LBL-9538

Lawrence Berkeley Laboratory
UNIVERSITY OF CALIFORNIA, BERKELEY

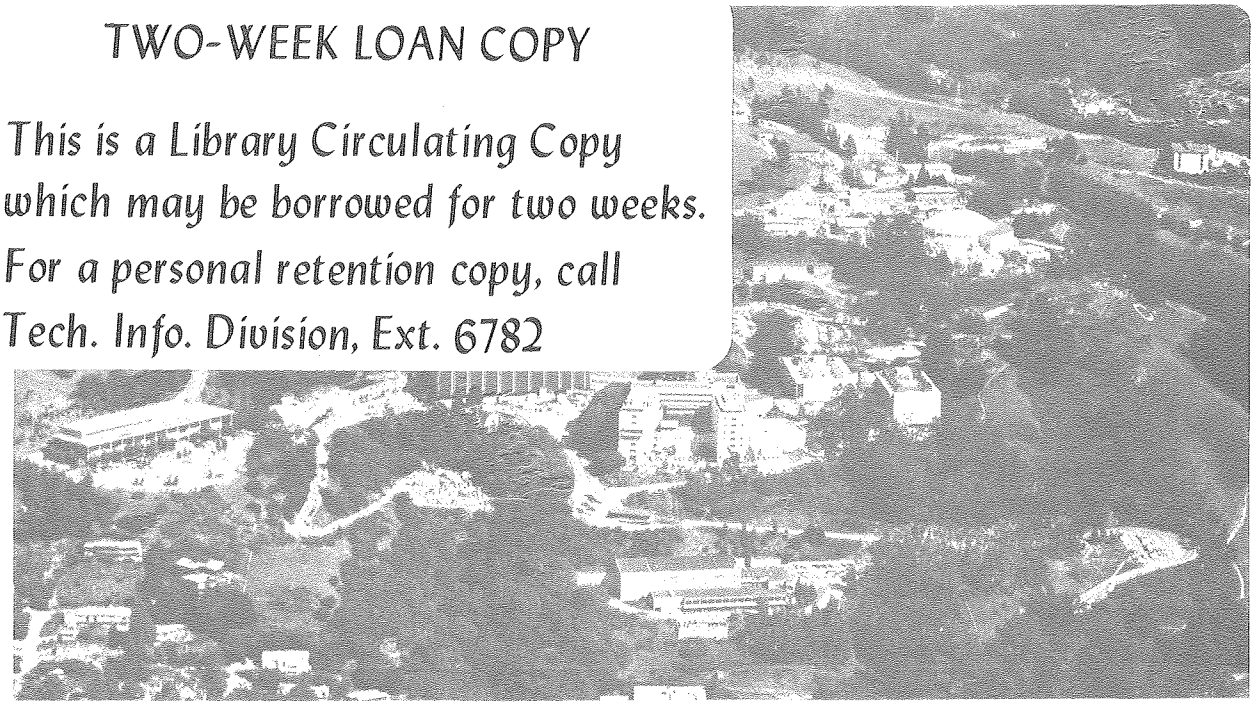
Engineering & Technical
Services Division

To be submitted to the International Federation for Information Processing Congress 80, Tokyo, Japan, October 6-9, 1980 and in Melbourne, Australia, October 14-17, 1980

SOME CAUTIONARY APHORISMS FOR USER-ORIENTED COMPUTER MANAGEMENT

David F. Stevens

July 1979



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

DISCLAIMER

LBL-9538

Some Cautionary Aphorisms for User-Oriented Computer Management*

David F. Stevens
Lawrence Berkeley Laboratory
University of California
Berkeley, California

July, 1979

Introduction: One sign of the approaching maturity of the computing craft is a growing awareness of the importance of the users. A consequence of this awareness is the growing interest in the development of user-oriented, rather than system- or center-oriented computer management. This note is directed to those who have achieved a certain measure of competence in the latter and who wish to make the switch to the former. It attempts to highlight those aspects of User-Oriented management most likely to frustrate the experienced Center-Oriented manager.

