# On State-Dependent Workload Characterization by Software Resources

Günter Haring

Institute for Information Processing
Technical University
A 8010 Graz - Austria

## ABSTRACT

A method for the characterization of computer workload at the task level is presented. After having divided the workload into different classes using a cluster technique, each cluster is further analysed by state dependent transition matrices. Thus it is possible to derive the most probable task sequences in each cluster. This information can be used to construct synthetic scripts at the task level rather than the usual description at the hardware resource level.

## 1. INTRODUCTION

Computer based information processing can be viewed as a communication between a data processing system and its environment. In this context the term workload comprises the sum of all information processing requirements offered from the environment to the system during a given time interval. These requirements consist of programs, data, commands, etc. /1/. As the performance of a computer system depends on the work loaded on the system, the characterization and modeling of the workload play a central role in all performance-orientated questions of computer system management, such as performance improvement, selection and design of computer systems, capacity planning, etc. /2/.
The workload of a system can be described at different levels: functional specification level, logical resources level or hardware resources level; these can be further subdivided, if necessary. The selection of the right level for the characterization depends on the goal of investigation. For example, to replace a complete system the hardware level would not be the right one, the characterization must rather be done at the program level. The difficult problem in workload characterization is , that usually there are not enough data available at the right level - if at all /3/. As a consequence workload is very often characterized at the wrong level, using values of hardware resource consumption, which are available on nearly all systems. Whereas there are a lot of papers dealing with workload characterization on the hardware level (/4/ - /11/), there are only few contributions that consider higher-level characterization using quantitative methods (/11/ - /13/). Workload characterization, used for computer system procurement or for the evaluation and assessment of different time sharing services, should be system independent (/14/ - /19/). This is not the case, if the level of characterization is related to hardware resources of the existing system. A consideration of higher levels is especially important if a synthetic script must be derived from the workload characterization to evaluate different systems. Within such a script a sequence of tasks must be constructed and included, which will then be executed on the systems to be tested. The hierarchical structure of a script is shown in fig.1 in Jackson-like terminology. The script consists of a sequence of tasks, and each task in turn consists of a sequence of statements, which must be specified in detail. This structure leads to a stepwise development of the script. The construction of a script at the task level is the background of this paper. At this point one should notice that usually there is enough information available for a characterization at the higher level. On the contrary, there is no adequate information available at the lower level, i.e. to complete the workload characterization, one needs information on the logical resources within a task, such as the special edit commands used, the execution frequencies of statements in higher programming languages and the access patterns and execution

sequences, etc. Usually, this information is not available, except within special, selfmade instrumentation packages (e.g. /22/, /23/). Therefore the lower level is very often inadequately replaced by the task dependent values of hardware resource consumptions.

