Strategies for problem prevention

by J. Newton

A philosophy of preventing problems from occurring in a data processing installation rather than reacting to problems is becoming increasingly necessary. The institution of comprehensive and formally managed testing strategies is an important step in this direction. Such strategies are discussed, and it is shown that they also support disaster backup/recovery plans.

The increasing dependence of business enterprises upon information systems, particularly those which have a high on-line component, means that short-duration system outage incidents (the type of incident normally referred to in discussing decreased availability) are having an increasing financial impact upon an increasing number of areas in many business enterprises.

There are a number of categories of problems, the incidence and impact of which will increase due to increases in the complexity of system and application interactions and the load and stress on these systems. If we broadly categorize problems causing short-duration system outages into hardware, software, operational support, and environmental categories, of these, the common experience is that the software problem category [applications and subsystems such as Multiple Virtual Storage (MVS), Information Management System (IMS), Customer Information Control System (CICS), etc.] has the highest impact and is the hardest for the Management Information Systems (MIS) function to address effectively. This area therefore provides the greatest potential for improvement in application availability and thereby improved financial return from the applications integrated with and supporting the activities of the various segments of the business enterprise. Discussion of the various problem categories can be found in the literature.[1]

In addition to the problem of short-duration outages, exposure to long-duration outages exists because of the possibility of various forms of disaster. Again, because of the increasing dependence of business upon MIS services, the financial exposure here is rising at a rapid rate and is already such that in some types of business, occurrence of a disaster without a disaster backup/recovery capability would lead to failure of the business.

A primary point of this paper is that there is a nexus between high availability and disaster backup/recovery in an area—comprehensive testing—critical to both, which in the case of high availability prevents software problems and in the case of disaster backup/recovery assists significantly in proving the availability of necessary equipment, software, and data (backup) and also in proving recovery capability. Comprehensive testing on a test bed driven by a Teleprocessing Network Simulator (TPNS) at a second site is discussed as a key strategy for addressing these two major issues in common.

The author recognizes that establishment of a second site with equipment suitable for comprehensive testing and disaster recovery is an expensive process and that in the case of the majority of installations it is difficult to justify in operational or tactical (budgetary) planning cycles.

The second site test bed approach pointed to throughout this paper can be justified, but only with

the involvement of senior business management. The issues of availability and disaster recovery are addressable for most installations only in strategic time frames (three years or more) because of financial reasons and the lead times involved in planning and developing a second site. This paper proposes the second site test bed solution as a model or ultimate approach to addressing the availability and disaster recovery issues. As is normally the case in dealing with major issues, for most installations the process of achieving this goal would be via a series of steps over a significant period of time. It is worth noting, however, that over that period of time the availability and disaster recovery issues will become more critical, and the ultimate resources required will become increasingly easy to justify.

There is no intention in this paper to get into any detailed discussion of contingency planning and its elements since these are addressed elsewhere.[2] Discussion of this issue, as with availability, will focus upon the significance of comprehensive testing.

In this paper we will

- Propose that Information Systems (IS) organizations now and in the foreseeable future need to develop a significantly greater emphasis on the prevention of problems as distinct from the "reactive" mode of problem resolution.
- Propose that a (arguably *the*) key to software quality and systems stability is the institution of comprehensive and formally managed testing strategies. The author suggests that the development of such a methodology, supported by the necessary management practices, is no longer optional in installations where there is a significant on-line load either presently existing or in the process of emerging.
- Show that resolution of the key technical issue of disaster backup/recovery is also supported by the development of a comprehensive and formally managed testing process.
- Support the view that communication with senior business management and end users on high availability and disaster backup/recovery is essential and should be framed in financial terms and illustrate some approaches to doing this.

The necessity for increased emphasis on problem prevention

In many installations it is and will be increasingly necessary to emphasize addressing the availability issue by problem "prevention" techniques and to move away from the "reactive" approaches often inherent in the methodologies, resource allocations, and management practices of organizations using MIS.

Problem prevention from a technical viewpoint. The technical need for a problem prevention approach is driven by the fact that the higher-impact software problem categories are simply not amenable to the reactive approach to problems. Stress-induced software problems are a case in point. Examples in this category are regularly encountered by large users of on-line subsystems such as IMS or CICS. In such subsystems, stress creates a variety of problems. They range from the well-known capacity problems caused by running out of a resource such as buffers or virtual storage to the subtler type of problem exemplified by the situation in which incorrect subsystem software code that prevents deadlocks from occurring is invoked and fails, leaving the resource control chains scrambled so that a subsequent request for a resource finds a "should not occur" condition with a consequent failure of CICS or IMS.

The stress-induced category of problem is particularly intractable.

Deadlock handling code is one type of code (restart code is another) that is more prone to fail than other types of code because of the difficulty of testing it in development laboratories in a comprehensive way. If one pauses to consider the vast number of environments, varying levels of software maintenance, differences in operational environments, and all the other factors that make each MIS installation unique, it can be seen why intractable software defects induced by stress will continue to occur. These defects will occur even if improved self-diagnostic facilities are built into the code and "fail-soft" technologies are used.

The stress-induced category of problem is particularly intractable in that the resultant diagnostic ma-

terial (e.g., a dump) will often reflect, not the *original source* of the problem (incorrect code), the traces of which have vanished, but a second-level problem (inability to use resource chains because they have been scrambled). Such a problem may take months to diagnose, since traces have to be built and tested, recurrences of the problem have to be analyzed, interaction with maintenance teams has to take place, and so on. The consequence of such a problem is that over an extended period of time the IMS or CICS subsystem will fail on a regular basis (often at the worst time—the time of greatest load and stress) and availability targets will not be achieved. Human productivity is impacted heavily, and further software change is delayed. A vicious circle develops and availability falls even further.

