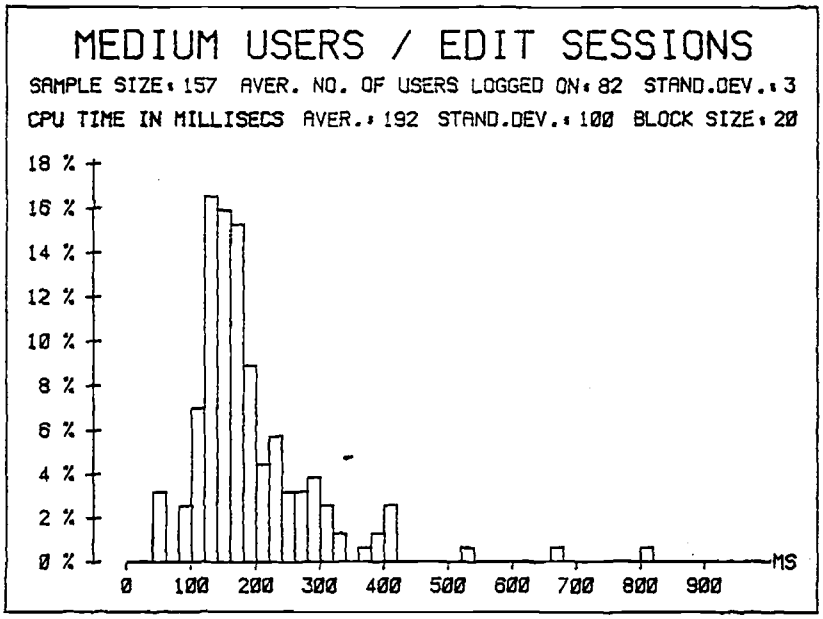

Figure II.51

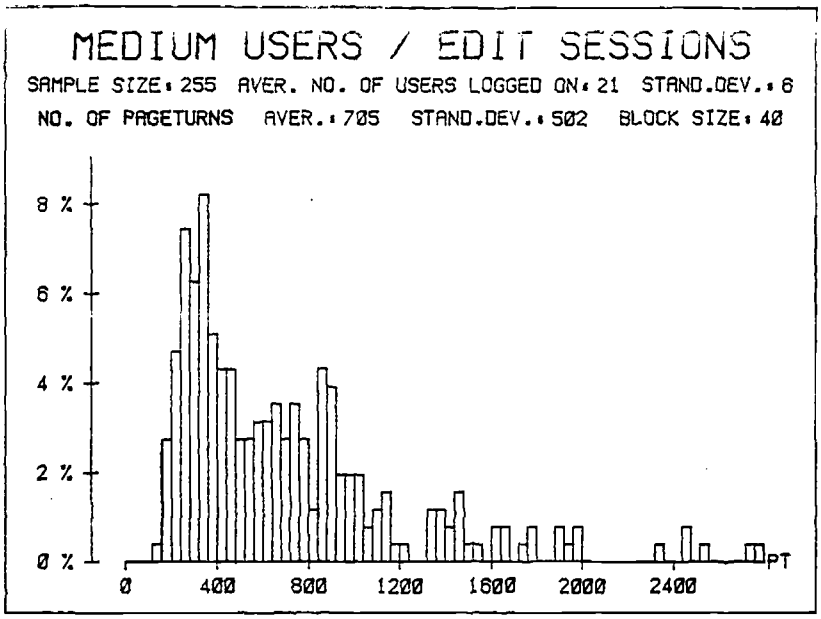

Figure II.52

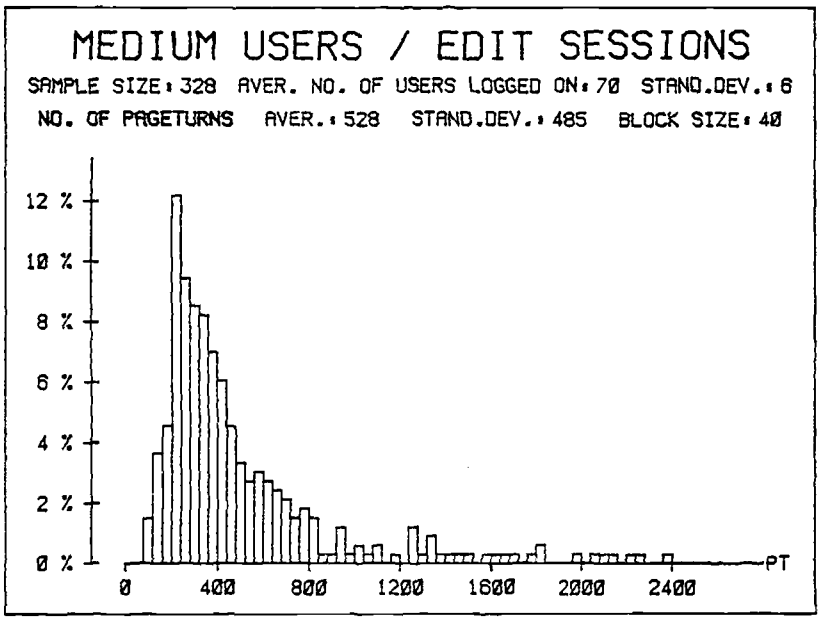

Figure II.53

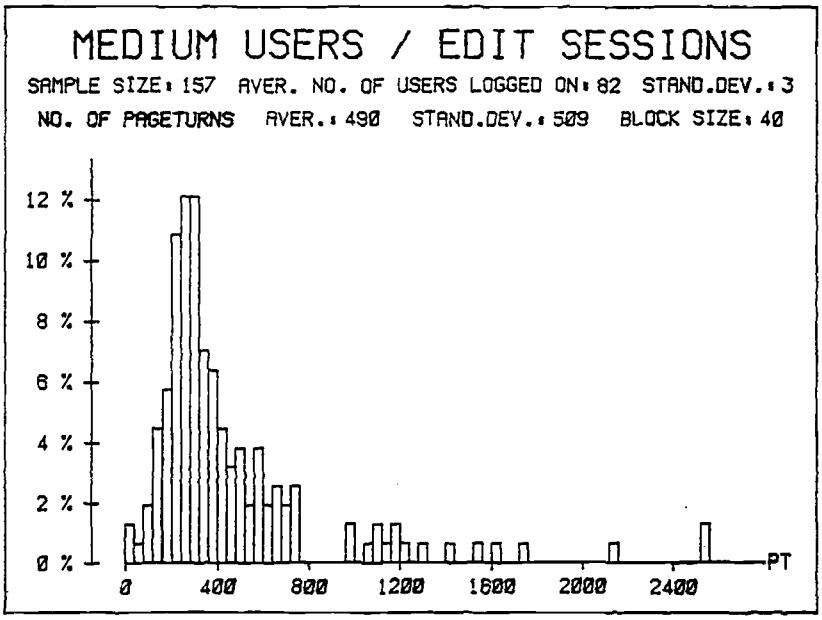

Figure II.54

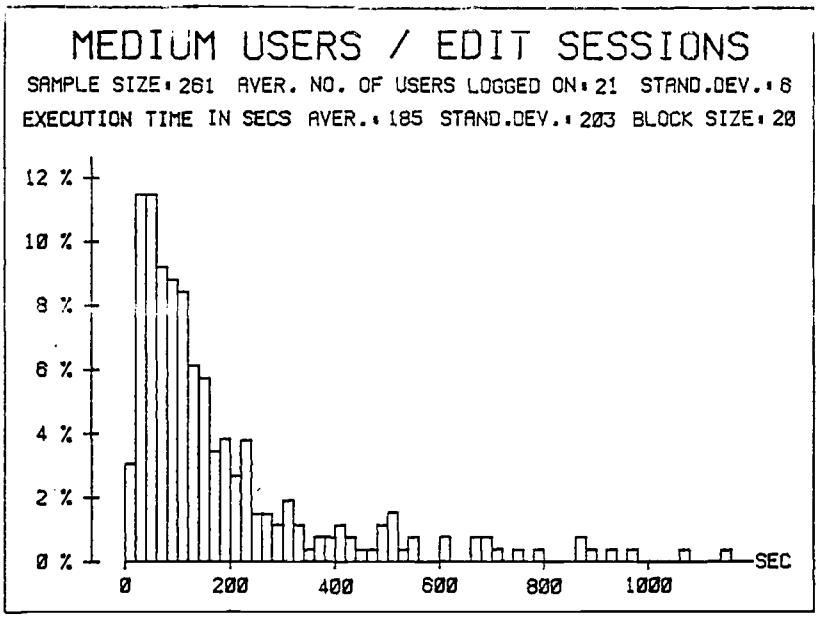

Figure II.55

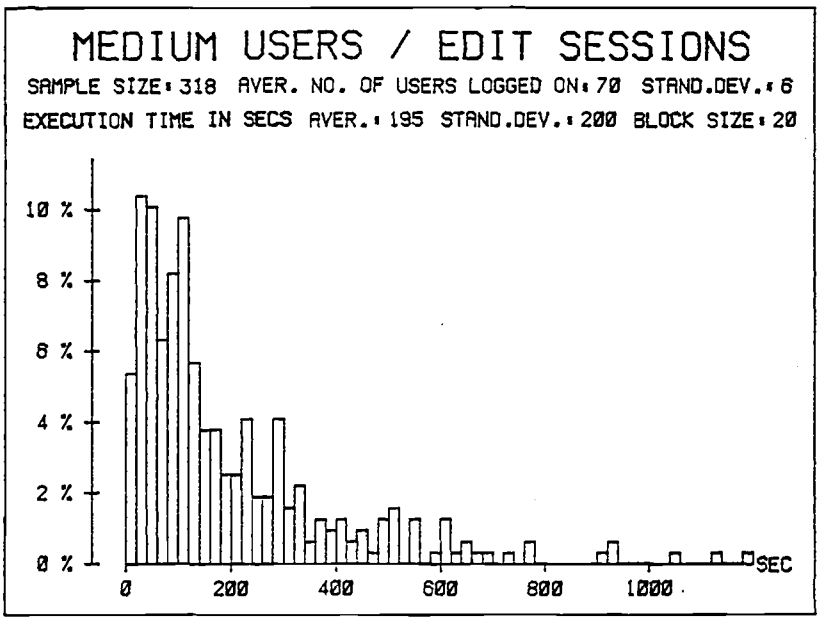

Figure II.56

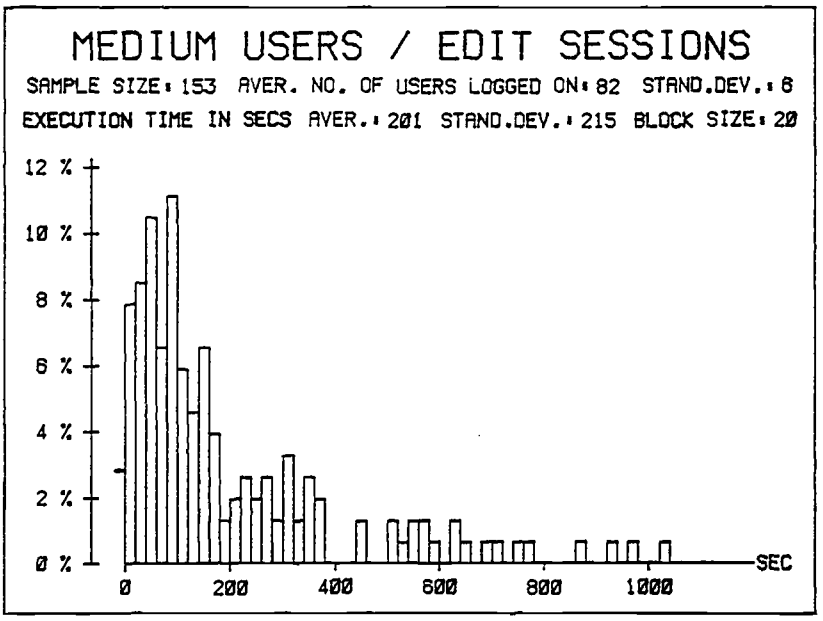

Figure II.57

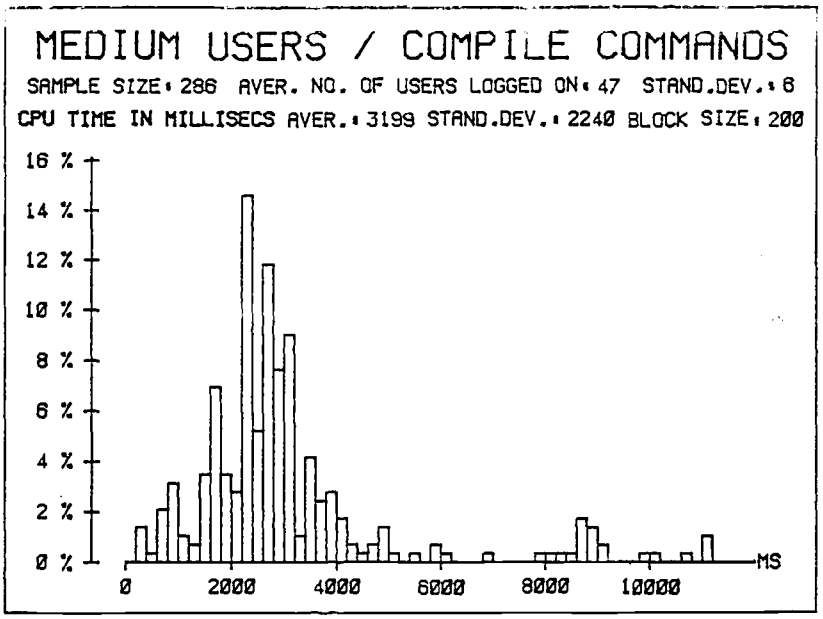

Figure II.58

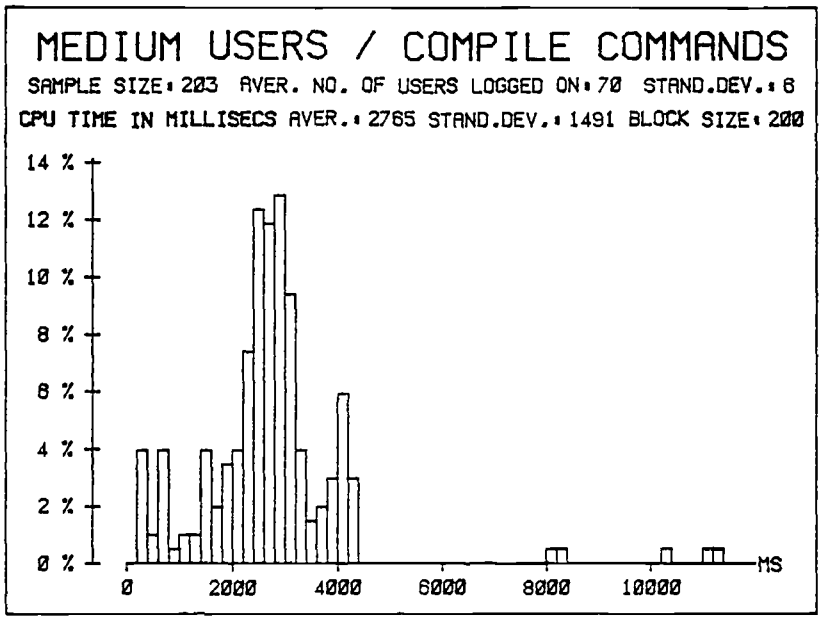

Figure II.59

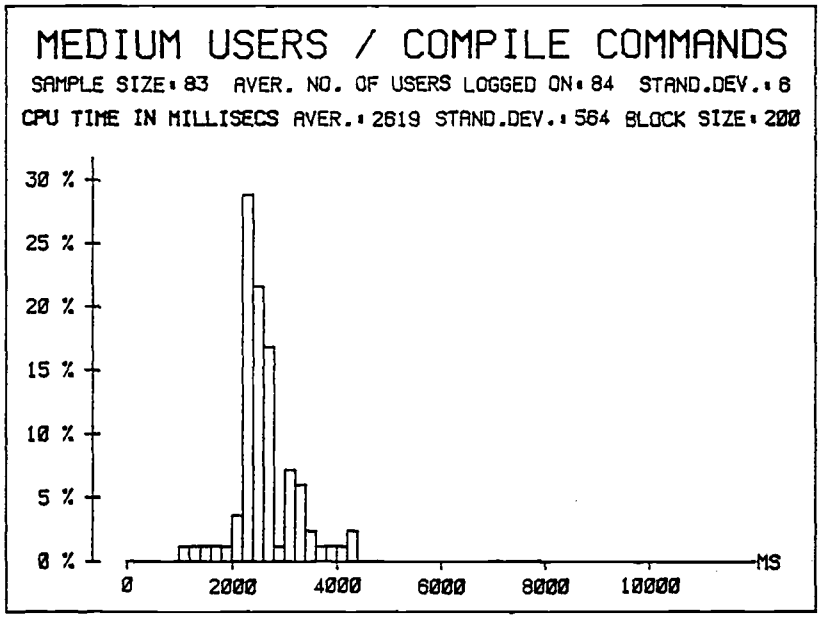

Figure II.60

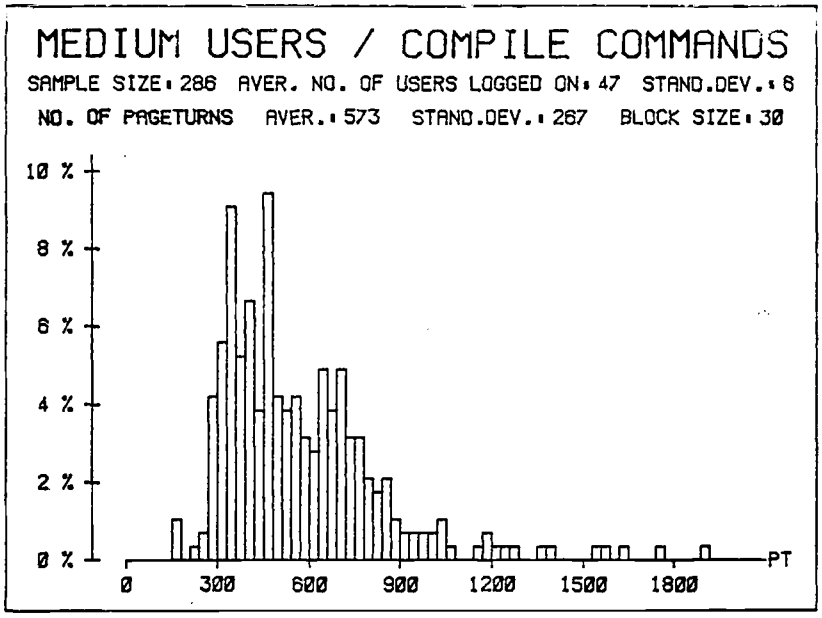

Figure II.81

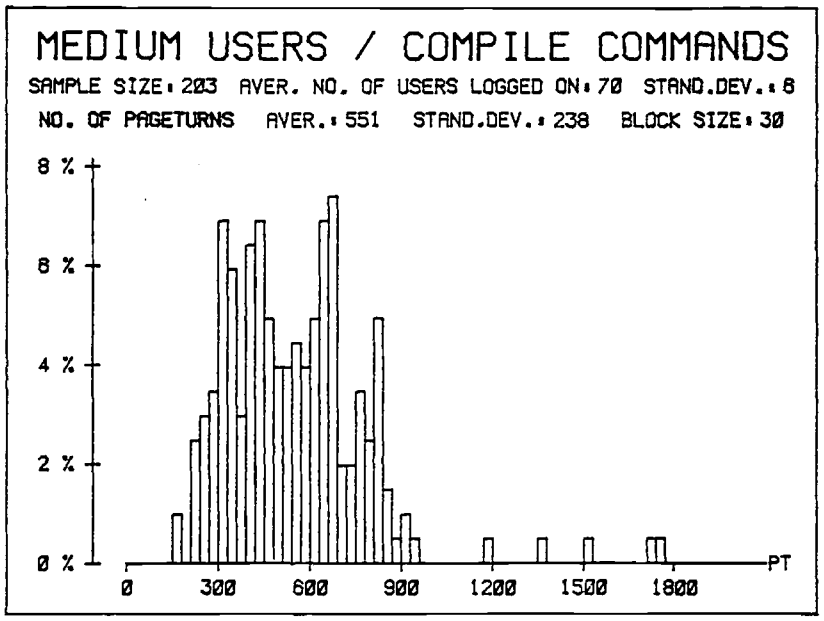

Figure II.82

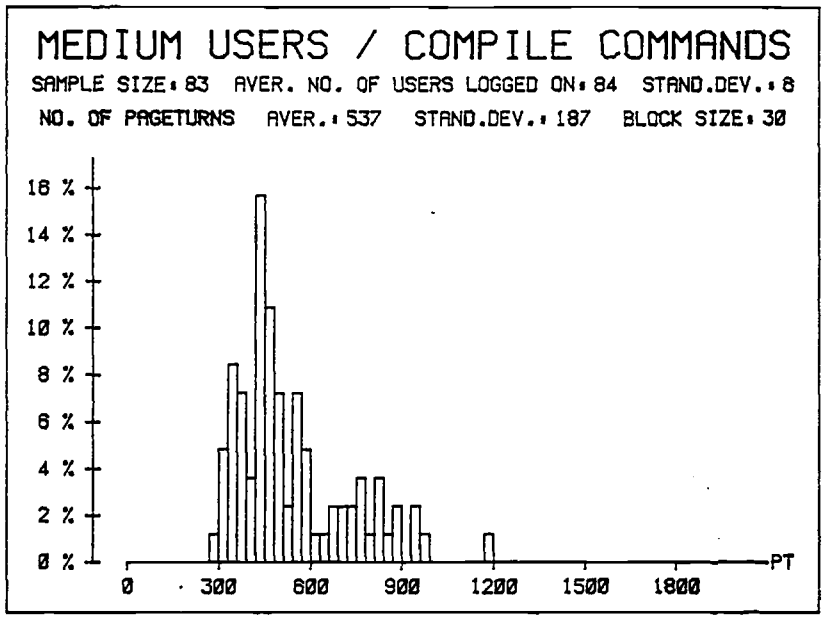

Figure II.83

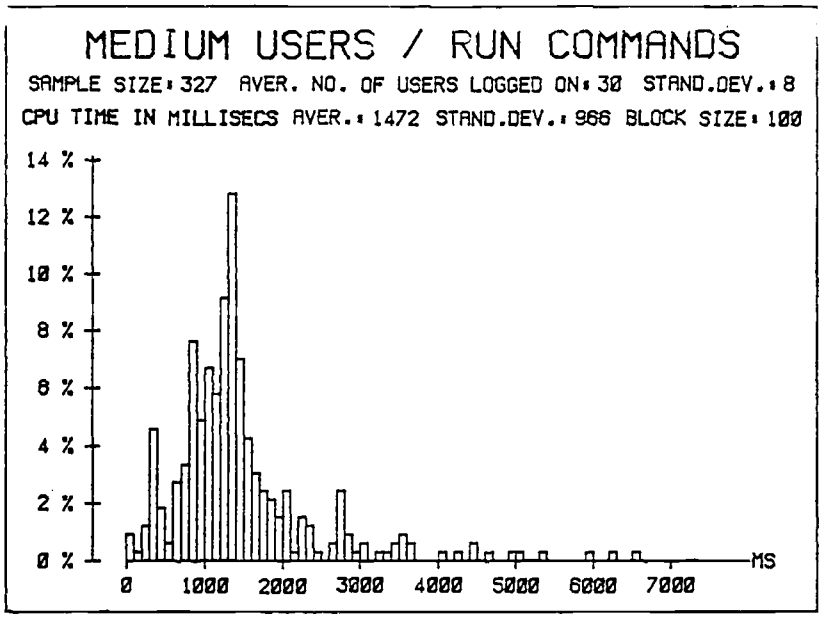

Figure II.64

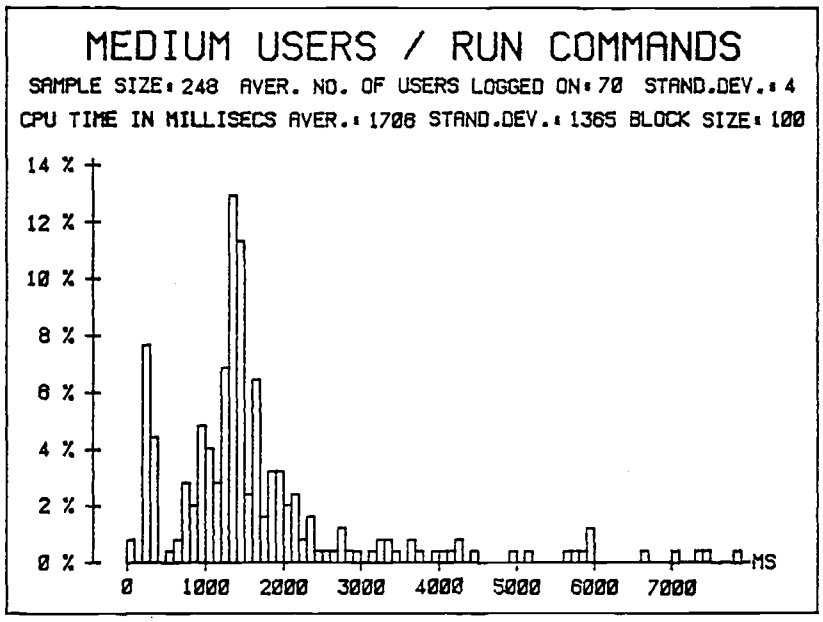

Figure II.65

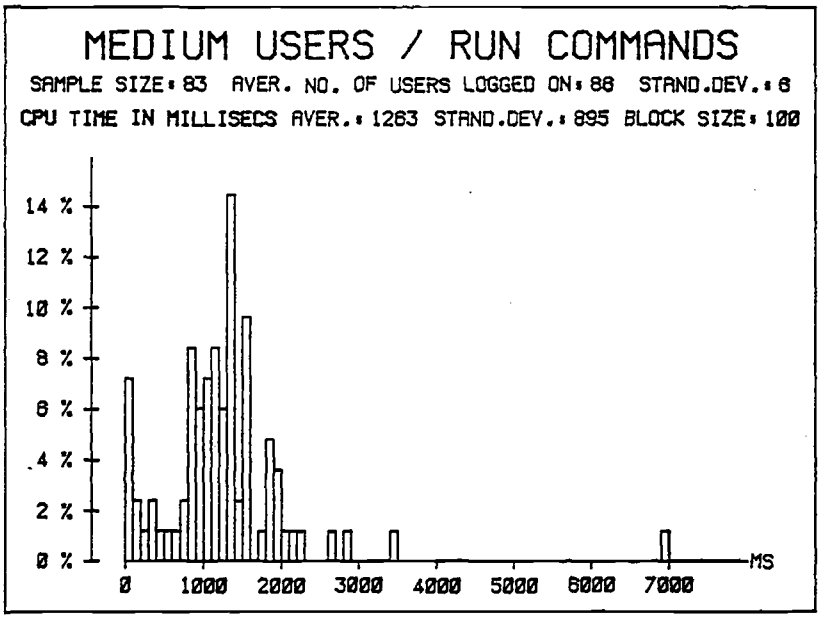

Figure II.66

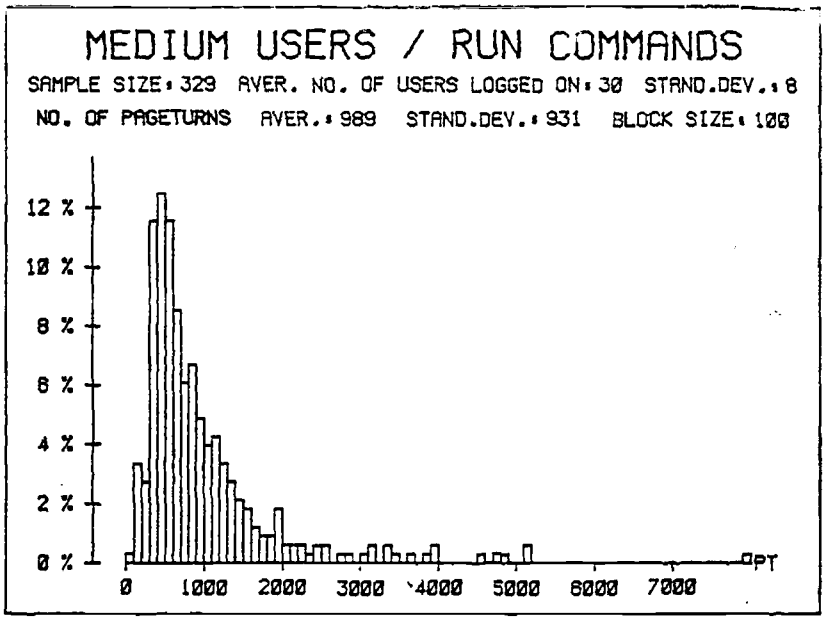

Figure II.67

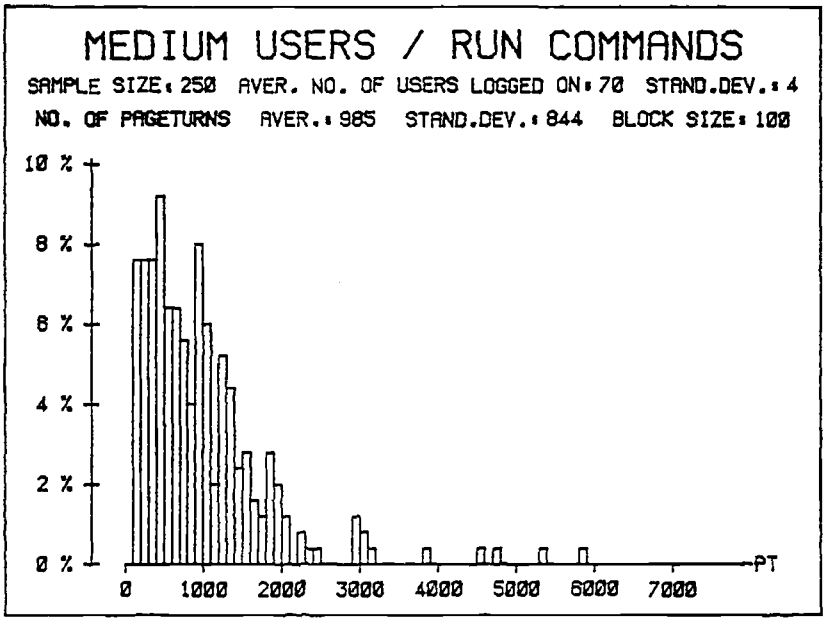

Figure II.68

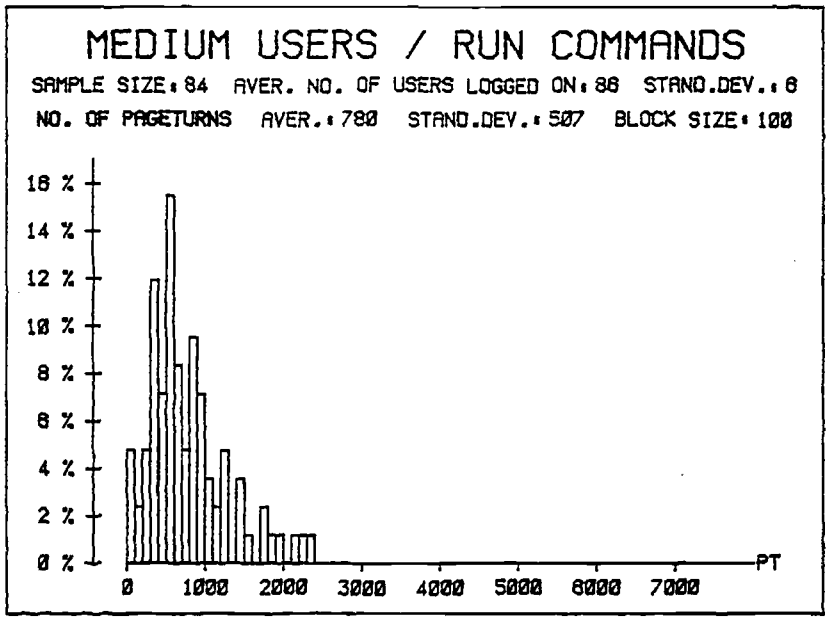

Figure II.69

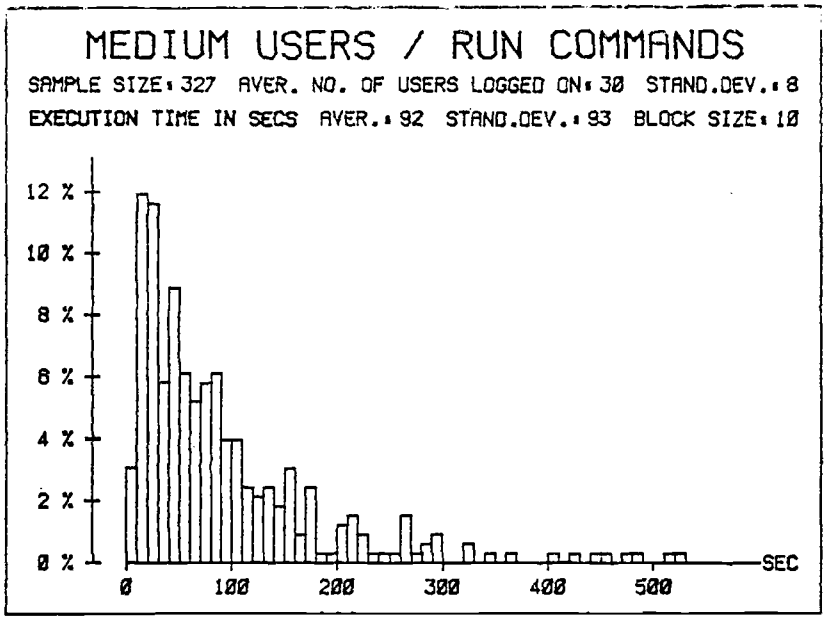

Figure II.70

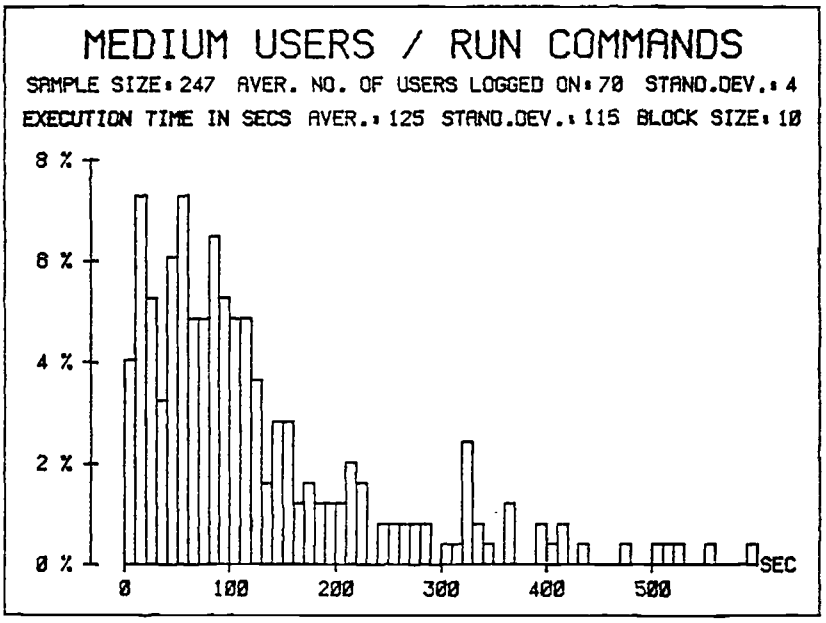

Figure II.71

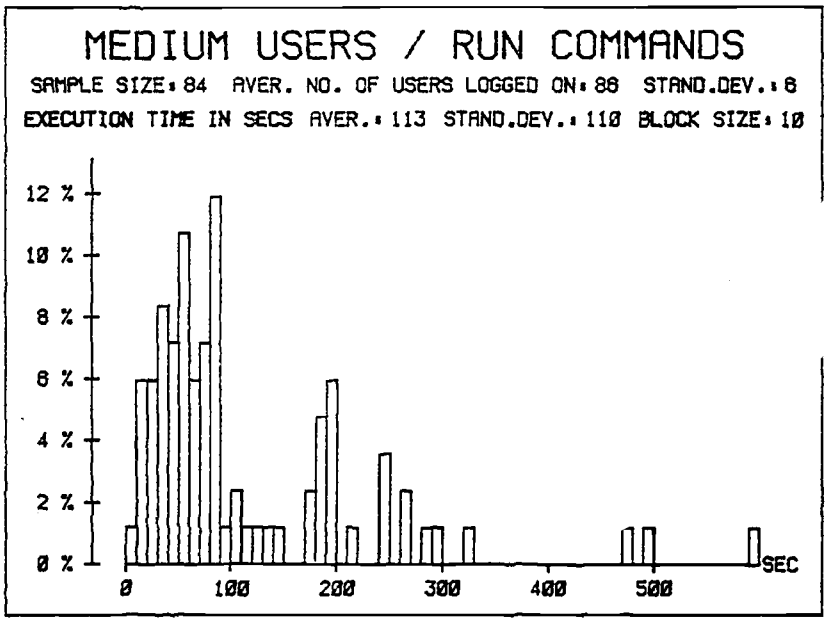

Figure II.72

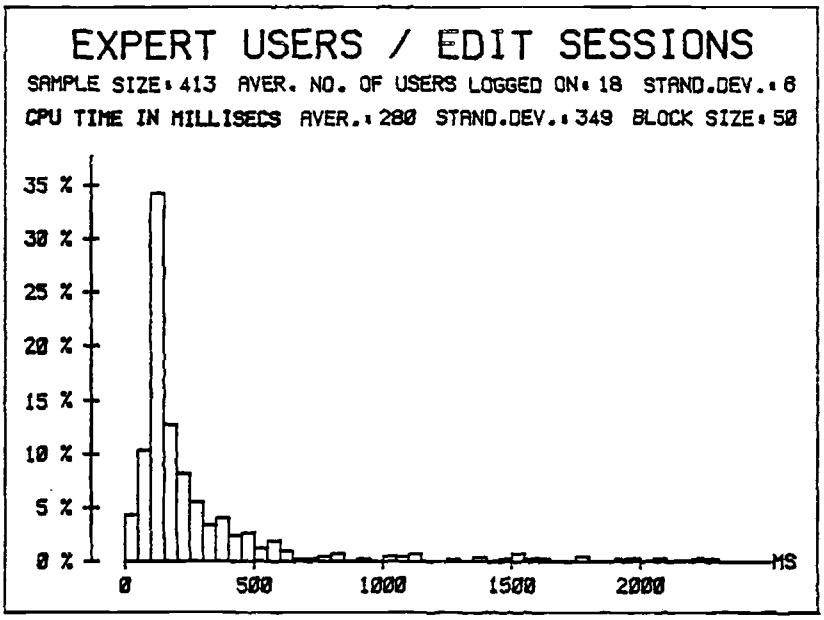

Figure II.73

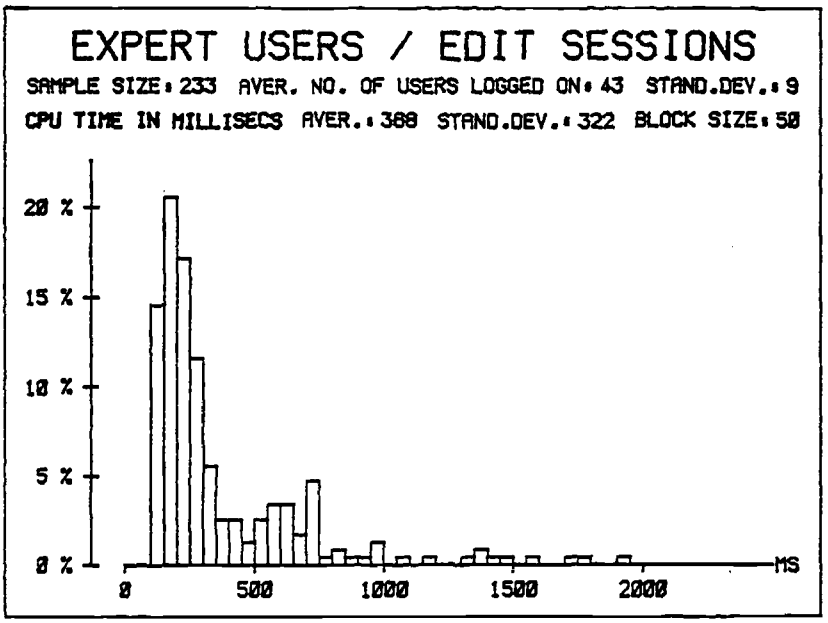

Figure II.74

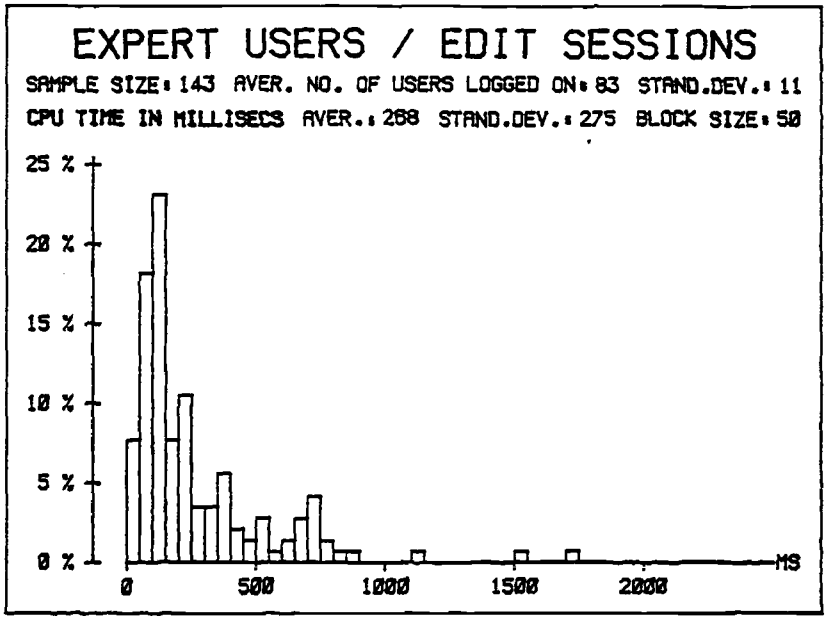

Figure II.75

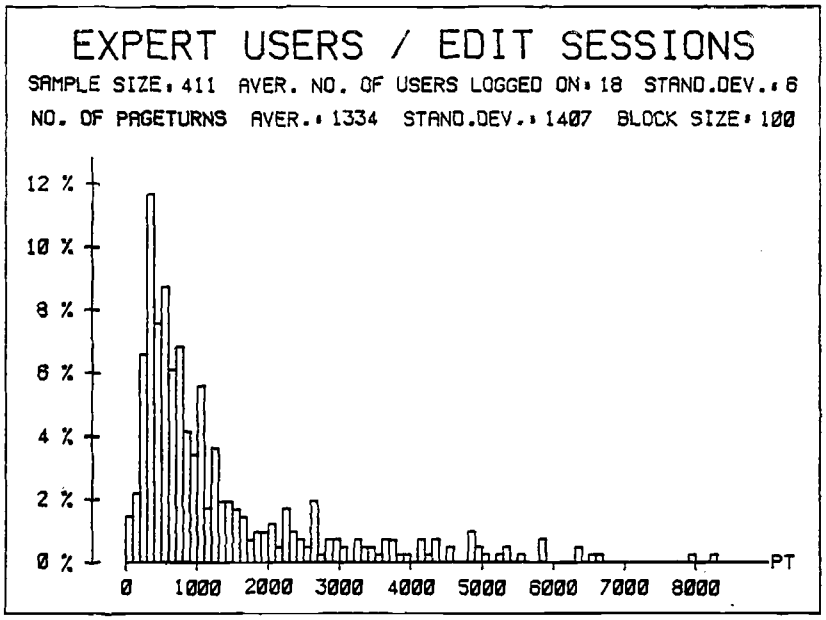

Figure II.76

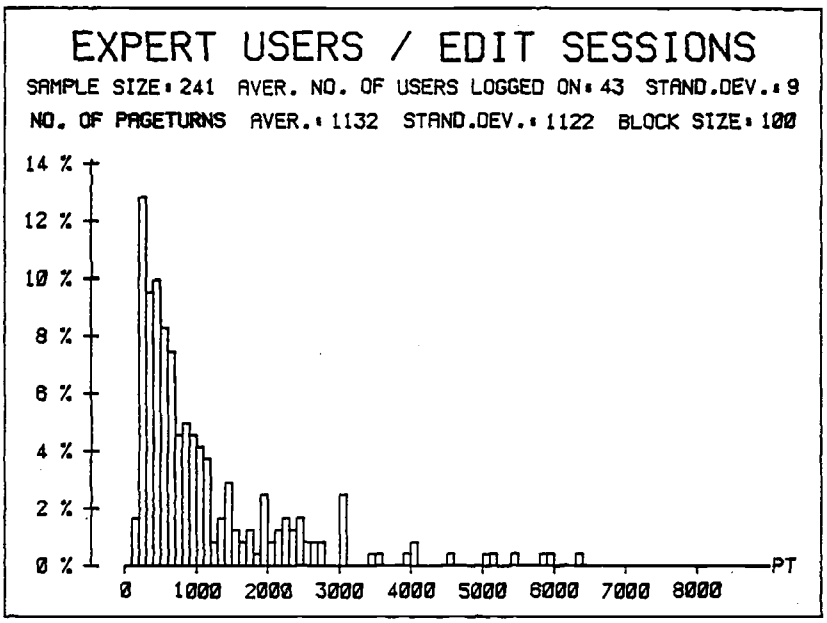

Figure II.77

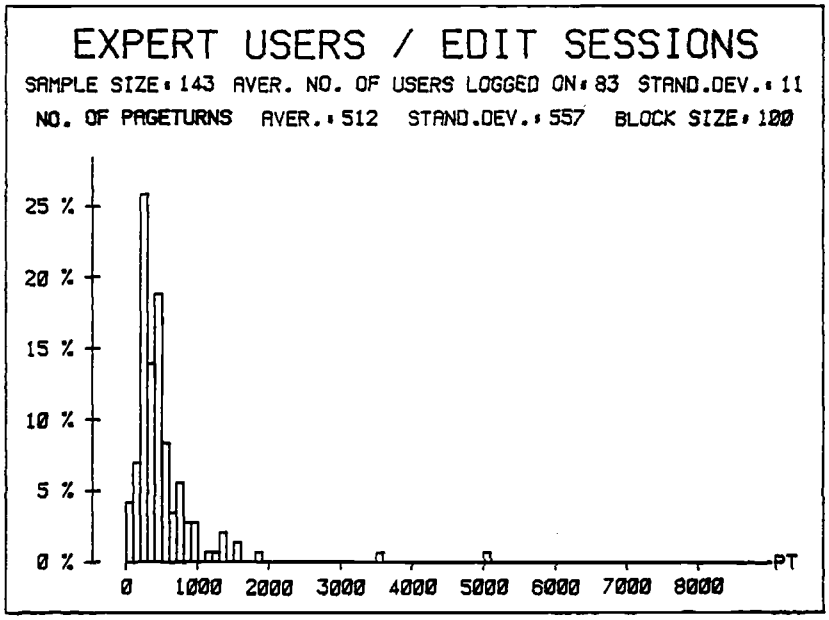

Figure II.78

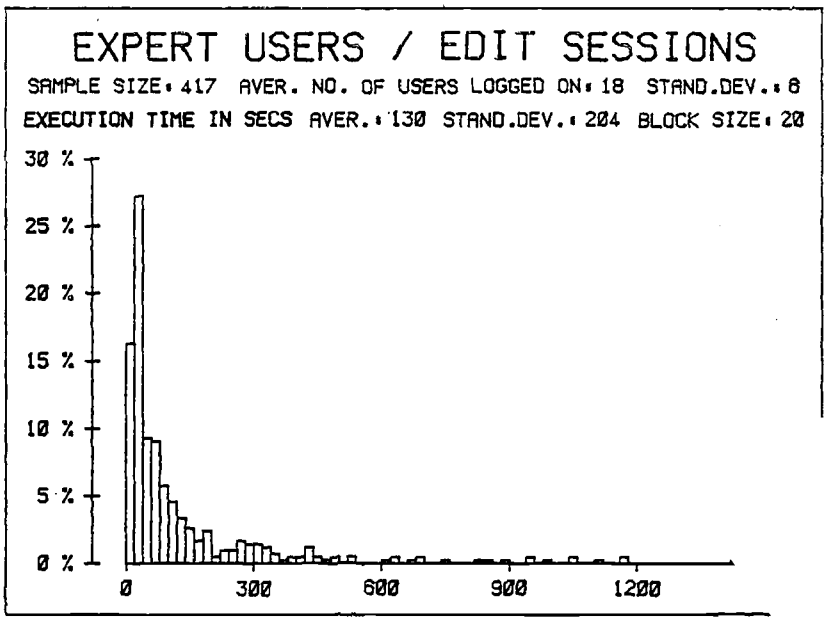

Figure II.79

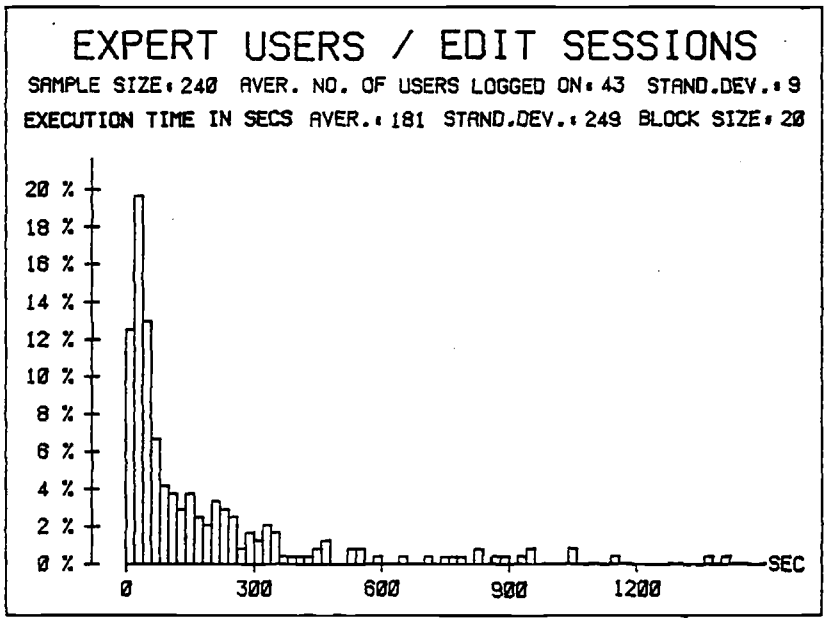

Figure II.80

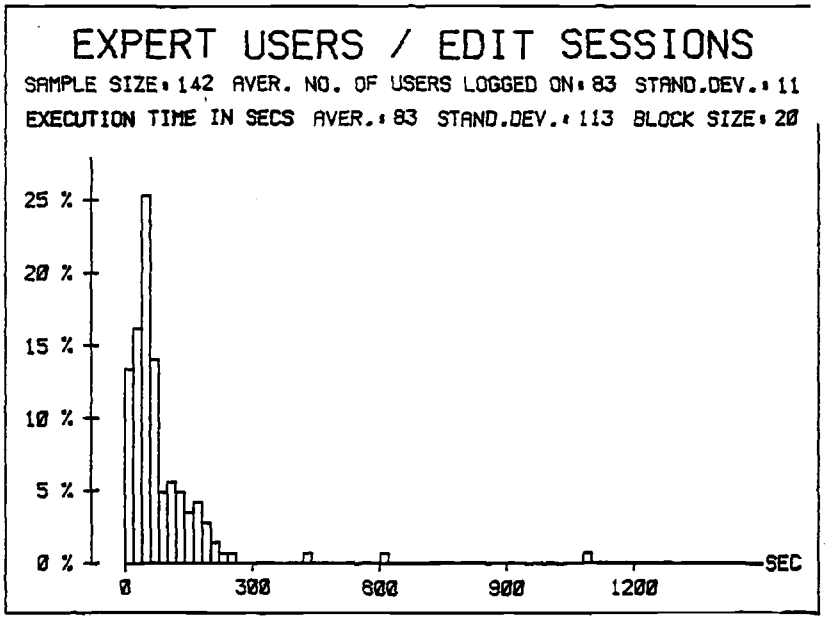

Figure II.81

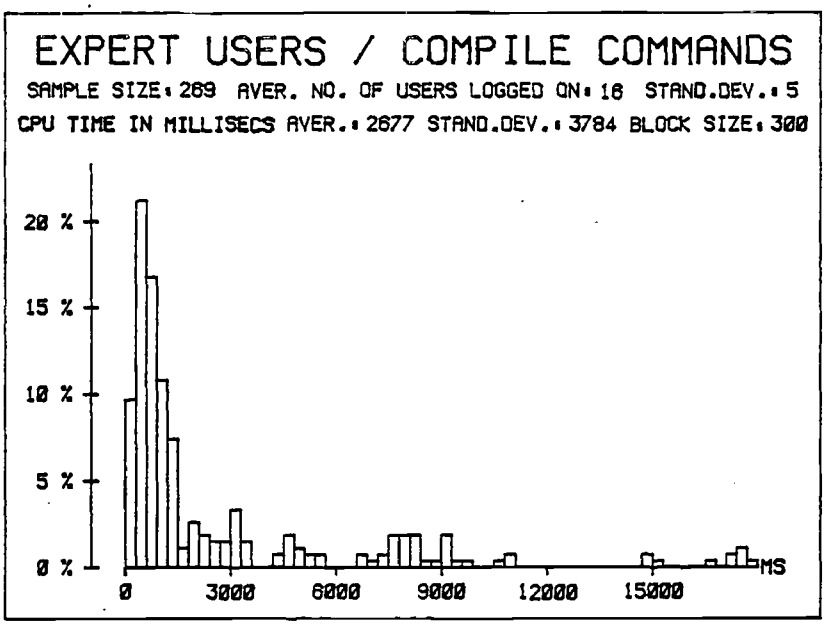

Figure II.82

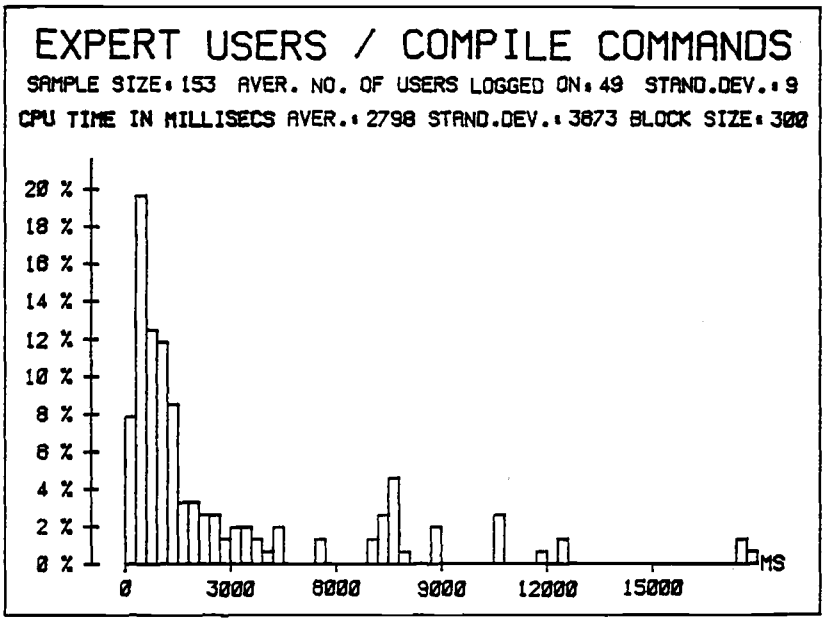

Figure II.83

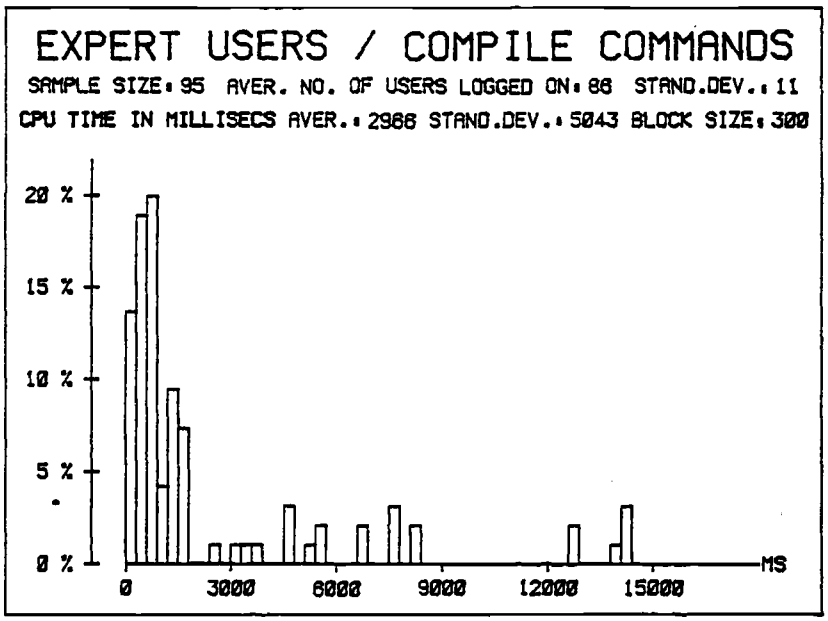

Figure II.84

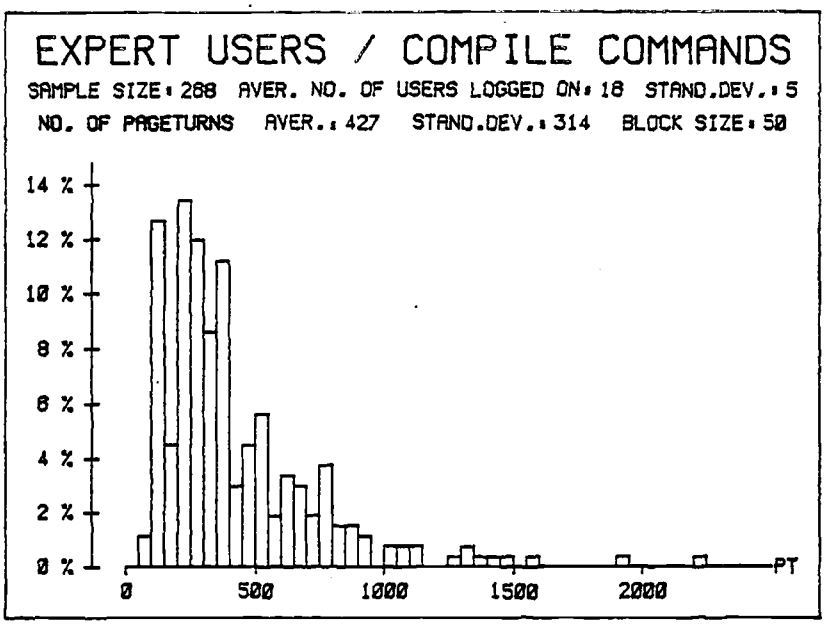

Figure II.85

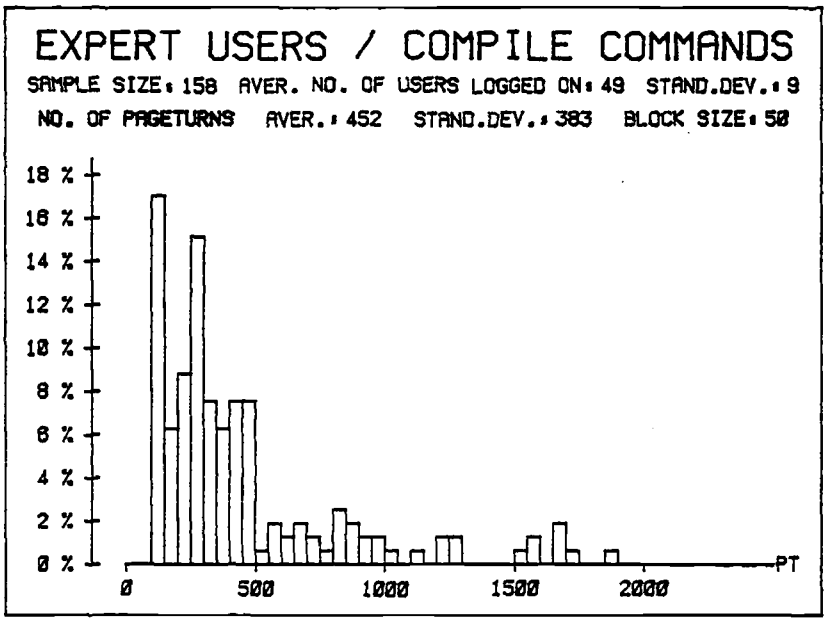

Figure II.86

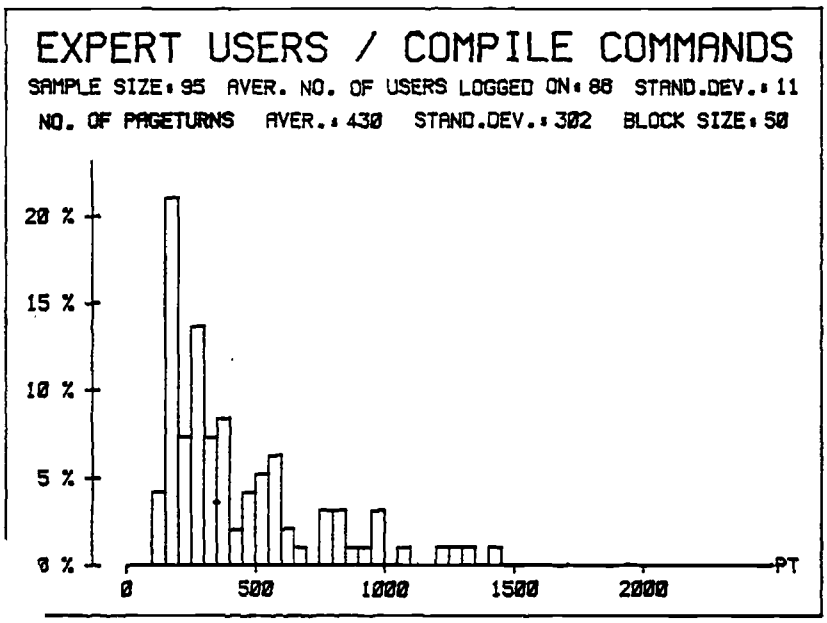

Figure II.87

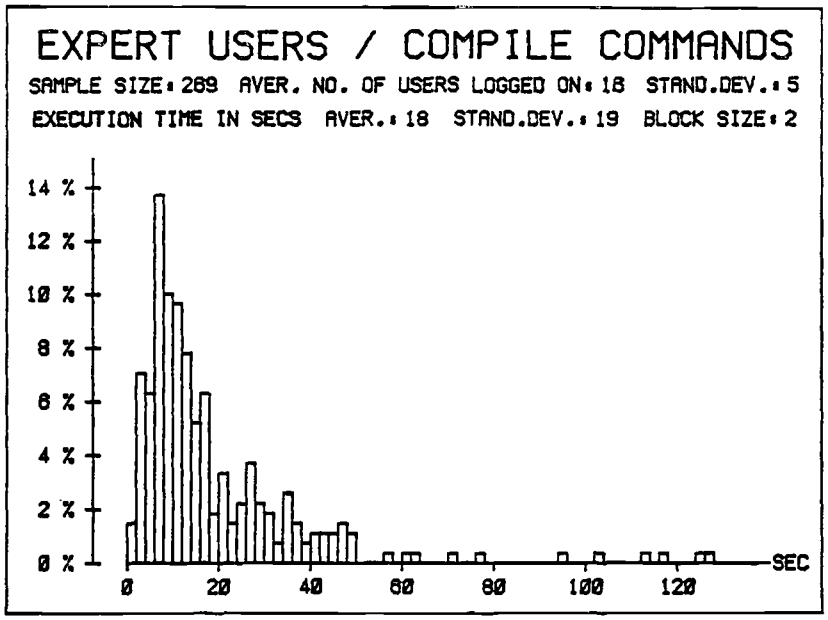

Figure II.88

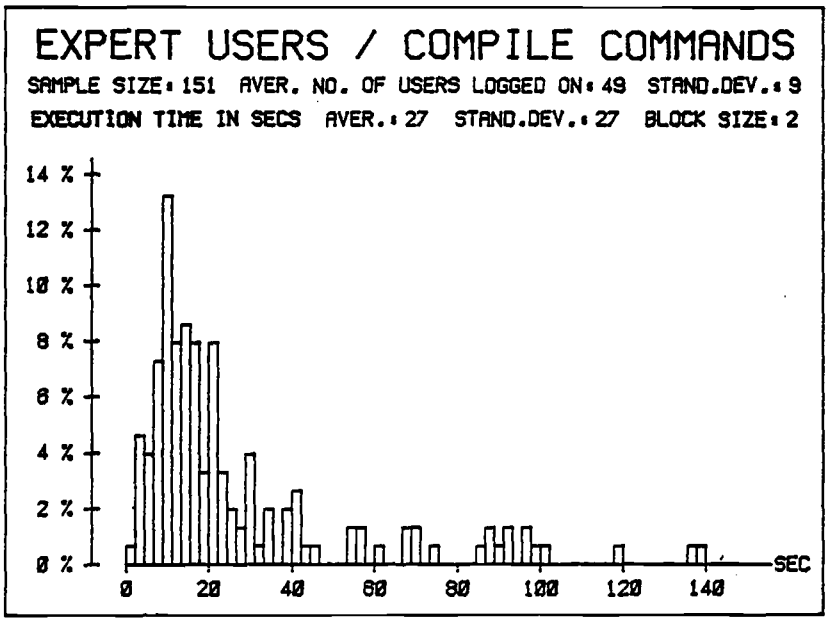

Figure II.89

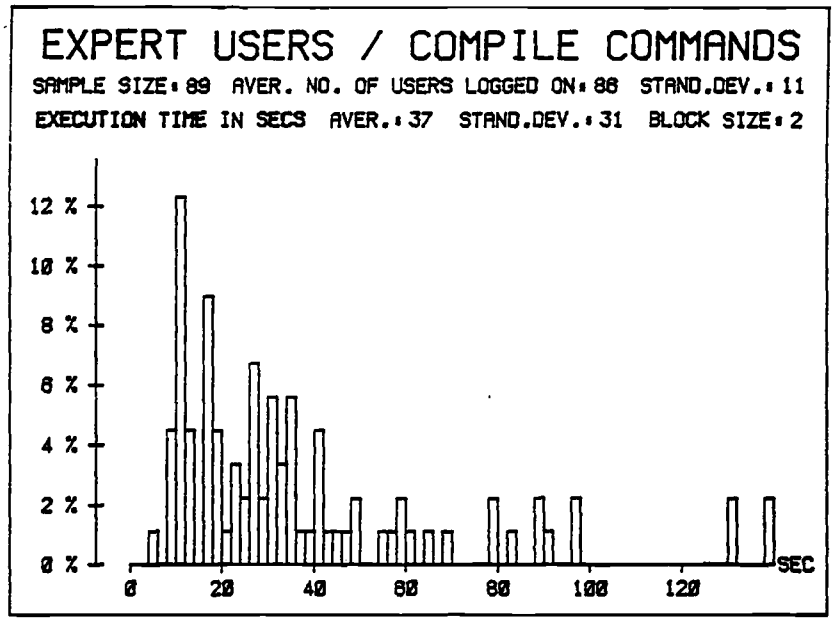

Figure II.90

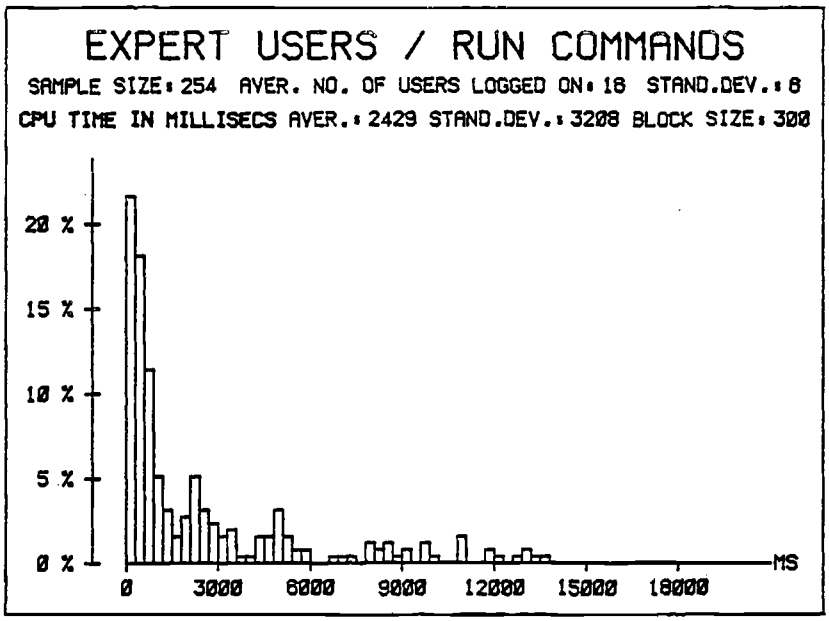

Figure II.91

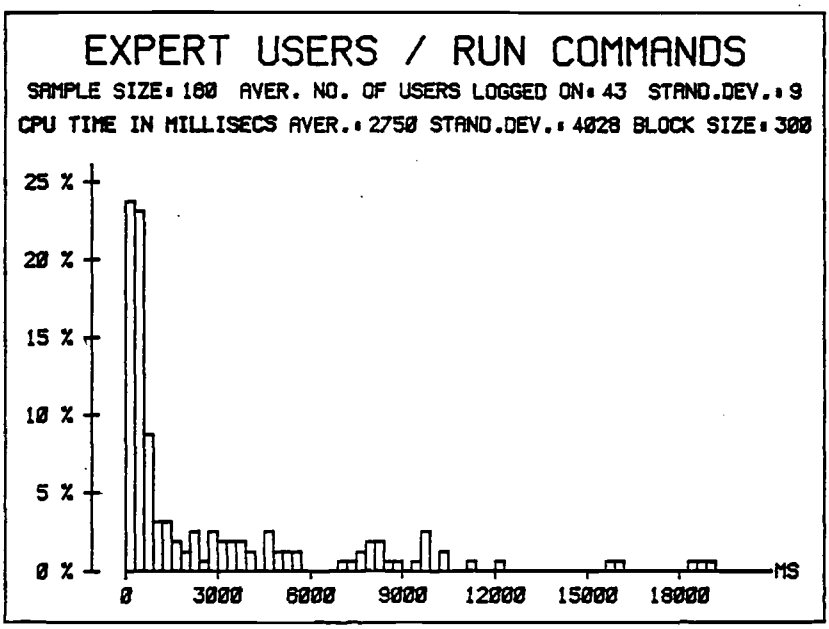

Figure II.92

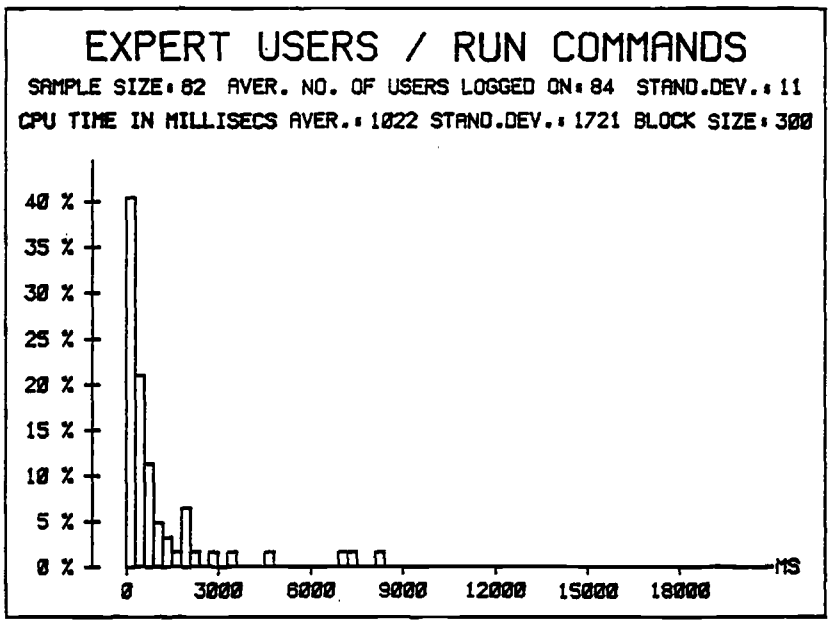

Figure II.93

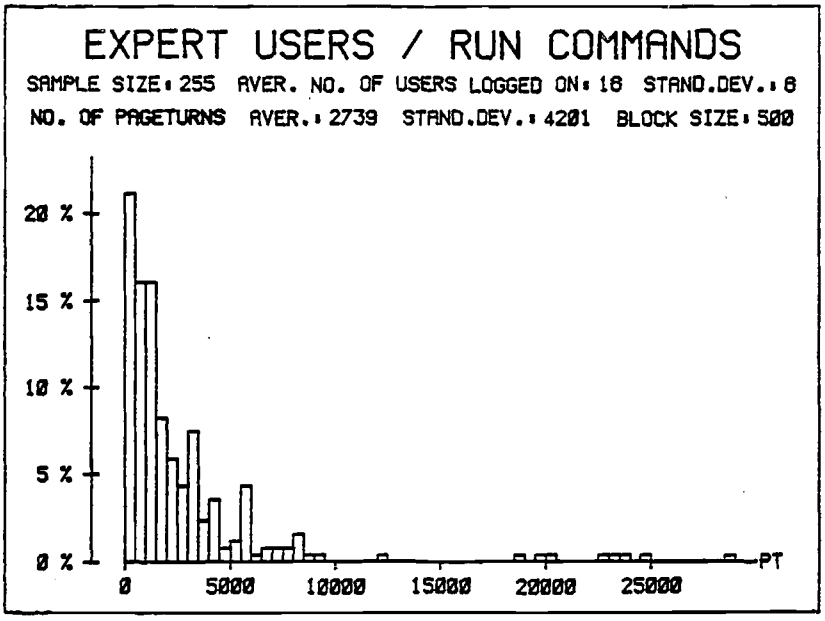

Figure II.94

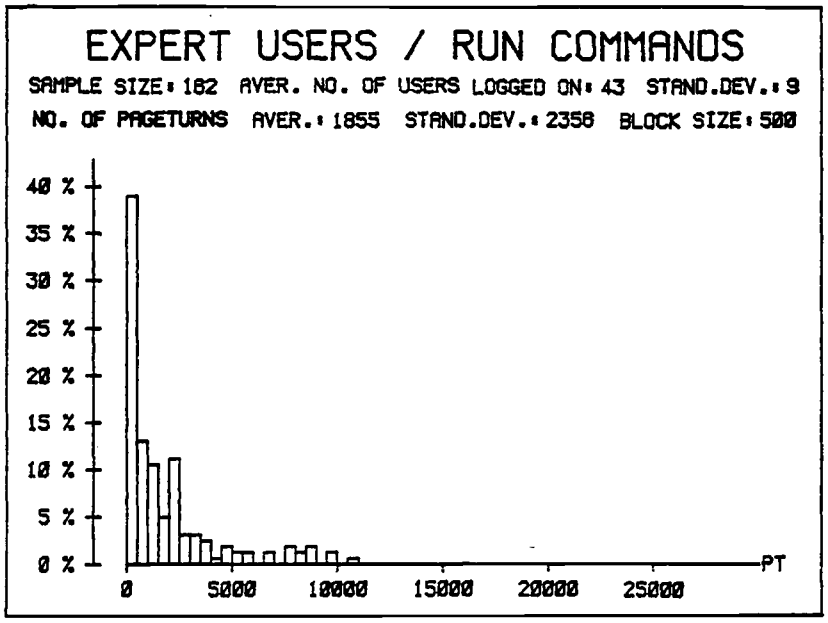

Figure II.95

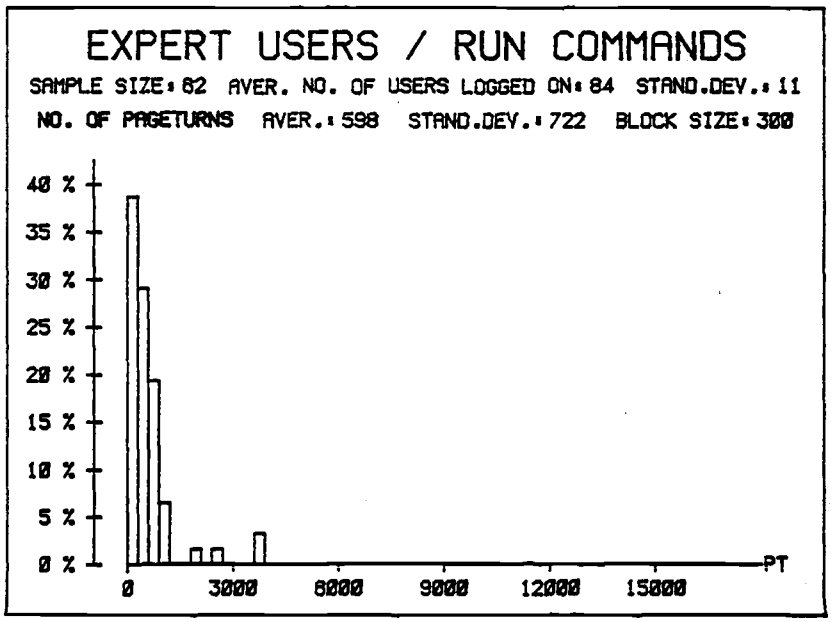

Figure II.96

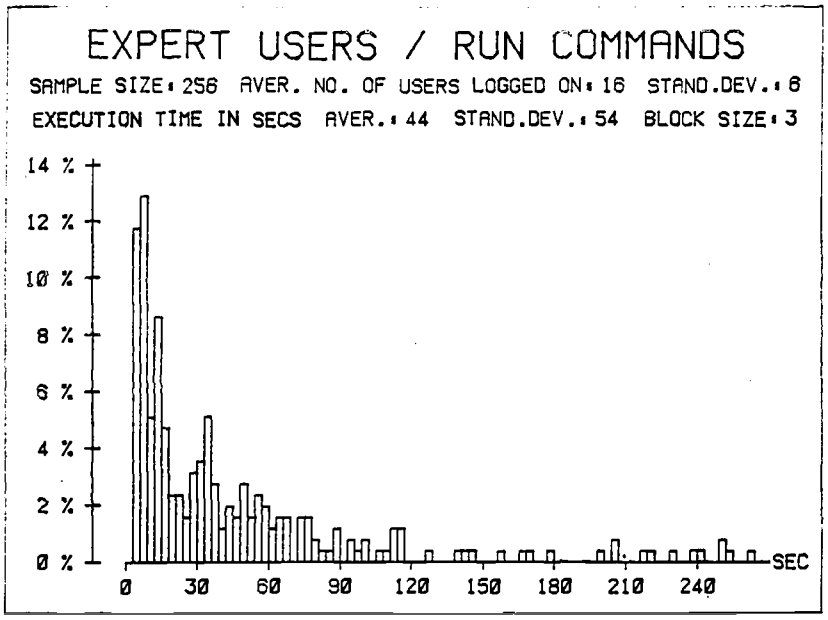

Figure II.97

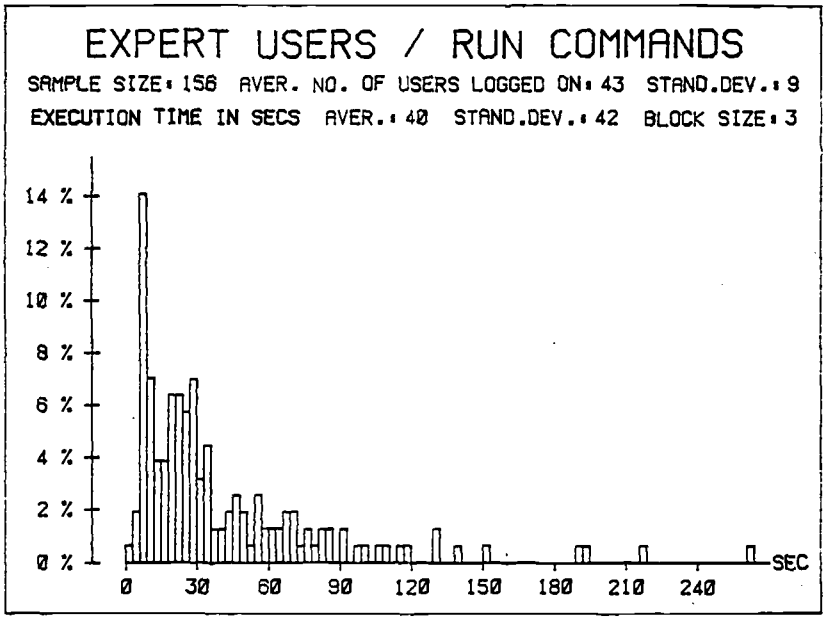

Figure II.98

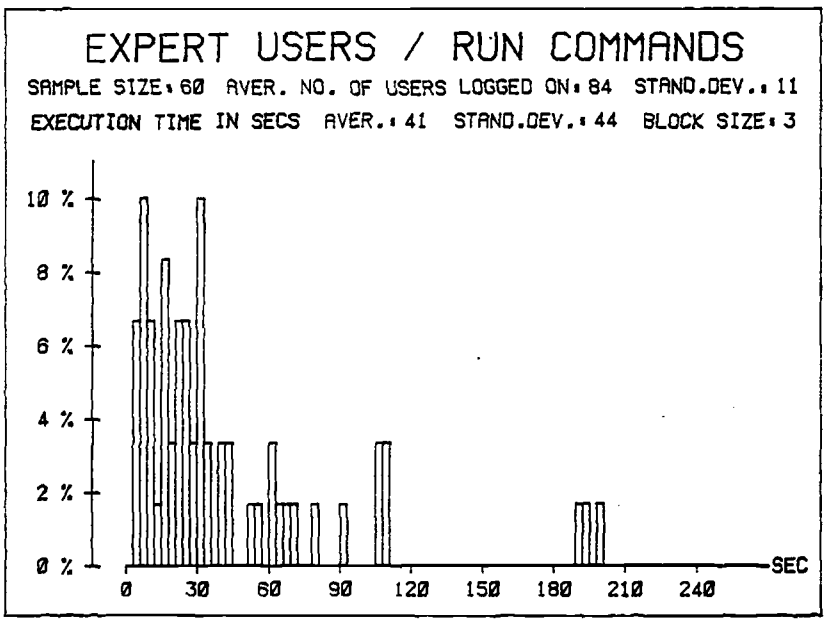

Figure II.99

C H A P T E R T H R E E

III. THE AUTOMATIC WORKLOAD GENERATOR

The results presented so far are going to be used to define or generate realistic user scripts. These user scripts then serve as input for the Edinburgh Remote Terminal Emulator [ADA1,ADA2].