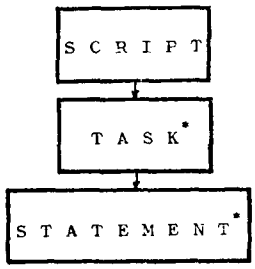


Fig.1 Hierarchical structure of a synthetic script

Serazzi used in /11/ a functionally - oriented approach at a low level of detail, which produces classes of workload components. He subdivided all of the program-steps of the workload into 9 categories depending upon the programming language used and the two types of work performed, i.e. compilations and executions. His goal was to show the equivalence between this functionally-oriented approach and the hardware resource-oriented approach, according to hardware resource utilizations of the distinct job steps, i.e. to show that the program-steps of each functional category have a typical hardware resource utilization pattern, though the resource utilization patterns corresponding to the various functional categories are not completely separate.
Agrawala and Mohr used a markovian model of a job in /13/. This model can be used to characterize a sequence of states, i.e. the sequence of job steps of a job. The transition probabilities in the model are homogeneous. Before applying this modeling technique, the jobs of their workload were classified by a cluster algorithm based on hardware resource consumptions.
In section 2 our approach to workload characterization using software resources is described. The workload classes which result from this characterization, using cluster analysis techniques, are further analysed using state dependent transition matrices in section 3.
Different from the approaches in /11/ and /13/ we did not use hardware resource consumptions for the preclassification of the jobs. Instead we used the frequencies with which functional components, like editors, compilers etc. are called by the jobs, in our cluster analysis. Similar to the approach in /13/ we represent each job as a sequence of job steps. Furtheron we assumed - as an essential extension to /13/ - nonhomogeneous transitions between the states (job steps) of a job.
Each run is represented as a sequence of tasks which must be executed in that order. The next task executed could depend not only on the current task but also on some prior tasks and on the step index. The execution of a run corresponds to the transition between distinct states of the run. A state corresponds to a software resource used by the run, like editor, compiler, etc. The results of this investigation are discussed in section 4 with some further remarks and notes on further research in section 5.

## 2. CHARACTERIZATION AT THE TASK LEVEL

A first attempt to characterize workloads on a level higher than that of hardware requirements is done by describing this workload by the software resources which are used by the jobs, like compilers, editor, linker, etc. In a first step the interactive workload should be subdivided into classes according to their consumption of software resources which then allows the derivation of characteristic terminal sessions, like those for data input, program development, production runs, etc. This partitioning can be achieved by using quantitative methods like cluster analysis. Notice that this method of description reflects some aspects of user's behaviour - e.g. sequence of tasks in a session - which is normally not considered in workload characterization if it is done on the hardware level. But these user aspects have an essential influence on the load and the performance of a system.
The data used in this investigation are from the interactive workload on a UNIVAC 1100/81 computer system used by the two universities at Graz. During a period of about three weeks 3830 runs were executed. The logfile of the system contains an entry both for each call of a software processor or a program and its termination. In our case the following types of software resources (functional categories) were identified:

1. FOR ... including all Fortran compilers
2. EDIT ... including all edit and data processors
3. MAP ... link and load functions
4. FILE ... file operations
5. SUSRES ... directing the output from a terminal to a printer
6. PRSP ... including all other programming language processors except those for Fortran
7. XQT ... program execution

There are several methods available to subdivide the total workload into classes

with different usage of software resources. The most practical solution seems to be the use of clustering techniques based on the distribution of calls for software resources per run, expressed as percentages. For the clustering procedure the well known KMEANS algorithm /24/ can be used, giving spherical clusters on scaled data. One should note that there is no problem with scaling of the data or the treatment of outliers - usually two critical points in cluster analysis when applied to workload characterization on the hardware level. Applying this method to our data - as described in /21/ - results in a subdivision of the total workload into seven clusters, which allow a very clear and simple interpretation. The average number of calls of the distinct software resources in the clusters and the transition matrices within each cluster give an information which is valid for a high percentage of the runs belonging to the cluster. The software processors and programs within a run can be seen as states of the run. A transition from one state to another corresponds to the termination of one processor (e.g. Fortran compiler) and a call of the next one in the task sequence of this run (e.g.editor). The state before the run starts corresponds to a fictitious state START. Similarly, a fictitious, absorbing state FIN is entered upon the termination of the last task in the run. The elements $q_{ij}$ of this transition matrix represent the number or frequency of transitions from state i to state j.
As an appropriate method of display a transition diagram can be introduced, which is derived from the transition matrix of the cluster. This diagram shows the most probable run structure in each cluster. The states are shown by rectangles, the most frequent transitions by thick lines and frequent transitions by thin lines.
To illustrate the results we take cluster 3 which comprises 10.67 % of all runs. With about 93.1 % calls of the editor and 4.4 % file operations, this cluster represents runs that perform primarily data preparations. The transition diagram is given in fig.2, the percentage of the calls of the distinct software resources and the transition matrix is given in tab.1.
In the first part of table 1 the average percentage of calls of the distinct software resources by a job belonging to cluster 3 is given. For example, on the average 93,1 % of the calls for software resources in a job of cluster 3 are requests for the editor. For the distinct values also the standard deviation, the coefficient of variation and the minimum and maximum within a job are given.
In the second half of table 1 the transition matrix of jobs belonging to cluster 3 is given. The entry $q_{ij}$ in the i-th row and j-th column denotes the total number of transitions from state i to state j summarized over all jobs of cluster 3.

