A quality assurance program for software maintenance

by JOHN W. CENTER
Medtronic Incorporated
Minneapolis, Minnesota

ABSTRACT

This paper presents the description of a quality assurance (QA) program applied to software maintenance projects. The QA program is a set of checks or inspections overlaid on the steps of the maintenance project. The relationship between the QA program and project management is shown. The paper includes a brief discussion of waivers and deviations to standards and control documents. The QA checks are delegated to three levels. The rationale, scope, and authority for each level are discussed. A list of sample criteria used for each QA check or inspection is given.
INTRODUCTION

Recently a great deal of interest has developed relative to quality assurance (QA) techniques and their application in the environment of management information systems (MIS). Problems associated with controlling software maintenance have also been recognized. The MIS department of a manufacturing company is usually responsible for applications with diverse characteristics. The applications cover the range of finance, planning, inventory control, personnel, process control, and automatic test equipment. With such differences in applications, the requirements and risks of maintenance are different. A means to control the maintenance process and reduce the risks would be of distinct benefit. If the MIS department has or is starting a QA function, a QA program applicable to software maintenance would be most useful.

Fischer and Mendis give excellent descriptions of general QA programs applied to software environments. Though the articles describe programs intended for new software development, they establish excellent background useful in understanding how a QA program works and what the objectives are.

Lientz and Swanson have a new book on the topic of software maintenance. Roberts presents the software maintenance problem in a very practical light. A set of plans is a major aspect of any QA program. Included are test plans, assurance plans, configuration control plans, etc. Buckley and Dunn and Ullman recently gave excellent presentations of QA plans for software projects.

The QA program presented here should be treated as an example. Readers who attempt to implement an identical one in their departments may find certain problems and inefficiencies. However, readers should be able to establish a software maintenance QA program by using the major features and concepts presented in this paper as guidelines.

THE PROGRAM

There are several components to the software maintenance QA program. The program is based on a set of checks or inspections. The placement or execution of inspections is determined when the software maintenance project is broken into steps. The checks or inspections are placed according to a set of rules used by traditional quality assurance management. The rules have evolved over time. The checks are further classified by type, in-line or off-line, and by three levels. There is often difficulty in applying current standards to older software being maintained. Waivers and deviations become an important component of the QA program as mechanisms to document difficulties in the universal application of standards.

Each of the following sections discusses the various components of the QA program in more detail. The description of the rationale, authority, etc., of each level of QA check is discussed. A list of the criteria used for each inspection or QA check is also included.

Project steps

There are dozens of ways to break a software maintenance project into separate steps. One might treat maintenance of sufficient size or magnitude the same as traditional new system development. It would then make sense to use steps given in one of the current books or tutorials on systems development methodologies. The QA program for very large maintenance projects would be the same as the QA program used for new development.

One could also consider the maintenance of a single program. Perhaps the number of lines on a report page is being changed. A project of this size is so small, and would be completed so quickly, that the benefit of a formal QA program is questionable. Projects in the emergency maintenance class could also be too small to require the formal QA program suggested in this paper. One must remember that emergency maintenance relates to the occurrence of a failure condition. There should be a separate formal QA program for failure investigation and corrective action.

Thus, the QA program discussed in this paper is oriented toward the middle-size maintenance project. The vast majority of maintenance resources are expended on projects in the middle group. Books and articles have been published recently on the topic of software maintenance. However, there appear to be few articles directly related to breaking the maintenance project into separate steps. There is one interesting potential source of information on this aspect of software maintenance projects. A major computer hardware conversion has strong similarity to the maintenance process. A publication provided by Sperry Univac to assist in the conversion process gives an excellent breakdown of steps in a conversion project. Some of the quantitative results of using a QA program during a conversion were presented and published at the 1981 National Computer Conference. The QA program for software maintenance is almost the same as one used during that conversion.

The box labeled “Establish Need for Maintenance” in Figure 1 represents the first identification of a potential or real software maintenance project. This need might be determined by the user department when a new law is passed, a new format for ZIP codes is used, etc. The need might be determined by the MIS department when it becomes apparent that much higher processing efficiencies might be obtained, a constraint on the physical hardware has been reached, etc. Usually a formal request or form is generated. The request is
processed through the MIS channels to place the maintenance project in the queue.

