To be submitted to EDP Performance Review

LBL-7962
Preprint c.2

TOWARDS USER-ORIENTED PERFORMANCE MANAGEMENT

David F. Stevens

August 1978

Prepared for the U. S. Department of Energy
under Contract W-7405-ENG-48

LBL-7962
c.2

Towards User-Oriented Performance Management

ABSTRACT: Conventional computer performance management efforts are founded upon a professional DP point of view. The user's view is quite different, containing no technical detail and concentrating on only three aspects of the system. Each of these three aspects possesses user-oriented performance attributes which, in turn, suggest the kinds of performance information necessary to develop a user-oriented view of system performance.

Keywords and Phrases: CPM, PMIS, (computer) performance management, performance information requirements.

Towards User-Oriented Performance Management [1,2]

David F. Stevens

Lawrence Berkeley Laboratory
University of California
Berkeley, California

August 1978

Introduction

Once upon a time there was a Systems Programmer who dabbled in the Black Art of Computer Performance Measurement. During the course of his investigations he noticed that many jobs languished in an unproductive wait state for want of attention from the Beautiful but much Sought-after Disks. Now, a characteristic of the System he nurtured was the CPU-Active nature of this Languishment. Nevertheless, he performed a Magickal Act to release the System from its thrall, with the result that there now was

1) less Languishment, and hence Better Service, but also therefore
2) less CPU Utilization.

He also took steps to unbind the CPU from the Languishing Job, so that CPU utilization dropt still further.

Question: If you, as a Modern Computer Performance Expert, installed a modification which reduced CPU utilization, would it be considered a Good Thing or a Bad Thing?

The question is not frivolous, but is intended to underline the point that the performance measures we apply to a system tend to direct the course of

[1] This work was supported by the U.S. Department of Energy.
[2] This work had its origin in a seminar sponsored by the Laboratoire de Metrologie in June, 1978.

system development. Associated with each measure is an understanding about which values are "good" values; system tuning and development activities are oriented towards approaching these "good" values. High utilization measures, for example, are generally considered to be better than low ones, whereas the reverse is true for turnaround measures. The following "law" is an obvious, but often overlooked, consequence of this orientation:

> If the goals of a performance management effort do not determine its measures, then the measures will determine the goals.

In other words, if the proper goals of performance management are forgotten, then the "good" values themselves become the goals, with the result that high CPU utilization, increased channel activity, instantaneous response time, and the rest of the standard repertoire all cease to be means to a clearly perceived end (improved service) and themselves become the ends towards which performance improvement efforts are directed. In particular, we often forget that the optimum value may be mid-range, rather than either maximum or minimum.

This effect can be minimized if the Performance Manager concentrates his attention upon those aspects of the service which are meaningful to the users. The balance of this note describes a simple model of the Users' View of the System designed with this in mind, and presents some samples of the kinds of performance information which are appropriate to such a model. It will be readily apparent that performance management activity based upon the suggested kinds of information will be much more user-oriented than the traditional forms seem to be.

Performance of WHAT?

Before embarking upon the main body of the discussion, let us spend a moment considering a fundamental question: What is the entity whose performance is to be managed?

There are within the DP Department at least three organizational levels at which performance management can be meaningful: the department itself, "the system", and the hardware. Most of today's CPM efforts are taken from one extreme of this small spectrum and aimed at the other, in the sense that hardware-oriented measurements are used as the primary line of defense against criticism directed towards the department as a whole: the traditional response to complaints about quality of service is a litany of impressive-sounding statistics about "availability", "CPU utilization", "overlap", etc. Whereas measurements of this type can be useful for pinpointing specific bottlenecks or other problem areas, and for providing the background for an effective system of exception reporting, they shed no light whatsoever upon those aspects of system behavior likely to be of interest to--or even comprehensible to--the users. In fact, as has been noted at some length elsewhere (in [1], for instance), they tend to be more obfuscatory than illuminating.

Users who complain about quality of service are interested in the performance of the system as a whole, and not in the performance of its component parts. They do not wish to become EDP experts. They recognize that hardware is a fundamental part of the system and that hardware performance is an important ingredient of total systems performance, but they have no interest in learning the often complicated and indirect relationships between raw hardware performance data and perceived system performance. Furthermore, they see very clearly that hardware is not the whole system. The system also includes software, people, physical environment, procedures, and policies, and the users interact with all of these things more directly and intensively than with the hardware. A measurement program which bypasses these aspects of the system will be incomplete, and the picture it paints will be distorted.