| | Average | Standard Deviation | Coefficient of Variation | Minimum - Maximum |
|---|---|---|---|---|
| FOR | 0.4 | 3.5 | 8.533 | 00.0 - 33.3 |
| EDIT | 93.1 | 13.5 | .145 | 57.1 - 100.0 |
| MAP | 0.1 | 1.7 | 16.917 | 00.0 - 40.0 |
| FILE | 4.4 | 10.9 | 2.502 | 00.0 - 41.2 |
| SUSRES | 0.2 | 2.4 | 9.574 | 00.0 - 33.3 |
| PRSP | 0.1 | 1.2 | 23.353 | 00.0 - 33.3 |
| XQT | 1.8 | 6.8 | 3.878 | 00.0 - 36.4 |

| | FIN | FOR | EDIT | MAP | FILE | SUSRES | PRSP | XQT |
|---|---|---|---|---|---|---|---|---|
| START | 0 | 0 | 896 | 0 | 33 | 2 | 0 | 11 |
| FOR | 1 | 0 | 15 | 1 | 4 | 0 | 0 | 0 |
| EDIT | 887 | 21 | 974 | 3 | 177 | 15 | 2 | 94 |
| MAP | 0 | 0 | 4 | 1 | 0 | 0 | 0 | 5 |
| FILE | 33 | 0 | 170 | 5 | 37 | 16 | 0 | 11 |
| SUSRES | 4 | 0 | 16 | 0 | 12 | 4 | 0 | 6 |
| PRSP | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 |
| XQT | 17 | 0 | 96 | 0 | 9 | 5 | 0 | 6 |

Tab.1 Percentage of calls and transition matrix for cluster 3

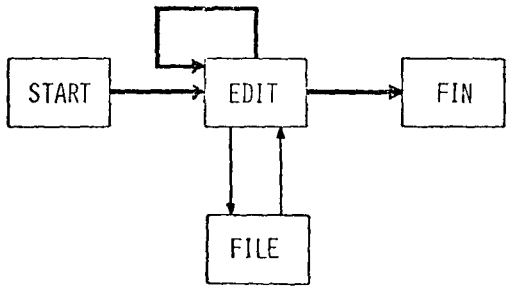


Fig.2. Transition diagram for cluster 3

Similarly the other clusters can easily be interpreted in the following way:

Cluster 1: Runs with control commands to control the work during night hours

Cluster 2: Development of Fortran programs

Cluster 3: Data preparation

Cluster 4: Administrative tasks like saving and organising activities for data and program files

Cluster 5: Production runs and preparation of data for the execution

Cluster 6: Generation and inspection of printouts together with program development

Cluster 7: Preparation of programs and data in other languages

Thus the application of a well known quantitative method leads to a very useful partitioning of the workload, which can be further analysed using state dependent transition matrices.

## 4. STATE DEPENDENT CHARACTERIZATION

The transition diagram which we used in the previous section does not explicitly depict transition steps. Therefore the so called trellis diagram is introduced. In the trellis diagram /25/ the transition step parameter k is explicitly shown. Each node in the diagram corresponds to a unique pair of state and transition step (fig.3). Any run of the workload determines a path in the trellis diagram.