The box labeled “Schedule Maintenance” is the first task of the newly assigned project leader. This task includes the when, who, and how of the maintenance project. The project leader determines which subsystems are involved, which people should do the specific tasks, whether there are any automated tools available, etc. It is assumed that the project is broken down primarily by information subsystem. The subsequent steps for each subsystem are concurrent. The project leader establishes the detailed schedule of subsystems on the basis of contingencies and requirements. The steps for a single subsystem are shown as a representation of many that might be in parallel.

The box labeled “Maintenance Preparation” represents the tasks performed prior to the actual maintenance work. This step includes the following tasks: (1) establishing the documentation package, detailed requirements documents, status checklists, forms, etc.; (2) gathering test data, sample files, or data specifically designed to test the (sub)system; (3) gathering file information or file descriptions, record layouts, and look-up tables for both the old and new versions; and (4) gathering program listings, cross-reference data, special or library routine descriptions, etc.

The box labeled “Modify Code” is the activity on which the entire maintenance project depends. Closely associated with the code modification is compilation. The box that directly follows code modification is error free or “Clean compile.” It is usually assumed there were no program coding errors prior to maintenance. A significant portion of software maintenance relates to data or file descriptions. When changes are made in the data descriptions, the data and files must be modified to match the new ones. The box labeled “Convert Data/File” represents this process.

After making the appropriate modifications to the program code and the data representations, the “new” programs have to be tested or checked out. In order to isolate problems quickly, each program should be tested separately. The process of testing the programs or routines is represented by the box labeled “Program Test and Debug.”

After all the programs have worked successfully, the subsystem is put back together and checked out. The box labeled “Subsystem Test and Debug” represents this process. Test failures mean looping back to modify code and convert data/files.

The “new” subsystem now has to be integrated into the system. The processing related to the integration is represented by the box “Preparation for System Integration.” This preparation includes a large number of administrative tasks. The following is a sample: (1) Generate new control language explanations or higher-level flow charts. (2) Generate new program documentation updates, new program listings, cross-references, etc. (3) Generate new data/file documentation updates, new data element, record or file descriptions. (4) Generate transactions to update the data catalog or configuration management mechanism. (5) Generate updates for operations documentation.

The new system is now ready to be put into operation. This process is represented by the box labeled “System Integration.” The entire information system is tested; this step is represented by the box labeled “System Test and Debug.” Problems and errors may be discovered during this process. This means that the maintenance for the subsystems with problems will have to loop back to modify code and convert data. With tight project control, the maintenance for the problem subsystems may have to start over completely.

The remaining process is represented by the box labeled “Documentation Update.” The preparations done at subsystem level should make it easy to actually perform this task. The final task is the box labeled “System Acceptance.” The user now gets involved in using the newly maintained system. The user will perform the acceptance test procedures that were generated early in the maintenance project.

The steps shown in Figure 1 are representative of the tasks performed during software maintenance. The remainder of this paper assumes that this set of steps is used to actually perform the maintenance and control the project. The QA program and associated QA checks grow around on the skeleton of these steps, or the project management methodology, used for software maintenance projects.

Placement of QA checks

One of the more critical aspects of developing a QA program is the placement of the QA checks or inspections. Experience from the long history of manufacturing QA gives some of the basic ground rules. Juran and Gryna give an excellent list of potential locations for manufacturing QA inspections. The following is a paraphrased version of that list, emphasizing software terms or analogies:

1. At receipt of software from vendor, called incoming inspection or vendor inspection.
2. Following the setup of the production process, setup approval, to provide added assurance against producing defective software. Sometimes the setup check is used to give prior acceptance of the software that goes through the subsequent process.
3. During the execution of critical or costly operations, usually called process inspection.
4. Prior to delivery of software from one processing group to another, called toll-gate inspection.
5. Prior to shipping complete software to the customer/user, called finished-goods inspection for hardware.
6. Before performing a costly irreversible operation.
7. At natural peepholes in the project flow.

The QA checks are represented by triangles on Figure 1. The names for the numbered triangles are (1) preparation, (2) compilation, (3) data/file conversion, (4) subsystem test and debug, (5) preparation for system integration, (6) documentation update, and (7) system acceptance.

