To be presented at the International
Conference on the Performance of
Computer Installations, Gardone,
Riviera, Italy, June 22 - 23, 1978

LBL-7256
c.2

THE NINE-FOLD WAY: A TUTORIAL IN SUCCESSFUL PERFORMANCE MANAGEMENT

David F. Stevens

January 1978

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

LBL-7256
c.2

To be presented at the
International Conference on the
Performance of Computer Installations,
Gardone, Riviera, Italy,
June, 1978

LBL-7256

THE NINE-FOLD WAY: A TUTORIAL IN SUCCESSFUL PERFORMANCE MANAGEMENT*

David F. Stevens

Lawrence Berkeley Laboratory
University of California
Berkeley, California

January 1978

ABSTRACT

Nine Principles of computer installation performance management are formulated and discussed, with examples from current practice.

* This work was done with support from the United States Department of Energy.

LBL-7256

The Nine-fold Way: A Tutorial in Successful Performance Management

David F. Stevens

Lawrence Berkeley Laboratory
University of California
Berkeley, California

January 1978

My intention in the next thirty minutes or so is to provide you with a foundation upon which you can erect a successful installation performance management program. Furthermore, I hope to convince you that though a knowledge of computers and computing might be useful in this endeavor, it is by no means necessary.... For success, as it is generally understood, and as I define it here, rests less upon technical competence than upon administrative excellence.

What, then, constitutes success? A computer installation may be considered as an organism; like any other organism, if it is not in a state of growth it is dying. Success thus consists of growth, and growth in a computer installation has two aspects: amount of installed computing power, and number of people employed. Faithful adherence to the nine principles outlined here will guarantee you a satisfactory rate of growth in both categories _even in the absence of real growth in the institution supported by your computing system_.

First Principle: Seek the advice and assistance of your mainframe vendor.

It is important to realize at the outset that your vendor desires your success as much as you do, for it is in your success that his opportunities arise. His advice on all matters is motivated by his sincere desire to assist you to augment your computing power: his configuration recommendations, choice of operating systems (and even their design), tools for measurement, suggestions on what to measure, are all directed at ensuring that your requirements for computing equipment continue to grow. His company has man-millenia of experience in the utilization of huge numbers of people to create monolithic systems with insatiable hardware demands. Use the manufacturer's systems, do things the manufacturer's way, follow the manufacturer's recommended procedures, and your success is guaranteed.

If so, why the other eight principles? For two reasons. First, it is better to be the master of your own fate than to be totally dependent upon the beneficence of another, however well-disposed. Secondly, it is frequently the case that you cannot rely wholly upon any one manufacturer to supply all of your needs. It is thus always desirable and frequently necessary either to do some of the work in-house, or to coordinate a number of external efforts. It is in those situations, as well as in developing a deeper appreciation for the many contributions of your mainframe vendor, that you need to develop your skill in the application of Principles 2-9.

(As near as I can discern, the First Principle is the only one not applicable to general management situations. Principles 2-9 should thus be somewhat familiar to you from other contexts. But it is precisely the function of tutorials such as this to show you how to apply these general principles in the specific arena of computer installation performance management.)

Second Principle: Increase management activity.

An active manager is one who gets deeply involved in the tasks under her supervision. Such involvement includes fact finding, goal setting, dialogue with subordinates, and other praiseworthy activities. These activities all take time. Under a passive manager they may occupy 0-10% of the total activity of the workforce; under an active manager, especially a creative active manager, they may occasionally take even more than 100% of the total activity of the workforce. (Suitable redefinition of goals, alone, can ensure that.)

In applying this principle to a computer installation, you must remember that not all of the important managers are human: all schedule-creation and other resource-management modules are managers in this sense, and the more active they are, the better for you. The most fruitful areas of overmanagement in current systems are those involving storage: virtual storage systems and garbage collection (even in single-level systems) are especially recommended as areas of involvement.