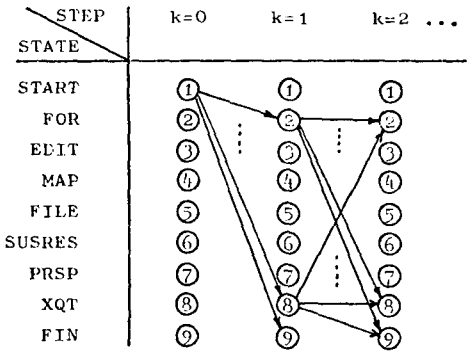


Fig.3. Trellis diagram

Till now we assumed that the transition matrix is homogeneous, i.e. independent of the transition step; this is not true in practice. On the contrary the transition matrix shows a strong dependence on the step level. Therefore the entries of the transition matrix will be noted by $q_{ij}(k)$. We define the system state by a vector $\underline{n} = \{n_i(k),\ i = 1(1)9,\ k = 0,1,2,\ldots\}$, where $n_i(k)$ represents the number of runs which are in the state i (1 = START, 2 = FOR, 3 = EDIT, 4 = MAP, 5 = FILE, 6 = SUSRES, 7 = PRSP, 8 = XQT, 9 = FIN) at step k. If the probability that a run is in state i at step k is denoted by $p_i(k)$, the probabilities can be calculated according to the following equation

$$(1)\quad p_i(k+1) = \sum_{j=1}^{9} p_j(k).q_{ji}(k) \quad \forall\ k \geq 0$$

with $p_1(0) = 1$, $p_i(0) = 0$ for $i = 2(1)9$, since each run is initially in the START state. Note also that $q_{j1}(k) = 0$ for $j = 1(1)9$ and all $k \geq 0$. State 9 (= FIN) is an absorbing state; therefore $q_{9i}(k) = 0$ for all $i = 1(1)9$ and all $k \geq 0$. Calculating the probabilities according to equation (1) yields $p_9(k)$, the probability that the run will finish after step k. The total probability, that the "system" is in the state FIN after step k is equal to

$$\sum_{j=0}^{k} p_9(j).$$

## 5. DISCUSSION

This technique of state dependent characterization can be applied to each of the clusters in section 2 and can help to identify the "right" sequence of tasks for each script.
E.g. cluster 3 shows the following trellis diagram (fig.4), where each entry in the table gives $p_i(k)$. The last line shows the total probability for the state FIN. About two thirds of the runs have only one edit task. Three quarters of the runs have two tasks at the most, with a high probability that both are edit tasks. 86 % of the runs have three consecutive tasks at the most, with a high probability, that all of these three tasks are edit activities. If the second task is not an editor call, it will comprise file operations or a program execution. But the probability for these two is rather small.
Fi.5 shows the percentage of jobs finishing after k transitions.

| k | 0 | 1 | 2 | 3 | 4 .... |
|---|---|---|---|---|---|
| START | 1.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| FOR | 0.00 | 0.00 | 0.01 | 0.00 | 0.00 |
| EDIT | 0.00 | 0.95 | 0.21 | 0.21 | 0.11 |
| MAP | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| FILE | 0.00 | 0.04 | 0.08 | 0.03 | 0.02 |
| SUSRES | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| PRSP | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| XQT | 0.00 | 0.01 | 0.03 | 0.02 | 0.01 |
| FIN | 0.00 | 0.00 | 0.67 | 0.07<br>0.74 | 0.12<br>0.86 |

Fig.4. Trellis diagram for cluster 3

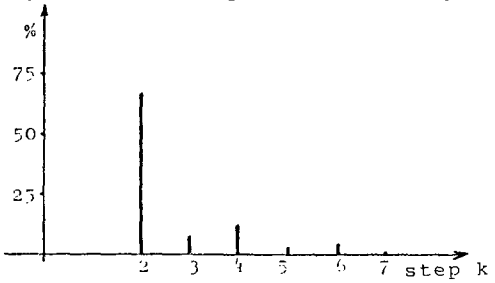


Fig.5. Percentage of runs finishing

Similar conclusions can be drawn in a more complex cluster like that for program development (cluster 2). The normal transition diagram is shown in fig.6. Primarily editors and Fortran compilers are involved in this class together with some occurrences of MAP, XQT and FILE.