1. REMOTE TERMINAL EMULATION

Remote Terminal Emulation is a method of testing computer systems; a workload driver is implemented external to and independent of the system. A remote terminal emulator is also very helpful at many different stages during system developments, i.e initial checkouts, acceptance tests and tuning [SPIE,DUKE].

The driver can be connected to the target system directly or through a communication network. Thus the target system cannot detect any difference between the driver and real users at real terminals. Performance characteristics can be determined by analysing a log containing data of driver/system interactions.

Many of the workload characteristics mentioned in chapter I.4 apply to this kind of workload modelling. One of the major advantages of this kind of executable artificial workload model is its flexibility and reproducibility. This means it can be presented to the target system from time to time; performance differences are then caused by the system alone.

AREAS OF USE OF AN RTE

As already mentioned above an RTE can be used in many different ways within a computer evaluation study, for example for tuning a system. Under the assumption that a fairly realistic representation of the workload is given, it is possible to change scheduling parameters and find out optimal settings.

If used as a tuning tool, a system's overall performance can be significantly improved. Another area of interest is system development. If changes in the software or hardware are made an RTE can be employed to test the modified system before it is released to users. An RTE can be used at many stages of a system development because once developed and properly set up it provides a powerful way of presenting a system with a repeatable workload. This may help to find errors or performance bottlenecks.

In experimental studies an RTE provides a very efficient way of presenting different kinds of workload to a target system, as the workload presented can easily be changed. If a predicted, future workload can be presented to a target system, future deteriorations in performance may be predicted and preventive measures taken.

2. THE EDINBURGH REMOTE TERMINAL EMULATOR

The Edinburgh Remote Terminal Emulator [ADA1] is such a general purpose workload driver which can be employed for the purposes mentioned above.

THE HARDWARE

ERTE runs on a DEC PDP11/40 computer with 32K words of store, and a RK05 cartridge disk unit (1.2 Mbytes) holding the system software.

A 66 Mbyte Ampex DM980 diskdrive holds the script files and monitor log data. The target system is usually the Edinburgh multi-access system (EMAS), but experiments are also performed on a VAX 11/780 under VMS. The communication is through a DQS11-E synchronous line interface at a line speed of 9.6 KB to the RCO-net. Figure III.2 gives an overview of the network and the position of ERTE.

THE SOFTWARE