The category of problem normally referred to under the topic of performance is often likewise a high-impact one, not so much because it can be difficult to diagnose given the right kind of skill but because its presence is often not evident until its impact has been significant. Its diagnosis and resolution can be lengthy.

Similarly, regression problems, if not always intractable to resolve, may appear in tested components as a result of a change in some other component. This type of problem is difficult to prevent by testing methods that concentrate upon individual software components.

Other classes of problems exist which are likewise difficult to address. One such class is to be found in the operations area where, in complex systems, the sheer volume of messages—amounting to a high background "noise"—makes it probable that system conditions leading to failure or performance degradation will go unrecognized and eventually create high-impact situations. Automation of operations can assist here, but to date there have been few generally available systems that substantially address this area.

Problem prevention and increasing complexity. In many installations the complexity of systems will escalate because of the widening range of services provided, their increasing interdependencies, and the rising loads upon them.

In this context it is worth noting that ten years ago few installations had peak transaction loads exceeding five per second. Many are now in this category. Some are of the order of one hundred per second, and a few can discern a future in which one thousand per second must be managed. Examples of the latter are most evident in the finance and retail industries, where the impact of Electronic Funds Transfer/Point of Sale (EFT/POS) operations on a major scale is widely predicted. (Note: These huge peaks are caused by the pattern of retail trading, in which large proportions of business are transacted in a few days of the year. Instead of the normal peak-to-average ratios of 1.6:1, ratios of 5:1 or even 10:1 are probable.)

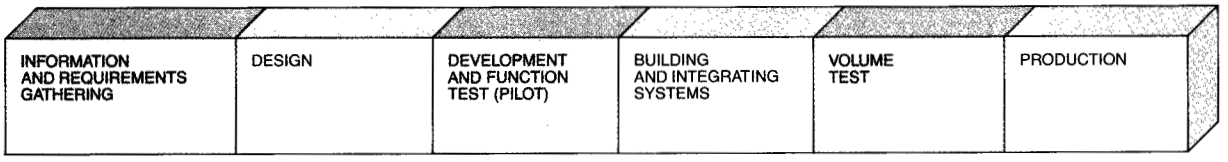
The reactive approach to dealing with problems will be less and less effective.

Note also that software subsystems are themselves increasing in complexity in attempting to deal with the emerging issues. The functional capability of data base/data communications systems is increasing, for example, in order to deal with availability and continuous processing issues. Increased function in these systems breeds its own complexities and, thereby, instability.

Problem prevention and business environment factors. In addition to the above, the nature of business itself is changing to adapt to the increasing expectations that business customers are developing. For example, this change is reflected in extended on-line operating hours, which have the effect of lessening the amount of time available for batch processing and software and hardware changes. Continuous processing (seven days per week, 24 hours per day) or at least extended processing hours are becoming more probable in a wider range of industries.

Behind these technical developments lie the business needs which increase the use of Automated Teller Machines (ATMs) and convenience banking of many forms, drive the need for global networking among banks in international financial dealings, and cause the steady extension of business hours in the retail industry.

Figure 1 Processes involved in issue/problem resolution



Problem prevention summary. A combination of many factors will tend both to destabilize systems and to make more acute the impact of short-term outages. Even given the efforts of vendors to improve the function of software, to use development techniques of a more structured form, and to improve the self-diagnostic capability of the software, the reactive approach to dealing with problems (particularly software) will be less and less effective.

The potential financial impact of a disaster will rise. The time to recover from disaster will decrease, and the premises upon which contingency plans need to be based will change. These factors point to the need for increasing emphasis upon problem prevention and for powerful tools to implement the problem-prevention process. In the key areas of software stability and disaster backup/recovery, the need for comprehensive testing as a major tool in problem prevention and for assurance of the workability of contingency plans will be increasingly apparent.

Because of the length of time involved in fully developing such a capability, there is a strong argument for at least taking the initial steps in the short term and planning for the more advanced elements in strategic time frames.

Another way of putting the point is to say that comprehensive testing should be considered within the context of the questions posed by MIS installations regarding the availability and contingency planning issues. These can be paraphrased as follows:

- What is the financial impact of short- and long-term application unavailability? In the case of short-term application unavailability, an effective and well-documented approach can be found in an IBM publication.[3] In this paper an approach to looking at the financial impact of long-term outages is discussed.
- What problem prevention techniques are available to counter the higher-impact problems? "Techniques" in this context refers not only to technical methods and tools but also to management practices and information for decision making, together with MIS organizational structure.
- How can these problem prevention measures be implemented?

An approach to examining the short- and long-term outage issues

Since protecting a business from the financial impact of short- and long-term outages of application systems or vendor subsystems such as IMS, CICS, etc., is possible at every stage of design and implementation, it is useful to remind the reader of what these stages or processes are in order to provide a context for discussion.

