# The Place of Self-Repairing Facilities in Computers with Deadlines to Meet

LOUIS FEIN†

## Introduction

FROM the title of this paper, one might expect to find in it exclusively a detailed account of self-repairing facilities in computers. However, the objective here is to identify the *place* of self-repairing facilities in computers with deadlines to meet. The objective is not to discuss details of self-repair. How does self-repair fit in with the host of techniques used and proposed for attaining the performance required of equipment with severe deadlines to meet?

In order to find a place for any activity with respect to related activities, one must set up a framework, a classification schema, allowing for the proper placement or classification of this activity. Thus, this paper contains a classification of used and proposed techniques for attaining high computer reliability, here called performability. In addition, a concept of deadline is quantitatively defined. Finally, the place of self-repair as a technique is identified. Economic implications of these techniques are not discussed here.

## Deadline Situations and Some Practical Examples

"Computers with deadlines to meet" is the theme of this conference. We suspect that what motivated the organizers of this conference was a more specific concern about computers with *severe* deadlines to meet—those computers that must operate in real time, for example. In fact, this example seems to predominate here. But it is clear that all operating computers have some kind of deadline to meet; they are not all severe, but they are deadlines. We recognize that these deadline situations are important, yet the notion of deadline is still rather qualitative. One is thus motivated to characterize such deadline situations quantitatively; *i.e.,* this characteristic situation parameter that entirely describes the deadline situation in terms of its quantitatively defined elements should be isolated, named, and defined, and should have a unit selected for it. Results should, of course, be harmonious with intuition and experience. It would be attractive to be able to assign a number to the severity of a deadline situation both to measure the severity of the deadline and to make a quantitative comparison among deadline situations having different degrees of severity.

A deadline situation contains several important elements. The elapsed time from start to end of a computation is significant in determining the severity of a deadline, and it is intuitively clear that the amount of work to be done during this available time, and the capability and reliability of the equipment are also significant parameters whose values contribute to how we feel about the "toughness" of a deadline. This deceptively simple description identifies those situation characteristics that should be named and defined, and for which units should be selected. Thus, the concept of deadline can be handled quantitatively, in terms of time (the variable usually considered exclusively), and in terms of a relation between relevant characteristics of the computer and relevant characteristics (the application parameters) of the job to be done by the computer. (One of the fundamental problems in the computer field is to identify, name, define, and select units for the characteristic parameters of computers, as well as the characteristic parameters of representative applications, and finally to find functional relation among them. It will be noted later that in the case under discussion here, we indeed have a simple illustration of computer parameters like power and up-time, of an application parameter, like the time available and required to do the job, and of a functional relation among these parameters, namely the deadline ratio.)

It is sufficient for our purposes to borrow some basic concepts from elementary mechanics. One is inclined to name the characteristic that would measure the severity of a deadline the "deadline ratio" or "deadline coefficient," and to define it as the ratio of the time $T_j$ required by a given equipment to do a job, divided by the time $T_L$ available or scheduled to do this job, *i.e.,*

$$\text{deadline ratio} = T_j/T_L.$$

Thus, when a job is to be done within the capabilities of the equipment and within a given time ($T_L$), the situation is characterized by a deadline ratio less than or equal to one; without these conditions, it would be greater than one.

It will be found useful to rewrite the expression for the deadline ratio in terms of machine and time parameters. The work $W_{max}$ that can be done by a given equipment in available time $T_L$ may be written as the product of the equipment power utilized on this job $P_u$ and $T_L$:

$$W_{max} = P_u \cdot T_L,$$

where $P_u$ is the time rate at which work is actually being done by the computer.

The amount of work W required to be done on the job requires time

† Consultant, 431 Ferne Ave., Palo Alto, Calif.

$$T_j = \frac{W}{P_u} + t_d; \qquad t_d = \text{down time.}$$

Thus,
the deadline ratio

$$= \frac{T_j}{T_L}\left(\frac{W}{P_u} + t_d\right) \Big/ T_L = \frac{W}{P_u T_L} + \frac{t_d}{T_L}$$

$$= \frac{W}{W_{\max}} + \frac{t_d}{T_L} = \text{work ratio} + \text{down-time ratio},$$

For a situation wherein the work to be done in available time $T_L$ is to be the maximum work that can be done by a machine operating at a given power ($P_u$), the down time $t_d$ must be zero, and $W = W_{max}$; *i.e.,* the deadline ratio is one.