The emulator software consists of several co-operating tasks running under the DEIMOS [GILM] operating system which is a general-purpose multitasking system supporting a number of general user programs. The programs in the system are run as separate tasks in their own 32K word virtual memories. DEIMOS is based on a small, fixed kernel and performs 4 main functions.

a) it allocates CPU to user programs

b) it stores and forwards messages

c) it enables tasks to map onto other task's VM to facilitate the passing of information

d) receives interrupts from all devices attached to the system forming them into messages for interpretation by the respective device handler tasks.

Other system functions are carried out by standard tasks.

SCRIPT TASKS

Each script task emulates a set of up to 16 virtual users per task. There are usually several scripts in an ERTE configuration. At the moment up to 6 script tasks each supporting up to 16 users can be simulated. The

number of simulated users on ERTE is therefore restricted to 96 users.

The script tasks which simulate terminals, are controlled by a parameter file containing the number of users for this task, their typing speed and a delay factor between the initial user logon-times. A script file name and terminal line speed is also connected with each virtual user. For each simulated virtual user a sequence of sessions is defined. After completion of these sessions or scripts the script can be restarted and executed again.

The software structure of ERTE is shown in Figure III.1.

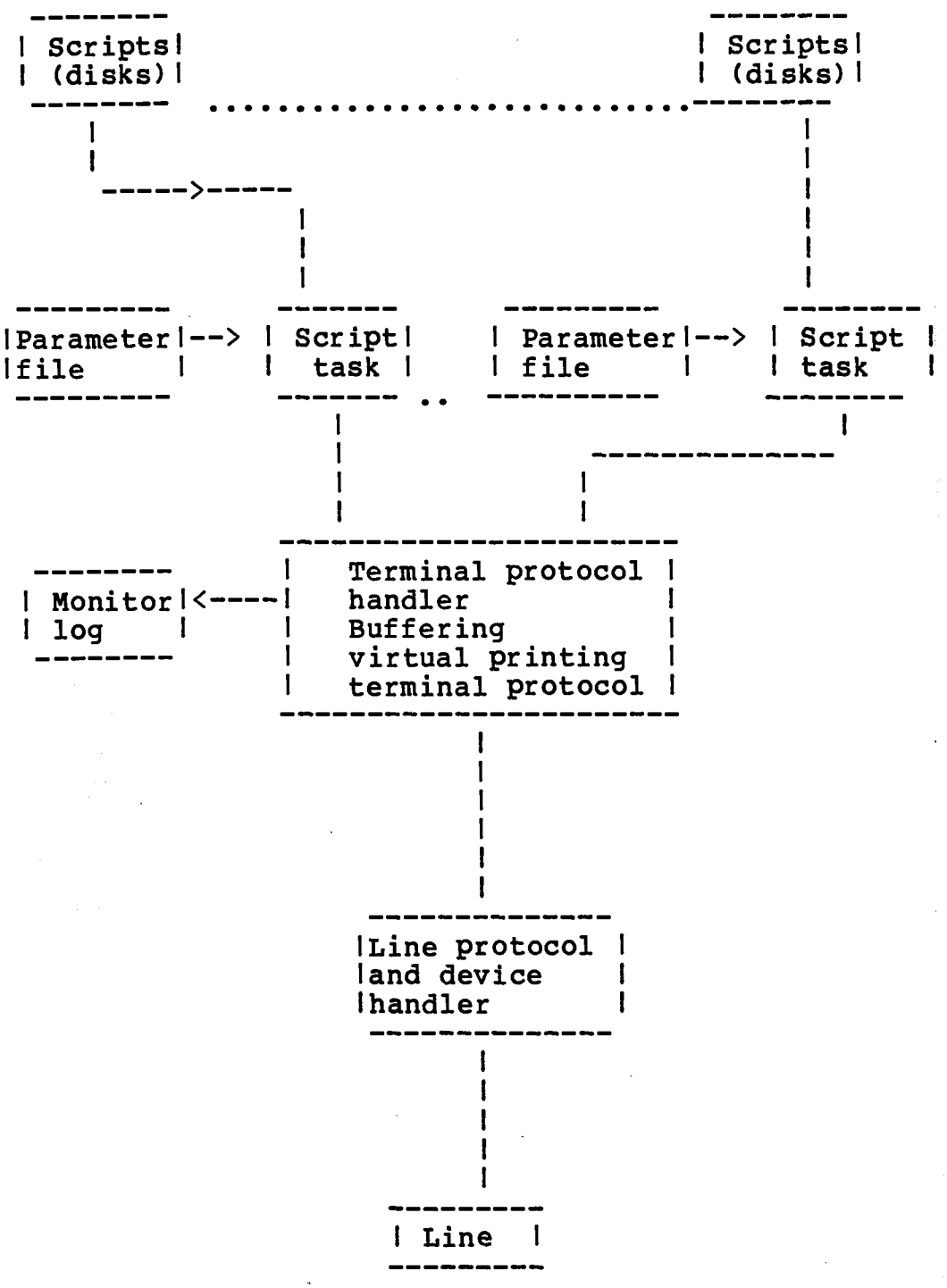

Figure III.1. Software structure of ERTE

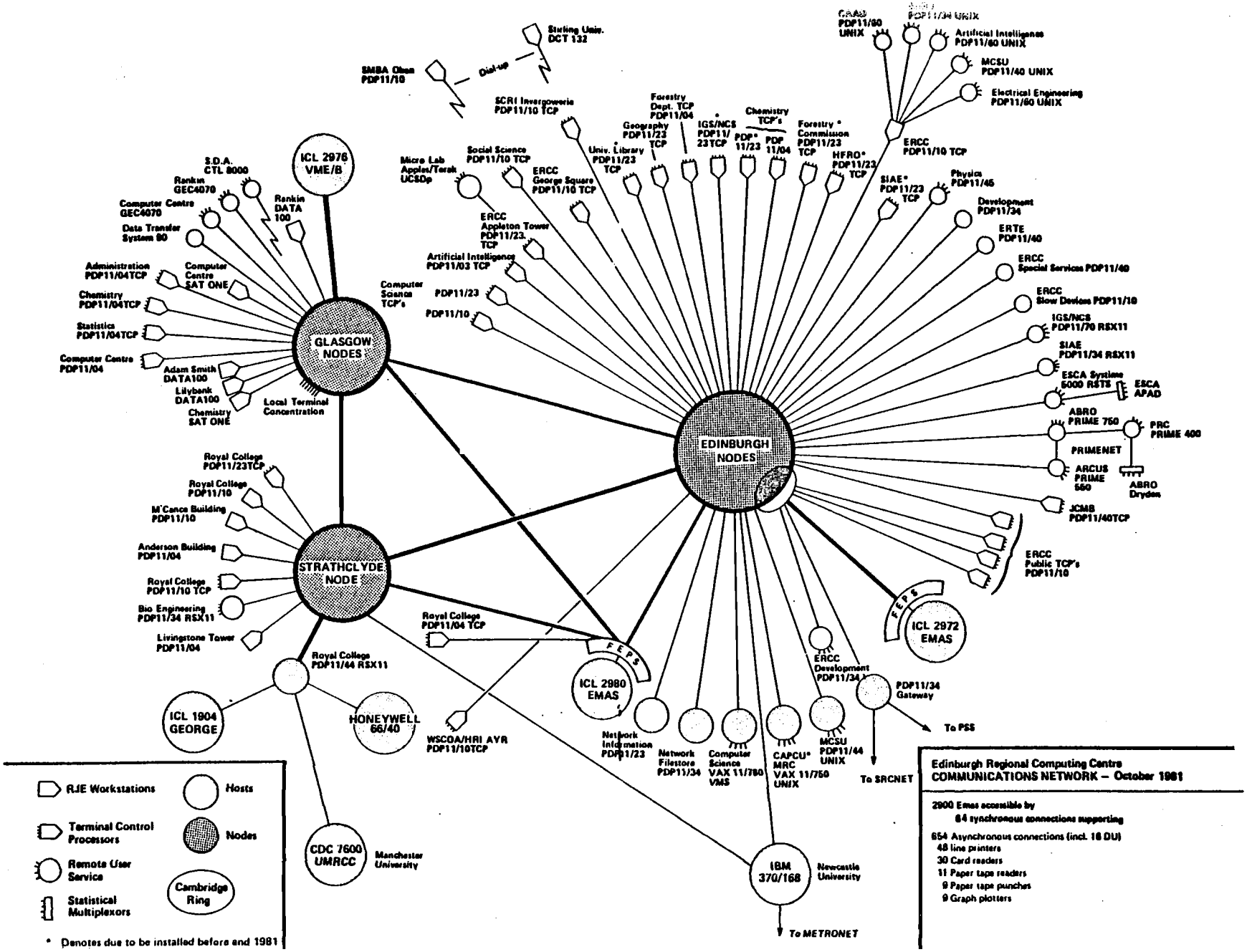
ICL 2976 VME/B
GLASGOW NODES
EDINBURGH NODES
STRATHCLYDE NODE
S.D.A. CTL 8000
Rankin GEC4070
Rankin DATA 100
Computer Centre GEC4070
Data Transfer System 80
Administration PDP11/04TCP
Computer Centre SAT ONE
Chemistry PDP11/04TCP
Statistics PDP11/04TCP
Computer Centre PDP11/04
Adam Smith DATA100
Lilybank DATA100
Chemistry SAT ONE
Local Terminal Concentration
Stirling Univ. DCT 132
Dial-up
SMBA Oban PDP11/10
SCRI Invergowrie PDP11/10 TCP
Micro Lab Apple/Terak UCSDp
Social Science PDP11/10 TCP
ERCC George Square PDP11/10 TCP
Geography PDP11/23 TCP
Univ. Library PDP11/23 TCP
Forestry Dept. TCP PDP11/04
IGS/NCS PDP11/23TCP
Chemistry TCP's
PDP* 11/23
PDP 11/04
Forestry * Commission PDP11/23 TCP
HFRO* PDP11/23 TCP
ERCC PDP11/10 TCP
GEAD PDP11/60 UNIX
PDP11/34 UNIX
Artificial Intelligence PDP11/60 UNIX
MCSU PDP11/40 UNIX
Electrical Engineering PDP11/60 UNIX
SIAE* PDP11/23 TCP
Physics PDP11/45
Development PDP11/34
ERTE PDP11/40
ERCC Special Services PDP11/40
ERCC Slow Devices PDP11/10
IGS/NCS PDP11/70 RSX11
SIAE PDP11/34 RSX11
ESCA Systime 6000 RSTS
ESCA APAD
ABRO PRIME 750
PRC PRIME 400
PRIMENET
ARCUS PRIME 550
ABRO Dryden
JCMB PDP11/40TCP
ERCC Public TCP's PDP11/10
Computer Science TCP's
PDP11/23
PDP11/10
Artificial Intelligence PDP11/03 TCP
ERCC Appleton Tower PDP11/23 TCP
Royal College PDP11/23TCP
Royal College PDP11/10