Fig.7 shows the trellis diagram for that cluster with the path of a very probable run, i.e. START-EDIT-FOR-EDIT-FOR-EDIT-EDIT (FOR)-EDIT-FIN. About one quarter of the runs have finished after eight transition steps or less. Fig.8 presents the percentage of jobs finishing after k transitions with quite another distribution compared to fig.5.

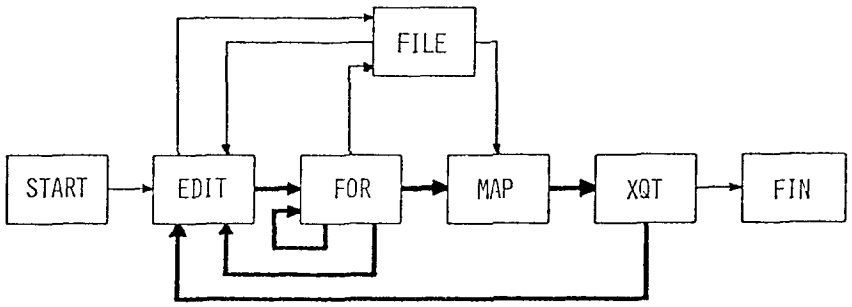


Fig.6. Transition diagram of cluster 2

| k | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 .... |
|---|---|---|---|---|---|---|---|---|---|
| START | 1.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| FOR | 0.00 | 0.07 | 0.46 | 0.22 | 0.31 | 0.19 | 0.21 | 0.18 | 0.17 |
| EDIT | 0.00 | 0.64 | 0.22 | 0.33 | 0.20 | 0.26 | 0.22 | 0.21 | 0.18 |
| MAP | 0.00 | 0.01 | 0.03 | 0.11 | 0.14 | 0.11 | 0.11 | 0.08 | 0.08 |
| FILE | 0.00 | 0.19 | 0.16 | 0.20 | 0.12 | 0.15 | 0.12 | 0.14 | 0.13 |
| SUSRES | 0.00 | 0.02 | 0.03 | 0.03 | 0.04 | 0.03 | 0.05 | 0.06 | 0.06 |
| PRSP | 0.00 | 0.00 | 0.01 | 0.00 | 0.01 | 0.01 | 0.00 | 0.00 | 0.01 |
| XQT | 0.00 | 0.07 | 0.08 | 0.08 | 0.11 | 0.14 | 0.14 | 0.14 | 0.13 |
| FIN | 0.00 | 0.00 | 0.01 | 0.02 | 0.04 | 0.04 | 0.04 | 0.04 | 0.05 |
| | | | | 0.03 | 0.07 | 0.11 | 0.15 | 0.19 | 0.24 |

