Quality assurance and maintenance application systems

by BARBARA J. TAUTE

Time Inc.

New York, New York

ABSTRACT

Modifications to application systems in production can have a devastating effect on the environment if the changes are not handled correctly. A comprehensive quality assurance (QA) approach can help minimize this potentially harmful effect. This approach involves all groups: users, data processing center, applications programming, and quality assurance.

The QA approach should address four areas:

1. Phased approach
2. Procedure flows
3. Maintenance guidelines
4. Implementation

This paper describes the QA phased approach successfully developed at Time Inc. The phased approach consists of the definition and implementation of eight phases, envisioned in a circular life cycle. Emergency processing is considered separately. Procedure flows consist of diagrams and charts listing responsibilities of the participants. Maintenance guidelines contain helpful hints and checklists and provide direction to the participants.

The benefits are noteworthy: The phased QA approach consolidates groups, forms a standard for maintenance procedures, and increases productivity.
INTRODUCTION

Computer application software systems in production exist in a volatile environment. An environment is unstable when data, hardware, systems software, and usage are in flux. Environmental changes will probably create a need for modification to the application system itself.

Production application systems are in high-risk environments simply because they are in use. The user community expects and needs correctness to meet deadlines (in contrast to a system in development, which is not yet in the users' hands). Any change in a system can have an adverse effect on its operation.

This paper will address a methodology for understanding, controlling, and benefiting the maintenance environment. The environment is best perceived as affecting four groups: users, data processing center, applications programming, and quality assurance. This approach document, developed by the Quality Assurance Department of Time Inc., continually reflects the responsibilities of all four groups. Only through a combined coordinated approach encompassing the four groups can changes in production application systems be controlled and made manageable.

TIME INC. ENVIRONMENT

Time Inc., with headquarters in New York, is a diversified company encompassing publishing, forest products, and video. The Application Programming Department supports the development and maintenance of applications for the departments of Magazines, Books, SAMI (Selling Areas—Marketing, Inc.) and Corporate Staff offices. It is composed of 240 people, split mainly between Chicago and New York. A liaison role of user of record provides an application interface to the user department. The Quality Assurance Department provides methodologies and structured approaches to all departments.

Because the people in applications programming were of varied backgrounds (multiple groups), disciplines (software packages to in-house development), experience levels (1 to 15 years), and sophistication (hard coded assembler to test data generators) a need was perceived for a methodology to consolidate these efforts. This methodology was to be created by the quality assurance team, which possessed an independent, experienced, state-of-the-art view for programming structure.

The methodology was to address several areas. The first area was to pertain to maintenance systems, the next to development systems, the third to testing, the fourth to measurement of all processes, and the fifth to productivity. This paper deals with the first approach, that of maintenance systems, which is defined as follows.

DEFINITIONS OF MAINTENANCE

A maintenance application system is defined as being in production and in use by the user groups. Since maintenance (back end) was being organized first (before a development approach), the quality assurance department was not able to define a specific set of test, documentation, operation, recovery, or security requirements that would need to be met before a system was officially considered to be in maintenance. This sort of requirement would compose the turnover criteria from a development effort. Since no standard existed, programs showed varying levels of completeness in their production environment. The original design criteria no longer existed, and in many cases system documentation had not been kept up to date. Yet these systems were providing accurate, meaningful data and output results to the user departments. In order to include all systems in the approach, it was necessary to define maintenance as pertaining to any system in a production environment.

The Quality Assurance Department, however, was able to influence the definition of a maintenance life cycle. It is defined to consist of eight structured, related phases with definitive criteria and responsibilities shared by four groups (users, data center, application programming, and quality assurance).

MAINTENANCE STRUCTURED PHASES

The maintenance life cycle (Figure 1) is represented as a circle with one phase leading into the next. Its eight phases refer to the following:

1. Request phase—An expressed desire for a change to a system
2. Estimate phase—A calculation of effort to complete the change
3. Schedule phase—An identifiable release date planned for the change
4. Programming phase—The modifications to a controlled source copy of the system
5. Test phase—The verification that the change performs as expected
6. Documentation phase—The modification to system, user, and run specifications
7. Release phase—The replacement of the old system with the changed system
8. Operation phase—The day-to-day usage of the system
Request Phase Definition

A request for a change, whatever its form, initiates a process that will affect an existing production system. A change request is a request for an investment, since there will be time and therefore money spent by various groups. A change request may occur for numerous reasons. If there are no change requests, the environment is stable; and if a system is not used, no changes are necessary. A system in production will by definition have changes made to it, because it is being used. The reason for a system change is often a function of the time a system is in use and is not peculiar to one system or another nor an indication of how poorly or well it was developed.