M'Cance Building PDP11/10
Anderson Building PDP11/04
Royal College PDP11/10 TCP
Bio Engineering PDP11/34 RSX11
Livingstone Tower PDP11/04
Royal College PDP11/44 RSX11
Royal College PDP11/04 TCP
WSCOA/HRI AYR PDP11/10TCP
ICL 1904 GEORGE
HONEYWELL 66/40
ICL 2980 EMAS
FEPS
ICL 2972 EMAS
ERCC Development PDP11/34
PDP11/34 Gateway
To PSS
To SRCNET
Network Information PDP11/23
Network Filestore PDP11/34
Computer Science VAX 11/780 VMS
CAPCU* MRC VAX 11/750 UNIX
MCSU PDP11/44 UNIX
CDC 7600 UMRCC
Manchester University
IBM 370/168
Newcastle University
To METRONET
RJE Workstations
Terminal Control Processors
Remote User Service
Statistical Multiplexors
Hosts
Nodes
Cambridge Ring
* Denotes due to be installed before end 1981
Edinburgh Regional Computing Centre
COMMUNICATIONS NETWORK – October 1981
2900 Emas accessible by
64 synchronous connections supporting
654 Asynchronous connections (incl. 18 DU)
48 line printers
30 Card readers
11 Paper tape readers
9 Paper tape punches
9 Graph plotters

Figure III.2

3. THE CURRENT SCRIPTS PRESENTED TO ERTE

Successful work with an RTE depends heavily on the workload model presented to the target system. The current workload model was constructed many years ago on a ICL 4/75 and there is no evidence that this workload model is any longer an accurate model of the real workload. The method applied to define these scripts is different from our method. Distributions of commands have been determined over a 24 hour period with an included, continuously busy period. It was found, that there have been discrepancies of the order of 1 per cent. It was thus decided to look at analyses of mid-afternoon sessions (2 hours, 3-5 p.m.).

To construct the scripts information was available for example on average number of concurrent users, CPU time spent in user process, number of input/output characters, total number of commands issued, average CPU time per command, typing rates, 'think times' (interval between receipts of an output message or prompt and input of the next command). The available information refers to a 2 hour period and no distinction was made between different user groups or types of sessions. The scipts were then defined individually, no automatic procedure was available.

The scripts now analysed with the algorithm described above shows that the session mix is different from what

we now find on EMAS (Table III.1). We do not find communication oriented sessions and file oriented sessions are represented by only 15% of all sessions. Edit oriented sessions are represented by only 25% of the sessions, which is roughly in accordance with our measurements for medium and expert users. For novice users, however, we find more than 40% of all sessions to be edit oriented. Unduly frequent are compile and compile/run oriented sessions which account for 15% of the observed sessions. Such sessions are only found with inexperienced novice users (about 5%).

The kinds of sessions we find are presented in Table III.1.