The triangles were placed on the project step flow on the basis of the list of potential locations given above. The preparation check is called a set-up approval. The checks done for compilation and data conversion are called process inspections. In some cases an additional set-up approval QA check should be established prior to data conversion, especially when the conversion itself is irreversible. The check
done for System Test and Debug is a toll-gate inspection. Additional formal checks could be done for Program Test and Debug as process inspections. The check done at Preparation for System Integration is a toll-gate inspection. Documentation Update is a natural peephole. The check called System Acceptance is a finished-goods inspection.

The QA checks were placed assuming a normal project, based on the scales of size, level of criticality, magnitude of costs, etc. Additional checks are certainly not precluded. When any additional checks are desired, consider the list of potential locations. Extreme care must be given regarding the removal or consolidation of the seven locations listed above.

Project management milestones

Placing QA checks in the stated locations gives an additional project management benefit. QA checks are associated with each of the major tasks of the maintenance project. Passing a QA check provides an absolute milestone. The project leader can truthfully say that the documentation has been updated only when the associated QA check has been completed. Additional QA checks provide the project leader with more detailed project milestones when needed.

The relationship between project management and QA checks has been recognized before. The NBS Special Publication by Fif6 gives a brief but excellent presentation of this relationship.

In-line QA checks

Some of the project steps have distinct boundaries, are very critical, have a serial relationship, etc. The QA checks associated with these distinct steps are placed in the line of the project flow. These checks are known as in-line checks. They may also be known as gates. These checks are called mandatory hold points in the draft by the Canadian Standards Association on software quality assurance programs.

When the project encounters an in-line QA check, progress is technically stopped. No further work can be done on the maintenance project until the inspection or check is considered accepted. Generally, these are the steps where further progress with undetected errors or problem conditions would
be costly in terms of time, staff, etc. Corrections can be made when it is still cheap. Since these are placed in the more critical positions, they make excellent project milestones.

**Off-line QA checks**

Some of the steps of the software maintenance project are of a detail, recursive, or parallel nature. The QA checks associated with these steps are often a quick inspection or observation for a particular condition. One example involves errors detected during compilation. The code for other routines can be modified or compiled while the cause for a detected compile error is being researched. The QA checks for steps with these characteristics can be done off line of project flow. This class is therefore known as off-line checks. They might also be called monitoring checks.

The off-line checks usually relate to conditions that can be quickly repaired or changed. The off-line checks are good for determining the effectiveness of the task being executed. The successful completion of the off-line checks is a good means to determine the project or task status, schedule impact, etc. They are good milestones internal to the major project steps.

**Waivers and deviations to standards**

Most MIS organizations have standards, test plans, procedures, and other control documents. QA programs require adherence to the appropriate control documents. However, maintenance takes place on systems that were probably developed under few or no control documents, under a different set of control documents, or in a different environment. Standards and procedures now used may not be applicable. The old standards may even conflict with those now used for development. Maintenance projects should be treated in ways similar to those employed for new development. The system should be brought up to current standards. The desire for bringing the systems into adherence is not practical in some cases.

Waivers are conditions where a standard, plan, procedure, etc., is completely dropped. Deviations are cases where temporary or very isolated changes are made relative to control documents. With documented waivers and deviations, the test plans, standards, etc., can be referenced by exception. It would be assumed that the normal or standard procedures were executed unless a waiver or deviation was requested.

There is risk associated with each deviation. Documentation of deviations is necessary to properly assess the risk, determine need for document changes, and make later maintenance more effective. The method of quantifying software quality discussed by Mendis is based on careful documentation of deviation and failure conditions. Only the QA function can approve the maintenance waivers and deviations.

**LEVEL OF QA CHECKS**

In Figure 1 there are seven triangles symbolic of QA checks or inspections. In the lower right corner of each triangle is a letter, A, B, or C. These letters designate the level of QA check.

There are several spectra to which the term level refers. These include detail, impact and scope, experience of personnel, degree of personnel responsibility, depth of personnel knowledge, breadth of personnel knowledge, and significance of results.