A disclaimer (or two)

At this point I might do well to point out that I do not advocate abandonment of the traditional performance measures. As indicated above, I believe that they have their uses and their place....But that place is within the councils of the EDP Department and not in the public eye, and those uses are as diagnostic tools and not as defensive shields (or, worse yet, offensive weapons).

The job of the Performance Manager is to know what system performance is and what it should be, and to map out a journey to take the system from the one to the other. Traditional performance measures can help the PM to identify likely causes of subpar performance and to distinguish between promising paths and blind alleys; they should be exploited as fully as possible to those ends. But they do not address the question of what system performance actually seems to be from the users' point of view....And performance is what it seems to the users to be. In the balance of this note you will find some suggestions on how to go about collecting information which does address the question of the users' view of performance. You will not, however, find a cookbook for "user-oriented performance management". Each user community is different, and all change with time; you must adopt those methods which fit your installation and adapt them to your own user community. You will find the effort rewarding.

A user's view of the system

In order to determine the kinds of performance information which will provide an accurate portrait of the performance perceived by the users we must first look at the system from the users' point of view. Different kinds of users see different things when they look at the system; the following model is based upon the viewpoint of an end-user with a real job of work to do, and for whom the computing system is a possible tool.

This user would like to view the system as a black box, the inner workings of which are totally irrelevant. His concerns are concentrated in three areas:

- access
- interface
- product/result

We will consider each of these areas in turn, indicating the characteristics of good performance, and suggesting the kinds of performance information which will allow the Performance Manager to develop a realistic picture of the service as seen by the users. Having discussed the individual areas, we briefly discuss some aspects of the performance of the system as a whole. And finally, we attempt to formulate a general principle to provide some foundation for the practice of user-oriented performance management.

Access

A tool is of no use unless it can be used, i.e. unless it is available and in good working order when needed. Good performance in the area of system access consists of high availability, reliability, and dependability, where these words are to be taken in their natural English senses, rather than in the (non)senses in which they are most often encountered within the performance measurement community. Thus, to a user, a facility is available when, and only when, it is ready for use by her. If "the system is up", but there are no available ports, or any required device is down, or the phone lines are all busy, or the input queue is closed, then it is not available. An access is reliable if, and only if, once granted it remains available as long as needed. Access is dependable if its availability and reliability conform to expectations. You should note in particular that it may not be enough to conform to schedule: you will be judged according to the users' expectations, not yours! You should also note that dependability by itself is not satisfactory: there exist systems which are dependably bad.

The following kinds of information will provide you with the raw material necessary to construct a user's-eye view of system access:

- accesses denied
  - number and kind (devices and services desired)
  - date and time
  - source of access (site, path, user, and method)
  - reason(s) for denial
- out-of-service periods
  - number
  - date/time
  - duration
  - cause
  - number of people affected
- accesses interrupted
  - number and kind (devices and services involved)
  - date/time
  - source of access (site, path, and user)
  - cause
  - cost
    - unrecoverable consumables
    - system time invested in partially completed job steps which must be restarted
    - redundant mount and demount effort
    - system resources dedicated to cleanup and (if necessary) to diagnosis
    - system resources dedicated to backup resources and procedures
    - perturbation of the remaining schedule
    - lost user time
    - severe emotional cost, both to users and to DP personnel

access patterns
    who, when, to what, from where, how, in what sequence
    trends

Interface

The user/system interface is probably the most crucial of the three aspects of the system with which the user is concerned. For while the other two determine the usefulness of the system, the interface determines its personality (or, more likely, impersonality).

At this point a reminder is in order that "the system" includes people, physical environment, procedures, and policies as well as hardware and software. Each of these entities contributes to the interface between "the system" and the user, and so the following discussion applies to all.

The most common complaints about human/computer interfaces are that they are unnatural, mysterious, inflexible, inconsistent, unforgiving, unfriendly, etc. Thus the prescription for a good interface is to remove these deficiencies. It is recognized that the DP department may not have the authority or the resources to replace some of the major existing software interfaces, such as the job control module, with reasonable ones. They should have the authority and resources, however, to mask some of that inherent ugliness with humane policies and procedures.

Where the installation <u>does</u> have control, it should insist upon language which is natural to the user, avoiding such hardware-oriented entities as hexadecimal or octal numbers, or such implementation crutches as Julian date: they are part of no natural language. Remember that there are many more users than implementors; when the interests of the two classes conflict, those of the user should dominate. Devise simple, straightforward procedures; use simple syntax; provide sensible, natural defaults; correct correctible errors. Above all, be courteous. The system should ask, rather than demand; it should be forgiving and avoid arrogance; and it should speak when spoken to: few occurrences are more disconcerting than requests to which the system gives no visible response.