Since changes can come from so many different sources (users, data processing center, programmer, manager, auditor), it is necessary to have a standardized format representing requests. It is not possible to evaluate changes or to assign priority without established criteria. A change request form helps pinpoint symptoms of a problem, level of need, and time of involvement; and appropriate levels of approval can be expressed and retained.

In the environment at Time Inc., several change request forms were being used. The Quality Assurance Department consolidated the best portions of these forms to devise a new form for all groups.

The completed change request form specifies the deliverable for this first phase. When it is completed, it is transmitted to the librarian. This transmittal initiates the next phase, the estimate phase.

Estimate Phase Definition

The estimate phase has its own significant steps. Upon receipt of the change request form, the librarian assigns it a unique number. Then, if the change request concerns an application system error, the suspect error needs to be verified prior to estimation. It is hoped that this will help eliminate reported problems that are the result of nonadherence to operational procedure, user misunderstanding, or invalid data. Next, a person from Applications Programming needs to evaluate the change request and estimate the number of hours it will take to correct the problem. The project manager must always review this estimate for concurrence. The estimate must be as thorough and as accurate as possible, since this number will be used as the basis for a scheduling process. The estimate is also reviewed with the user of record.

The process of internal reviews helps to eliminate incorrect estimates and permits several individuals to learn from the estimation process, rather than just a few. At Time Inc., levels of review established were based on fixed hours. These limits were refined as upper management modified their level of review requirements and as personnel became more confident in the estimate process.

The completion of the estimate and review portions (including signatures on the change request form) determines the end of the estimate phase. The schedule phase follows.

Schedule Phase Definition

Functional enhancements compose a significant portion of the maintenance workload. In order to handle these changes as efficiently as possible from the standpoint of all departments, a scheduled release concept is used. Releases of systems are numbered and given predetermined dates based on their business cycle, such as every 60 or 90 days. Nonemergency changes are then assigned to a specific release number. Each release contains multiple changes and forms a new replacement system. The advantages of this process are

1. Consolidation of changes
2. Increased stability of the production system
3. Reduction of training requirements
4. Better planned and managed workloads

The change requests provide the basis for a project tracking system. A summary entry for each outstanding change forms the agenda for a meeting of all four departments. Periodical sessions, called Change Control Board meetings, are held to discuss the various needs and priorities of groups. After the meeting the project tracking system and the change request form can be updated with a scheduled date and release number.

When a scheduled release accommodates multiple changes, it represents many added hours of effort. There is an increase in the risk of malfunction for that release. Therefore an approval process similar to the review process in the estimate phase must be followed. This process helps guarantee that management will maintain proper awareness of releases of a predefined magnitude.

At Time Inc. the process of scheduled release was a new one. There was initial skepticism about its success until one group installed this process, with close guidance. The process
was an overwhelming success: Users were pleased to have expected dates for changes, requests were dealt with and not put aside because of time commitments, and application programming was favorable to fewer releases in number. A 3-month release cycle was initially established, but this was ultimately changed to a 2-month cycle to be more responsive to the natural business cycle.

The assignment of a release date and time frame for a change marks the completion of the schedule phase. The next phase is the programming phase.

**Programming Phase Definition**

Modifications of application systems in production differ from changes in systems in development, since production systems are currently operating as an integral part of the user’s business. It is essential to insure that modifications are being made to the current operational version of a system. Thus, a procedure to control the source should be followed. The program phase consists of creating a test version of the controlled production source code of the system, making the programming changes according to the approved design, and initiating the test process and updating the documentation. Associated version numbers and dates will help guarantee that the proper source code files are being modified. A program log at the start of each program will help to identify the changes and ease the next modifier’s job. While modifying the test version, care must be taken to maintain the integrity of the production system. Files can be lost or data can be extensively altered if the integrity of the systems is compromised. In addition, the programmer must be cautioned to make only authorized changes. When a segment of code is opened up, it is not the time to make a “nice little fix.” These fixes can be disastrous (in error), can cause extra test time (invalidating estimates), or can just be undesirable (unnecessary enhancement) to the users.

At Time Inc., various existing support systems (SLIM, PANVALET) required the retention of varying source copies. An attempt to consolidate these was not deemed advisable, because of varying form as well as location (varying cities, varying locations). Therefore, all groups agreed upon a minimum number of storage repositories and upon adherence to defined control procedures. A long-range paperless repository is currently being planned.

At the conclusion of the programming phase, a set of program and system changes exist. This marks the beginning of the test phase.