Figures III.3-III.11 show the distributions for CPU times, pageturns and execution time (i.e. length of edit sessions for edit commands) for different command groups. For the old ERTE scripts no distinction was made between different user groups. As the distributions for different user groups and commands differ in many cases a realistic user mix should be taken into account for a comparison between our data and the data we get from the ERTE scripts. Fortunately we find that the average values, distributions and execution times for the different commands groups are generally lower compared with our data. For edit sessions, for example, we find an average of 130 MS of CPU and around 468 PT per session. For our novice users

we find values of 286 MS of CPU and 1568 pageturns. For medium users the values are about realistic. For many of the other commands we find a similar effect: the ERTE model roughly matches the lowest average values and distributions found for our three user groups. This means that the general load imposed on EMAS via ERTE is too low and thus not representative of the real workload on EMAS.

One reason for not rewriting the scripts was that no reliable data and techniques were available to generate more accurate workload models. Other script models suggested in the literature [NOLA,SALT] are too crude and do not provide a picture of the various kinds of sessions that can be expected on a system. They do not seem to be based on sufficiently detailed measurements and there is no analysis of interactive user sessions either. Thus they are no real help in defining user scripts.

Another reason is that it is a very time-consuming task to work out and type in an ever increasing number of individual sessions. Thus, our aim is to write a program that generates the user scripts required as input for ERTE based on observations and requirements.

Table III.1. Groupings found for the standard ERTE Scripts

Membership %	General Comm %	Specific Comm %	Edit Comm %	Compile Comm %	Run Comm %	Other Comm %
10	16	30	1	7	3	43
	14	15	3	7	6	14
6	65	0	0	0	0	35
	8	0	0	0	2	9
7	7	17	0	4	25	47
	9	7	1	6	9	11
1	0	0	100	0	0	0
	0	0	0	0	0	0
6	9	6	64	11	3	8
	13	7	7	14	7	11
18	8	5	39	11	8	29
	11	6	7	9	8	16
29	5	14	19	16	14	31
	5	7	5	5	5	8
4	52	6	17	6	0	20
	12	8	6	6	3	9
4	3	0	22	32	7	37
	5	3	7	8	8	15
12	13	4	0	32	14	47
	13	7	1	10	11	12
1	0	0	0	50	50	0
	0	0	0	0	0	0
2	7	0	0	93	0	0
	9	0	3	10	3	0

Table III.1

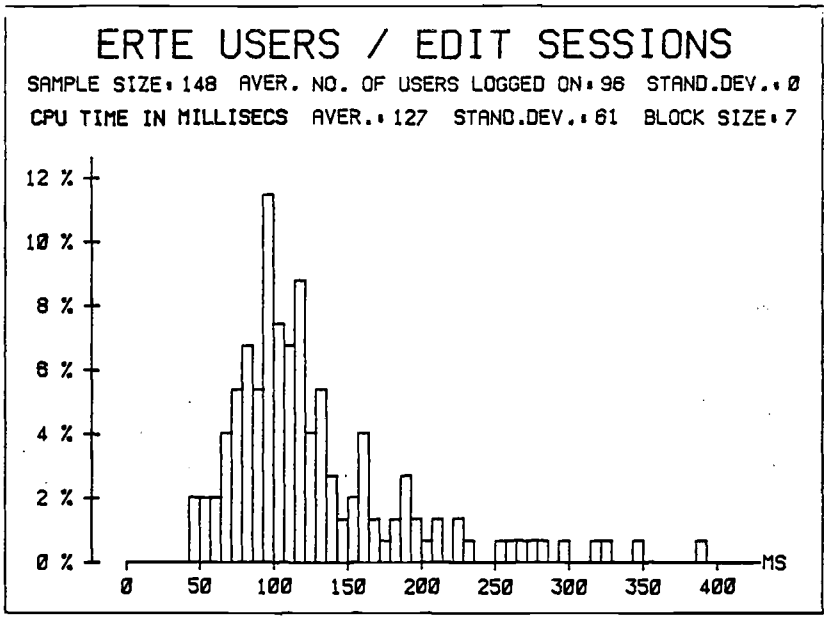

Figure III.3

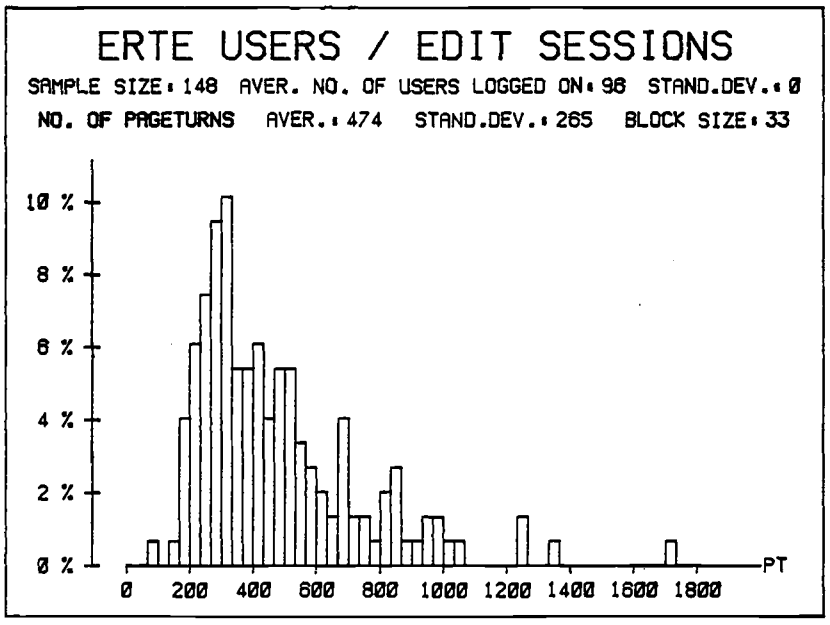

Figure III.4

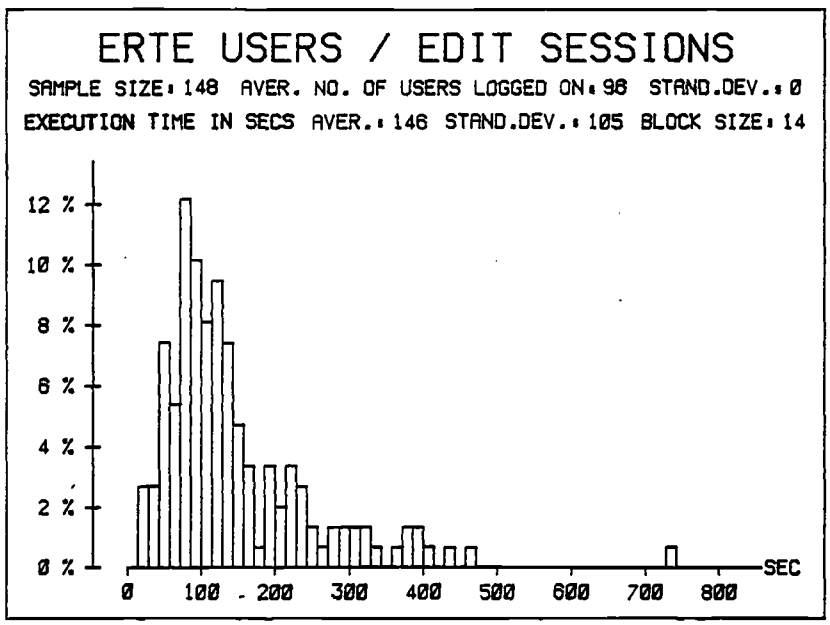

Figure III.5

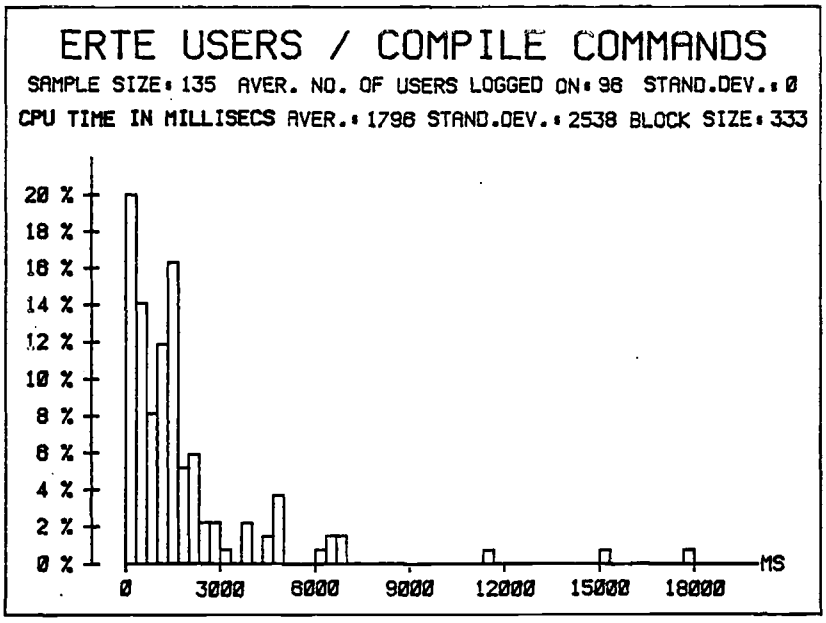

Figure III.6

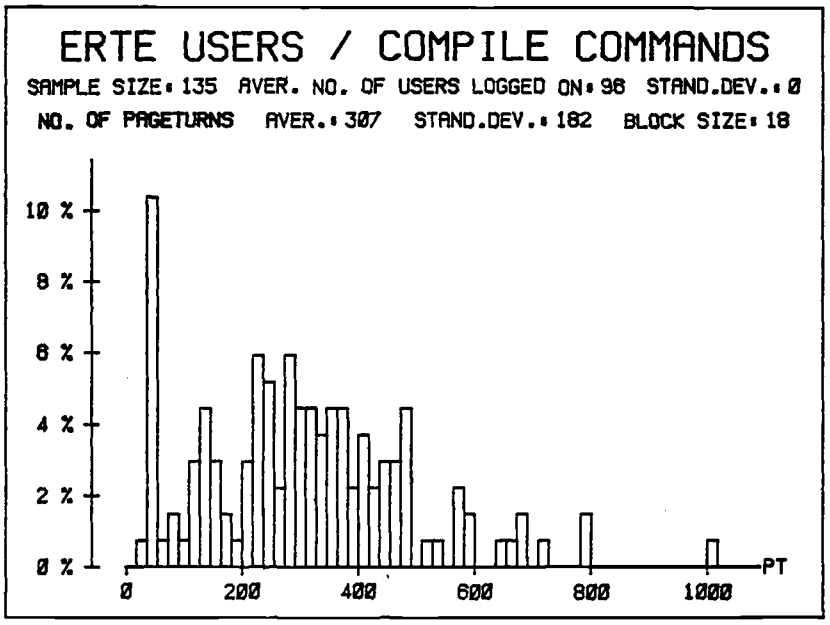

Figure III.7

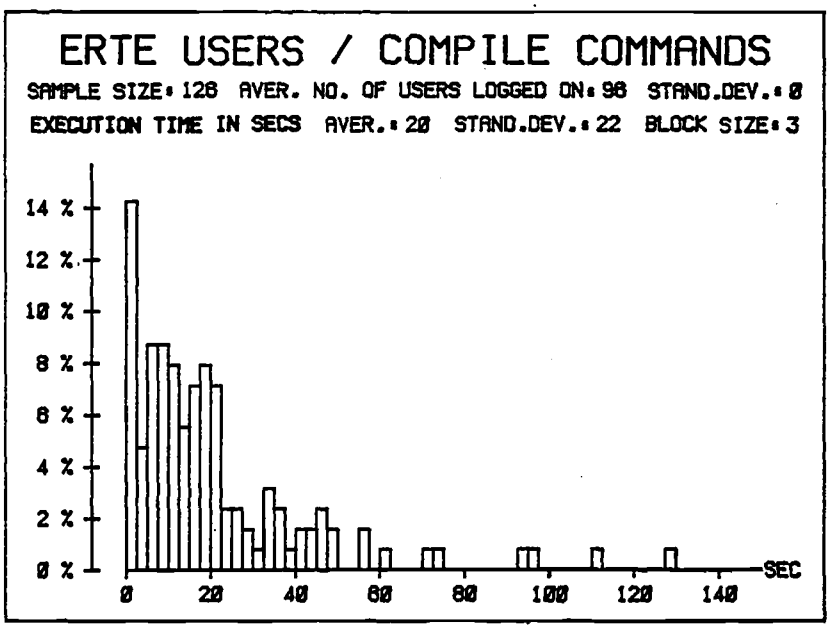

Figure III.8

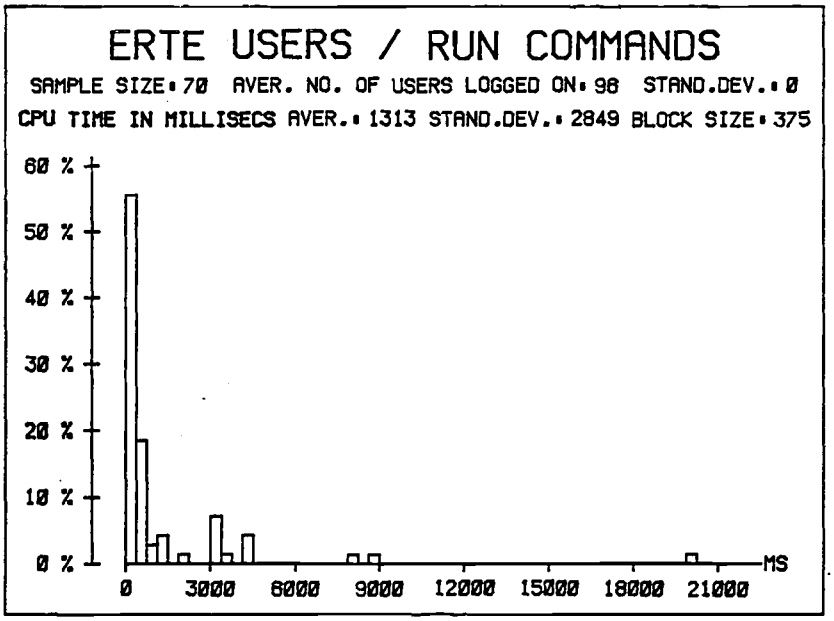

Figure III.9

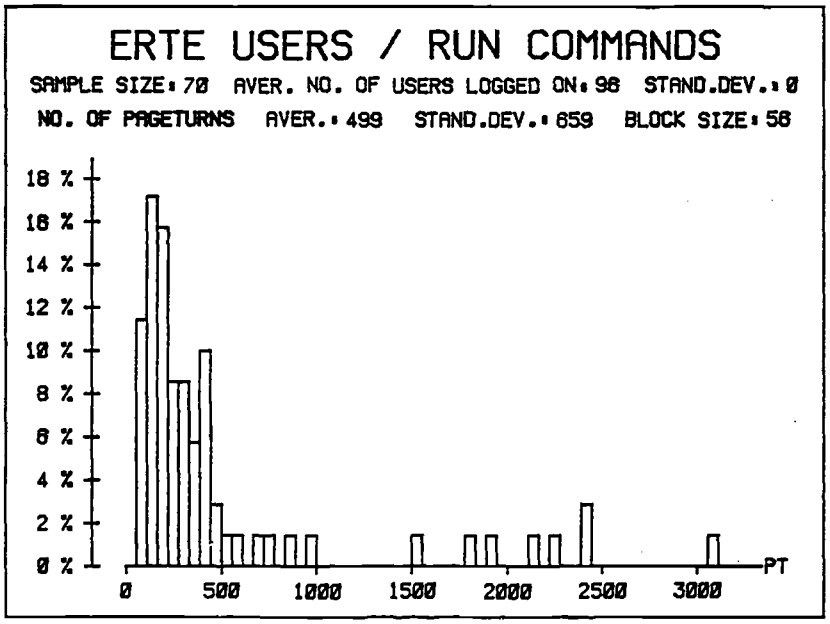

Figure III.10

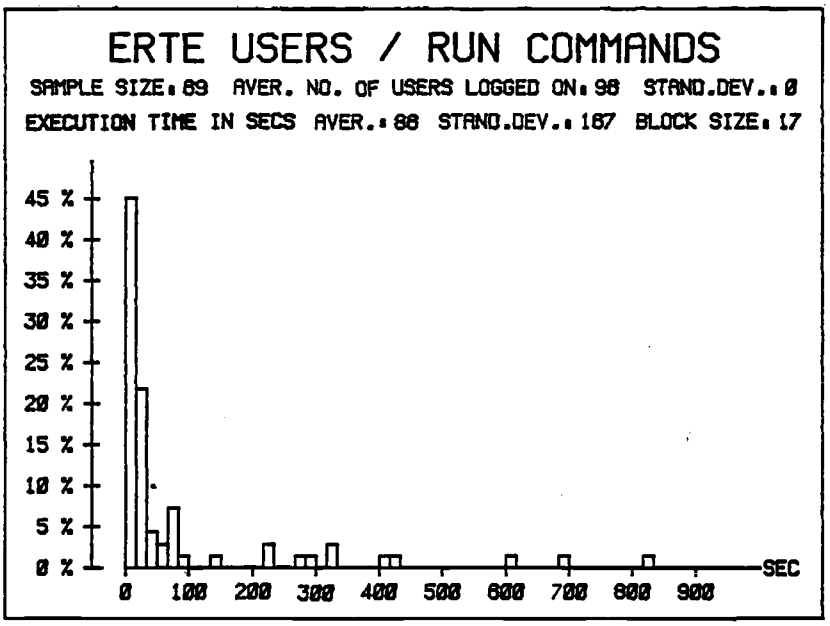

Figure III.11

Table III.2. Average values for thinktimes, execution times, CPU times and pageturns per command

ERTE SCRIPTS

Think time	31 16
execution time	24 79
CPU per command	511 708
Pageturns per command	260 194

Table III.2.

4. REQUIREMENTS OF AN AUTOMATIC WORKLOAD GENERATOR

Depending on the performance study the workload models presented to the target system vary significantly. In a tuning environment the workload model may be more stable, whereas in a system development phase a range of workloads is used to seek errors under the widest possible sets of conditions. In an experimental study the workload has to be changed if, for example, future workloads have to be presented to the target system.

To make the fullest use of an RTE in the most efficient way it is therefore desirable to provide facilities to generate the workload models desired for the different kinds of studies.

Therefore our automatic workload generator (AWG) should have the following characteristics:

1) It should generate automatically workload models which can be presented via ERTE to the target system without further modifications.

2) The data used as input for the automatic workload generator should be based on data from real workloads.

3) The automatic workload generator should be highly parameterized to make the generator as flexible and

tunable as possible.

4) The accuracy of the workload model should be demonstrated

5) The workload generator should be easy to use.

With such a workload generator it is possible to generate not only credible models but also many different kinds of models the user of this generator wants to generate. This may be of interest in experimental studies or in a development phase of a system. Our workload generator should take this into account and be able to produce different varieties of accurate workload models.

Parameters required by the ERTE

Basically ERTE requires descriptions of users' sessions. The parameters to describe these sessions are:

1) the users' think times

2) their typing efficiency

3) the commands and command sequences they issue at system-command-language level and in the editor

The automatic workload generator is designed to generate the commands and command sequences at both system command language and editor level.

5. INPUT PARAMETERS FOR THE AUTOMATIC WORKLOAD GENERATOR

The following parameters are needed as input for the workload generator.

5.1 General parameters for the workload generator

- The number of different user groups

Currently three different user groups can be generated, i.e. novice users, medium advanced users and expert users.

- The number of users for each user group to be generated

These numbers are quite important as they define the user mix working on the system. As mentioned above, the maximum number of users being simulated is currently restricted to 96.

- System command language commands

The third parameter is the group of system command language commands available on the system which ERTE is

driving, in our particular study experiments are performed with EMAS.

- EDIT commands

Next the commands of the EDITOR running on the system under study are provided.

These 4 different parameters apply for all users. The parameters described below have different values for different usergroups and are thus provided separately for the user groups to be generated.

5.2. User group specific parameters

- The command selection

Different user groups choose different kinds of commands. More advanced users tend to choose from a wider selection of commands whereas novice users stick to basic commands. We therefore provide probabilities with which commands are issued for system command language commands and edit commands.

- Thinktimes

Thinktimes also differ from usergroup to usergroup. We provide for each user group the distribution of thinktimes. These distributions are calculated from all commands and no distinction is made for different command groups.

- The session mix

The session mix indicates the percentage of the the different kinds of sessions for the particular usergroups. Each time a new session is to be generated, the AWG decides according to these percentages which kind of session is to be generated next. A session mix is provided for the use of system command language commands and edit sessions.

- Distributions of CPU time and pagefaults per run commands

If a run command is to be generated, the AWG defines the CPU time required and selects an appropriate procedure to run which produces this amount of CPU and pagefaults.

5.3 Session specific parameters

- Distribution of number of commands issued

The number of commands issued varies for different kinds of sessions and is therefore provided for each type on both system command language level and editor level.

- Transition probability matrix

A transition probability matrix determines which command is to be issued after having typed in the current one. Transition probability matrices are provided for session types at the sytem command language level and editor level.

6. GENERATING USER SESSIONS

To simulate the dynamic behaviour of user sessions we generate the user sessions with the help of an automatic workload generator. The probabilities of parameters are stored in vectors and matrices according to their frequency of occurence. For example, if we find that about 25% of all sessions are communication oriented (indicated by 1), we assign 25 times the value 1 to the vector containing the information about the session mix, 10 times the value 2 for file-oriented etc.. A random

number x between 1 and 100 is generated and picks the value at the x-th place in this vector. Depending on this number a communication oriented, file-oriented or edit-oriented session is generated.

Then the number of commands c for this type of session is defined in a similar way. The AWG then loops c times to generate c system-command-language commands. Before defining the actual command, the thinktime is calculated. In the case of an edit command the type of edit session and the number of commands for this type of edit session has to be defined. The edit command sequences are then calculated in the same way as for commands on the system command language level. For compile commands and run commands CPU times and pagefaults have to be defined to select a file from the file library.

After having generated c commands the AWG calculates the next type of session and proceeds in the same way as described above. At the moment ERTE allows up to 44 sessions to be generated for each user. When the required number of users and sessions have been generated for a particular user group, the AWG generates user sessions for the next user group in the same way with different probability values for the required parameters.

7. EVALUATIONS OF THE AUTOMATIC WORKLOAD GENERATOR

The automatic workload generator as presented above is a very helpful way of generating executable artificial workloads. Its major strength is the ability to generate a wide range of workloads for different kinds of studies: as outlined in previous chapters, the user population is flexible and quite versatile:

- there is a wide range from complete novice users to expert users
- novice users change from novice to expert users
- users exhibit different variations under heavy load
- the user mixes change

As user mixes, session mixes, distributions etc. can easily be changed the workload generator can generate the kind of session that is needed for the particular study. The CPU time needed to generate 288 session (3 sessions for each simulated user) is around 4 minutes on a VAX 11/780.

Another important aspect to mention is the fact that there are about 2400 users liable to use EMAS, around 1500 students and 1000 staff members. Every week about 5 new users get a user number, some users (figure unknown) give up their work on EMAS. From this big and changing user community we monitored about 200 users. Novice and medium advanced users were monitored for 4 weeks, expert users over a time period of several months.

Considering all these parameters we have to ask whether is is possible to generate credible interactive workloads. Let us first discuss the sample size of the user population.

Firstly, as already mentioned, it was not possible to monitor the whole user population, because the kind of data we needed for our study is strictly private and each user had to give permission to be monitored. It should be mentioned that many users do not like to be monitored for various reasons and finding an acceptable sample size was not easy. Besides this there was another reason to keep the number of users monitored as low as possible. On EMAS there is the option of giving ALIAS names to any command, and users do use this option. To group the commands found it was necessary to know and ask the users to clarify the meaning of their commands.

Because of these limitations we had to select the users monitored carefully to find a representative picture of the user population. The users we monitored do work regularly on EMAS. The students write programs for their practical assignments on EMAS and thus did intensive work on EMAS. Under heavy load (around 100 users logged on) most of the users are students and thus they do contribute significantly to the overall load. The expert users we chose were also working regularly on EMAS, most of them do logon at least twice a week, some

of them daily.

For these reasons we think that the sample of users monitored does give a representative picture of the user community. This question is important as the workload we generate simulates 96 users working simultaneously on EMAS.

It may still be argued whether it is justified to model interactive workloads in the way we suggest because the kind of workload generated for a thirty minute run on ERTE may never exist exactly like that in reality. In this connection we can ask the question if a workload model would be more credible if we had monitored and modelled a user community for a whole day or part of a day (which was not possible). This question, too, cannot be comprehensively answered. As we emphasized several times users do change (novices become expert users), users write different programs etc.. These factors change the kind of workload imposed on the system. Let us also discuss the question whether monitoring at so-called 'representative' hours would provide more credible data. A representative hour might be between 2 and 3 o'clock in the afternoon sometimes during mid-term. The question arises whether these 'representative' hours always have stable, similar workloads.

To answer this question we again look at the user community during term time. At that time many student users who are working on EMAS have deadlines for their programs. Deadlines are different for different user groups and just before deadlines users tend to use a machine quite extensively. Different user groups have different programs, different demands on resources and thus at two 'representative' hours at different days we may well find different workloads. To find this out, frequent measurements are necessary, in many cases this is difficult to perform.