The kinds of information which tell you how successful your interfaces are include such things as

- error patterns
  - kind of error
  - frequency of occurrence, by module
  - kind of user (level of experience, mode of use, etc.)
  - environment
- module use patterns
  - which modules are avoided? why?
  - which are over-used? why?
  - how many are system make-work (such as conversion or re-formatting)?
- the complaint log
  - which complaints are most often repeated by new users? by experienced users?
  - how long does it take to produce a "satisfactory" answer?
- the question log
  - which modules are the most mysterious?
- the refund log
  - what are the most common causes of refund requests?

It is important not to rely too heavily upon mere statistical reports in these areas, especially the last three. The weaknesses in the system are often rather subtle, and their elucidation demands intelligent reading of the data. (This can be an extremely useful side-product of a competent and highly-motivated Consultants' or Programming Assistance group.) Another source of input which can be tapped for information on the quality of your interface is the user community itself, directly, via users' meetings of one kind or another. The success of such meetings, however, depends upon the willingness of the users to speak frankly in public, and the willingness of the DP Department to accept public criticism and to react reasonably.

Product/Result

As I have already indicated , the nature of the process is of relatively little interest to the user, so long as it gives the right result. The "right" result is the one the user really wanted, not necessarily the one he asked for; i.e., the user wants the system to pay attention to what he means. (The difficulty the user encounters in trying to express his meaning in terms the system can understand is properly an element of the interface rather than the result. It is mentioned here to emphasize that the mere absence of system-detected errors is no guarantee of user-oriented rightness in the result.)

In general, the results should be truthful, complete, accurate, never empty (i.e., no process should vanish without a trace), and meaningful to the user. This last is particularly true, but often neglected, for error messages. Results should also be obtainable within a reasonable time and at a reasonable cost.

(There may be a temptation among those whose installations do not impose explicit charges for computing to feel that "cost" is a meaningless concept: not so. The list of subheads under "accesses interrupted: cost" [in the Access section, above] is dominated by costs which exist whether or not the user pays real money for computer time. Nevertheless, it must be admitted that costs which are expressed in real money have a way of attracting the attention of the user...and that one way of increasing the user-orientation of an installation is to make it dependent upon the real money that users are willing to pay.)

The dimensions of "reasonable time" and "reasonable cost" are, of course, highly subjective. What appears as a drop in one user's financial bucket may exceed another's entire yearly budget for computing, while a comfortable speed for the second may well appear snail-like to the first. Within the context of interactive computing, for example, one encounters all four of the following points of view:

1) fastest response is best;
2) response which is too demanding, i.e., too quick, creates tension which causes errors;
3) response which is too slow creates impatience which causes errors;
4) response time should be invariant.

Whereas each of these may suit a subset of the user community, it is unlikely that any one of them will make all users truly comfortable. The situation is further complicated by the fact that the same user will have different preferences in different situations.

There is, therefore, no information which is readily available which will give the Performance Manager a reliable and timely picture of performance with respect to the speed aspect of the process. The best she can do is keep a careful eye on whatever mechanisms are used to reduce delays. Since these mechanisms are often informal, unofficial, and contrary to policy (special treatment by the operators, for instance), this is not always an easy thing to do. In those cases where turnaround time is governed by a priority scheme, a growing use of high priorities is a lagging indicator of dis-satisfaction with the speed of the process.

Suitable performance information addressing the other desiderata of product/result is nearly as difficult to acquire, for most of it cannot be obtained automatically. A good Consulting, or Programming Assistance, staff can provide much of what is necessary, however. In general, the perceived quality of this aspect of the operation can be inferred from such indications as:

- misunderstood messages
  - circumstances
  - kind of user
- missing results
- hardware and software error logs
  - kind of error
  - module affected
  - users affected
  - amount of work lost

<u>System as a whole, or Whatever happened to the Service-Level Concept?</u>

The standard performance management tactic most directly concerned with the performance of the system as a whole is the Introduction of Service Levels, i.e. the establishment of specific numerical criteria for "success" in any of a number of specific performance areas. Unfortunately, the service-level concept is very nearly meaningless in a truly user-oriented operation. I fault it on three counts:

1) it is not user-oriented at all (despite its occasional use of "user production units");
2) it assumes an adversary relationship;
3) it is a classic example of the substitution of numbers for judgement.

Let us take these charges one at a time. First: user-orientation. My notion of a user-oriented system is one which allows the users to be individuals and which addresses their individual problems; a system using service levels is center-oriented and sees the users only as statistics.