Please note that this is not a guidebook to success, but rather a sort of psychological comforter. When, in your attempt to become the compleat User-Oriented Computer Manager, you have suffered a particularly distressing setback, a review of these aphorisms may help to remind you that such disappointments are constant, continual, and universal.

Much of the content of this compendium is drawn from that mysterious well of wisdom known as "common knowledge". That means I generally don't know the source of the specific formulation used for a given aphorism. (Some of them may even be original.) Sources I do know are acknowledged in the text and the references.

---

* This work was supported by the U.S. Department of Energy under contract No. W-7405-ENG-48.

1. The human capacity for random behavior is always greater than expected.

The failure to allow for this aspect of human-ness has caused the downfall of innumerable systems. As an example, consider the presumably simple matter of names. Most (U.S.) standard forms with space for a person's name require, in some order, "first name", "last name", and "middle initial". The following is a list of problems which have been encountered with this traditional format. (It is not exhaustive.) All of them have caused difficulty for more than one system.

1) there is only one name
2) "first" name occurs last
3) "last" name occurs first
4) "last" name occurs next-to-last
5) compound names
6) first name consists of only an initial
7) two (or more) middle initials
8) no middle initial
9) choice of "middle" changes from time to time
10) prefixes (de, von, ..., and also Dr, Mr, Ms, Mrs, Prof, ...)
11) postfixes (Jr, III, ...)
12) non-English orthography (accents, umlauts, cedillas, ...)
13) insufficient space for one or more of the names

This list illustrates a few of the many sources of human randomness, for some of these "anomalies" are of national origin, some are cultural, some familial or regional, and some individual. A user-oriented computer manager must maintain an awareness of this human potential for randomness, and be sure that the design, implementation, and operation of the systems

under her control can accommodate it. Four specific aspects of human randomness which are frequently encountered in the computing milieu are noted below.

1.1 Systems involving humans are non-deterministic.

A deterministic system is one in which the same set of circumstances always yields the same results. A system involving humans contains random elements and so, by definition, is non-deterministic. Moreover the non-determinism is expressed at unpredictable times and in unpredictable ways, not only via unexpected technical activity, but also via unexpected personal response.

For example, human tolerance for the inconveniences encountered when dealing with systems varies widely from person to person, and for any given person it varies just as widely from occasion to occasion. Service which is tolerable under normal circumstances may become unacceptable in times of crisis; a particular idiocy may be acceptable the first eleven times it is encountered but may cause violent reaction at the twelfth instance; a person who is normally placid may become a tiger when his boss, or his girlfriend, is looking over his shoulder.

You cannot be expected to anticipate all of the specific vagaries you will encounter, but you must anticipate that vagaries will occur. If you cannot cope with them when they do, your system may fail, and will certainly fail to please.

A recognized aspect of randomness is the ability to err; humans carry this aspect one step further:

1.2 To err deliberately is human.

Users are especially fond of exploration and of limit-testing. This activity--called "programming by the experimental method"--can occasionally have serendipitous results: some of the more interesting machine instructions in IBM's 704x and 709x computers were discovered by users who meticulously tried all of the "forbidden" and "undefined" possibilities on predecessor computers. Programming by the experimental method can also lead to spectacular failures. The most famous of these in the 704x/709x case is XEC* * ("execute indirect to self" [1]), the execution of which can best be described by attempting to follow the instructions on a card which says

Turn this card over and
follow the instructions found in the
place specified on the other side
of this card.

on the front, and

The other side of this card.

on the back.

The machine got so involved in turning the card over and over it forgot how to do anything else; the only way out of the the bind was to turn off the machine. A system which must self-destruct to undo the consequences of a simple oversight is not very human-tolerant.

Programming by the experimental method is encouraged by two common system defects: poor documentation and unbearable constraints. In the first case, the experimental method is the only way to determine the consequences of an unfamiliar action; in the second, it is employed in the belief that there must be some way to get around a particular limitation. Eliminating

these two defects, even if 'twere possible, would not eliminate deliberate errors, however: Human curiosity and cussedness are just too great.

