16 Critical Software Practices
for Performance-Based Management

Jane T. Lochner
U.S. Navy

The 16-Point Plan™ focuses on effective management and technical processes for improving the bottom-line: detecting defects, managing complexity, reducing rework, eliminating excessive and unnecessary costs, and increasing productivity. It addresses three primary areas of software management: project control, product construction, and product integrity. The practices were forged in the crucible of real-world pressure to succeed and represent the combined experience of successful program managers and industry leaders. Recognizing that change is difficult, the 16-Point Plan recommends small but powerful steps that can be introduced into an established program.

THIS ARTICLE DRAWS ON information in the Software Program Managers Network's (SPMN) new program manager's guide titled, *16 Critical Software Practices for Performance-Based Management*, which is due for initial release in Fall 1999. The 16 Critical Software Practices™ for Performance-Based Management (the 16-Point Plan) are applicable to all large-scale, software-intensive projects (i.e. projects relying on the full-time efforts of 12 or more people). The practices that comprise it, however, are scaleable, all or in part, to smaller projects and to different software project environments. The 16-Point Plan was developed by SPMN with assistance from members of the Airlie Software Council, about 20 of the nation's leading software experts convened to assist SPMN in the identifying of industry best practices. The guiding principles used in developing each of the practices were that each practice be:
- applicable to all types of software and life cycle models
- flexible
- nonproprietary
- specific
- measurable
- realistic and attainable
- readily implementable

The practices identified satisfy these guidelines, making the 16-Point Plan a firm foundation for project success. This plan should be used according to the circumstances and environment of a given project, including where it is in its life cycle when the 16-Point Plan is first adopted. All practices are generally applicable to both government and industry projects and to nearly all domains. The plan is focused on effective management and technical processes, including techniques for finding defects as they occur, managing complexity, reducing rework, eliminating excessive and unnecessary costs, increasing productivity, and other beneficial effects.

The practices that comprise the 16-Point Plan are termed "critical" because software project managers and organizations, whose bottom-line performance is consistently better than average, use these practices and consider them essential. The bottom-line that the buyers of software development are interested in consists of:

- end-user satisfaction
- development and maintenance cost
- time-to-market
- quality
- predictability of final cost and schedule

Each of these critical practices is supported by metrics from past large-software-intensive system development and maintenance projects. These practices have been forged in the crucible of real-world pressure to succeed.

The 16-Point Plan is not presumed to be an exhaustive set of practices. However, the plan represents the combined experience of successful program managers and industry leaders. It will go a long way toward engendering success in any software development or maintenance effort.

As illustrated in Figure 1, the 16-Point Plan addresses three primary areas of software management:
- **Project Control**. It includes those practices that result in the identification of basic project constraints, expectations, and metrics. It also encompasses practices to plan and implement a project environment to predictably satisfy customer expectations and constraints.
- **Product Construction**. Includes those activities that specify the basic product requirements; maintain traceability to these basic requirements; and control content, change, and use of the many artifacts and deliverable products that are produced to satisfy user and customer requirements and expectations.
- **Product Integrity**. This ensures that defects, which occur as part of the software process, are identified and removed in a timely fashion. Product integrity ensures that testing is complete and effective and results in the right product consistent with agreed-to requirements and actual expectations.

While the practices that comprise the 16-Point Plan are individually useful, their complementary nature provides a strong synergistic effect when used as an integrated set. Using them will not guarantee success, but they can help facilitate it.

Those familiar with process improvement models, such as the Capability Maturity Model (CMM®), will quickly realize

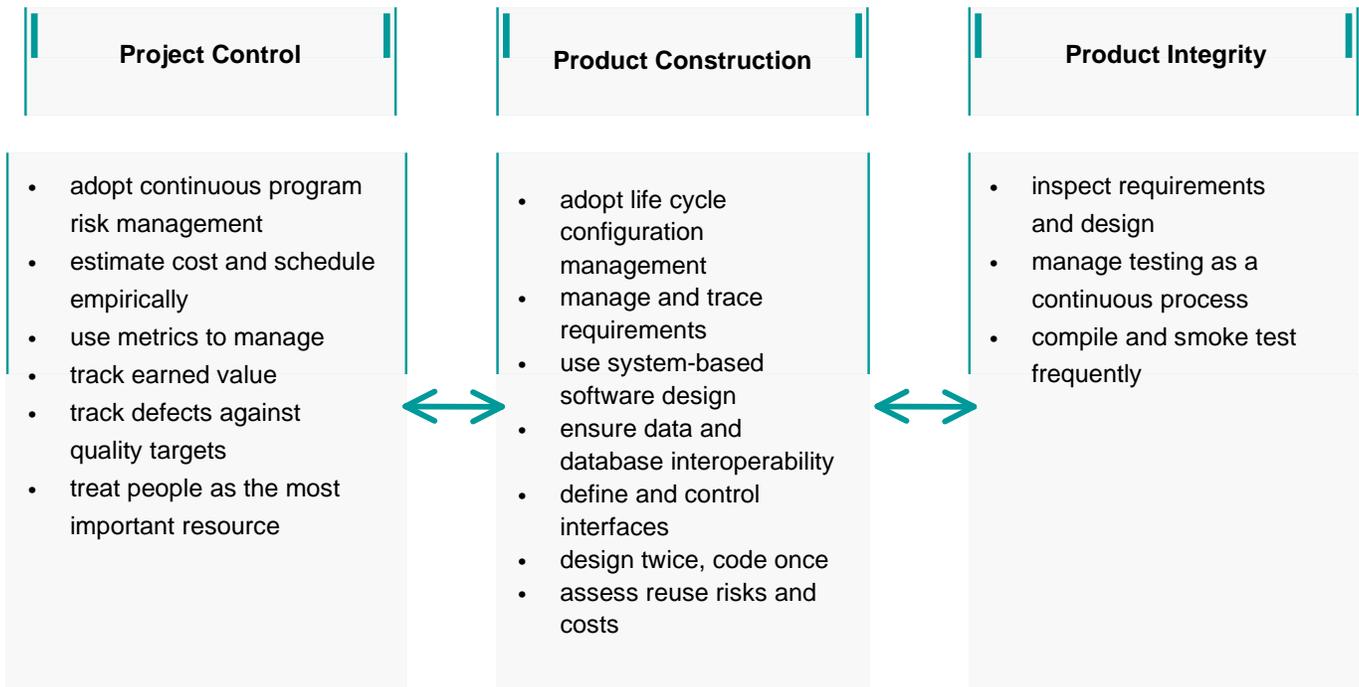| Project Control	Product Construction	Product Integrity
• adopt continuous program risk management • estimate cost and schedule empirically • use metrics to manage • track earned value • track defects against quality targets • treat people as the most important resource	• adopt life cycle configuration management • manage and trace requirements • use system-based software design • ensure data and database interoperability • define and control interfaces • design twice, code once • assess reuse risks and costs	• inspect requirements and design • manage testing as a continuous process • compile and smoke test frequently