The MIS process steps are illustrated in Figure 1. If we look at the development and implementation of systems at each of these stages and ask ourselves the question, "How can we improve each stage to address the short- and long-term application unavailability issues?", a number of points will emerge:

- We are missing relatively simple chances to prevent application outage, most often not because the possibility is not recognized, but because it would require a change in practice and is regarded by the person who identifies it (normally a technical rather than a management person) as being "too hard" or "too costly" (without that person having the time to gather the data to support or disprove these initial thoughts).
- There are major deficiencies in the process of implementing function which are difficult to grasp because they surface in every part of the MIS function. In these cases powerful justification is required for acceptance of methods that would remove the deficiencies, not only because of the resources involved, but because of the wide-reaching ramifications for MIS as a whole in implementing solutions.
- The simpler and less powerful methods of improving the short- and long-term outage situation often

are still basically reactive, while the preventative methods require resources that are powerful and more difficult to justify and that, moreover, require major changes to the way MIS operates.

The more powerful methods must be identified for longer-term (strategic) planning purposes. Methods of determining their costs and benefits must be implemented. The less powerful methods should also be understood and implemented, but only as seen against the background of the more powerful methods and used as a complement to them.

Prevention of some categories of problem will be possible within tactical decision-making time scales.

Let us use the model in Figure 1 as a vehicle for defining some of the methods of addressing the availability and disaster recovery issues, taking each of the processes in turn.

Information gathering and design. In order to address the issues at the design level, it is necessary to know what level of availability is required for specific transaction sets within the application and what time is available for recovery from disaster by transaction set. By defining availability Service Level Agreements (SLAs) with the user at the information/requirements-gathering stage and basing these on transactions within the application, it is possible to have an effect upon availability and the time available to recover from disaster at the design level. A designer knowing these parameters can then direct his design to the critical functions with both availability and disaster recovery in mind.

Examples of design which take account of these lead to

- Ensuring that key transaction functions are duplicated in lower-level processors. An example of this type of processing is to be found in designing ATM function, where the basic cash dispensing and deposit acceptance transactions are implemented both within the host and in an IBM 4700 level of processor. This enables very high availability for these key functions and can extend the time available to recover from disaster for the application as a whole.
- Designing the duplication of credit card checking into a store-level processor where the checking is done against a "hot card" file for items over a certain value in retail systems where credit checks are done for all host-processed transactions (zero floor limit). Again, the availability of the host and the speed of recovery from disaster become less critical for the application as a whole.

The above may be fairly obvious. The point, however, is that the need for such design information for availability and disaster recovery is often not reflected in requirements gathering from the users of such techniques as SLAs at the predesign stage.

From the perspective of MIS management, there is a need to get information on the impact of short-term outage in financial terms as a precursor to decision making (both tactical and strategic). A good method of determining the financial impact of short-term application outage is to be found in Reference 3. This answers the question, "What is the value of availability for my existing applications?" This type of exercise is essential to provide the catalyst and justification so that MIS management can focus upon the availability issue.

Beyond this phase of assessment there is then the need to answer the MIS management question, "Where do I put my resources to get the most benefit (best return on investment)?" To answer this question it is necessary to design a problem management system which enables categorization of problems by *method of prevention* and financial impact of the problem category. This financial impact is measured not only in terms of user impact, but also of the impact upon MIS productivity of people and the slippage of schedules.

Such a reporting mechanism enables management to look at each category in turn to determine which category, if addressed, would provide the best return on investment. Typical "prevention method" categories would be stress testing, virtual storage capacity monitoring, regression testing, function testing, uninterruptible power supply, maintenance of software at current levels, DASD space management, etc.

Prevention of some categories of problem will be possible within tactical decision-making time scales (e.g., annual budgets), whereas others will be addressable only in strategic time frames with input from users and senior business management.

An information gathering process of fundamental importance to addressing the contingency planning issue is that which assesses the financial impact of disaster in senior business management terms.

In the author's experience the disaster recovery issue cannot be resolved while it remains solely in the province of MIS, principally because the resources which may ultimately be required are outside the range of MIS tactical (annual) budgets, and because MIS in looking at the issue as a sole agent cannot *credibly* assess the financial exposure represented by disaster without recovery capability. Senior business management will not—with good reason—accept MIS assessments as to business financial exposures, since this is clearly the province and area of expertise of senior business management itself. Moreover, MIS management cannot judge the degree of "insurance" that the enterprise can provide to ensure recovery from disaster.

A powerful and simple method to acquire this information is for a representative of MIS to go to senior business management and explain the exposure to disaster inherent in MIS operations and to raise the important point that the senior business management level contains the "owners" of the applications that MIS runs to support the business segments they manage. The MIS representative should point out the fact that disaster recovery is potentially a business *survival* issue rather than one in which risk analysis techniques are relevant. Risk analysis techniques (financial cost of event multiplied by probability of event equals exposure) are not appropriate where business survival is at issue. This point can be illustrated by looking at the financial cost of losing an entire enterprise valued at, say, $1 billion as the result of an event whose probability of occurrence is 10 million to 1—an exposure of $100. Risk analysis techniques are relevant where survival is not an issue.

One initial question to be answered is whether key application loss (individual or collective) would result in loss of the enterprise. To this end an important step is to interview the senior business management "application owners" to determine the business costs associated with application loss for varying periods of time. The following example illustrates the output of such a review.

In the case of a particular business, a total of approximately 40 applications were identified and 15 executives were interviewed. Of the 40 applications, eight turned out to be key ones. The impact of unavailability of these (applications A to H) is illustrated in Figure 2.