The limiting case of this phenomenon is

1.3 Everything will be tried.

Finally, although it has been anthropocentrically remarked that randomness seems to be a necessary component of intelligence, we have

1.4 In the game between the system security officer and the fool, the fool always wins.

Parker [2], Gilb [3], and Murphy (in the exposition of [5]) have all noted that fools attack any system more often, and are more difficult to defend against, than criminals. The latter attack only those parts of the system containing something of value, and in ways designed to shield themselves from detection. Fools, on the other hand, attack all parts of the system with equal intensity, and with an ingenuity which is beyond imagination. There is no defense against fools. You cannot predict where or how or when they will defeat your system, but you can safely predict that they will defeat it. Your only recourse is to do your best to keep defeat from necessarily equalling destruction.

Under no circumstances should you ever be swayed by the argument that "nobody would ever be foolish enough" to do something. Human folly knows no bounds.

2. No system should depend essentially upon human reliability.

(This is a slight re-statement of one of Gilb's laws [3].)

This is another facet of non-determinism. Just as your users will act in unpredictable ways, so will those people who are a part of your system. This does not mean that you should avoid the use of people in your system; it simply means that you must take their lack of consistency and perfection into account.

3. Systems run better when they run downhill.

This is a slight modification of one of Gall's observations on Systemantics [4]. It suggests that when your system is helping people to do what they want to do, it has some chance of success. It further suggests that the chance of success will be enhanced if its methods and means (as well as its goals) are sympathetic to the users. Gall's formal characterization of this principle (also in [4]) is rather more pretentious, but nonetheless to the point:

> "Systems aligned with human motivational vectors will sometimes work. Systems opposing such vectors work poorly or not at all."

4a. To a small boy with a hammer, everything looks like a nail.

4b. If you have to drive a nail, every tool looks like a hammer.

The first of these complementary statements is known in many variants, and has been rather widely quoted (see, for instance, Kaplan's Law of the Instrument, in [5]); the second is seldom stated explicitly. Between them they address the extreme philosophies employed when a human has a problem and a tool which are ill-matched.

The boy with the hammer provides a reminder of the human tendency towards a well-known kind of psychological myopia in which our perception of a

problem is distorted by the tools at hand: the fact that we know some statistical analysis leads us to consider user satisfaction a statistical problem; the presence of a hardware monitor encourages us to view a probe point library as the key to system efficiency. A somewhat catchier name for this syndrome, which is frequently encountered in the data base environment, is "product-oriented solution": a "solution" in which the user is required to convert his problem into the particular type of nail for which the system's hammer was designed.

4a also says that the systems you give your users determine the kinds of problems they can solve conveniently, and thus have great influence on the kinds of problems they will attempt. If your users need to do more than just drive nails, you'd better give them something more than a hammer.

4b, on the other hand, reflects the ingenuity your users will display in using systems for their purposes, whether or not those are the purposes the designers had in mind during development. (The fact that Fortran has no character manipulation features does not prevent Fortran programmers from doing character manipulation in Fortran.) And once having twisted the systems to their purposes, the users will proceed to complain, often bitterly, that these misfit systems don't satisfy their desires efficiently.

5. The Sisyphus Axioms

(These nine principles are so named because in sum they appear to lead one to the conclusion that User-Oriented Computer Management is a hopeless endeavor, a never-ending struggle against insurmountable odds. In the sense that the voice of user criticism is never stilled, that conclusion is valid, as these axions proclaim in excruciating detail. But, as I hope

becomes clear in what follows, criticism and quality can co-exist.)

5.1 There is always a user for whom the system is not designed.

5.2 There is always a purpose for which the system is not designed.

5.3 The full set of user requirements is always inconsistent (i.e. cannot be satisfied by any system).

5.4 Complicated systems cannot be forced to work.

5.5 Simple systems cannot be expanded. (Expansion fails.)

5.6 Every feature is in use. (Contraction fails.)

5.7 There is no such thing as a transparent change. (Doing anything fails.)

5.8 Somebody is unhappy. (Doing nothing fails.)

5.9 Advance warning is always insufficient.

Perhaps these axioms can all be summed up in a conservation law:

> In any interaction between a well-managed system and a user community, the amount of user misery is conserved.