A subtler form of overmanagement, but one which requires advance planning (for these gambits can rarely be introduced into a running system), is the use of fully-managed, general-purpose, slow-but-sure procedures in place of specific, efficient, special cases. Two examples of this behavior which I have encountered are:

1) backspace implemented as rewind + read
2) execution-time calculation of off-set in such Fortran statements as A(1,1,1) = B(2,3).

Active human management can be invoked in all of the usual ways; anyone with extensive experience in the military or any other large organization is familiar with the possibilities here, so I will content myself with discussing a single tactic which is particularly useful in the computing milieu: the scheduled interrupt. The more interruptions you can schedule, the smaller the duty cycle of the installed equipment, the more equipment you need. Some of you, of course, are fortunate enough to have vendors whose products provide enough unscheduled interruptions to satisfy your needs. If you need some additional interruptions, but are already overworking the two basic justifications (preventive maintenance and system development), here are three less familiar possibilities:

1) reconfiguration
2) system cleanup (some systems help you out here by failing to reclaim released space in various situations)
3) preservation of order and accountability across shift changes.

Third Principle: Complexify.

(This is in some sense the essence of overmanagement, but is sufficiently well-defined to merit individual consideration.)

The more complex a procedure, the less productive is the unit which uses that procedure, and hence the more units--people or machines--are necessary.

This is, of course, the area in which vendor software is unsurpassed. Vendor success is largely a consequence of the Law of Large Numbers (see the Fourth Principle), but there are a number of opportunities open to you, even with your more limited resources. Among the devices which have achieved success in many installations are

1) suggestively incomplete documentation, in which the reader is invited to draw an "obvious" conclusion....which turns out to be false;
2) subtly incompatible contiguous systems, whether contiguous in time or in space (i.e. whether as successive variants running on one machine, or as simultaneous slightly specialized variants running on several);
3) error-intensive syntax, especially for job control:
    a) positional parameters instead of keywords (e.g. RUN,,,,,,,,,A.)
    b) multiple (contradictory) meanings for a single keyword (e.g. R = Read, Ring, Rewind, no-Rewind)
    c) multiple keywords for the same meaning in different places (e.g. SL, E, R all meaning "tape label exists")
    d) letter-number confusion (e.g. O = 0)
    e) long, non-mnemonic, nearly identical names (e.g. BXQZ1A, BXO2IA).

With respect to procedures in such human areas as work submission, I once again bow to the real world: you can do no better here than to emulate your government (whichever one it might be).

Fourth Principle: Iterate as necessary: "Two heads are better than one."

This is based upon the Law of Large Numbers as it applies to the process of system design:

$$O \propto N^2$$

i.e. overhead is proportional to the square of the number of members on the software development team. (This is, by the way, a conservative statement of the Law; some observers claim that the exponent should be $2\sqrt{M}$ or even 2M (where M is the number of managers) instead of a simple 2.)

I believe that everyone is so familiar with the operation of this Principle that further exegetical remarks are unnecessary.

Fifth Principle: Design for the ages.

This may appear to be somewhat off the subject of performance management, but in fact it is not. Design is a significant element of the performance of any system, and full performance management involves management of design. This is too large a subject to do more than touch upon, and it has been the subject of intensive discussion recently. This is another area where you have much to learn from your vendors, for the surest way to develop systems with all of the attributes which contribute to success as I have defined it is the classical vendor approach:

1) smother all problems with numbers of bodies;
2) insulate your designers from all distracting influences (users are distracting influences);
3) insulate your implementers from all distracting influences (designers are distracting influences);
4) insulate your maintenance personnel from all distracting influences (implementors are distracting influences);

5) design every conceivable thing into the system;
6) fix the (complete) design before implementation starts (and allow no redesign as a result of actual experience).

Others are more competent than I to address this aspect of performance management; suffice it to say that if you embrace monolithic, fixed design and eschew incremental design, then your systems <u>will</u> promote installation growth.

<u>Sixth Principle</u>: Direct your system evaluation effort.