This is characteristic of many so-called "real-time" situations. In this context, the following remarks are addressed to some problems of the design of equipment useful in situations whose deadline ratio is one or very close to one for both short and sustained periods of time. Let us first note some practical examples of such situations as well as the relevant activities of designers in this aspect of the field.

A most significant military deadline situation with deadline ratio equal to one is the situation in which the SAGE system is designed to operate. Here the value of $T_L$ is large indeed. The central component of the SAGE air-defense system is a digital computer. Radar and other data are accepted and processed by the computer and a complete description of the air situation is prepared and presented to operators. The equipment is interrogated and gives commands to the external environment. In this application, as already noted, the system requirement is for *continuous* operation.

Every branch of the service has requirements for equipments that will operate without error during offensive or defensive operations. Inasmuch as the frequency and duration of these operations are, on the whole, unpredictable (*i.e.,* $T_L$ may have a wide range of values) the problem of designing equipment to meet these requirements is, in a sense, no different from the problem of designing for continuous operation.

Some business-type applications impose requirements that are also severe. If payrolls are delivered late, or if some checks are incorrect, there will be grief, consternation, discontent, etc. When computing equipment is called upon to handle these deadline situations, one is again faced with the task of getting a prescribed amount of work done every day. In commercial-account banking applications there is the double requirement that a prescribed minimum amount of work must be done with the equipment each banking day, and some of the work must be done within certain prescribed hours of the day. For example, laws in certain areas require that checks that are to be refused payment must be returned by a specified time of the working day. Statement day is important to some of us. Here again one can imagine that it is imperative for the equipment to provide a certain minimum of error-free operation for a prescribed period prior to statement day.

In industrial control applications where an operating staff normally controls the factory processes inside the factory, many situations are characterized by deadline ratios of one for a certain period within the working day, which may be twenty-four hours in some cases. Arguments concerning the use of computers in industrial control revolve about the problem of obtaining equipment that can provide the required error-free operation. As long as such equipment is not available, it seems that the proper function of the computer can not be to control a plant directly but only to provide information to operators, managers, foremen, etc. Then if the computer fails, the factory, not being connected to the computer, will not blow up, and the people who are connected to the machine will remain intact and can continue operation and control of the plant on the basis of experience, meter readings, etc.

It is clear that there is and will be a large market for computers in situations characterized by deadline ratios of one for longer and longer durations. The characteristics of computers usable in these situations will also, incidentally, provide a basis for minimizing the cost of maintenance of these computers.

## Performability

From the point of view of sustained performability[1] (usually called reliability), today's electronic-computer systems have been and are erratic, on the whole. The few who warned about the situation were shouted down, on the basis of the presumed initial high cost of implementing facilities designed to increase performability and maintainability. This charge might be proven about a substantial segment of the electronic industry. It would be interesting to know the cost to the Department of Defense for electronic equipment that was not economically maintainable or failed to work at all, in the last ten years. Some published estimates indicate that it has taken from 10-100 times the initial cost to maintain military equipment. The tide is turning, however. What is of deepest significance is the fact that it is presently quite in fashion to be concerned about performability. Whereas people until a year or two ago smiled benignly on the misguided zealots who insisted on the necessity for performability considerations in the design stage, today there is a 50:50 chance of their getting a hearing—in the military, that is. Commercial organizations are only now beginning to pay attention to the importance of performability considerations, mostly because of the belatedly recognized need for equipments to handle situations with deadline ratios close to one, and also because of the now officially recognized high cost of maintenance. Nevertheless, these computer manufacturers' sales departments still insist on and win the point that an

[1] It is recommended that the term "perfectly reliable" be reserved for *components* or *modules* of a system that are "failure-free" and that a *system* producing "error-free" results be designated "perfectly performing."

extra initial cost for performability will make their computers less competitive. It may also be noted in passing that the trend toward the concern for performability is partly a reaction to a situation where electronic equipment could not be operated economically and sometimes not at all, even in situations with deadline ratios much less than one.