The QA function has the ultimate accountability for all QA checks. The actual execution of the check and the associated responsibility can be delegated to other people or organizations. The delegation is an attempt to place the responsibility for detecting and correcting problems, deficiencies, and errors at the functional level closest to the source and most directly affected by the discovered problems. The use of levels allows the QA checks to be performed more cost-effectively.

**Level C**

The Level C checks are the lowest level of the QA checks. This level of check is delegated by the QA function to the people or organizations actually performing the maintenance. These checks are usually performed by the people with the least general experience, the least management responsibility for the maintenance project, and the narrowest direct scope of responsibility. The checks at this level are inspections of the tasks or functions that have the most constrained impact on the project. Referring to Figure 1, all the Level C checks are off-line or monitoring steps.

The Level C check could be done by the programmer, by a supervisor or senior programmer, or by a QA inspector assigned to and closely associated with the task. The NBS Special Publication by Brandstad is a good tutorial on checks at this level. The Level C checks are normally performed in or near the same physical area where the task is executed.

The programmers physically performing the maintenance should not check or inspect their own work. Independence can be obtained by having the programmers rotate or exchange work. On a periodic basis, the pattern of exchange needs to be changed, because the effectiveness of the “independent” check may deteriorate if it is not changed.

**Level B**

The QA checks performed at Level B are delegated by the QA function to the project leader. The results of the checks done at this level usually have significant impact on the schedules and budgets of the task or project. There are three Level B checks in Figure 1. Two of the Level B checks are gates or in-line checks. The other is a monitor or off-line check.

The QA checks done at Level B are usually not performed in the same physical area where the work or task is done. This type of check would be done in a project work area or in the office of the project leader. The Level B check could be a consolidation of checks done at Level C. It might be a check of consolidated work or a consolidation task itself. The Level B check is a project milestone in most cases.
Level A

The QA checks at Level A are kept within the QA function. The checks done at this level are the most crucial, have the widest scope, and have the biggest impact on overall schedules and budgets. The checks done at Level A are also very informative in a project management sense. They are used to determine the adequacy and completeness of the checks done at Level C and Level B.

The Level A checks are similar to those done at Level B. A great deal of the effort of the Level A check is consolidation of previous checks. However, the final system acceptance evaluation is an extensive system execution. The system test would probably be executed against a standard or special set of test data.

The QA check done at Level A would seldom, if ever, be performed in the same physical area where the maintenance work was done. It would probably be performed in a work area dedicated to the QA function or in the office of the assigned QA personnel. Separation is used to maintain the independence and integrity of the Level A inspection or evaluation.

QA CRITERIA

The primary purpose of QA checks is to determine whether maintenance was performed properly. A secondary purpose is to determine the magnitude and associated risk of any deviation. The determination is made against a set of objective standards. Each step of the maintenance project process is different and has different goals. The various QA checks need different criteria oriented toward the object of the task being inspected.

The following sections give a representative set of criteria for each of the QA checks identified in Figure 1. When the QA program for software maintenance is implemented, those responsible for QA, system maintenance, etc., will have to determine whether the proposed set of criteria is correct for their department, systems, and people.

Preparation inspection criteria

These criteria apply to the QA check shown as Triangle 1 on Figure 1. This check is in-line or a gate, and is at Level B. The objective is to make sure that information needed in the later steps of the project is available. It is expected that the maintenance project leader will take a few extra minutes to check and double-check the status of the required documentation. Any oversights or omissions could cause significant delays during the execution of the maintenance.

The following criteria are the bare minimum. The project leader is given the authority to make additional checks and add criteria based on the case, people, system, and situation. The criteria are as follows: (1) Complete, or sample, data files must be present or storage location must be referenced. When sample or test data are to be used, the source and general content of the data file are to be documented. (2) File and record descriptions are to be present, referenced, or otherwise accounted for. (3) Program descriptions, lists, specifications, and other appropriate information are to be present, referenced, or otherwise accounted for. (4) Status, background, and detail information necessary to execute the maintenance properly and to insure that it was done, must be available. (5) Test plans, system acceptance criteria, etc., must be present or available.