Another point to mention is that measuring and modelling is a labourious task and the validated and calibrated model is rarely available a couple of days later. In the meantime new software packages or a user group using a resource intensive program package may be found on the system. Again we find a different workload which may be very different from the workload just being modelled. Even if we modelled a one hour workload as closely as possible on one real workload, there would be no guarantee that this kind of workload would still occur frequently or at all the same day, week or month later.

This means any workload model is an approximation of the real workload currently found on a system. In this context we have to ask to what degree it is possible to generate 'accurate' or 'credible' workloads. In chapter

I we gave several definitions for accuracy of workload models. To validate our model generated by the automatic workload generator we are going to apply a mixture of all three definitions.

- CPU-times and Pageturns found for each command issued are compared with those ones found for real users.
- Functions in terms of commands are compared with those ones found for real users.
- Execution times are compared with execution times for real users.

Applying these three definitions we can only validate our model against the values we measured for our sample size and extrapolate it to the whole user community. Models for a whole user community based on data from a whole user community can only be validated for this day or period of time and are assumed to be valid on different days. This, too, is doubtful. On the other hand we discussed in chapter II the fact that users do exhibit different variations under different loads and we can suspect that there is a relationship between performance and user activity which yields a varying, but stable load.

To mimic this is, of course, beyond the reach of the automatic workload generator. In this sense, it is not possible to mimic users' activity. The target of the workload generator is to generate an accurate (with

restrictions) , stable, repeatable load for experiments where a realistic workload is needed and a stable, repeatable predictive load for experimental studies. Under this aspect the automatic workload generator is a very powerful tool for performance evaluation studies.

The first target is to generate a workload which yields similar execution time, CPU time and pageturn distributions for various command groups as we found for real users. The validity - with restrictions outlined above - of the the model is demonstrated in chapter IV. After that we generate two predictive workload models and compare them with the results for our 'realistic' interactive workload model.

C H A P T E R IV

IV. EXPERIMENTAL RESULTS

In this chapter we validate and study interactive workload models generated by the automatic workload generator. We demonstrate how user mixes and outliers influence the system's performance in terms of elapsed execution time for various commands. A workload model is presented that matches in a satisfactory way the values we found for real users. This workload model may be used as a standard model for EMAS. Experimental runs are presented to demonstrate how different user mixes and outlier sessions, i.e. users sessions with jobs requiring large amounts of CPU time or generating large numbers of pageturns, affect execution times.

1. MODEL VALIDATION

Validating and calibrating interactive workload models for time-sharing systems is a very tedious and tricky enterprise. Difficulties with validating models have been described in chapter I. The first problem we are confronted with is the question whether we can simulate the real world without exceptions or whether we have to make compromises.

It is obvious that such a large and diverse user community as we find on EMAS cannot be simulated in every detail. To keep a model concise and manageable it is necessary to make compromises. We now outline the major aspects where we do not (to keep the model sufficiently compact and manageable) or cannot (because of lack of data) simulate interactive workloads in every detail.

- The user mix

The user mix on EMAS varies widely over the year, because of changes in user characteristics (novice users do become more experienced, vacations, deadlines for undergraduate users etc.). The actual user mix during the measurements was not known, and we therefore have to make approximations and guesses based on sporadic checks. On EMAS a command called NEWUSERS lists the user numbers currently logged on to the system.

- The session mix

There are many different kinds of sessions to be found on EMAS (see Chapter II). To keep our model concise and easy to handle we decided to restrict the number of different kinds of sessions to be generated to 6, namely

a) communication oriented sessions
b) file oriented sessions
c) edit oriented sessions
d) edit-compile oriented sessions
e) edit-compile-run oriented sessions
f) run oriented sessions

- File library

To simulate interactive workloads we have to provide the synthetic users with a collection of programs and files. As we are simulating 96 users it is desirable (from a computing centre's point of view) to keep the amount of store used by these users as small as possible. Another option is to give as many users as possible the same files and programs, although different users or usergroups need different files and programs. Novice and medium users have the same files and can thus behave like novice and medium users. If another user mix is to be simulated we do not have to assign different files to the artificial user processes every time we want to change the user mix. To do this we also need a dedicated 2972 and dedicated machine time should be kept as low as possible.

The real user community has a much wider variety of files. This is reflected in the distributions for CPU times or pageturns for run and compile commands, which do not have many gaps but are quite smooth. For our purpose we wrote synthetic programs that the artificial users can work with and some files and programs were provided by the users we monitored. A first idea was to collect programs from real users, but this proved to be more difficult than expected. Users do not necessarily like to give their programs away and if so, they did not do what we were looking for.

With a limited number of files and programs the smooth distributions could not be simulated, the synthetic

programs used tend to approximate the distributions but they do have gaps. How we cope with these 'inaccuracies' when validating and calibrating the model will be described later in this chapter.

- Edit sessions

Editing is quite a major activitiy of the user community. On EMAS a user can choose between many different editors. From a modeller's point of view, this too, is quite an unfortunate feature, because editors have different performance characteristics and influences on the system's overall performance are not known. However, very popular editors are ecce,vecce and edit. Currently implemented for the workload generator is ecce and vecce.

- Terminal characteristics

Another problem is the terminals. EMAS can be accessed via videos or hardcopy terminals. It is not known from which terminals the users logged on and what the baud rate was that they used (this is assumed to be 1200 in most cases). It is also not known whether and how much different terminals influence users' behaviour and the load imposed on the system.

- Long sessions

For ERTE runs dedicated machine time should be kept as short as possible. Long sessions at system-command-language level or at the editor level are therefore not worth modelling, as they would not finish during the normal run.

For our model validation and calibration we first compare CPU time, pageturns and execution time distributions for Compile and Run commands. The comparison is done in the following way. We take the distributions of the original user data and the synthetic user data and order them according to size. These arrays are then normalized and the differences in per cent for corresponding array entries are calculated. The differences are shown in diagrams and average values for differences are provided.

The next question is "when do we consider the model 'accurate' ?". This question is difficult to answer because no generally accepted answer can be given and measures suggested are to a certain degree subjective. Our approach to this problem is the following:

We consider the model satisfactory if the experimental data differ from the original data on average less then 25%. This 25% needs some more explanation. Because of all the restrictions described above it is not possible to match all parameters and deviations need to be accepted. As we are interested in a performance of a system our main attention is drawn to execution times of various command groups.

Our attempt to match CPU times and pageturns was in most cases successful but there are examples where we deviate from the original data for more than 30%, for instance for run commands of the novice users. However,

novice users do not need many pageturns for run commands and 30% changes (in our case 30% more!) does only account for a very small percentage of the overall load. Furthermore, pageturns are dependent on the actual scheduling strategies and it was decided not to overvalue the number of pageturns if the workload imposed on the system was small.

For the model validation we define a user mix consisting of 40 novice, 30 medium and 26 expert users. Our novice users were monitored at the beginning of an academic year when many novice users were logged on to the system. This mix seems a good approximation to our sporadic checks. The model is validated against novice users as the sample size for this user group under heavy load was the biggest. For this validation we select novice user sessions with 92 to 100 users logged on to the system.

For an ERTE run we need dedicated machine time; ERTE runs are therefore done during the night. For these experiments we have to set up a comparable environment to that during our measurements. During our measurements a Distributed Array Processor was not installed but is in use now, so the 2972 had to be reconfigured and used without this DAP. Our model users also had to use the modified standard subsystem with the extra monitoring information. The interactive terminal activities of the model users were also written to a circular 64 KB file. These data are then analysed and compared with those we found for

real users.

The ERTE runs lasted on the whole for 45 minutes. During the first 5 minutes users log on to the system and this time is not included in the analysis.

The following six figures show the changes of the distributions in per cent for compile and run commands. The dotted function gives a cumulative frequency to indicate the base to which absolute values these deviations in per cent refer. Execution times for all commands do not include execution times for edit sessions as those values tend to be considerably longer than for the rest of the system-command-language commands. The actual work done for real novice users was on average 0.44 commands per minute. Our model novice users finished 0.49 commands per minute. These values include edit sessions. The diagrams show that the distributions of execution times found for real users are simulated. In some cases exceptional values have been scrapped because these values would have changed the average value signifcantly. We consider this model an accurate match of the real workload and take this model as a basis for comparison with the next experimental run.

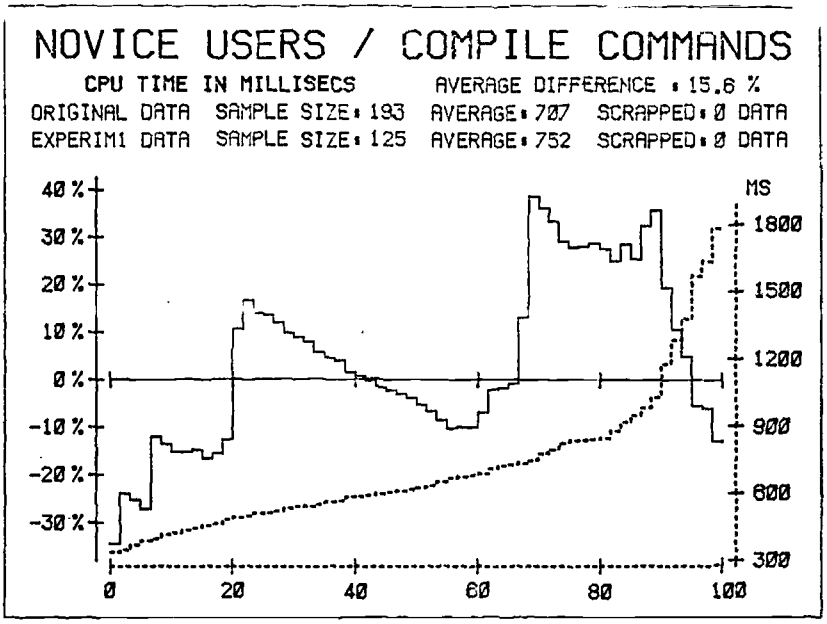

Figure IV.1

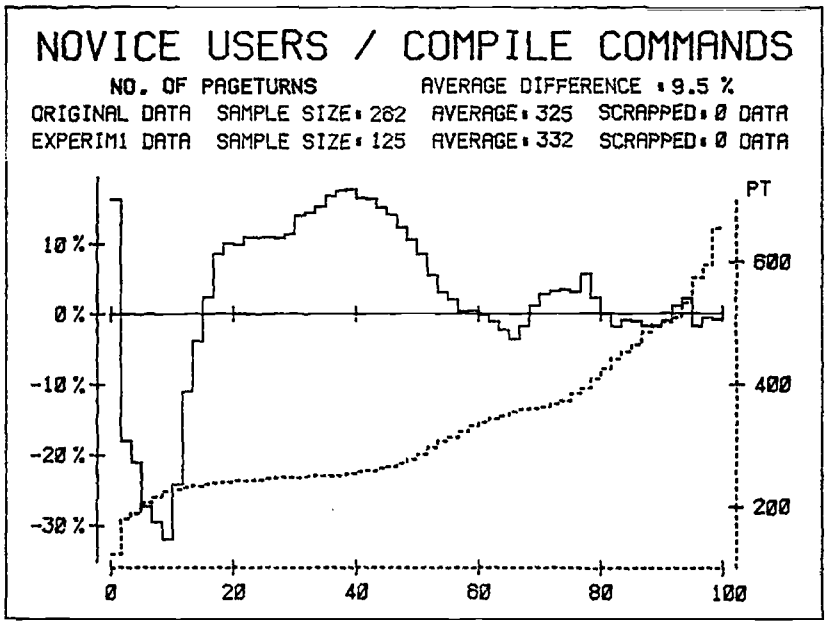

Figure IV.2

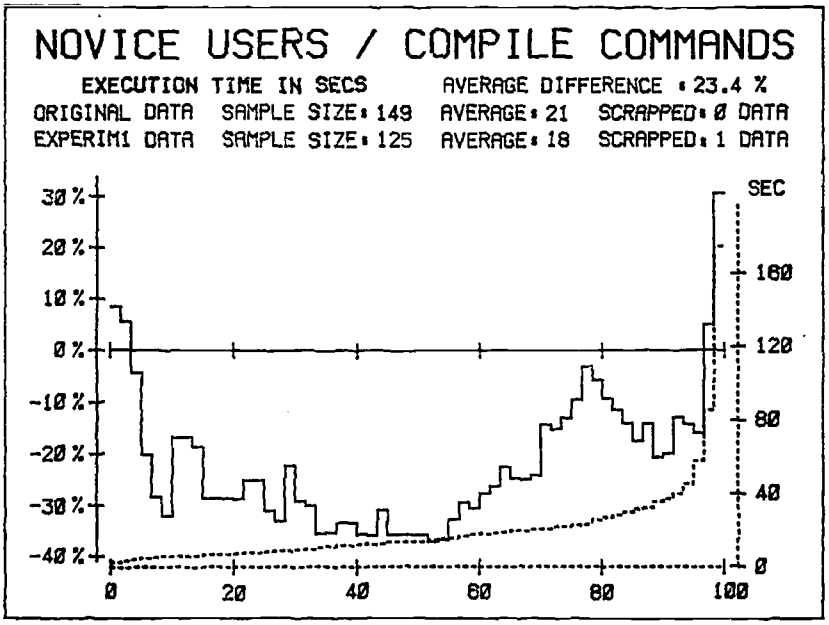

Figure IV.3

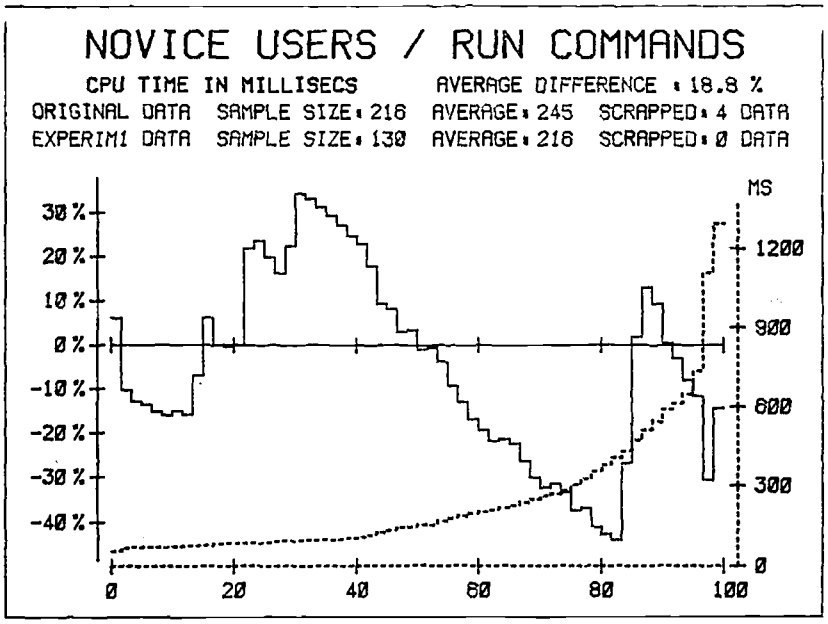

Figure IV.4

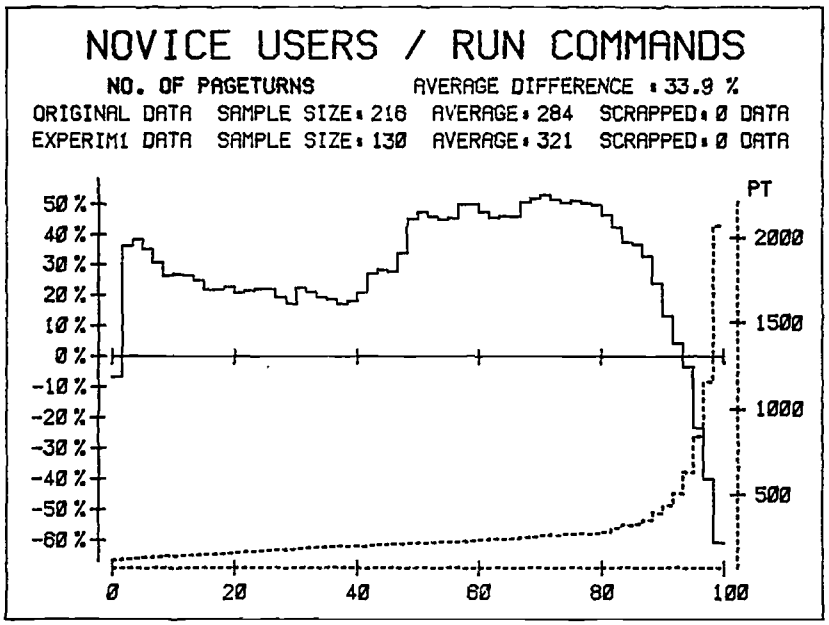

Figure IV.5

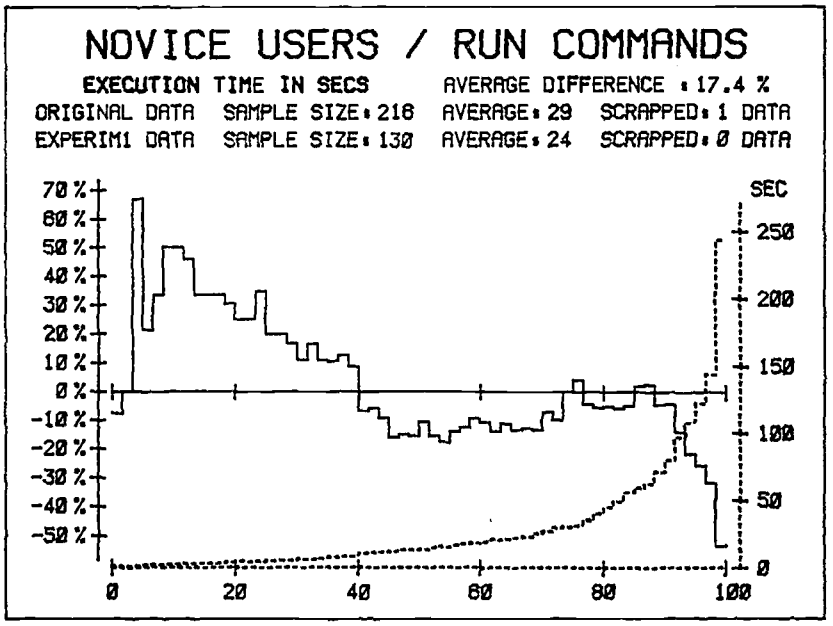

Figure IV.6

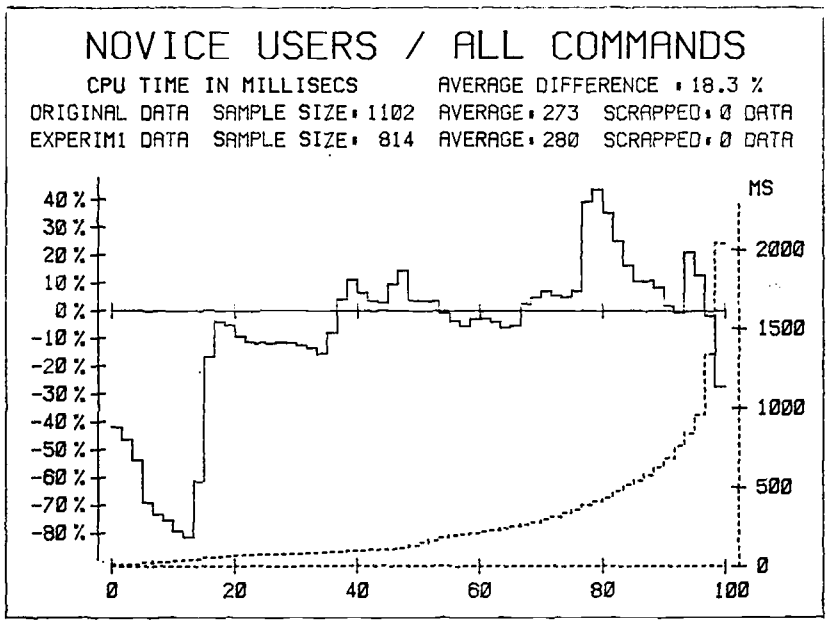

Figure IV.7

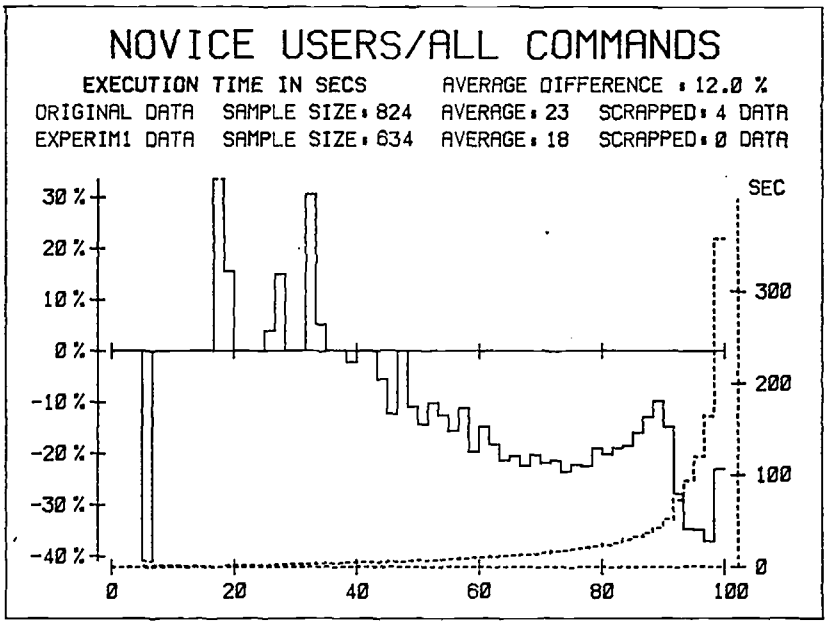

Figure IV.8

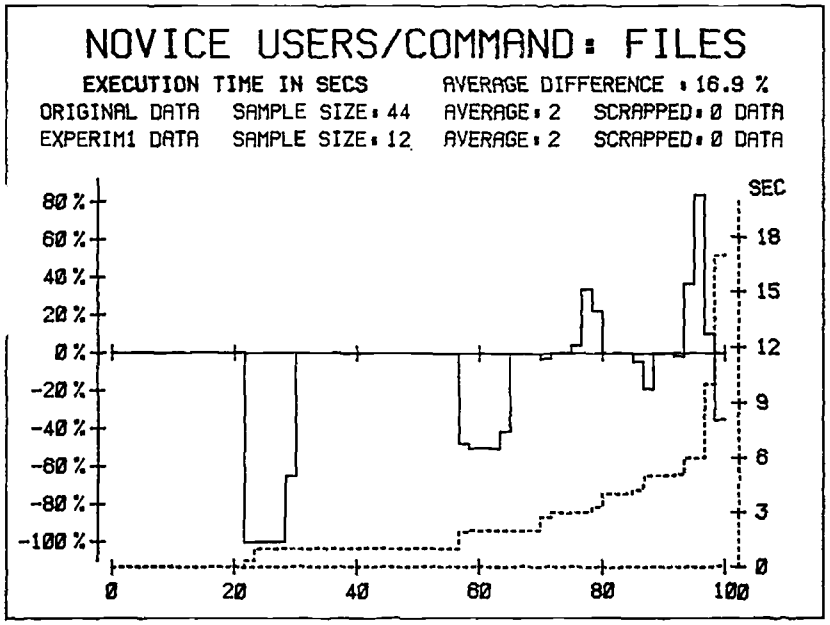

Figure IV.9

2. EXPERIMENTAL RUNS

To demonstrate the applicability of the automatic workload generator we now describe two experimental runs and study how execution times are affected if we change the user mix and introduce outlier sessions.

2.1 Changing the user mix

On EMAS the user mix changes over the year. At the beginning of an academic year we find many 1-st year students working on EMAS - so called novice users. In the course of the year these users change and become medium users and thus change the load imposed on the system which in turn might affect execution times.

To find out how a different user mix affects execution times we simulate 16 novice, 54 medium and 26 expert users. The rest of the experimental environment is unchanged. We now compare the medium user group of experimental run 1 with the medium user group of experimental run 2. Medium users have more resource intensive run and compile commands and execution times for these command groups go up while CPU-time and pageturn distributions stay rather constant (Figure IV.10-18). The average time for file commands increases drastically, but this is caused by LIST commands to send output to a terminal; I/O intensive work is slowed down and I/O capacities are apparently overloaded. Execution times for destroy, copy or rename commands were found to be quite stable.

However, the execution times lie below the real observations (see Figures II.72-74). This phenomenon can be explained by the fact that medium users also have more communication and file oriented sessions which do not impose a heavy load on the system.

Another explanation may be that at the end of the year users may well have tight time-limits and may be under pressure to get their work done, which might cause them to submit heavy run jobs from an interactive terminal.

Our next experimental run studies the effect of 'outlier users'. Expert users we monitored tend to avoid busy periods, because we found smaller numbers of logons for very busy periods. For novice users we found about the same sample size under any load. From this observation we may conclude that during busy periods more inexperienced users logon to EMAS.