Thus whatever misery you manage to remove from one user is distributed among the rest. (Please note that the qualification "well-managed" is a necessary element of this law, for it is certainly possible to introduce misery-amplifying changes into any system.)

These nine statements are not independent, but each contains elements occurring in none of the others. They are so closely related that it is

most convenient to consider them as a group.

5.1 has two aspects: The first is that there are always more kinds of users than you thought about when you designed the system (those character manipulators in Fortran, again); the second is that at least one kind will be more numerous than you thought possible. 5.2 is closely related to 1.1, 4b, and the first aspect of 5.1; it is also included here for completeness and symmetry. 5.3 recognizes that different users have different demands. Rich users, for instance, tend to want instant service, poor ones want cheap service; progressive users want new features, conservative ones want no changes; adventurous users want license, timorous ones want safety. (Lincoln's form of this axiom is the most well-known: "You can satisfy all of the users some of the time, ... ") (5.1, 5.2, and 5.3 are derived from Ethnotech observations [6].)

5.4 is the negative form of the Laissez-Faire Principle ("if it ain't broke, don't fix it"); it highlights the futility of attempting to force a complex system to act according to the designer's wishes. Once a system has achieved a certain level of complexity, it either works or it doesn't. If it works at all, it works according to its own rules. If those rules are not the desired ones, the system must be scrapped and rebuilt from the ground up: any attempt to force it into some other mode of behavior will run afoul of Weinberg's Strong Connection Law ("a system is a collection of parts no one of which can be changed"), or Udall's Fourth Law ("every reform always carries consequences you don't like"). (5.4 is one of Gall's Systemantic Laws [4]; the Weinberg and Udall formulations are found in [17] and [18], respectively.)

5.5 recognizes the futility of simple expansion: Gall's formulation [4] is somewhat less pessimistic, noting merely that an expanded system will act

according to its own rules, and not according to the rules of the smaller system it replaced. In view of 5.4 and 5.8 the present version seems more appropriate here. 5.6 recognizes the assiduousness with which the users apply themselves to the discovery and exploitation of every possibility offered by the system. The elimination of even the least of them will cause at least one user's job to fail. (5.6 is a consequence of 1.3.)

5.7 and 5.8 sum up the apparent hopelessness of the situation: you can neither do nothing nor do anything which will eliminate the misery experienced by your users. It may help to bear in mind the fact that users' definitions of misery change with their expectations and experience. As you improve the system you elevate their expectations; the result is more discriminating users. The more discriminating the user, the smaller the disappointment which will be sufficient to provoke misery.

Your goal should not be the elimination of misery (for quiet users lead to apathetic management), but a change in the kind of event which brings it on. Thus, if a user today expects his job to complete sometime before next Monday, your goal should be first to elevate that expectation to tomorrow morning, and then to two hours from now, and then to two minutes. You will notice that in the latter case a single minute's tardiness will cause the same sort of disappointment as total loss of the job in the first case. You will have eliminated no misery, but you will certainly have improved your system.

5.9 is most well known in its US Navy version, which is of such venerable antiquity that when John Paul Jones uttered his famous rallying cry--"I have not yet begun to fight" [16]--a Marine sergeant in the fo'c'sle was heard to mutter "There's always some dumb s.o.b. who don't get the word." No matter how frequently, forcefully, or forehandedly you try to warn the

users of an impending change, there will always be someone who doesn't get the word, or who doesn't believe it, or who knows it wasn't intended for him.

6. In a saturated system, work to maximize throughput. In an unsaturated system, work to minimize turnaround time.

This is Baker's Basic Law of System Management [7]. It is the rational response to the fundamental change in character which saturation imposes upon a system. A saturated system is one which has more work to do than it can accomplish. (A saturated system is not necessarily full in the sense that all resources are fully committed: a single over-committed resource is enough to cause saturation.) The distinguishing characteristic of a saturated system is the presence of lengthening queues. In this situation the strategy of least discontent is to adjust the system to do as much work as possible. (If you are extremely clever--or lucky--you might even adjust it well enough to eliminate saturation.)