**Test Phase Definition**

Testing of a production system is one of the most critical and important phases of the entire change process. Testing helps insure that the replacement system will function properly and not disrupt the user environment. The quality of testing is a function of both the thoroughness of the test plan and the quality of the test data. The test plan should be comprehensive and should include unit testing, integration, and system testing of the changed elements and their interfaces. Good test data start with a good test base, which is kept current as production systems change. If a test base does not exist, create one. Regression testing (verifying changed systems run correctly with known data) can only be accomplished through maintaining a controlled test set.

The Program Manager has the ultimate responsibility for the correctness of system changes, but other groups may also become involved. Often users will test for functionality, and the data center may verify via parallel runs. This involvement can provide further confidence in system integrity.

At Time Inc., through the initiation of this methodology, the programmers were made aware of the importance of keeping a good test base to achieve time savings, cost savings, and accuracy.

The completion of the test phase is evidenced by correct operation of the changed program and by a test approval signature on the change form. Prior to a system going live (after a correct test run) documents must be updated. The next phase discusses the documentation issues.

**Documentation Phase Definition**

Three forms of documentation are considered for maintenance systems: system, user, and run.

Since documentation must be accomplished before the system is released to production, it is therefore represented in the life cycle immediately before the release phase.

System documentation is maintained to help a programmer learn a system and its elements. This information describes the system in its past phases (history) and its present state and is used as the basis for future changes in the system. These items, at a minimum, should be continuously maintained: high-level system flow, system functional description, self-documented program source compilation listing, and documentation describing data flow. System documentation should be kept in an accessible but secured library area with a checkout procedure. The documentation should be retrieved by the programmer or analyst, updated, and replaced in the library. The amount and level of detail for controlled documentation must be maintainable, since obsolete, incorrect documentation is more confusing than no documentation at all.

User documentation should be composed of user manuals, error lists, and functional descriptions for the use of the system. A change in a system may very well change the way the system appears to the user. An increase in functions can increase the capabilities of the system. Remember: If users do not know how to use the system, they cannot use it and will not use it. Any change in a system should be evaluated for the possible necessity of correcting or updating user documentation. These modifications to the user documentation should be made before the system is released.

Run documentation is the combination of materials (Job Procedures, JCL, restart/recovery instructions, etc.) required by the data processing center to operate the system. Therefore, the minimal requirements need to be set by the data center. Often a system change will result in some modification in the way the system is run. This means that existing oper-
ating instructions should be changed at the time of release. If the data processing centers have standards or procedures, these should be followed for form and content of run information.

Reviews by the project manager or systems analyst can help guarantee that all documentation is complete. The user of record should help determine if user documentation is thorough and clear.

Training may not be required for relatively small system changes, but it is certainly necessary for larger ones. If there is a functional change, users need to be retaught how to use the function. New operating procedures may need to be taught to the data processing center. The type and amount of training is very system-dependent. The development of good documentation can facilitate this training process.

At Time Inc., documentation levels varied considerably among applications programming groups. Some groups had required comprehensive documentation, and very detailed user manuals were found to exist. For others, the manuals were out of date and had not been maintained. One data center had existing standards; for another they were still in development. The introduction of this methodology helped to standardize all documentation.

The completion of the documentation phase is evidenced by signoffs on the change request form and in the tangible documents themselves. The subsequent phase is the release phase.

Release Phase Definition

The release phase is the natural last step of the change process as well as the most critical step in the life cycle of a change. It is at this time that users become excited about the increased functionality that exists, programming is enthusiastic about their system release, analysts become concerned about the unforeseen impacts of the change, and the data processing center looks for a clean, easy installation of the changes. This critical period can be greatly eased by a well-controlled and well-communicated release procedure. A pre-release review conducted by the Quality Assurance Department with the participation of all four groups can ease this process. Quality assurance should verify the following:

1. System installation readiness
2. Adequate system testing
3. Adequate approval testing
4. Completeness of documentation
5. Installation requirements specified
6. Transmittal form supplied

Once these characteristics are verified, the system can be considered ready for release to production. These criteria should be presented punctually for review so that the review does not postpone the release process. After the review, the system is officially turned over to the data processing center, and the user is informed that the system is ready for use.

A controlled release process insures the definitive release of systems. In the absence of this process, systems may never officially be turned over to production but continue to be run as test systems. This can lead to multiple versions and confuse users. At Time Inc. this controlled process eliminated some of these redundancies and solidified the release process.

The turnover to production marks the completion of the release phase. This is signaled by the running of the production jobs by the new system, and also officially by the signatures on the change release transmittal. The last phase is the operation phase.