Do not adopt performance evaluation techniques indiscriminately. Systems (and people) respond to the measures used for their evaluation. Thus, if you wish to have a fully utilized system, multiply your system manager's base salary by the percentage of system utilization. Complexity is usually related to size; you can guarantee a suitable level of complexity by evaluating your programmers on the basis of lines of code. Choose your measures carefully, and performance will automatically adapt along your selected lines.

<u>Seventh Principle</u>: Name your measures carefully.

(We have now moved out of the province of the management of performance per se into that of the management of the measurement of performance. The management of performance alone is not sufficient to guarantee success. Not only is it desirable that you achieve your performance goals, but you must also provide objective proof of superlative performance. That is the function of the management of performance measurement. Some practitioners go so far as to say that management of performance is unnecessary in an installation which truly understands the management of performance measurement.

The Seventh Principle will be seen to be a subset of the Eighth, but it was practiced long before the other forms of obfuscation came into use, and is therefore considered separately.)

The importance of a measure in terms of the weight it is accorded by the outside world is often a function of its name more than of its content. People will rarely (if, indeed, ever) question the definition of a measure, especially if it has a reasonable and comfortable sounding name. Some of the prime examples of this phenomenon are

1) "availability", which measures scheduled uptime instead of availability to the end user;
2) "MTBI" ("mean time between interruptions"), which measures average scheduled uptime between unscheduled interruptions instead of the mean service interval;
3) "percent saturation", which measures percent of "capacity", under some static, and hence unrealistic, definition of capacity (for capacity is dependent upon workload and scheduling considerations), and which in fact ignores the presence or absence of saturation;
4) "degree of multiprogramming", which counts the number of initiators and not the degree of concurrency.

It makes little difference what you measure so long as its name reflects your purpose.

<u>Eighth Principle</u>: Obfuscate.

Obfuscation is the casting of shadow instead of light. If you are to achieve success, you must demonstrate that your overloaded equipment is used efficiently and effectively. The easiest way to approach the problem is to ensure that high measurements have positive names, e.g. "CP efficiency" for "CP utilization". But that is only one element of the obfuscator's arsenal.

1) measure the wrong things: utilization instead of throughput, MTBI instead of service interval;
2) measure the right things in the wrong way: <u>existence</u> of overlap instead of <u>depth</u>, <u>means</u> instead of <u>medians</u> and <u>distributions</u>;
3) measure things of no significance: average response time.

A general rule to follow is that the easier a measure is to obtain, the more likely it is to be obfuscatory. So take those measures which come readily to hand (which includes nearly all averages and percentages), give them jazzy names, and you're on your way to success!

This Principle, of course, is based upon the fact that while figures maybe don't lie, the truths they tell may be irrelevent, immaterial, insufficient: obfuscatory.

<u>Ninth Principle</u>: Numbers are an acceptable substitute for judgement.

This Principle underlies the success of the previous two. Upper management would rather be swayed by numbers than exercise judgement, for numbers are reassuringly unarguable; you can adopt the same philosophy. Thus, for instance, you need not attempt the difficult task of evaluating the quality of a programmer's work when you can merely count the lines of code he produces, nor attempt to assess the satisfaction of your users when you can point to 99.5% CPU utilization.

Remember your goal is growth, and growth is measured with numbers.

Appendix: A brief reading list.

This is an indicative list which points to a few of the places where some of the more technical aspects of some of these Principles are considered at greater length.

1. The Computer Manager's Guide, D. F. Stevens, DATAMATION, June, 1976
2. The Mythical Man Month, F. J. Brooks, Jr., Addison-Wesley, 1975
3. How to Succeed in Software?, S. Michaelson, IFIP 68 (Invited Paper)
4. Systemantics, J. Gall, Quadrangle, 1975
5. Obfuscatory Measurement, D. F. Stevens, 1977 Sigmetrics CMG VIII, LBL-6115, Rev. 2, July, 1977
6. How to Improve your Performance through Obfuscatory Measurement, D. F. Stevens, NCC 78, LBL-7250, January, 1978
7. The Use of Statistics in Performance Measurement, Parts I and II, G. Carlson, EDP Performance Review, August and September, 1977