Fig.7. Trellis diagram for cluster 2

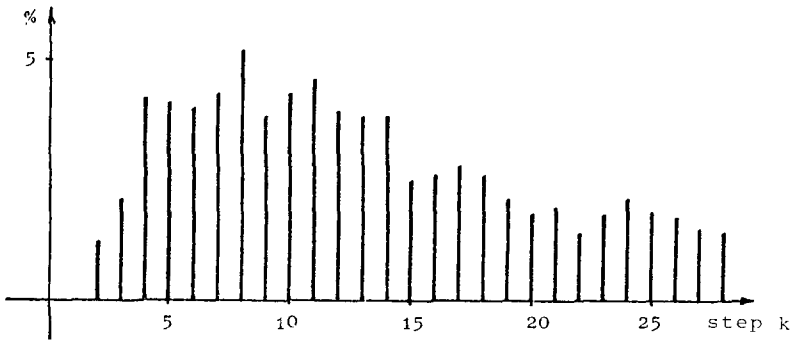


Fig.8. Percentage of runs finishing

The difference between trellis diagram and the general transition diagram for cluster 2 results from the fact that usually a run in this cluster (program development) consists of a sequence of calls to editors and Fortran compilers alternatively at the beginning, to FILE-, MAP-, XQT-phases in the middle and toward the end of the run. Whereas this characteristic can not be seen very well in the general transition diagram, it becomes obvious inspecting the trellis diagram over a longer period of steps. This shows the clear advantage of state dependent characterization.
Whereas the programs in cluster 2 (program development) show rather long task sequences, the runs in cluster 5, which contains the production runs, have short sequences; about half of the jobs have finished in five steps or less. All the steps are primarily executions of user programs (fig 9 - 11)

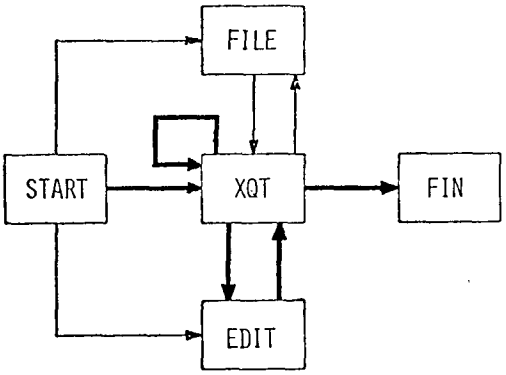


Fig.9. Transition diagram of cluster 5

| k | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| START | 1.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| FOR | 0.00 | 0.01 | 0.03 | 0.02 | 0.01 | 0.01 |
| EDIT | 0.00 | 0.28 | 0.13 | 0.16 | 0.14 | 0.12 |
| MAP | 0.00 | 0.01 | 0.02 | 0.03 | 0.02 | 0.02 |
| FILE | 0.00 | 0.13 | 0.11 | 0.07 | 0.06 | 0.04 |
| SUSRES | 0.00 | 0.02 | 0.02 | 0.02 | 0.02 | 0.02 |
| PRSP | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| XQT | 0.00 | 0.55 | 0.54 | 0.38 | 0.37 | 0.31 |
| FIN | 0.00 | 0.00 | 0.15 | 0.17 | 0.06 | 0.10 |
| | | | | 0.32 | 0.38 | 0.48 |

Fig.10. Trellis diagram of cluster 5

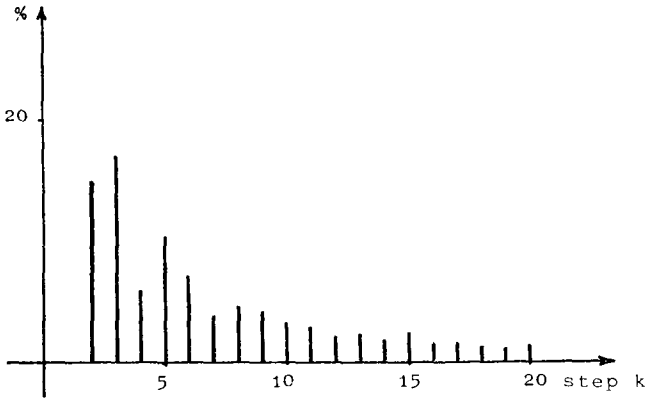


Fig.11. Percentage of runs finishing

One should notice, that the state probability vector p can be used together with a generator for uniformly distributed (0,1)-random numbers for automatic selection of tasks in the synthetic script /26/.

## 6. CONCLUSION

This paper presented a method for workload characterization at the task level. With rather simple and well known methods like cluster analysis and state dependent description a good and expressive characterization can be found. This information can be used to construct synthetic scripts at the task level.
Further investigations should consider transition matrices of higher order and include sojourn times within the tasks. The resource usage of different tasks at distinct steps should be tested for significant differences like in CPU-time, i/o-activity, etc.