Figure 2 Business impact of application loss

DURATION OF OUTAGE	BUSINESS COST OF APPLICATION LOSS				
	< $100K	$100K–$1M	$1M–$10M	$10M–$100M	> $100M
UP TO 4 HRS	ABCDEFGH				
UP TO 1 DAY	CDEFGH	AB			
UP TO 3 DAYS	CEFG	H	ABD		
UP TO 1 WEEK	CF	EG	BDH		A
UP TO 1 MONTH			CEFG	BDH	A
UP TO 3 MONTHS			C	BDEFGH	A

Such review material makes clear the magnitude of the exposure faced by the business. In some cases it is immediately clear that the enterprise will fail if there is a disaster without backup and recovery capability.

Such a review in its own right may well produce the justification for a second site and associated equipment in the judgment of senior business management. From this review will be derived the exposure level and from that the level of "insurance" that business management is willing to pay.

For many businesses the reconstruction time (without a comprehensive contingency plan supported by the necessary resources) will, even if it is possible, be too long not to have at least a massive impact upon the business.

In some cases the business impact will *appear* containable via some relatively inexpensive approach (e.g., reciprocal processing) when in fact it is not if the review looks beyond *today's* situation three to five years into the future, taking into account the changes that will take place in that time (review what has happened over the last five years).

The premises upon which a contingency plan is to be designed, built, and tested must hold true for a long time. If the approach chosen is a "shell site" provided and maintained by an independent operator, contractual obligations will lock the installation into this approach for some years.

In the case of reciprocal backup, similar obligations apply, with the added problem of having to synchronize the MIS planning of the installations involved over a long period. Dealing with one's own MIS plan is a difficult enough problem without having to integrate this plan with one or more other MIS plans.

Building the contingency plan around mutual backup between two or more regional processing complexes may be a reasonable approach, particularly if partitioning of the load can be achieved so that all critical applications are in production at both sites. Even here, very careful consideration of the future must be made, since, for example, the advent of continuous processing at either or both sites obviates the possibility of practical testing of the (two) contingency plans.

Unshared second site premises may appear expensive when viewed only as a contingency plan. This may well not be the only reason for a second site. The contingency plan may be integrated, for example, with a plan to move development and systems programming functions into the second site at which development and complete testing are the main functions. These functions can normally, in most cases of disaster, be regarded as expendable for a limited time. The availability issue can be addressed by complete and formal testing of software at the second site, the disaster backup complex being an ideal vehicle for tests done at the higher levels of load (stress, performance, capacity, and regression testing as well as disaster recovery testing itself). There will be more discussion of this point later in this paper.

Development and function testing groups can assist in improving the quality of code by a variety of techniques.

From a problem prevention perspective, the point here is that the design of a contingency plan is dependent upon a range of current *and future* factors which influence the viability of the basic approach chosen. Also, only the unshared second site approach will provide a strategic solution for a number of MIS installations, particularly those who have or will have significant on-line networks and where continuous processing or extended processing are possible.

One other key item of information that comes from the review is the *time available to recover.* As seen in Figure 2, application A must be recoverable in less than three days for the business to survive. This time becomes a key design point for the contingency plan. To ensure a three-day recovery the contingency plan would need to target a lesser time. If, for example, two days proves impracticable, the alternative is to design the critical elements of the application so that they can "keep going" independent of the host for a longer period of time.

Should the management of a business decide to do nothing regarding the disaster recovery issue after exposures have been identified in the review, an auditor will often request that the risk be accepted explicitly, via risk acceptance documents signed by each application owner (senior business manager), and that the review be done regularly to reassess the (increasing) exposures.

Development and function testing. In the context of availability, development and function testing groups can assist in improving the quality of code by a variety of techniques. In the development area, techniques such as structured programming, walkthroughs, standards, etc., will assist. These have been thoroughly discussed in the literature. In the functional testing area there is often room for improvement. Techniques that assist the analyst to define comprehensive matrices of function versus test cases can be implemented, and the test case product can be automated via a medium such as TPNS. The discipline of comprehensively defining the functions (normal and abnormal) to be tested will improve even manual function testing.

Building and integrating software releases. In order to test software (subsystem and applications) comprehensively to ensure the software stability that many installations find to be at the heart of improving availability levels, it is necessary to create a process that produces a suitable target for software testing.

At present, the targets of testing are frequently restricted to individual software components ranging from applications to vendor subsystems such as MVS, IMS, etc. These components are tested functionally to a greater or lesser degree in environments which are continually changing—environments which exist on development and test machines. (In the case of subsystems this functional testing is sometimes referred to as installation testing.) After this functional testing, applications may be tested by some process involving nonreproducible "volume" tests to which the input is manual and difficult to control. Then the applications are allowed to "settle down" over some indeterminate period of time in the development and test machine environment prior to cutover to production.

With use of such a method it is impossible to effectively test subsystems at anywhere near the load that will be experienced in production. Moreover, the configuration of the software environment surrounding the tested subsystem or application bears little resemblance to that supporting the production environment. To call such a process "testing" is a singular leap of imagination (perhaps optimism may be a better word). The primary deficiencies of this so-called testing process include the following:

- The testing is restricted to functional testing at best.
- The environment for testing is usually in a state of flux so that the results of tests done two days ago are invalidated by yesterday's change.
- The target of testing is not the system that will finally go into production but fragments of it independently tested at different times in different environments.
- The testing is manual and therefore unreproducible, so that even if such testing does reveal problems, there is no guarantee that a fix has in fact resolved the problem even if retesting does not reproduce it.
- Manual testing at any level of volume is difficult to organize, can take long periods of time, and is often invalidated from one test to another by changes in the background environment.