Compilation inspection criteria

A check is performed to insure that the changes made as part of the maintenance project did not contaminate the original source program. This check is shown as Triangle 2 on Figure 1. The QA check, Level C, is performed by the programming personnel in an off-line or monitoring mode.

If no error messages are generated by the compiler, the program or routine is considered to have passed this Clean Compile check. Error messages are to be cleared or disposed of on the basis of criteria given below. The classification of error messages is based on the ANSI COBOL compiler used on the Univac 1100 series of computers.14

1. Leveling: Violations of ANSI or Federal Standard COBOL.15,16 This error message is conditionally acceptable. A request for deviation is needed. The request should discuss the reasons for using the extension. Include plans or suggestions to remove the condition where applicable.

2. Remarks: Actions taken by the compiler, but not necessarily an error in the source program. This class of error message is generally acceptable. Where possible, and reasonable, the condition should be removed for the sake of clarity.

3. Minor: The compiler generates code based on assumptions. Attempt to remove all error messages in the minor class. When it is considered necessary and reasonable to keep the condition, a request for deviation is to be filed. The request should include a plan to remove the condition where appropriate.

4. Serious: The compiler is unable to make reasonable assumptions, and no code or incomplete code is generated for statement. Errors in the serious class are not acceptable. Corrections will be made until all errors of this class are removed. When the serious error cannot be removed after reasonable effort, a compiler specialist will investigate the problem and provide a solution. If the specialist feels the risk is warranted, a deviation can be issued until solution is found.

5. Fatal: No code for the program or routine is generated. Errors in the fatal class are not acceptable. Corrections will be made until all errors of this class are removed. When the fatal error cannot be removed after reasonable effort, a system or technical specialist will investigate the problem. The subsystem, and possibly the project, will not be allowed to proceed until a solution is found.

6. Compiler Errors: Internal conditions that generate errors during the compilation. This class is obviously not
acceptable. The compiler specialist will work with the experts at the vendor to remove the condition.

Data conversion criteria

The QA check associated with the data conversion step is shown as Triangle 3 on Figure 1. This is a Level C check performed in an off-line mode.

Typically, the data are converted by using a system utility, a local utility, or a special program. The utility routines are executed by using directives that give parameters, selected options, etc. The following criteria associated with the use of utilities and directives must be met: (1) There must be no syntax or format errors in the directives. (2) There must be no obvious execution errors associated with the directives.

In most cases, a test is made for input and output record counts and control totals within both utilities and special routines. The following criteria apply to the record counts and control totals: (1) There are to be no unexpected mismatches in the record counts or control totals. (2) There are to be no unexpected changes in the data size parameters like record size, blocking factor, and file size.

When the specified criteria are not met, an investigation is to be made. Disagreements involving the acceptance or rejection of a data conversion QA check will be resolved by the project leader. It is assumed the project leader will consider technical advice from a data or system specialist.

Subsystem Test and Debug Criteria

The QA check for Subsystem Test and Debug is indicated by Triangle 4 on Figure 1. This check is a Level A check done in an off-line mode. Though the QA function does not actually perform the tests, the status of success or failure is of prime interest and concern. The QA function is to see that all the tests required by the appropriate test plan are carried out. The presence and adequacy of the test plan was one of the criteria points in the QA check for preparation. The NBS Special Publication by Adrion is a brief tutorial on testing procedures. Smith is an author of some other books and articles published recently on the topic of testing and validation techniques. The QA function acts as an overseer and expeditor. Any disagreements that arise about the acceptance or rejection of tests will be settled by the QA function.

The test and debug process for the subsystem is executed after all the routines and programs have been successfully tested. There is no formal QA check during the debug phase of programs and routines. It is assumed that all the programs and routines have passed their tests, so any problems found at the subsystem level are considered very serious. If the fault cannot be quickly cleared and resolved, one of the previous steps will become the point for starting maintenance over.

One of the techniques often used to determine the success of a subsystem test is a file and report comparison. This approach is very effective when maintenance does not result in major changes in the data file structures or report contents. It is often easy to use a system utility or a local utility to do the file comparison. The following criteria are to be met when comparisons are used to evaluate success: (1) Files must have no unexpected mismatches, and (2) reports must have no unexpected mismatches. Expected differences can be cleared by documenting them with a request for deviation. Unexpected differences that cannot be easily removed are to be documented for further investigation.

