RISK AND CONTROL OF THE SOFTWARE MAINTENANCE PROCESS

By Frederick Gallegos, CISA, CDE
U.S. General Accounting Office

The increased demand for new or improved computer information systems application has created a number of problems for the DP professional, the users, and management. Staff and budget constraints limit the ability of DP managers to respond to this demand. Antiquated or poorly designed systems have created severe maintenance and performance problems, leaving little time to develop new applications or improve existing systems. Thus, the risk and control of maintaining the software is an important goal, an institutional goal toward which all must work.

In response to this goal, the National Bureau of Standards issued its Federal Information Processing Standards Publication Number 106, entitled *Guideline on Software Maintenance*. The Guideline provides general guidance for managing software maintenance. It does an excellent job of communicating that improvements in the area of software maintenance will come primarily as a result of the software maintenance policies, standards, procedures, and techniques instituted and enforced by management.

Controlling the software maintenance process is an institutional goal, which management, users, and DP professionals must collaboratively work toward. Controls involve how well they work together in performing the software maintenance process illustrated in Figure A.

Everything that is done to software affects its quality. Thus, measures should be established to aid in determining which category of changes are likely to degrade software quality, especially impact to the user. The primary purpose of change control is to assure the continued smooth functioning of the application and its orderly evolution. Examples of such controls are described in Figure B.

FIGURE A

THE SOFTWARE MAINTENANCE PROCESS

1. Determination of need for change
2. Submission of change request
3. Requirements analysis
4. Approval/rejection of change request
5. Scheduling of task
6. Design analysis
7. Design review
8. Code changes and debugging
9. Review of proposed code changes
10. Testing
11. Update documentation
12. Standards audit
13. User acceptance
14. Postinstallation review of changes and their impact on the system
15. Completion of task

FIGURE B

CONTROLLING SOFTWARE MAINTENANCE

1. Review and evaluate all requests for changes.
   —Require formal (written) requests for all changes.
   —Review all change requests.
   —Analyze and evaluate the type and frequency of change requests.
   —Consider the degree to which a change is needed and its anticipated use. All changes should be fully justified.
   —Evaluate changes to ensure that they are not incompatible with the original system design and intent. No change should be implemented without careful consideration of its ramifications.
   —Emphasize the need to determine whether the proposed change will enhance or degrade the system.
   —Approve changes only if the benefits outweigh the costs.
2. Plan for and schedule maintenance.
   —Assign a priority to each change request.
   —Schedule each approved change request.
   —Adhere to the schedule.
   —Plan for preventive maintenance.
3. Restrict code changes to the approved work.
4. Enforce documentation and coding standards through reviews and audits.

The inherent risks of software maintenance are many. If organizations do not properly manage and assess the process from an institutional standpoint, systems may evolve to a state where the organization's ability to meet the information demands of its users and decision making will in fact affect its profitability. The risks are having systems develop characteristics such as those shown in Figure C. The greater the number of characteristics found in a system, the greater the potential for disaster and need for redesign.

FIGURE C

CHARACTERISTICS OF SYSTEMS WITH HIGH RISK AND ARE CANDIDATES FOR REDESIGN

1. Frequent system failures
2. Code over 7 years old
3. Overly complex program structure and logic flow
4. Code written for previous generation hardware
5. Running in emulation mode
6. Very large modules or unit subroutines
7. Excessive resource requirements
8. Hard-coded parameters which are subject to change
9. Difficulty in keeping maintainers
10. Seriously deficient documentation
11. Missing or incomplete design specifications

Software maintenance is an institutional issue which all participants must work towards in achieving controllable systems in these times of dynamic technological change. It is a goal which can be reached if supported by all concerned.