By contrast, an unsaturated system can handle all of the work which is submitted. You can then turn your attention from simply doing the work, to doing it in the nicest possible manner. This normally means adjusting the system to complete the individual jobs as quickly as possible.

Note that reducing individual turnaround time is a far different thing from reducing total turnaround. This can best be illustrated by a slightly contrived example. Consider a system capable of running four jobs at once, and assume that four identical one-hour CPU-bound jobs enter the system at the same time. The best one can do for that total set of jobs is four hours. Most simple round-robin schemes would come very close to that minimum, with the four jobs progressing simultaneously through the system

like a well matched team of horses, and all finishing at about four hours after submission. By contrast, a very old-fashioned sausage-style scheme (which allows only one job at a time to run, but where a job once started runs to completion) would also require four hours to complete, but three of the jobs would have finished much earlier (one at the end of each hour).

The two schemes have identical total turnaround times; the user-oriented scheme, however, has a 2.5-hour median while the other median is 4 hours.

(Do not be misled by the example into abandoning your multiprogramming system, however.)

7. An on-line system is underutilized, overloaded, or both.

This is a law of the excluded middle, there being no possibility of a happy medium; it is a corollary to the Fourth Law of Hebditch [8]. (It also owes something to 5.1.) It derives from the fact that a successful on-line system carries with it the seeds of its own destruction. Success begets popularity which begets overcrowding.

That both attributes can be present arises from the existence of multiple viewpoints. Viewed from the austere confines of the comptroller's office almost all on-line systems are underutilized. Viewed from the bleak wasteland of the users' terminals almost all on-line systems are overloaded.

8. The length and accuracy of a user's memory of any event is proportional to the square of the damage inflicted on the user by the event.

8.1 A user never remembers a beneficial event [9].

Understanding and acceptance of this aspect of Murphy's Law are essential to the continuing sanity of the Computer Manager. It is well known, having found wry expression in popular culture by means of the "What have you done for me lately?" sort of joke, but it is easily forgotten. Whenever we are tempted to consider this attitude somewhat unjust, we should remember that a single system error can wipe out not merely the memory of many successful prior experiences, but also their accumulated results. Eastern Airlines has found this principle important enough for integration into their recent advertising: "We're only as good as your last flight."[10] That is an excellent motto for the user-oriented manager in any field to adopt.

9. Silence connotes apathy rather than acceptance.

While it is true that happy users do not complain, it is also true, as we saw with the Sisyphus Axioms, that it is not possible to keep all of your users happy over an extended period. Continued silence must therefore result from some less desirable circumstance.

User comment is driven by two motivating forces: the experience of distress and the hope of redress or improvement. Of the two, the second is more important, for the first alone is, in the long run, insufficient. Continued inattention will silence even the most persistent of naggers.

One of the things a user-oriented manager must do is keep the dialogue between the system minions and the users on a free-flowing and constructive level. It is when things appear to be going well that you most need a friendly and sensitive user who has some expectation of a respectful hearing, for only such an one will bring you the subtle early signs of encroaching saturation or of eroding service. If you have taught her that her warnings go unheeded, she will take great delight in allowing disaster

to come upon you unawares.

A manager's best friend is a knowledgeable, critical, demanding, vocal user. If knowledgeable, he knows the significance of the signs he reads; if critical, he will not let you slip into careless habits; if demanding, he will inspire you to better performance, no matter how good you think you are now; if vocal, he will give you the benefit of his informed and constructive criticisms. The elimination of criticism is not a proper goal of user-oriented management; the conversion of diatribe to dialogue is.

10. Performance is as perceived not as measured.

Despite Lord Kelvin [11], numbers are no substitute for judgement. That is not to say that one cannot--or should not--measure performance, but that measurement won't necessarily pinpoint the problems the users see. Users evaluate a system by comparing its performance with their expectations and their needs, not by checking its performance numbers. A weak system which is available when needed is better than a powerful one which is never available.