This is not to say that there has not been any activity in the performability field, for there has. No doubt, many will recall the differences of opinion that raged in the early days—and newcomers to this field fight the battles all over again—among proponents of marginal checking à la Whirlwind, proponents of built-in checking for "everything" à la Univac and Raydac and later, Norc, proponents of duplication à la Binac, and proponents of programmed checking à la everyone who didn't have other facilities. Diagnostic and exercising routines and other techniques were later added to the arsenal of techniques for increasing performability. In recent years, more and more attention has been given to the development of reliable "long-life" components.

These techniques are categorized under various names: preventive maintenance, built-in checking, accuracy control, programmed checking, etc. Unfortunately, except for the effort to obtain perfectly performing equipments by having perfectly reliable components, connectors, and connections, none of the techniques that were developed and adapted for use in computers were part of an intgrated program to attain maximum performability. Even superficial observation will show that these techniques are each bits and pieces. By themselves, they help to do part of the required job.

To develop this argument, it may be useful to consider a detailed operational definition of performability of equipment. First, consider this definition. A perfectly performing equipment is one that produces error-free results during the period that the equipment is operated at its utilized power $P_u$. It follows from this definition that one would consider equipment perfectly performing if no errors resulted, even though components failed, the design was poor, cross talk prevailed, etc. One would consider that an equipment was not performing perfectly if errors resulted in spite of no component failure but because of an external transient, for example. Thus, the key consideration in the determination of performability from the user's viewpoint is the freedom from error in the results —in the output. It is *not* primarily in those faults of components or design or whatever that are usually responsible for errors. If we assume, on the other hand, that one has a well-designed equipment, that steady-state errors all result solely from components that failed, and that transients do not appear too frequently, then we can restate our definition of performability to include the designer's (and maintenance) viewpoint as follows: (ideally), a perfectly performing equipment is one that produces error-free results during the period that the equipment is operated at its utilized power with $t_d$, the down time, equal to zero, *i.e.*, uninterrupted continuous operation. It should be noted especially that we do *not* concentrate attention here on detailing the activities one engages in during the period designated "down time," but merely on how much time is involved in executing the activities necessary for getting the equipment back on the air. Thus, the implication of this definition is that one must consider that an equipment is performing perfectly—even if errors are actually made within the equipment—when one can do whatever is necessary to correct these errors (before they show up at the output) in zero time. One would have to consider that the equipment is not performing perfectly at utilized power if errors were not made, but there was a delay in obtaining results for some external reason.

If we detail and classify the activities necessary to put the equipment back on the air (the maintenance procedures), it will 1) allow us to write a detailed operational definition of performability from both the user's and the designer's viewpoints, 2) help to classify the activities that workers in the field have been engaged in, and indicate how to evaluate these as contributions to an integrated program to maximize performability, and 3) help to identify, classify, and set criteria for evaluation of any proposal for increasing performability of computer equipment.

## Lost Time—Lost Work

Assuming a fault-free design and a fault-free program, down time on a computer consists of the time taken to detect errors, to locate the fault causing the error, to repair the fault, and to prepare for restarting the computer so that it can continue from the point of error. If there is reconstituting to be done, because the error was detected and the machine stopped during the execution of an instruction beyond the one in which the failure occurred, then one might have not only "lost time" but "lost work." If some work must be redone, then the time to redo this work must be added to the lost time due to error detection, failure location, and failure repair. To take this situation into account, we will modify our definition of deadline ratio (or deadline coefficient).

$$\text{Deadline ratio} = \frac{W}{W_{\max}} + \frac{t_d}{T_L} + \frac{W_r}{W_{\max}};$$

$W_r$ = amount of rework.

= work ratio + down time ratio + rework ratio.

A perfectly performing equipment operating at a given power $P_u$ on a job will be characterized by having

$$W/W_{\max} \leq 1 \text{ and } \frac{t_d}{T_L} + \frac{W_r}{W_{\max}} = 0.$$

Thus, a detailed operational definition of perfect performability is the following: A practically perfect performing

equipment is one that produces error-free results during the period that the equipment is operated at its utilized power with the total time taken for error detection, failure diagnosis, and failure repair vanishingly small, and where the amount of rework is vanishingly small.