In order to overcome these deficiencies, the requirements are first that a static target be tested, second that this target contain all elements of the final production software configuration, third that the tests be reproducible, and finally that the testing be automated. The only process that fulfills these requirements includes building a software release that contains all the application and subsystem changes to be introduced in a given period in a package, uses this as a static target for testing, and is driven by TPNS at variable levels of load.

This "build and integrate" process creates the software release that is to be the target of testing. This process must be integrated with supporting change and problem management systems for optimum control.

A reasonable question that is often asked is "Accepting that applications function testing is the responsibility of the MIS installation, why cannot the vendor use the practices you suggest to eliminate volume-induced and functional problems in the subsystem software he supplies?" The answer to this is that the collections of software making up the total software systems in larger installations are sufficiently complex that an almost infinite number of permutations and combinations of configurations is possible, and in fact every installation of any complexity is unique.

The best that the vendor can do is to test his system functionally in a very limited range of environments. Moreover, the situation is in practice not improving a great deal. Though progress in software development methodology is continually being made, offsetting factors such as complexity and stress together with the sheer difficulty—from the vendor's viewpoint—of comprehensively testing the almost infinite number of permutations and combinations of software collections to be found at varying levels of maintenance in the larger installations ensure that software failures will continue to occur at much the same or an even greater rate for some time.

Volume testing needs to be done by the MIS installation itself.

Should the reader feel that this claim is unwarranted we suggest the following. Look at the trend in application software failures in your installation and also, where this is available, look at the trend in subsystem software failures. From the viewpoint of vendor software the customer will normally have access to the information defining the number and severity of software problems found in *production* installations around the world. In our opinion, it will be found that the number of reported problems of high impact that are difficult to resolve (e.g., stress-induced problems in IMS, CICS, etc.) and difficult to prevent by conventional testing is rising.

In practice volume testing needs to be done by the MIS installation itself. The implication of this fact is that, particularly in the case of complex system users each of whom will have a unique environment, comprehensive testing of total software releases using corequisite automated testing tools is an increasing necessity. Therefore, the proposition is that there should be a formally managed test environment supported by the requisite tools and methodologies.

By formal testing is meant testing that is formally managed (using explicit criteria and controls for entry and exit in an environment where testing standards are well defined) and that is on a stable rather than shifting system base. Changes are not made to the system during testing unless necessitated by problem detection.

A formally managed test process should recognize the need to emphasize problem prevention and the need to implement complete testing. In addition to function testing, the process will provide for stress testing, performance testing, capacity testing, regression testing, testing of key operational procedures, and disaster backup/recovery testing. The process should recognize that there must be objective criteria for exit from each level of testing done and that these will be enforced; e.g., exit from stress testing will not be complete until X problems are found in IMS and Y problems are found in CICS where new releases are being tested. In addition, whatever the criteria, exit from testing will occur *immediately* when the criteria are met (no indeterminate settling periods) so that high productivity in the change process can be achieved.

Another key point is that testing should be done against a static target, and that target should duplicate as closely as possible the production environment in which it is ultimately going to run. In order to ensure that the software system to be tested is protected from *ad hoc* changes, the testing process should provide that the test libraries are securely protected and under control of a test group who act as a filter for change.

Major improvements in the management of software change are possible, and while improving the efficiency of MIS generally, they will have a particularly marked effect on the productivity of systems programming groups, potentially reducing the time for implementation of subsystem software from some months to a few weeks. Change and problem management are essential tools for ensuring the success of the approach. Volume testing using a fixed release provides the control necessary for effective change management, since only what is contained in the release will go into production. This means that release build and test cycles must be relatively short (e.g., monthly). This further points to the need for automation of testing.

Problem management is the key to validating the effectiveness of testing practices and to pointing the test group at the categories of high-impact problems which most often occur, thus providing the material to allow the test group to focus upon them.

The development of a release approach to software implementation where all software changes (subsystem and application) are made at one time and tested together without further change through all the levels of complete testing leads to a phased approach along the following lines:

- A "development" phase where unit and function testing are done using a comprehensive test methodology on individual software elements (application and subsystem software).
- A "build" phase where, having passed these initial (low-load) levels of testing, the subsystem and application changes are collected together into a release. An appropriately implemented change management system should be used to ensure that no changes to the release were subsequently made. Any changes to be implemented after the release build date had passed would have to be put into the next release.
- A "volume test" phase where the release is tested through all levels of volume testing. The testing would, for the most part, be done at levels of load equaling or exceeding the levels experienced in production, using copies of *production data* and *production transactions*. The type of tool needed is typified by TPNS.
- Production acceptance. No testing is required here, but it is necessary to have controls that ensure that only the tested release is taken by operations. This mechanism could be the test group's final fully tested release to which only the test group had access. Operations would accept software changes only via releases signed off by the test group.

Regression testing is often poorly done.

Volume testing of software

The elements of volume testing. Assuming that function testing has been performed as part of the development process and assuming also that a build process has collected together all application and subsystem software into a release which can be used as the (static) target of testing, a TPNS system can be used as a driver system, using copies of production data and transactions collected from the primary processing complex as the medium for testing the new release.

The levels of testing to be done in the volume testing environment (to ensure complete testing) can be defined as follows:

- Regression testing—Regression testing is done to ensure that modifications which have been function tested do not impact existing unmodified function.

  The types of problem we are trying to prevent are exemplified by the following:

  - Tests to ensure that code supporting a modified transaction still interfaces correctly with unmodified transaction code.
  - Tests to ensure that implementation of a new release of IMS/DB does not affect the CICS interface with IMS/DB.
  - Tests to ensure that application of maintenance to MVS does not affect the IMS or CICS interfaces to the Virtual Storage Access Method (VSAM) or Virtual Telecommunications Access Method (VTAM).