Operation Phase Definition

Operation is the day-to-day activities of any system. Even though a system is stable, various elements can cause the erosion of a system solely through its use. Not all these elements can be planned for or controlled because of the vast, unknown combinations of actions that can occur. What is helpful, however, is to plan for an orderly description of the event if an error does occur. Users and the data processing center must be trained and educated about the functions and use of the system. The more operational aids written into and about the system, the more closely the environment can be handled by the users and the data processing center, and the less programmer assistance will be necessary. Testing should be planned. It is costly, but remember that thorough testing can improve the systems operation.

After the system has gone through one complete production cycle, the four groups should convene and conduct a postrelease review, led by the project manager.

A system in operation runs until it is outdated, is too difficult and costly to support, no longer functionally serves its purpose, or needs extensive changes. A change leads back to the request phase; a need for a rewrite leads into a new system development process.

As in most companies, Time Inc. has numerous systems in production and several in development. This process helped all groups by controlling the maintenance process and defining its components.

A phase included separately is emergency processing. Because it is treated in a unique fashion, the process should be discussed by itself.

EMERGENCY PROCESSING

Every attempt should be made to keep emergency releases to a bare minimum. In fact, the by-product of a well-controlled maintenance cycle ought to be the practical elimination of emergency releases. However, they still exist in the real world and therefore must be planned for. An emergency is a change of such importance or impact it must be considered immediately and out of the normal maintenance structure. Emergency processing expressed for the eight phases is described below.

The emergency request phase can be originated by a change request form or a "midnight" phone call. A change request form is completed as soon as possible to record all work.

The emergency estimate phase follows the standard structured process if the emergency change does not need an immediate fix. The programmer or analyst must immediately evaluate the change, if it is a highly critical fix, and proceed directly
to program and test. This can apply to both a temporary and a permanent fix. If the correction cannot be accomplished rapidly, a manager’s approval is necessary. If the permanent fix must be made during extended working hours, the same phases should be followed; but the time can be compressed and the modifications can be installed outside the maintenance release.

The schedule phase for emergency problems falls outside the scheduled release concept. The schedule and time will depend on the complexity of the change.

The program phase is no different for emergency problems. A source still needs to be extracted from a library and modifications made to it.

The test phase is perhaps more important for emergency problems. Since a bad situation already exists, extreme care must be taken to make sure the fix improves the situation rather than aggravating it. For changes to be installed immediately, all groups involved should approve the testing prior to its release.

The emergency documentation phase must be conducted in the same manner. However, emergency changes may require different timing. Temporary documentation may need to be developed before the permanent updates are released.

The emergency release phase will be under control of the data processing center once the proper approvals are received. The transmittal remains the same.

The emergency operation phase follows the same guidelines once the emergency fix is in place.

Thus, the description of the phases and the emergency phase give an understanding of the cyclical nature of maintenance systems.

PROCEDURE FLOWS

A visual representation of the phases of maintenance is best provided through diagrams and responsibility charts. The format as shown in Figure 2 is used to describe visually the functions of the participants. The action column briefly describes the event in English text. This event is represented by a symbol, such as a listing, tape, or paper form, which is placed beside the heading of the participant. Lines drawn connecting the symbols indicate multiple participant input or output. In this way all interfaces are shown in a clearly serialized fashion. A chart should be constructed for each phase of the maintenance approach. The level of detail is optional, depending on levels of need for any one particular group or desired summary level for another. This form of phase representation clearly identifies each participant group’s activities and their interface groups. The second portion shows the responsibilities of the participants. Each title is listed and English text used to delineate their activities during each maintenance phase. A set of procedure flows was constructed for Time Inc.’s environment.

GUIDELINES

Guidelines provide procedures for handling various aspects of the maintenance process. These guidelines should be somewhat flexible in order to encompass subsequent suggestions or considerations for the maintenance process. They should not be hard-and-fast or demanding rules. As with any guidelines, they are not meant to be a replacement for good judgment. In many cases checklists are provided to help participants increase their level of understanding about their responsibility for the phase. In other cases, helpful hints and known areas of concern are identified for the participants. A set of guidelines for Time Inc.’s environment was constructed.

IMPLEMENTATION

For a methodology to be successful, planned action and attention must be given to its installation. At Time Inc. a separate document was produced that addressed the implementation of the maintenance approach. Before the approach was implemented, program control library, automated tracking system, move to production procedures, and required system documentation were addressed.

The implementation document itself had the following chapters:

1. Introduction
2. Installing the Procedure
3. Training the Participants
4. Monitoring the Effort
5. General Installation Schedule

SUMMARY

A controlled, phased maintenance approach can simplify and assist in the delivery of timely, correct versions of applications systems software. This process is defined as a circular life cycle with the collaboration of four groups to insure its success.

Time Inc. has defined a methodology for its environment and is in the process of a successful installation and implementation of the procedures.