Preparation for system integration criteria

The QA check for this step is Triangle 5 on Figure 1. It is a Level B check performed in a gate or in-line mode. The objective of this QA check is to make sure that all the documentation required for system integration is present. A second objective is to insure that all previous QA checks have been performed to a satisfactory degree.

The check is to make sure that the system can be reintegrated from the maintained subsystem. The check is also used to make sure that the new documentation can be easily generated. The project leader is expected to take sufficient time to check, double-check, and even triple-check the status of documentation. Any omission or oversight would cause major setbacks. The project leader should take enough time to insure that there will be no major problems found at system test.

The project leader is given the authority to make additional checks and add criteria based on case, people, system, and situation. The following criteria are the bare minimum: (1) Flow charts are to have adequate explanations. (2) Updates to the file and records descriptions must be present or referenced. (3) Updates to programs and routine descriptions, lists, and other appropriate information must be present or referenced. (4) Updates for generating operations documentation must be present or referenced. (5) All deviations and waivers are to be approved. (6) All documented problems are to be cleared.

Documentation update criteria

This QA check is the last step performed prior to final operational testing. The check appears as Triangle 6 on Figure 1. It is a Level A check performed as an in-line gate.

The QA check is to insure the presence of documents that will define the system at a later time. The following criteria are to be used for the check: (1) All internal MIS documents for the system must have been replaced or updated. (2) All entries in the data catalog, or configuration management mechanism, must have been updated. (3) User documents must have been updated, retrieved and replaced, etc. (4) The system history must have been brought up to date.

System acceptance criteria

The QA check for System Acceptance is shown as Triangle 7 on Figure 1. The system acceptance is the last step in the maintenance project. This check is at Level B and is really integrated into the acceptance procedure itself.

The users of the system play a major role in system acceptance. The criteria for the acceptance of a system will vary with
the features of that system. One of the criteria for the prepara-
tion QA check is the presence of system acceptance criteria.
The responsibility for controlling the execution of the accept-
ance procedure is left to the project leader. With well-defined
and adequate QA checks executed at each of the major steps
of the maintenance project, the system acceptance check be-
comes a simple procedure.

CONCLUSIONS

This paper has presented the structure and some details of a
QA program for software maintenance. The program is still in
its infancy. A QA program almost identical to the one
presented in this paper was used when our company converted
its systems from the computer of one hardware vendor to
another. Once the personnel have become accustomed to the
QA program, they should find it beneficial.

The QA program discussed in this paper can serve as a
model or prototype for other organizations. Those who intend
to implement a similar QA program should study the struc-
ture or features of the program rather than the details. The
most significant features of the QA program are the use of
three levels of QA checks, two types of checks, and the waiver
or deviation. These features make the program work, make it
effective, and make it acceptable to the personnel and man-
agement.

REFERENCES

Reading, Massachusetts: Addison-Wesley, 1980.
4. Roberts, T. J. "Maintaining Quality after the Software is Released."
5. Buckley, F. "A Standard for Software Quality Assurance Plans." Com-
puter, August 1979, pp. 43–50.
6. Dunn, R. H., and R. S. Ullman. "A Workable Software Quality and
Reliability Plan." Proceedings of 1978 Annual Reliability and Maintain-
8. Center, J. W. "Quantitative Measures of MIS Quality Assurance during
Hardware Conversion." AFIPS, Proceedings of National Computer Con-
10. Fife, D. W. Computer Software Management: a Primer for Project Man-
agement and Quality Control. NBS Special Publication 500–11, Washing-
dale, Ontario: Canadian Standards Association, April 1981.
Verification, and Testing for the Individual Programmer. NBS Special
1977.
Corp., 1978.
National Standards Institute, 1974.
17. Adrion, W. R., M. A. Branstad, and J. C. Cherniavsky. Validation,
Verification, and Testing of Computer Software. NBS Special Publication
Validation and Verification: A General Guideline. BCS-40342. Seattle,
Software: Validation and Verification. Document BCS-40343. Seattle,
Howden. "An Approach to Transfer Verification and Validation Tech-
(Vol. 50), pp. 367–373.