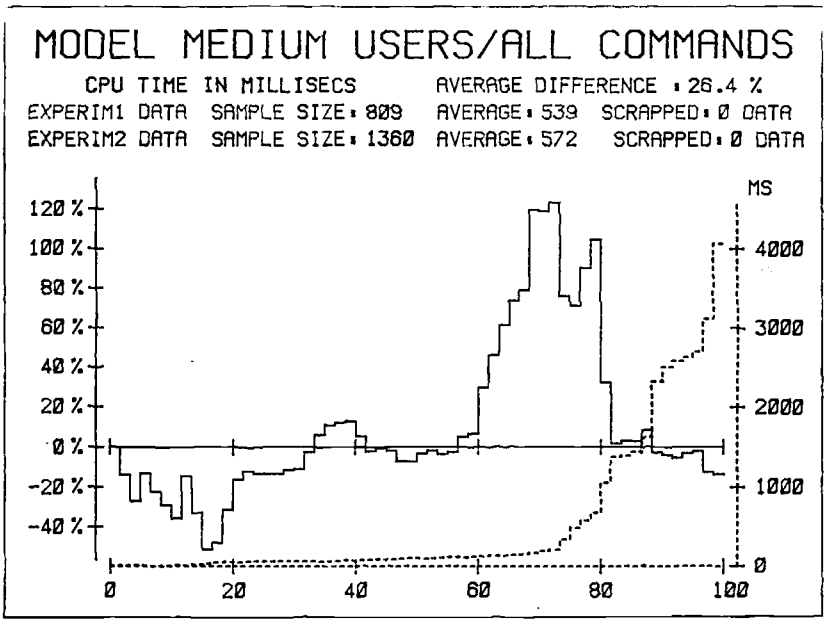

Figure IV.10

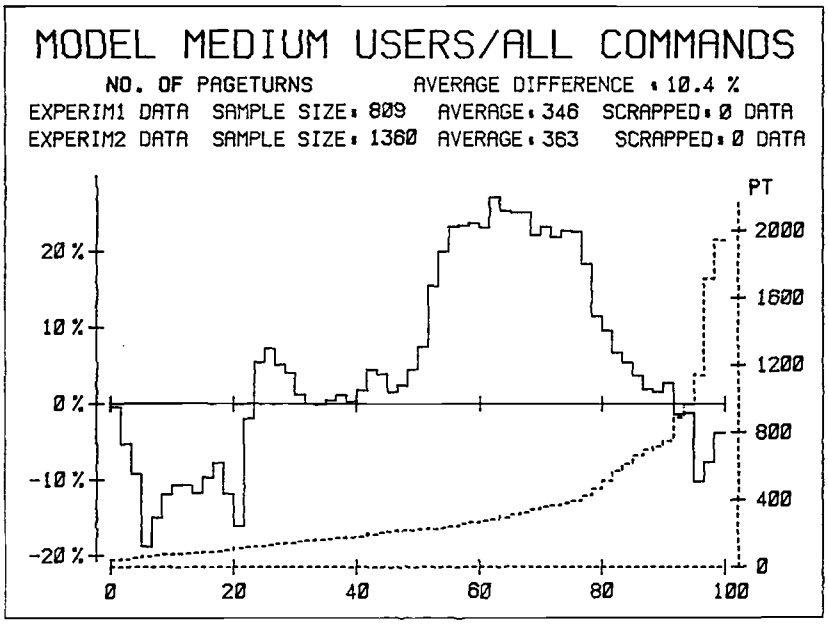

Figure IV.11

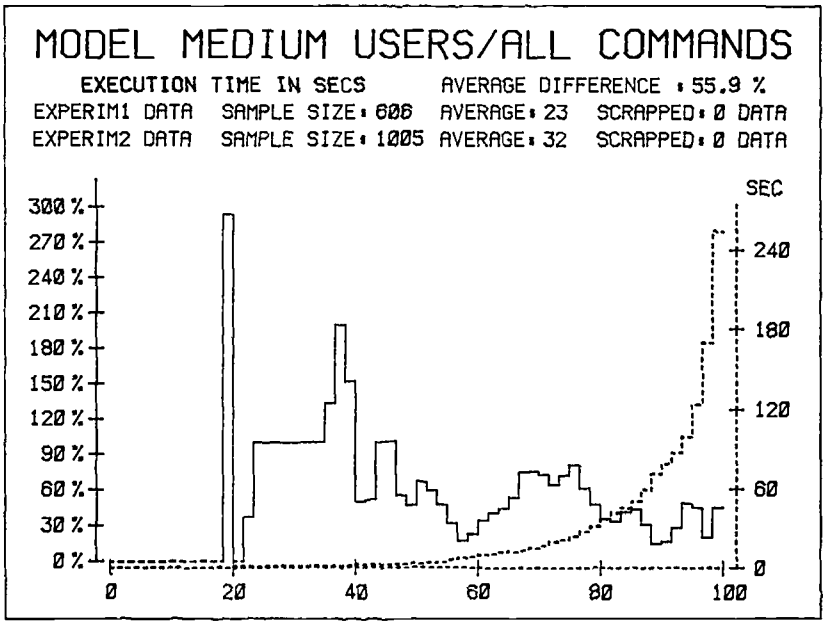

Figure IV.12

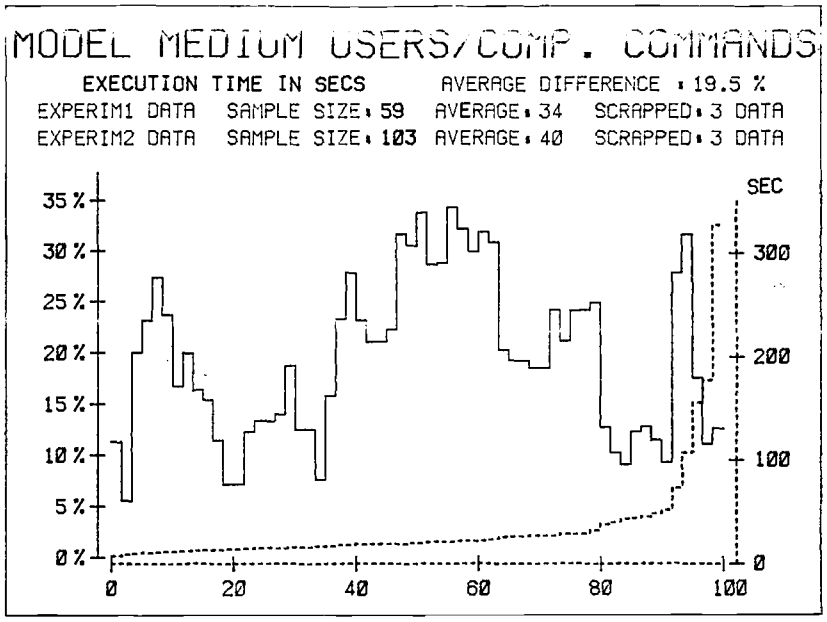

Figure IV.13

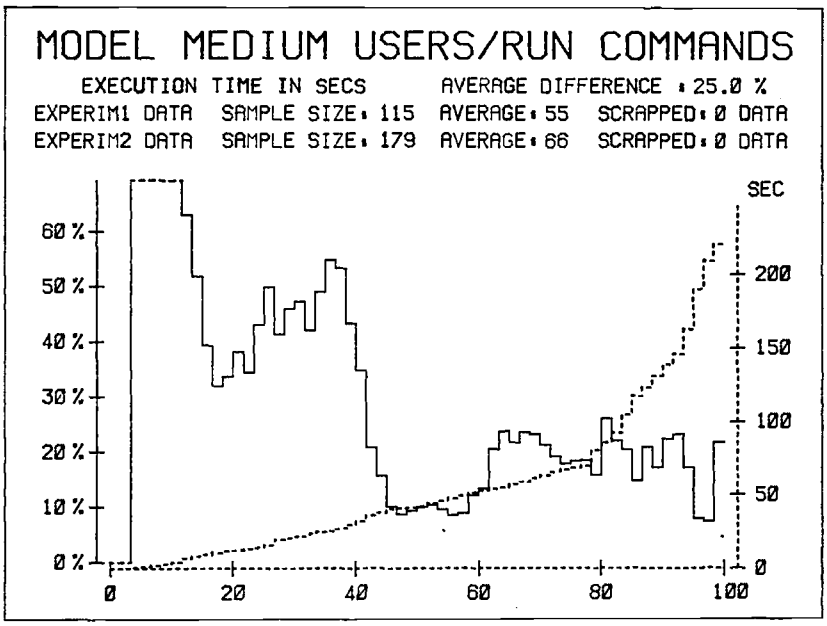

Figure IV.14

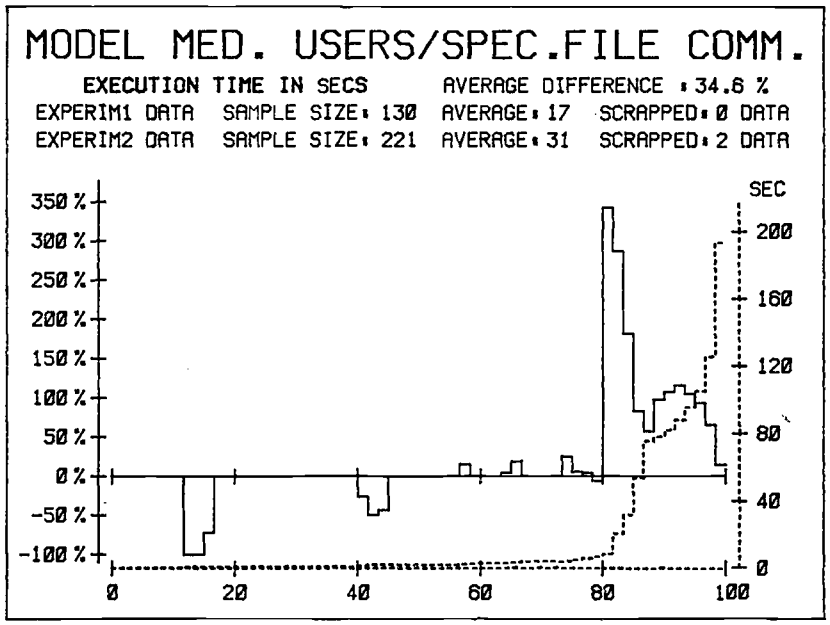

Figure IV.15

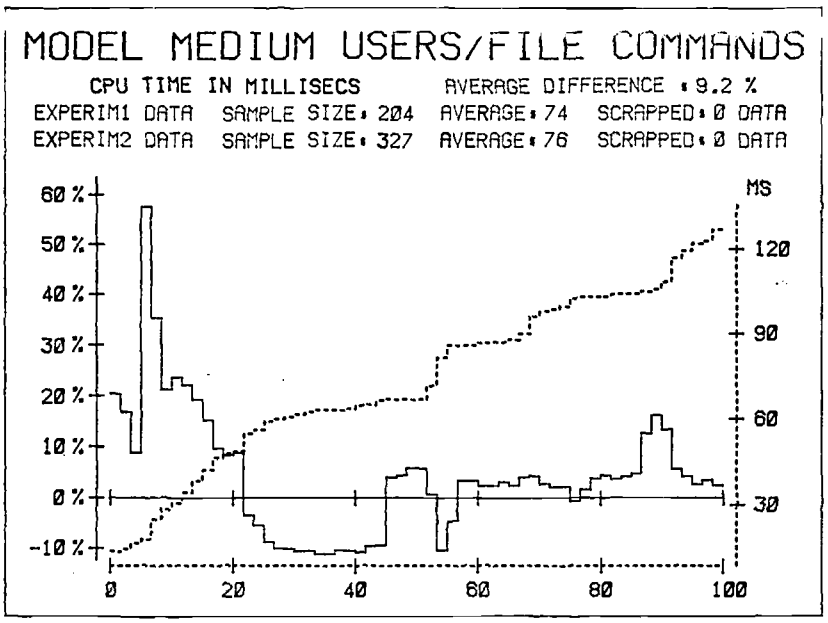

Figure IV.16

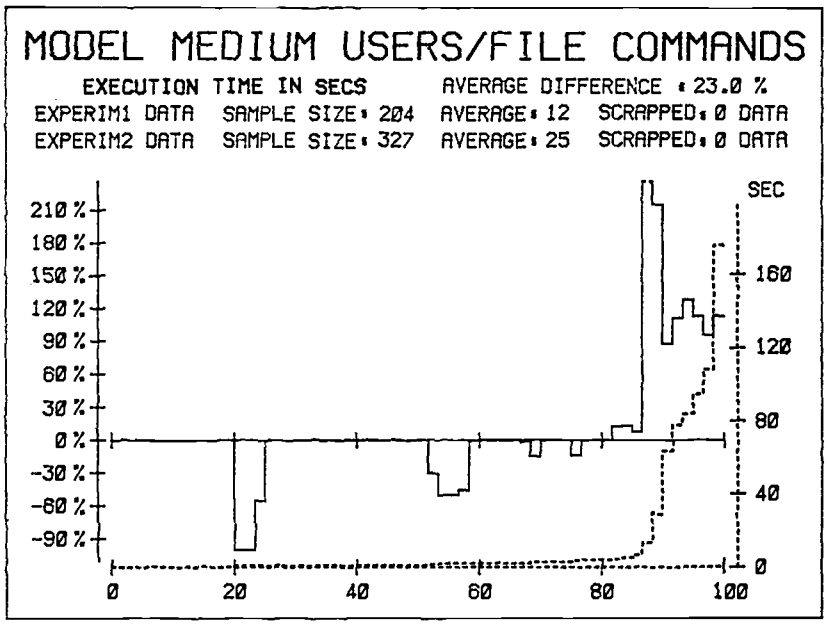

Figure IV.17

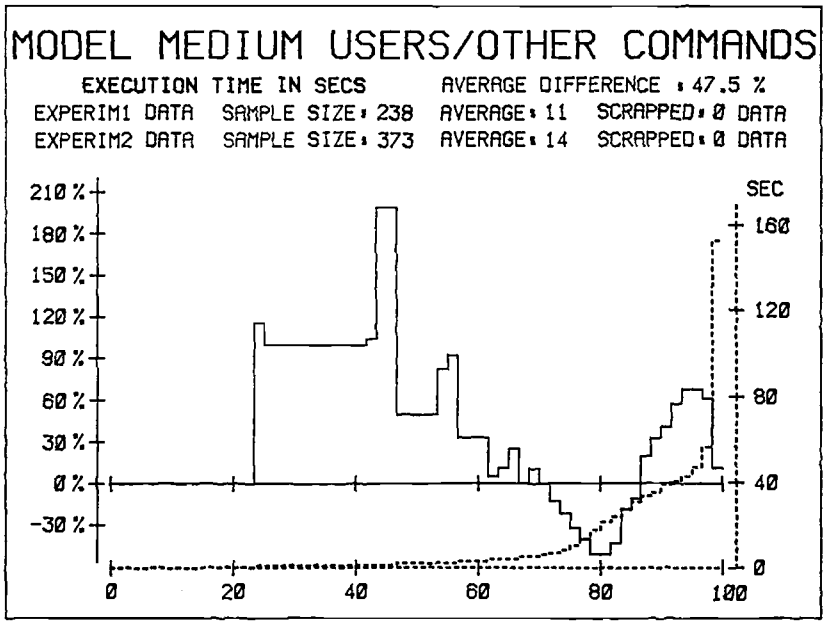

Figure IV.18

1.2 Introducing outlier sessions

To find out what effect outliers have on execution times we replace for the second experimental run two expert users by two 'outlier' users. These outlier users constantly run programs and do not do anything else. The run programs use between 1600 and 7000 MS of CPU time and produce between 1100 and 2500 pageturns per run. The rest of the artificial users remain unchanged. We again compare the model medium users of experimental run 1 with the model medium users of experimental run 2 because we have the biggest sample size of this user group.

We find that execution times increase between 20 and 30 percent for all command groups (Figures IV.19-27). The command TELL is used to send another user a message. Figure IV.27 shows average execution times for edit sessions which is fairly constant. The slightly lower execution time for the third run may be explained by the fact that we have a slightly lower sample size and some of the longer edit sessions could not be finished and thus reduce the average values for the execution times.

This experiment shows that users who have highly run oriented sessions can affect execution times. On a system like EMAS with around 100 users logged on such outliers have to be taken into account. For tuning a system effects of outliers are important, because dependent on the use of a system one might want to provide better service for users who submit small jobs. In this case outlier sessions should not affect other users. But it

should be pointed out that in our experiment the editor remained responsive and was apparently not affected by the outliers. As edit sessions are more time-consuming the work users get done is therefore only reduced by a few per cents.

However, in our experiment these outliers were present all the time, which may not be realistic for a real workload. In a real workload several outliers may occur at different intervals more frequently causing the system to slow down. If we study execution times of real novice or medium users, we find values way outside the average values for 'small' run or compile commands. These values may well be caused by several outliers being present at that particular time in the system.

Concerning the user mix on EMAS at the end of an academic year we may assume that the user mix does change and thus the load imposed on the system. The presence and absence of outlier sessions or jobs is an important influence on a system's performance and should be taken into account for a realistic workload characterization study. However, it is not possible to decide precisely what kind of user mix is present under heavy loads at different times. Even if we had monitored a whole user community for some time interval under heavy load this would not allow any general conclusions on user mixes. User mixes may change daily, and extreme outliers present on the one day may not be found at another day. Different deadlines for different user groups may also influence

mixes and performances, and these things, too, change quickly.

These experiments are only a selection of a wide range of possible runs. But these experiments demonstrate how changes in a workload can affect the execution time of various command groups and commands. For tuning a system or changing configurations experimental runs with different workloads are an invaluable help to find improved settings or to decide whether the changes will bring the desired effect. The automatic workload generator and an RTE are invaluable tools for this purpose.