### Maximizing Performability by Already Developed Techniques

We are now in a position to identify exactly what part of an integrated program for maximizing performability previously developed techniques are designed to handle. One would like to be able to anticipate, if not prevent, component failures causing errors during operation of a computer. Thus, pre-aging and preselecting components, marginal checking, and test and exercising routines are all examples of techniques for *prevention, anticipation,* and *extrapolation* wherein one attempts to treat or test components under prescribed conditions in order to extrapolate the behavior of the components at a later time. In light of the fact that a large effort today is going into the development of very "long-life" reliable components, one should note that these preventive techniques assume "unreliable" components.

Others of the techniques mentioned earlier fall into one of the three classes: *error detection, failure diagnosis,* and *failure repair.* Parity checking, echo checking, duplicate equipment, and programmed transfers on alternate paths are all designed to *detect* errors in transfer or storage of information. Duplicate equipment, check arithmetic or logical units that work with appropriate algorithms, and program checks are used to *detect* errors in arithmetic or logic. Circuits have been built and programs written to *detect* errors in control and timing and also in storage-address selection.

*Failure-diagnostic* techniques have included the use of programs, component indicators, module indicators, and routine dynamic trouble-shooting techniques. While the *repair* of components that failed is almost universally done by manual replacement, automatic substitution for faulty subsystems has been done occasionally.

### Other Techniques For Performability

We now come to a consideration of those approaches and techniques which if successfully implemented could indeed provide equipments with almost perfect performability and thence make them candidates in situations with deadline ratios equal to or close to one, *i.e.,* for computers with severe deadlines to meet. It has already been mentioned that perfectly reliable components can implement an adequately designed equipment with perfect performability, since then results should be error-free and the lack of component failure would result in no requirement for work rerun.

Error-correcting codes and facilities have the same property. If errors, although they do occur within the system, are corrected almost instantaneously and if little or no rerun is necessary, then systems with error-correcting facilities satisfy the criterion for practically perfect performability. Note that when the errors that are corrected are caused by components that failed, one can live with these in the equipment.

In the work of Moore and Shannon [1], and also of von Neumann [2], on the design of reliable systems from unreliable components, techniques are discussed that also have this characteristic in common with two other techniques mentioned below, namely that the correctness of the results is indifferent to faulty components that remain in the system. Thus, the equipment satisfies the criterion for perfect performability in spite of faulty components. Such circuits we shall call "indifference" circuits.

Other indifference circuits [3] are those characterized by a quad of diodes substituted for a single diode wherein the circuit behaves even with some diode failure exactly as a nonfaulty single diode would behave,

Another indifference circuit [4] has enough component redundancy so that output values coincide with the value of a majority opinion from among individual circuits performing identical functions. Here again the system is immune to some component failures, and erroneous results are not produced even though some components may have failed.

### Self-Repair

In every approach mentioned above, except for the one of trying to get perfectly reliable components in the first place, it is assumed that there will be component failure, and an attempt is made to design around this by selecting a kind of redundancy that makes the system immune to component failure, *i.e.,* the faulty components remain in the equipment. Usually, we replace faulty components. The question arises as to whether or not we can satisfy the criterion for almost perfect system performability and still replace, rather than live with, faulty components. A self-repairing facility with the characteristics described below will do just that.

Consider a computer made up of a number of standard module types. Build in sufficient redundancy in equipment and code so that the error-detection function is performed; *i.e.,* storage, transfers, arithmetic and logic operations, address selections, timing, and control are checked for accuracy instantaneously. Error detection is made an integral part of the design. Design each module in such a way that the dynamic operation of each module is continually and automatically monitored, and failure is instantaneously sensed and indicated. When a failure is diagnosed, automatically switch in an appropriate "hot" spare module that is in the rack, and automatically restart at the beginning of the micro-operation that was being executed when the error was detected. If the switch fails, arrange for a spare module to be switched into a predetermined position by having the switch that failed so designed. (It may be noted incidentally that the "tilt" indi-