Let us try to illustrate this with a familiar example from another field: salad service in a restaurant. One can postulate a number of sub-services to which the service level concept could be applied:

a) accuracy in dressing assignment

b) amount of salad

c) proportion of ingredients

d) extras (croutons, freshly-ground pepper, ice-cold fork,....)

e) delay time before serving

Let us now contrast a first-class full-service restaurant with high service-level goals (all of which are met) with a second-class restaurant with a salad bar. The salad bar allows the user who is so inclined to load up on onions and to skip lettuce entirely, or to have both blue-cheese and oil-and-vinegar dressing, or to select any of a very large number of usual or unusual options, and to do all of this at the user's selected time.

Which of the two is more user-oriented?

The second problem with the service-level concept is its assumption of an adversary relationship between the DP Department and the users. Instead of replacing this adversary relationship with a cooperative one, the service-level concept merely formalizes the nature of the struggle.

This is perhaps a good point at which to insert yet another disclaimer: While I do not consider the use of service levels to be particularly user-oriented in an absolute sense, I recognize that the introduction of the service-level concept often is a first step in an attempt to move an installation from a state of open warfare between DP and the users towards a cooperative program of user-oriented performance management. By formalizing the conflict one can often defuse it. By restricting the conflict to the definition and specification of formal service standards, one frees all parties--once the standards are established--to cooperate in their achievement.

But formalizing the conflict reinforces the notion that conflict is inevitable, and the formal nature of the process encourages the DP Department to strive for the establishment of standards which are certain to be attained--and hence which are equally certain to be sub-optimal. In an adversary relationship service standards are always in danger of being converted from milestones of progress into implements of war: those which are not met become weapons which are hurled at the DP Department; those which are, become defenses behind which the DP Department hides. User-oriented performance management cannot exist in such a state of war.

Finally, we have the problem of the substitution of numbers--statistical service levels--for judgement--an assessment of the <u>quality</u> of the service. The essense of user-oriented performance is user satisfaction. User satisfaction is NOT measured by means of statistical success ratios. Some users are more demanding than others and cannot be satisfied by normally acceptable performance. Other users are quite willing to sacrifice standard service in one area to gain exceptional service in another. No user derives any comfort from being told that the service is above reproach because his is the only output which was lost last week.

How, then, should one attempt to measure user satisfaction? There appears to be growing agreement that a well-thought-out attitude survey is a proper instrument.

(An attitude survey presents the survey population with a set of attitude spectra, such as

ease of use:	simple								complex
attitude of staff:	hostile								friendly
capabilities:	inadequate								ample

Respondents are asked to make one check mark in each spectrum. The method is discussed more extensively by Lyons [2] and Pearson [3].)

The attitude survey is certainly not a fool-proof method: An improperly designed instrument will avoid or even conceal the major sources of discontent. Installation politics may prevent full distribution to significant users. Past experience with DP failure to follow through may have rendered the users too apathetic to be bothered with yet another questionnaire. Nevertheless it offers a better handle on what users think of the quality of service than any purely numerical system can hope to provide.

For some installations there is another, even more powerful, measure of the deficiencies of the system as a whole; namely the applications that the users do elsewhere...especially if company policy makes such foreign relations difficult.

And, finally, here are two general areas of investigation which cover the whole gamut of the system:

- peak (not average) retry numbers (i.e. the maximum number of submissions required to successfully complete a job)
  - type of job
  - reason for resubmissions
  - user(s) affected

- total lost time
  - number of occurrences
  - causes and circumstances
  - users affected
  - approximate cost

General Principle

It should be clear by now that the most useful kinds of information for the user-oriented Performance Manager are the negative bits, the ones which spotlight the holes and deficiencies of the system. The reason for this is twofold. First, the good parts of the system are those which do what the user expects; for that very reason they are invisible. It is the less successful parts which determine the users′ view of the system.

Secondly, it is difficult to improve performance in a meaningful way when you don′t know where the warts are. The job of the Performance Manager is to improve performance. This is not done by highlighting ones successes or concentrating upon ones good features, but by seeking out the deficiencies and awkwardnesses of the system and eliminating them. The model given here of the users′ view of the system, and the collection of the kinds of performance information advocated, should help you to pinpoint those aspects of your system which cause your users the most trouble. It is what you do then that determines whether or not you are a User-Oriented Performance Manager.

References

1. Stevens, D. F., How to improve your performance through obfuscatory measurement, 1978 NCC Proceedings, AFIPS Press, pp 425-431 (LBL-7250).
2. Lyons, Michael L., Measuring user satisfaction: The semantic differential technique; Ethnotech, Inc., 1977.
3. Pearson, Sammy Wray, Measurement of computer user satisfaction; Doctoral thesis, ASU, August, 1977.