  Regression testing is often poorly done because it takes place in an environment that does not reflect the production environment in terms of software levels and configuration.

- Operational procedure testing—This type of testing is done to ensure that a function or group of functions can be supported operationally. Examples of such testing are

  - Tests which ensure that procedures validating the integrity of data operate correctly.
  - Tests which ensure that the procedures put in place for recovery of data bases or files after their corruption are working.
  - Tests which ensure that fallback (local) procedures are working.

- Stress testing—Stress testing ensures that the total system will operate reliably under the levels of load that will be experienced in *peak* production situations. In this type of testing, the production

environment is emulated in order to determine whether there are problems or "bugs" in the total system that are induced under conditions of load by contention for resources. Stress testing also reveals application design defects that only become evident under load. The testing objective here is to break some element(s) of the system by inducing stress.

- Performance testing—Performance testing ensures that on-line transaction response (and to a lesser degree batch turnaround) targets can be met at the levels of load that will be experienced in *peak* production situations. Performance is measured against Service Level Agreements (SLAS) made between MIS and the users of the service. For this type of testing the production environment must be duplicated.

  Examples of the questions asked during performance testing are

  - What resource lack or application design error is causing a transaction or application (set of transactions) to perform poorly?
  - Can Service Level Agreements be met on the new system?
  - What is the effect of changing software performance parameters?

- Predictive testing (capacity testing)—Predictive testing is done by executing the profile(s) of projected system loads in order to determine the point at which the major system resources such as CPU, real storage, virtual storage, etc., will be exhausted.

- Disaster backup/recovery testing—Although there is a range of possible disaster backup and recovery strategies (reciprocal arrangements, shell sites, etc.), this paper confines its discussions to those situations where an unshared second site is the chosen approach. In not a few cases reciprocal arrangements and shell sites can be ruled out because of the impracticality or even impossibility of testing the backup/recovery process. This is not possible with a shell site and impractical in most reciprocal arrangements where heavy on-line loads are involved over extended operating hours.

  In the author's view, a disaster backup/recovery plan which cannot be fully tested at regular intervals is at best suspect and at worst deludes management into believing that an effective plan is in place when this in fact is untrue.

Assuming a second site at which a processor complex capable of handling the main on-line and batch functions is in place, we can test major elements of the disaster backup/recovery process on a frequent and continuing basis as part of a complete test plan. The use of the second site complex for TPNS-driven testing of new software releases at the levels of load encountered in primary site production using copies of production data, transactions, and applications enables us to answer many of the important questions relevant to a disaster backup/recovery plan.

The following questions can be answered using such a testing process:

- Do we have all vital records (application and system data) at the second site?
- Do we have all software (application and subsystem) at the second site?
- Is the host component of the system operable?

It is not claimed that this level of testing covers testing of all the components of a contingency plan. It does not cover testing of network backup, people issues, or logistical issues such as report distribution. It does, however, ensure the backup/recoverability of the host base in a continuing way and allows tests which of necessity can only be irregular (network switch, logistics, etc.) to proceed as an increment upon the host testing, the latter being the most volatile component. In this context it is worth noting that the applications, data, and equipment that must be backed up are, over significant periods of time, quite volatile, and the effectiveness of backup must therefore be proven continually.

Splitting the disaster backup testing into dynamic and static components—the first of which is done regularly as an offshoot of complete testing and the second much less regularly—makes the overall plan easier to manage and provides a greater assurance that the the overall contingency plan will work. Testing the complete backup of host data, software, and procedures is, in the author's view, necessary because of the volatile nature of data and software and the considerable interdependencies that exist. The alternative approach of sifting out the "critical" applications and data on a continuing basis is complex and fraught with possibilities for error, particularly when viewed over the long periods of time for which the contingency plan is to be maintained.

Disaster recovery is slightly different in that most installations would recover applications on a priority basis with the most critical applications first. That is to say, one could argue that only the recoverability of the critical applications must be proven. If we take a long-term view, however, it is reasonable to suggest that a much higher percentage of applications will be "critical" than may be the case at present, and the scope of recovery will therefore necessarily be wide. Situations where close-to-complete recovery in a short time period is highly desirable if not absolutely critical will clearly be more likely in the future, so that in the strategic sense the contingency plan should ensure that a very wide range of applications can be proven to be simultaneously recoverable at the recovery site.

Complete testing at the second site continually proves the existence of complete backup and complete recovery capability for the host components.

An approach to performing volume testing. Assuming that at a second site we have equipment which mirrors the main on-line processor(s) and peripherals, we have the equipment for performing complete testing. Assuming also the use of the formal software release build and test approach defined above, we have the basis for implementing a comprehensive method of testing which is now outlined.

Before doing this it is necessary to reiterate a point alluded to in other parts of this paper. Because the costs associated with the ultimate implementation of the approach are large, particularly when seen in a tactical rather than strategic time frame, some readers will have the feeling that the approach is infeasible from the viewpoint of cost alone. The benefits, however, are high, and the approach will in any event have to be implemented over a long-term time frame, perhaps integrated with plans to build a second site for reasons other than disaster recovery (e.g., machine room and staff expansion). Note also that the approach can (and should) be implemented in a number of steps with a number of different paths to the final goal.