## 7. REFERENCES

/ 1/ D.Ferrari: "Computer system performance evaluation", Prentice Hall 1978

/ 2/ H.D.Schwetman: "Workload characterization: Why? What? How?", Proc.EUROCOMP on Comp.Perf.Eval. 1976, p.457-471

/ 3/ D.M.Conti: "Benchmarking: an imperfect science", Proc.ECOMA-9 Conf.1981, p.88-93

/ 4/ A.K.Agrawala, J.M.Mohr: "A comparison of the workload on several computer systems", Proc. CMG IX Conf.1978, p.177-183

/ 5/ A.K.Agrawala, J.M.Mohr: "A model for workload characterization", Proc. of Symp. on Simul. of Comp. Syst. 1975, p.8 - 18

/ 6/ M.L.Bolzoni, P.Mapelli, G.Serazzi: "A methodology for the classification of batch and interactive workload components", Proc. ECOMA-9 Conf. 1981, p.190-199

/ 7/ K.Terplan: "Workload representation in mixed environment", Proc. EUROCOMP on Comp.Perf. Eval. 1976, p.389-403

/ 8/ C.G.Crothers: "Workload determination and representation for on-line computer systems", MITRE-Corp.,F19628-73-C-0001,1973

/ 9/ J.A.Lockett:"Computer performance analysis: evaluation with mixed online-batch workloads", The Rand Corporation, Santa Monica, Rep.No. R-1275-PR, 1974

/10/ R.L.Mead, H.D.Schwetman: "Jobscripts - a workload description based on system event data", Proc. NCC 1978, p.457-464

/11/ G.Serazzi: "A functional and resource-oriented procedure for workload-modeling", Proc. Performance 81 Conf. 1981, p. 345-362

/12/ A.K.Agrawala, J.M.Mohr, R.M.Bryant: "An approach to the workload characterization problem", IEEE Comp.Magazine, June 1976, p. 18-32

/13/ A.K.Agrawala, J.M.Mohr: "A Markovian model of a Job", Proc. 14th CPEUG Meeting, 1978, p. 119-126

/14/ J.S.Cameron: "An approach to benchmarking terminal oriented systems", Proc. ACM Ann.Conf.1976, p.208-212

/15/ D.M.Conti: "Use of synthetic benchmarks for estimating sercice bureau processing charges", NBS Techn. Note 920, July 1976

/16/ D.M.Conti: "A method for estimating service bureau processing charges", Proc. 7th Int.Conf.of the CMG 1976, p.34-60

/17/ D.Ferrari: "Characterizing a workload for the comparison of interactive services", Proc. NCC 1979, p.789-796

/18/ S.A.Mamrak, M.D.Abrams: "A taxonomy for valid test workload generation", IEEE Comp.Magazine, Dec.1979, p. 60-65

/19/ L.E.Nolan, J.C.Strauss: "Workload characterization for timesharing system selection", Software-Pract.&Exp., Vol.4 (1974), p. 25-39

/20/ M.Jackson: "Principles of Program Design", Academic Press 1975

/21/ G.Haring: "Workload characterization at task level", to appear in Comp.Perf., vol.3, no.2 (June 1982)

/22/ L.D.Fosdick: "BRNANL, a Fortran program to identify basic blocks in Fortran programs", Univ.of Colorado at Boulder 1974

/23/ R.L.Sites: "Programming tools: statement counts and procedure timings", ACM SIGPLAN Notices, vol.13, no.12 (Dec.1978), p. 98-101

/24/ M.Anderberg: "Cluster analysis for applications", Academic Press 1973

/25/ H.Kobayashi, M.Reiser: "On generalization of job routing behaviour in a queueing network model", IBM Res.Rep. RC 5679, Oct.1975

/26/ Ch.R.Spooner: "Benchmarking interactive systems: producing the software", 1979 Conf. on Simul., Meas. and Modeling of Comp.Syst., p. 249-257