Users' expectations can be unrealistic. A part of your job, therefore, is to conduct a continuing course in realism. You must beware of the temptation to define "realism" as "the immediately attainable". A vocal user corps will help you to avoid this temptation, but only if a good relationship has been established.

11. Systems adapt to the measures used for their evaluation.

This is in some sense the system analogue of Darwin's ideas on the origin of species: the organism which survives is the one which adapts to the

exigencies of daily life. In the world of computing systems those exigencies seem more and more to have such names as "high CPU utilization", "depth of multiprogramming", etc. System changes which enhance these qualities persist, those which degrade them do not, and the system evolves. The following is easily seen to be a corollary to this law:

11.1 Unless the goals (of a system) determine the measures, the measures will determine the goals. [12]

Unless the measurement program is consciously applied as a means to an end the evolution of the system will cease to respond to true needs and, in accordance with 11, will evolve so as to optimize the measurements. This is a technical seduction which is very difficult to avoid.

12. The easier a measure is to obtain, the more likely it is to be misleading.

The measures which are easy to obtain are the ones your manufacturer wants you to take, the ones which make the system look "good" even while it saturates. The easy measures include averages instead of medians and distributions, and percentages of doubtful bases. This topic is covered in more detail in [13].

13. "Pay attention to what I mean."

Users often find it difficult to get us to pay attention to what they say; it is far more difficult to get us to pay attention to what they mean. The problem is that many of us, including users, have trouble communicating clearly. The fault may be in us or in the medium of communication. The User-Oriented Computer Manager must be sensitive to at least two areas

with such a feeling of comfort comes some chance of a satisfactory relationship.

* * * * * * * * * * * * *

L'Envoi: This set of aphorisms is neither complete nor independent; it well may be inconsistent. (That would merely reflect the complexity and fluidity of the task of the User-Oriented Computer Manager.) Much of it is self-evident perhaps even to the point of triviality. But truth is none the less truthful for being trivial, and we often lose sight of the self-evident. Perhaps it will serve as a useful reminder from time to time; I hope so.

References

1. IBM 7040-7044 Principles of Operation, Form A22-6649-2.

2. Parker, Donn: Computer security differences for accidental and intentionally caused losses, NCC-78.

3. Gilb, Tom: Laws of Unreliability, Datamation, March, 1975.

4. Gall, John: Systemantics, Quadrangle, 1975.

5. Dickson, Paul: The Official Rules, Delacourt Press, 1978.

6. Ethnotech seminar on Measuring and Increasing User Satisfaction, presented in Washington, DC, December, 1977.

7. Baker, James: Private communication, 1979.

8. Hebditch, David: The Ten Laws of Teleprocessing, Datamation, November, 1975.

9. This has also found expression as Zimmerman's Law of Complaints in [5].

10. From an Eastern Airlines radio advertising jingle, June, 1979.

11. Lord Kelvin, 1883: "...when you cannot express it in numbers, your knowledge is of a meagre and unsatisfactory kind." Quoted in Lehman, M. M.: Performance Evaluation, Phenomenology, Computer Science, and Installation Management, in Performance of Computer Installations, Ferrari, ed, North Holland, 1978; and in Gilb, Tom: Computerware, manuscript in progress, 1977.

12. Stevens, David: Towards User-Oriented Performance Management, EDP Performance Review, May 1979.

13. Stevens, David: How to Improve your Performance Through Obfuscatory Measurement, NCC-78.

14. Stemple, David: Oral comment at VIM-30, May 1979.

15. Zadeh, Lofti: Quoted in the Preface to Fuzzy Automata and Decision Processes, Gupta, Saridis, and Gaines, eds, North Holland, 1977.

16. In the battle between the Bon Homme Richard and the Serapis; 1779.

17. Weinberg, Gerald: An Introduction to General Systems Thinking, Wiley, 1975.

18. Udall, Morris: Quoted in Faded Fiefdoms, Dennis Farney, in the Wall Street Journal for May 3, 1979.

TECHNICAL INFORMATION DEPARTMENT
LAWRENCE BERKELEY LABORATORY
UNIVERSITY OF CALIFORNIA
BERKELEY, CALIFORNIA 94720