As part of a complete testing process for a new software release and also as part of a disaster recovery test plan, the standard procedure would be to send the copies of operational data bases to the backup site at, say, Monday night cutoff. The restored copies would be used in conjunction with transactions captured during Tuesday's on-line processing (the full day's transactions) to perform TPNS volume tests (stress, performance, and capacity) on a Wednesday (Tuesday's transactions would be converted into TPNS-generated scripts). We would be testing the new-level subsystem and application software as a total system.

Performance tests would be done under the load levels existing on the production system.

Performance tests would be done under the load levels existing on the production system (with various performance measurement tools running). These measurement tools would then be switched off and traces switched on for stress-level testing that would be at a somewhat higher load. Finally, these traces would be switched off, and selective measurement tools would be switched on to perform capacity testing, i.e., driving the system at projected load profiles to detect future capacity limitations.

In the latter stages of the volume test, key operational procedures could be tested. An example of such tests would be one in which a processor failure was duplicated by "pulling the plug" on MVS, IMS, or some other key subsystem. The ability to restart the system using documented procedures could then be tested.

At the end of Wednesday's test run, Tuesday night's production data bases would be (logically) compared with the updated test data bases to detect regression in function (regression testing).

The other aspect of this approach is that it provides a continuous test of disaster backup and recovery for the host system component. That is, we would be proving the availability of vital records at the disaster recovery site, proving the existence of necessary capacity, proving the availability of all necessary software, and proving the operational status of the recovery site on a regular basis. In this sense the availability and disaster backup/recovery issues can

be seen as interdependent and "complete" testing as a technique to address both issues.

The production process. Once the software release has been completely tested, according to the objective test exit criteria referred to earlier, there is no reason to delay its entry into production. In the first stages of implementing a complete testing approach, it may be prudent to separate change into subsystem changes in one part of the cycle, followed by applications changes in the next part, then back to subsystem changes, and so on. The tradeoff here is that change will take longer because of the lengthening of the overall cycle.

Complex issues have a significant degree of interdependence.

The foregoing process provides a framework for considerably reducing the impact of software instability and the exposures inherent in not having a proven disaster backup/recovery capability. It is not suggested that such a process could or should be implemented quickly. The approach defined provides a suggested framework for planning and implementation over a period of time in a number of steps which can begin with as limited a step as using TPNS on a back shift for limited stress testing.

The experience of the author in dealing with issues of the power and pervasiveness of availability and disaster recovery is that they must eventually be addressed with powerful methods and tools which allow a problem prevention philosophy to take root in the MIS organization, and that this can only emerge over a period of time.

There are less powerful (and less expensive) methods which can assist in improving availability and in facilitating recovery from disaster. Some of these lie within the province of operations, i.e., in the production area. These approaches include the following:

- Automation of operations to progressively higher degrees is and will continue to be a major area in which operations can improve the stability of the production process.
- Instituting a Help desk which is integrated with the problem management system. This approach is in common use in installations with networks of significant size. The effectiveness of the Help desk is greatly enhanced by integrating its function properly with a problem management system which provides information for MIS management to enable them to focus not only upon high-impact problems but upon categories of low-impact problems where the numbers are very large (problems with terminals, individual transactions, etc.).
- Uninterruptible power supplies (UPS) have a clear applicability. Moreover, if the financial impact of application outage is established, it is relatively simple to make a judgment as to whether or not these are justified.
- Storage management, particularly of DASD space, is another area where it may well be possible to justify the effort involved in avoiding failures induced by inadequate storage management. Storage management strategies are also important in ensuring the availability of vital records for disaster recovery purposes.
- The length of the batch processing and change "windows" available to operations is an area upon which operations should maintain a focus. Because on-line operating hours will increase, as will the amount of work to be done in the batch processing window, the likelihood of impacting the availability of on-line functions due to the inability to complete batch processing will increase. As a result, the size of the window available for change (of any type) will decrease and bring with it exposure to back-level systems and poor productivity to those who have the responsibility for implementing change.

Other dimensions of problem prevention

Although the identification of strategies for the technical resolution of individual issues is necessary, on its own this is insufficient since the MIS organization must also be able to recognize the interdependence with other issues, absorb the necessary changes to methodology, justify resources, and take full advantage of such solutions in managing the MIS process more effectively.

Planning for major issue resolution. One characteristic of complex issues which must be dealt with is

that they have a significant degree of interdependence. This is not surprising given that the impact of each individual issue is pervasive, often affecting and affected by all elements of the MIS organization. From a planning viewpoint the major issues must be dealt with on a collective basis as part of a strategic planning process.

The technical issues that need to be so addressed in larger installations include

- Availability
- Disaster recovery
- Continuous or at least extended on-line processing
- Very high transaction peak rates (particularly in industries with an EFT/POS potential)
- Automated operations
- Management of large networks
- Management of large data aggregates
- Use of advanced functions such as Structured Query Language/IBM Database 2 (SQL/DB2)-based software for broadly based implementation of application function
- Capacity planning
- The effect of all the above on applications and subsystems design

In a number of installations a considerable number of these issues have already strongly emerged, whereas others do not have so high a profile. All should be considered in any strategic (three-to-five year) planning process and their interdependencies understood.

Development of a full understanding of the interdependencies among the various issues is important to ensure development of technical architectures which are effective for the longer term. The linkage between availability and disaster recovery through testing has been the focus of this paper. Similar linkages exist between all the major issues.