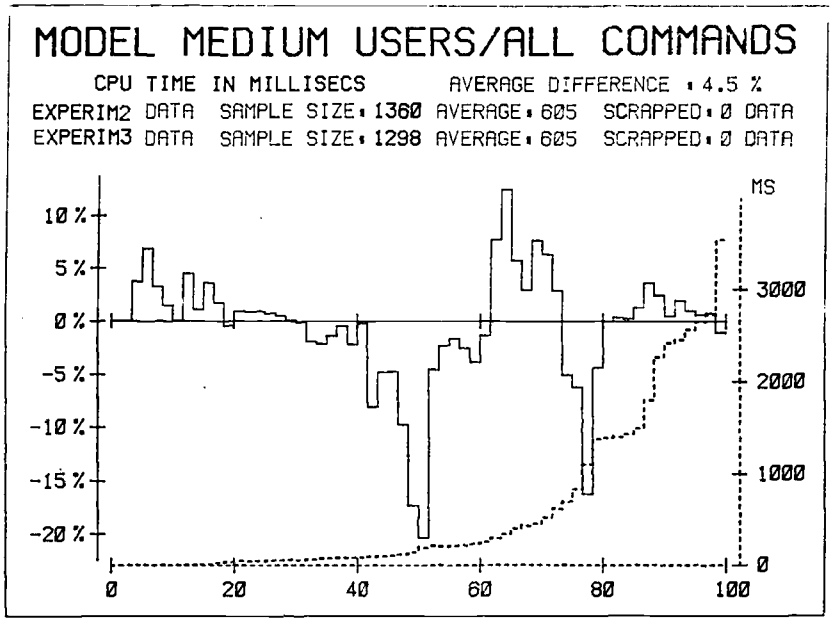

Figure IV.19

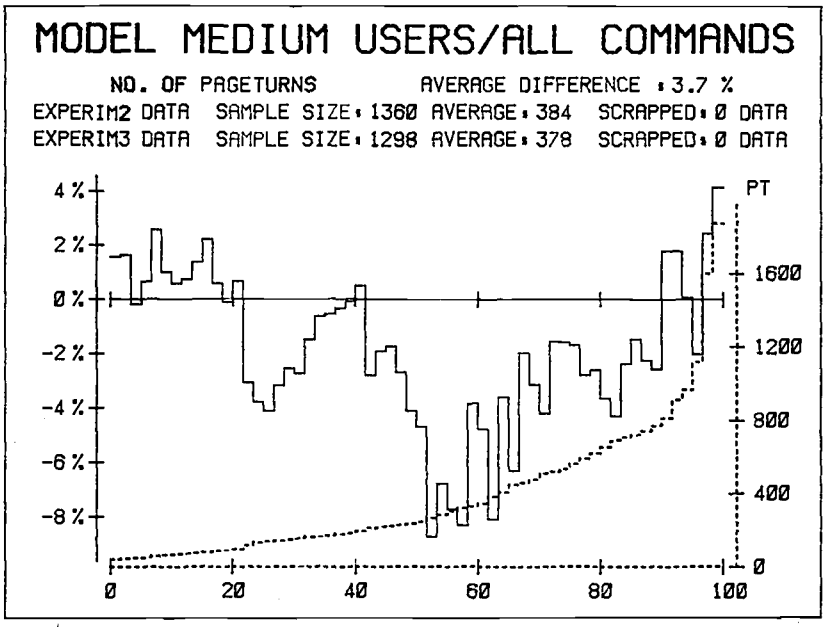

Figure IV.20

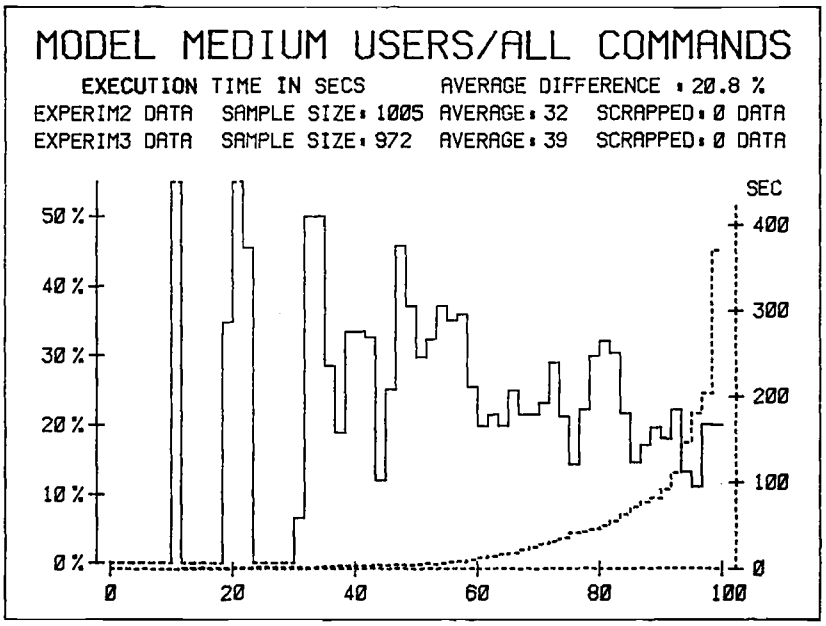

Figure IV.21

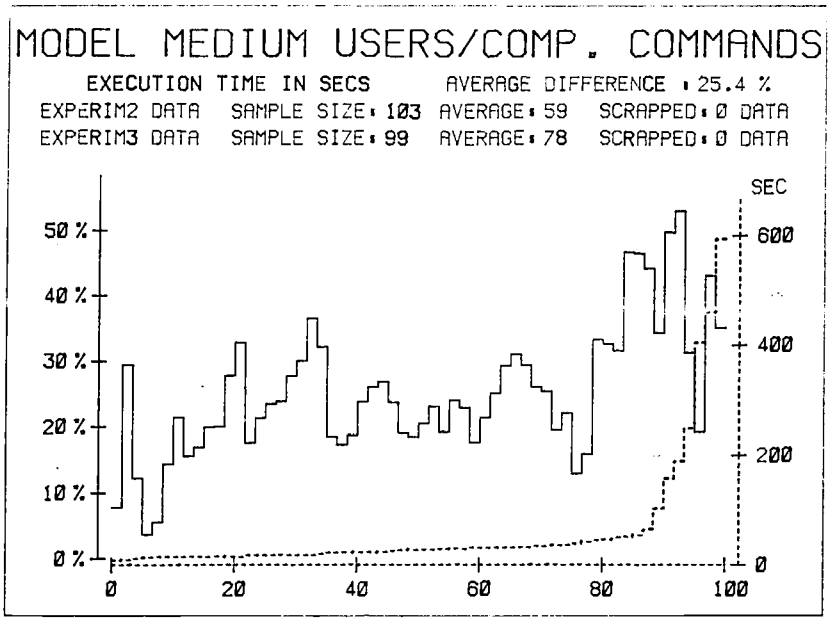

Figure IV.22

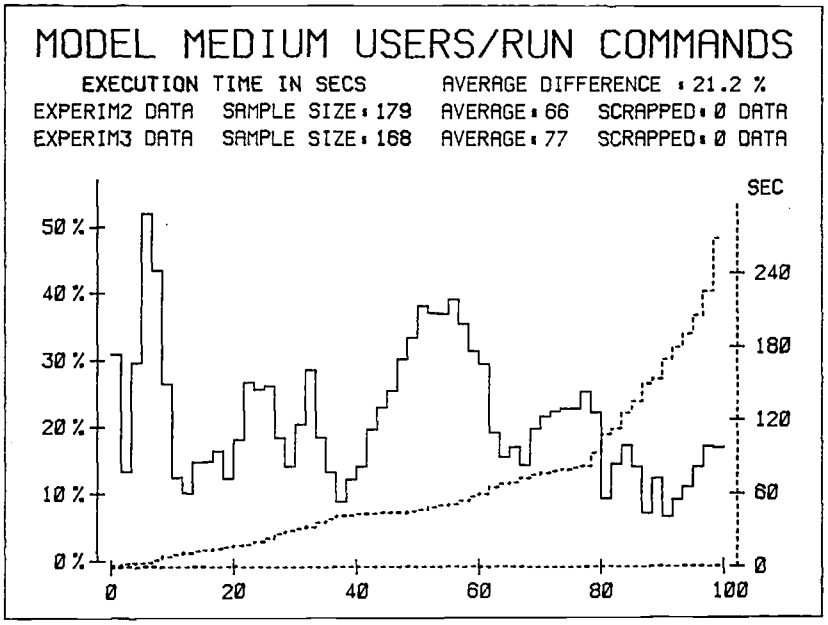

Figure IV.23

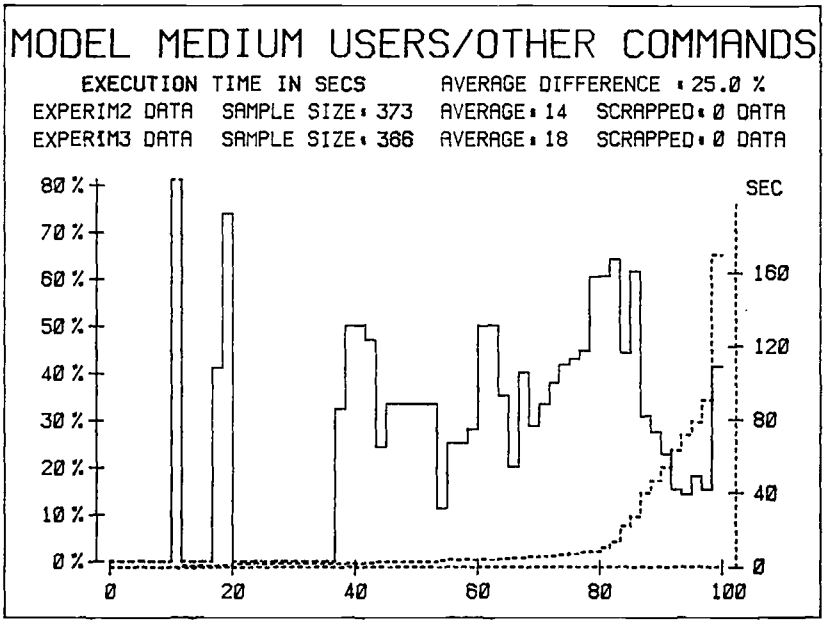

Figure IV.24

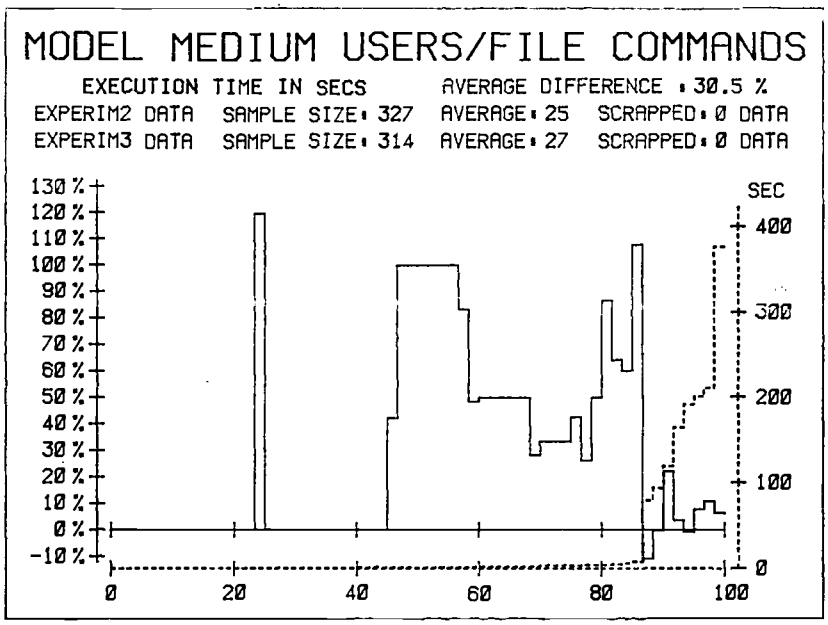

Figure IV.25

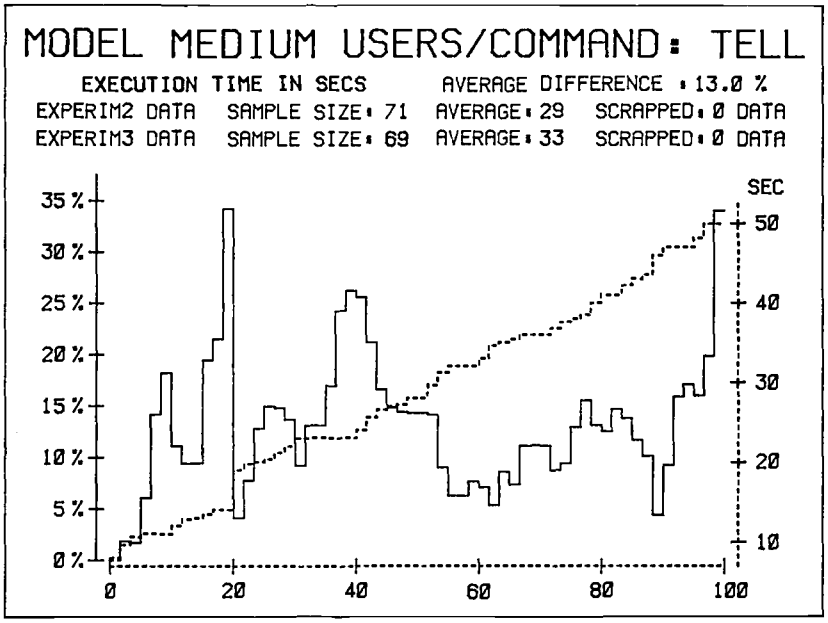

Figure IV.26

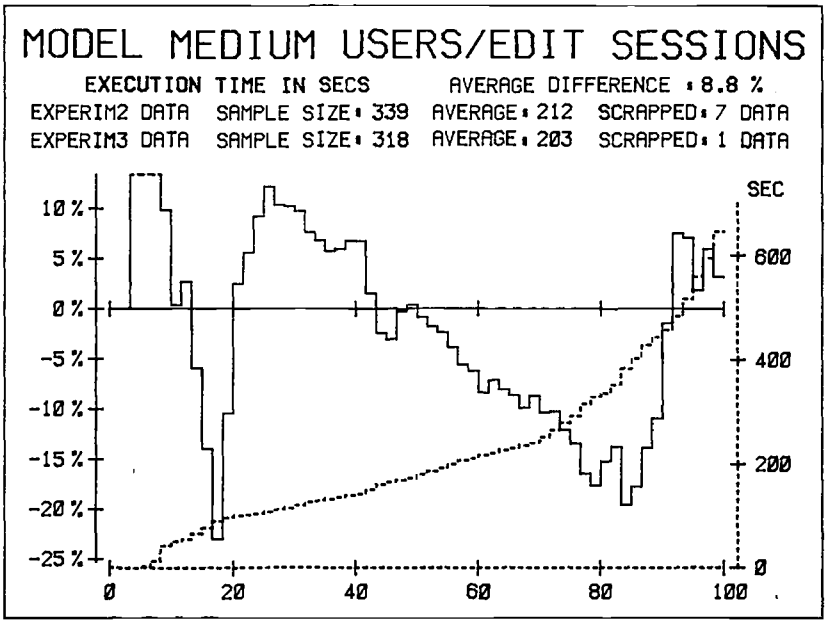

Figure IV.27

C H A P T E R F I V E

SUMMARY AND CONCLUSIONS

The target of this thesis was to provide a method of modelling and generating accurate and realistic interactive workloads for time-sharing computer systems. Accuracy of the workload modelled was validated against execution times found for real users with execution times observed for synthetic users on ERTE. The technique applied can be summarized as follows:

To be as system independent as possible the characterization was mainly done at system-command-language level. For this purpose we grouped the system command language commands available on EMAS into five groups : Edit, Compile, Run, File and Other commands. The users sessions were structured according to their percentages of different command groups. The kinds of sessions found are called edit oriented if there are predominantly edit commands, or edit-compile oriented if edit and compile commands are predominatly used. The following kinds of sessions were found :

- Communication oriented sessions

These sessions are very short and users may just log on and off or look for mail, make system enquiries etc.

- File oriented sessions

In file oriented sessions users may rename, destroy, send etc. files. These sessions are also fairly short and do not use much CPU time.

- Edit oriented sessions
- Edit-compile oriented sessions
- Edit-compile-run oriented sessions
- Run oriented sessions

For the last four kinds of sessions the number of commands issued per session varies widely, CPU time used and length of session varies widely, distributions found for these parameters are very much like the overall distributions. In addition we calculated for each kind of session the transition probabilities from command group to command group to simulate sequences of commands.

As the user population is diverse the structuring of users' sessions was done for three user groups separately, namely Novice users (i.e. computer science I students at the beginning of their first year), Medium users (i.e. Computer Science I student at the end of their first year) and Expert users (i.e. Postgraduates and Staff from various departments).

Differences between these three user groups could be observed. The more experienced users become the more

they use the computer as a communication system and the more file oriented sessions are observed. About a quarter of all expert users' sessions are communication oriented, from novice users' sessions we only observe less than a sixth of all sessions to be communication oriented. Novice and medium users do have a higher percentage of edit, edit-compile, edit-compile-run and run oriented sessions.

For each usergroup we also studied the distributions of the CPU time and pageturns of run and compile commands. For this study we were not interested in absolute values but more in the differences between user groups. It can be seen that novice users do submit very small jobs, jobs of medium users consumed more than three times as much CPU time and pageturns whereas expert users do submit many very small jobs but do have big 'outlier' jobs. The next step was to study differences for these parameters under different loads.

SYSTEM-COMMAND-LANGUAGE LEVEL

It is often assumed that users adapt to different system performance characteristics. This hypothesis can only be supported for some parameters. At system-command-language level differences are very small. For novice and medium users we hardly find any difference, for expert users a slight increase in edit sessions and decrease of run commands could be found. However, if editing is done, users tend to compile and

thus compiling inrceases, too.

THINKTIMES

Thinktimes do differ under different loads. For medium and expert users thinktimes do go down under heavy load. For novice users we observe a slight increase under heavier load.

CPU TIMES AND PAGETURN DISTRIBUTIONS

The distributions of the CPU times used and pageturns produced is quite similar under any load. 'Big' jobs are found under any load but those jobs are submitted slightly more often when the load is lighter.

Another interesting observation is that for medium and expert users the sample sizes we have decreases as the loads becomes heavier. This may indicate that the expert users we monitored do avoid busy periods and prefer to logon when the system is more lightly loaded. Novice users may yet not know and appreciate quick execution and response times or do not know how to get them.

CONCLUSIONS FOR MODELLING WORKLOADS

We modelled an interactive workload at system-command-language level and had a small selection of file and programs the artificial users could work with. Those programs were designed to match CPU times used and pageturns produced of real users. Our model

was validated against execution times for run and compile commands: we achieved a match within the range of +25% and - 25%. These encouraging results have proved that the selection of the parameters is suitable to generate accurate interactive workload models for time-shared computer systems.

As already pointed out it is often assumed that users adapt quite quickly to a system's performance. If this were so for every load a different kind of workload would have to be defined. In our studies the only parameter that really changes is the thinktime and of course the user mix. The use of the system-command-language and the kind of jobs submitted is fairly constant and has not to be changed for different workloads to be modelled. However, as the measurements and experimental runs have shown the presence and absence of outliers may have a considerable influence on execution times. If a user mix changes or the load becomes lighter, more outlier jobs may be submitted bringing it 'back to normal'. Those changes have to be taken into account to define interactive workloads which are sufficiently accurate and credible.

SYSTEM DEPENDENCY AND SYSTEM INDEPENDENCY

Another interesting question is to what degree these results can be transferred to other systems. Unfortunately there are no comparable studies to be found in the literature. However, there are indications

that some characteristics are system independent. For example the fact that very many expert users sessions are fairly short. The question might arise whether 64 K were not enough to get a realistic picture of users' sessions and more short sessions are fully recorded. This is certainly true to some degree.

But as explained in chapter IV this fact was no drawback for a workload characterisation, as extreme long sessions are not worth modelling anyway. In chapter I we gave a short example of a cluster analysis technique applied to data of an accounting file of the departemtal VAX 11/780. Among the feature sets chosen for clustering we also had the login time of each session. Both cluster analysis techniques applied did generate clusters with a significantly large amount of sessions (almost 1/2 for technique 1 and slightly less than 1/2 for technique 2) with a login time of one minute. It is stressed that these sessions were not logon failures but sessions which successfully logged on and off! On VAX we mainly find expert users and no novices, and for expert users on EMAS we find a somewhat comparable amount of short sessions.

Concerning thinktimes recent studies on VAX give an average of 12 seconds, with the most common found 2; this too matches with the thinktimes we found for expert users on EMAS. Another study done on an IBM/168 under SVS suggested an average thinktime (the definition of

the thinktimes are the same as for our experiment) of 28 seconds! This too could well match with the results for our experiment.

The measurements were done on a computing centre's system and thus may well have quite a big number of novice students logged on (no information was provided about user mixes). If we assume this to be the case the average and distributions for thinktimes do well match with those for a mixture of novice and medium users. It is admitted that this case is speculative but as there are not many thorough studies available this case is still worth mentioning as support for independency of thinktimes on different machines.

However, there are also unmistakably system dependencies to be considered. One is certainly the user mix, which may well differ substantially on different machines. The user mix on the Computer Science VAX 11/780 or EMAS for example is very different, as there are almost no novice users on this VAX. And user mixes do influence a system's performance. If a workload has to be modelled a good knowledge of the user mix is therefore crucial to get reasonable results.

Workloads imposed on a system are also influenced by software changes. New software facilities, utility routines, program packages or new programming languages

may influence the performance characteristics of a system. If user groups have to use resource intensive program packages we may well find a considerable change in the workload. Other aspects which may be important in the future are the network facilities. EMAS is attached to the RCO and can be used as host from various machines. This fact, too, may change the workload. Users may want to do their editing and compiling of small programs on smaller computers and just send big programs over the network to EMAS to run them there. Changes in workloads and performance aspects are still unexplored.

Another system dependency is certainly editors. These days on many machines we find a big number of all kinds of editors, with screen editors becoming more and more popular. The use of different editors and the influence on a system's performance is still unexplored and certainly more detailed investigation into this subject is needed.

MONITORING ASPECTS

Because of all these changes and restrictions workload models can only approximate real workloads. An important aspect to be considered is how to keep up to date with changes in the workload imposed on a system. A major difficulty when doing the monitoring experiments was the problem of privacy. The data we collected were strictly private and users had to be asked for

permission to monitor their work. However, it would be more desirable to automate this process and collect information at system-command-language level without these restrictions. To do this it might be possible to write the data necessary for our work to a seperate file, extract the information needed in such a way that it is no longer possible to identify the user. This solution is only acceptable for student users, because they do not have the possibility to set alias commands. For expert users who do use alias commands this solution may not be satisfactory and misclassifications of commands may lead to wrong results. To keep up to date efforts in measurement and analysis cannot be avoided. To study effects of workload changes it will be necessary to generate from time to time new user scripts and present them to EMAS via ERTE. The automatic workload generator will be an invaluable help for this purpose.

It should be kept in mind that an interactive workload model for time-sharing systems can only approximate real workloads because of the changes in workloads imposed on a system.

For users who do not have the time to do measurements and want to describe an interactive workload anyway the results presented in this thesis provide invaluable information for defining realistic interactive workloads. For users who can do the measurements to

define interactive workloads for time-sharing computer systems we have provided a method which proved to be succesful to define interactive workload for time-sharing computer systems.

REFERENCES

[ADA1] Adams,J.C.,Currie,W.S.,Gilmore,B.A.C. 'The Structure and Uses of the Edinburgh Remote Terminal Emulator',Software-Practice and Experience, Vol 8, 1978,pp 451-459

[ADA2] Adams,J.C.,'Performance Measurement and Evaluation of Time-shared operating systems',PhD-Thesis, University of Edinburgh,1977

[AGR1] Agrawala,A.K.,Mohr,J.M.,Bryant,R.M.'An approach to the workload characterization problem,Computer, Juni 1976,pp 18-32

[AGR2] Agrawala,A.K.,Mohr,J.M.,'Some results on the clustering approach to Workload Modelling', Proc. CPEUG, Oct 1977, pp 11-14

[AGR3] Agrawala,A.K.,Mohr,J.M.,'Predicting the workload of a computer system', AFIPS,1978,pp 465-471

[AGR4] Agrawala,A.K.,Mohr,J.M.,'The relationship between the pattern recognition problem and the workload characterization problem', Sigmetrics/CMG VIII, Nov 29-Dec 2 1977,pp 131-139

[ANDE] Anderberg,M.R.,'Cluster Analysis for Applications', New York, Academic Press,1973

[ARTI] Artis,H.P.,'Capacity Planning for MVS Computer Systems', Performance Evaluation Review, Vol.8,No.4,1979,pp47-62

[BARD] Bard,Y.,'A characterization of VM/370 workloads', in: H.Beilner,E.Gelenbe,'Modelling and Performance Evaluation of Computer Systems',North Holland,1977,pp 35-56

[BARB] Barber,E.,Asphyell,A.,Dispen,A.,'Benchmark Construction', Sigmetrics PER., Vol.4 No.4,1976

[BARN] Barney,G.C.,Pentzaropoulos,G.C.,Swindells,W.,'Performance criteria,measurements and analysis for interactive systems', Computer Performance,March 1982,pp.10-16

[BUCH] Buchholz,W.,'A synthetic job for measuring system performance IBM Systems Journal,No.4,1969,pp 309-318

[BUZE] Buzen,J.P.,Goldberg,R.P.,Langer,A.M.,Lentz,E.,Schwenk, H.S.,Sheetz,D.A.,Shum,A.,'Best/I-Design of a tool for computer system capacity planning',National Computer Conference,1978,pp 447-455

[CHAT] Chatfield,C.,Collins,A.J.,'Introduction to multivariate analysis',Chapman and Hall,1980

[DUB1] Dubes,R,Jain,A.K.,'Validity studies in clustering methodologies',Pattern Recognition, Vol.11,1979, pp 235-254

[DUB2] Dubes,R.,Jain,A.K.,'The User's Dilemma',Pattern Recognition, Vol.8,1976,247-260

[DUKE] Duke,M.O.,'Testing in a complex system environment', IBM Systems Journal,No.4,1975,pp 353-365

[ESPO] Esposeto,A.,Mazzeo,A.,Costa,P.,'Workload characterization for Trend Analysis',Performance Evaluation Review, Vol.10,no 2,pp 5-15

[EVER] Everitt,B.'Cluster Analysis',Published by Heinemann Educational Books Ltd.,1974

[FANG] Fangmeyer,H.,Gladen,R.,Larisse,J.,'An automatic clustering technique applied to workload analysis and system tuning', in: H.Beilner,E.Gelenbe,'Modelling and Performance Evaluation of Computer Systems',North Holland,1977,pp 427-432

[FER1] Ferrari,D.,'Characterizing a Workload for the Comparison of Interactive Services', AFIPS, National Computer Conference,1979,pp 789-796

[FER2] Ferrari,D.,'Computer Performance Evaluation', Addison-Wesley,1978

[FOXL] Foxley,E.,Ali,M.,'Techniques for the generation of simulated workloads',Computer Performance,Vol.1,No.1,1980

[GILM] Gilmore,B.A.C.,'DEIMOS-User manual',ERCC internal report 1975

[HELL] Hellermann,H.,Conroy,F.,'Computer System Performance', McGraw-Hill,1975,pp 380

[HONA] Honal,H.,'Erfahrungen mit der Analyse der IBM/168 unter SVS', Arbeitsgespraech der GI/NTG-Interessengruppe "Messung, Modellierung und Bewertung von Rechensystemen',Bonn,1981

[HUNT] Hunt,E.,Diehr,G.,Garnatz,D.,'Who are the users?-An analysis of computer use in a University Computer Center',AFIPS, National Computer Conference,1971,pp 231-278

[JARV] Jarvis,R.A.,Patrick,E.A.,'Clustering using a similarity measure based on shared neighbors',IEEE Transactions on Computers,Vol.C-22,No.11,1973,pp 1026-1034

[JOSL] Joslin,E.O.,'Application Benchmarks',Proc. of the A.C.M. National Conference, 1965, pp 27-37

[KERN] Kernighan,B.W.,Hamilton,P.A.,'Synthetically Generated Performance Test Loads',Proc. of the First A.C.M. S.I.G.M.E. Symp. on Measurement and Evaluation,1973, pp 121-126

[KOBA] Kobayashi,H.,'Modeling and Analysis',Addison-Wesley publishing Company,1978

[LUCA] Lucas,H.C.,'Performance Evaluation and Monitoring', Computing Survey, Sept. 1971,pp 79-91

[MCGI] McGillivray,N.,MSC-Thesis, Edinburgh, 1982

[MEAD] Mead,R.L.,Schwetman,H D 'Job Scripts- A Workload Description based on system event data',AFIPS, National Computer Conference, 1978,pp 457-464

[MERT] Mertens,B.,'A pragmatic approach to a quantitative performance evaluation',in: H.Beilner,E.Gelenbe,'Measuring,Modelling and Evaluating Computer Systems',North Holland,1977,pp 1-17

[MILL] Millard,G.E.,Rees,D J,Whitfield,H 'The Standard EMAS Subsystem', Computer Journal,Vol.18,No.3,1975

[MINE] Minetti,V.,'Performance Evaluation of a batch-time sharing computer system using a trace-driven model' in: H.Beilner,E.Gelenbe,'Modelling and Performance Evaluation of Computer Systems',North Holland,1977,pp 485-497

[NOE1] Noethe,V.,'User behaviour in a system command language environment',Computer Performance, Computer Performance, Vol.3,No.1,1982,pp 5-9

[NOE2] Noethe,V.,'An automatic workload generator',submitted for publication to Software - Practice and experience

[NOLA] Nolan,L.E.,Strauss,J.C.,'Workload Characterization for Timesharing System Selection', Software-Practice and Experience,Vol.4,1974,pp 25-39

[REES] Rees,D.J.,'The EMAS Director',Computer Journal,Vol.18,No.2,1975

[ROEH] Roehr,K.M.,'Modelling of Computer System Performance during development',in: H.Beilner and E. Gelenbe (editors), 'Measuring,Modelling and Evaluating Computer Systems', North Holland,1977,pp 905-433

[SALT] Saltzer,J.H.,Gintell,W.,'The Instrumentation of MULTICS', CACM, No.8,1970,pp 495-500

[SHAW] Shaw,S.,'ERTE results',ERCC

[SPIE] Spiegel,G.,'RTE - Past is Prologue',Performance Evaluation Review, Vol.10,No.2,pp 993-1008

[STEP] Stephens,P.D.,Yarwood,D.J.Rees,Shelness,N., 'The Evolution of the 2900 system',Software-Practice and Experience,Vol.10,No.12,pp 993-1008

[STON] Stone,H.S,(Editor),'Introduction to Computer Architecture', Science Research Associates,Inc.Chicago,1975

[SREE] Sreenivasan,K.,Kleinman,A.J.,'On the Construction of a Representative Synthetic Workload', Communications of the ACM,Vol.17,March 1974,pp 127-133

[SVOB] Svoboda,L.,'Computer System Measurability',Computer, June 1976,pp 9-17

[TATE] Tate,D.,'Pipelining:Design Problems and Implications', Computer Design,Infotech State of the Art Rep 17,1974, pp 253-280

[WHIT] Whitfield,H.,Wight,A.S.,'EMAS-The Edinburgh Multi-Access System', Computer Journal,Vol.18,1973

[WIGH] Wight,A.S.,'The EMAS Archiving Program',Computer Journal,Vol.18, 1975,pp 131-134

[WOOD] Wood,D.C.,Forman,E.H.,'Throughput measurement using a synthetic job stream',AFIPS Conf. Proc.,1971,pp 51-56

[WRIG] Wright,L.,Burnette,W.A.,'An Approach to Evaluating Time-sharing systems',Sigmetrics,PER,Vol.5 No.1,1976