Continuous processing affects the application design of an installation and the choice of subsystem function to be used (functions of the type specifically required for continuous processing are in evidence in the latest versions of IBM's IMS Fast Path). Continuous processing, availability, and disaster recovery all have an influence upon capacity planning. This influence is in some cases greater than the influence of on-line transaction growth.

Similarly, automation of operations has the potential to improve considerably the stability and thereby the availability of systems.

An effective MIS strategic planning process is necessary to address these issues, since they will be around

MIS management needs support in many areas.

for a long time and their significance can only increase and become more common across a wider spectrum of installations.

From the reverse perspective, failure to effectively address these issues on a collective basis will create a major obstacle to servicing the needs of the business as a whole. This could be seen as a potential disaster which must be prevented by investing in (strategic) planning.

MIS management and problem prevention. In addition to the strategic planning process referred to above, MIS management needs support in many areas. Important to the day-to-day operational processes and to the tactical planning process is the provision of information which enables decision making that is objective and that has a return-on-investment basis.

In the context of problem management, for example, there is often an unfulfilled need for information that enables management to make tactical (budgetary) problem prevention decisions based upon return on investment criteria as discussed earlier.

In the area of change management there is a need to provide MIS management with information that will enable them to make objective decisions as to when a change involves acceptable risk. The application of objective criteria for exit from testing will assist management in this process.

Again, Service-Level Agreements defined prior to design enable management to ensure that functions important to availability and disaster recovery are considered at the design stage.

MIS organization and problem prevention. In the author's view the necessity for much increased focus

upon problem prevention logically implies that a new major MIS group (in addition to development and operations/technical services) may emerge and that this should be recognized in tactical and strategic planning. The responsibilities of this group would include the assurance of availability and performance of systems through testing, implementation of security, provision of disaster recovery planning, auditing in general, assurance of data integrity, definition of standards, etc.

One essential reason why this group should be organizationally separate is that it will be in *necessary contention* with the other groups. For example, in order that a software test group be effective, it must take an aggressive approach to finding flaws in the systems and procedures of development and operations and resist the frequent development (for applications and subsystems) attitude to testing—a negative one born of a natural desire to get systems into production because of tight deadlines.

Another catalyst for evolution of such a group is that, not being burdened by the history and conventional wisdoms of the other groups, it will be in a better position to innovate and move MIS quickly in creating problem prevention practices.

Problem prevention and the user. A difficulty which MIS groups experience in judging the effectiveness of their efforts is matching MIS perceptions and those of the user.

MIS availability statistics based upon average percentages of host availability are often used by MIS as an indicator of success in the availability area. To the user such statistics are close to meaningless because they can hide significant outages causing significant financial impact. User perception is shaped not by the average situation often used by MIS but by the longer outages where the worst business impact is felt. A somewhat better guide (at the system level) is the *minimum* daily host availability over an extended period, since this indicates the level of service that MIS can guarantee. This also pinpoints the system conditions that cause the greatest impact and that offer the greatest potential for improvement if addressed.

From the user viewpoint, more relevant statistics are minimum daily availability figures for applications and transaction sets within applications. A further degree of refinement that enables targeting the best return on investment possibilities is to express the application availability figures in financial terms, developed from the type of review discussed in Reference 3 and integrated with problem management/availability reporting. At this level, user and MIS perceptions of availability will begin to mesh, and the user will be in a much better position to define and make Service-Level Agreements for new applications prior to design and to fund correction of existing system and application deficiencies, based upon an increasing awareness of the costs of application and even transaction set unavailability.

From the disaster recovery viewpoint as discussed in some detail earlier in this paper, a financial impact review with senior business management is a vital step in developing a tested contingency plan.

Conclusion

To proceed to the high system availability levels normally required in installations with a significant on-line load, it is necessary in most of the more complex installations to adopt an attitude toward problems which is preventative rather than reactive. Fully instituted, such an approach affects all MIS processes but in particular raises the requirement for complete, formally managed testing methods.

The disaster recovery issue can also be addressed in a significant way by the institution of complete and formally managed testing where that testing is done at the disaster backup site. Thus it is a powerful strategy to plan for these two issues together.

These issues, along with a number of others referred to but not discussed in this paper, can only be addressed if a long-term plan is developed. The involvement of senior business management is crucial to addressing the issues because the issues are beyond the power of MIS to address as a sole agent.

Acknowledgments

The author acknowledges P. N. Rose, senior consulting systems engineer with the IBM United Kingdom organization, for making available his views and approaches to the issue of disaster recovery. Contributions to the ideas expressed in this paper came from a number of people working at various IBM customer installations over many years, who are too numerous to mention individually here.

Cited references

1. *Improved Stability in Large Systems*, GG22-9051, IBM Corporation; available through IBM branch offices.
2. *Contingency Planning for Catastrophic Events in Data Processing Centers*, G320-6729, IBM Corporation; available through IBM branch offices.
3. *So You Want to Estimate the Value of Availability*, GG22-9318, IBM Corporation; available through IBM branch offices.

John Newton *IBM Australia Limited, P.O. Box 1798Q, G.P.O., Melbourne, Victoria 3001, Australia.*
 Mr. Newton joined IBM Australia in systems engineering in 1965. His experience since then has been in large IBM customer installations in the finance, process, and retail industries. Over the last ten years his work has focused on key customer technical issues, of which availability and disaster recovery are two. He spent three years on assignment at the IBM Santa Teresa laboratory performing testing work and developing testing methods for the Information Management System Fast Path product. Currently a Senior Systems Engineer, Mr. Newton has a Science degree from the University of Melbourne (1961).

Reprint Order No. G321-5252.