Figure 1. *The 16-Point Plan.*

that these practices supply tactical solutions to the model's strategic orientation. The practices map to many of the model's key process areas and should assist organizations striving to advance to the next CMM maturity level.

These practices are straightforward, readily implementable techniques. Although some practices may require training in basic skills, such as conducting structured meetings as a necessary foundation for formal inspections, they can, for the most part, be implemented without making investments in new equipment, technologies, or staff. Cultural resistance to the discipline inherent in these practices and to the management visibility that comes from several of the practices, is the biggest obstacle to successfully implementing these practices. Another stumbling block is that organizations inexperienced in some of the practices in the plan may think they are unnecessary. Because they do not use the practice at the present time, they may not recognize the value and benefits to their organizations that implementing these practices would bring.

The 16-Point Plan can be used with an established program. It recognizes that change is difficult and recommends small but powerful steps to initiate each of the 16 practices. Although the 16 practices cannot make successful those programs that are inadequately funded, without proper staffing, and faced with impossible schedule deadlines, implementing these practices can minimize damage.

The 16-Point Plan

The 16-Point Plan takes a two-dimensional approach to the critical practices. First, it takes a vertical approach, explaining the details of each practice. Each element is examined, identifying "practice fundamentals." These are key principles that outline the essence of the practice. Implementation guidelines also

are identified. These are practical steps that can help implement the practice in a given program. A set of "quick look" questions is provided to help the program manager make a crude assessment of whether his/her program has a potential problem in each area and a list of "alarms," which relate to lessons learned in each area. Each practice specifies associated metrics that the project manager should monitor. Finally, each practice concludes with a list of detailed questions that should be asked if a project is unable to satisfactorily answer the "quick look" self-assessment questions, following with the recommended corrective actions.

Then it takes a horizontal perspective, describing how individual practices might be sewn together into an effective program. The 16-Point Plan associates the 16 critical practices by program phase. Taken together, these two perspectives provide a model that can be applied to any project and move it toward success.

The following is an example of some of the practice details contained in the 16-Point Plan.

Project Control

To be successful, the right project environment must be established. This environment must be conducive to establishing a stable team, identifying, organizing, and coordinating tasks, and recognizing and mitigating risks that can impede success. Most of all, this environment must make it as easy as possible for team members to pull together toward success.

Project control brings together a series of six management practices that can help establish a success-oriented environment. These provide early indicators of potential problems, coordinate

The Capability Maturity Model and CMM are registered in the U.S. Patent and Trademark Office.

the work and the communications of the development team, and achieve a stable development team with the needed skills. These practices are essential to delivering the complete product on time, within budget, and with all documentation required to maintain the product after delivery.

Achieving project control requires developing a detailed activity network for all effort to at least the next delivery, an estimate of the cost and schedule for this effort, and allocation of the cost and schedule estimate. Planning is the basis of project control, establishing a method for conducting business, a management process, and a quantitative basis for monitoring progress and risk. The goal of planning is to establish a working project management environment, not the production of a plan for the sake of meeting a requirement to produce a plan. "Use Metrics to Manage" is one project control critical practice. Examples of "quick look" self-assessment questions that might be asked regarding metrics are:
- Have threshold values been established?
- When was the last time the metrics showed an anomaly (were not what was expected)?

Alarms that indicate that the metrics program is not being taken seriously include:
- A large price tag attached to request for metrics data.
- Rebaselining is frequently required.

Collecting metrics solely to collect metrics is not a best practice. It is, in fact, counterproductive. Since the value from a metrics program comes from the actions taken as a result of metrics analysis, one should track the percentage of decisions made based on metric data.

When the "quick look" self-assessment questions indicate a lackadaisical metrics program, the program manager should asked more probing questions such as:
- Are there threshold values for early problem indication metrics that trigger reporting to higher levels of management? If so, for each metric with such threshold values, what are the threshold values and to what level of management does each value trigger a report?

A worthwhile metrics program must measure the right metrics. The following steps can be used to identify the proper metrics:
- Define program issues/problems/risks
- Identify reporting obligations/needs
- Determine what indicators would show problem areas
- Sort indicators into metric categories
- Determine the delta from metrics currently collected and metrics needed
- Identify additional reporting collection delta
- Identify ranges/metrics

Product Construction

Projects need a common means of doing business as well as a common language process during construction to ensure communication among suppliers/developers, users, programmers,

analysts, project leaders, program managers, and the program executive officer.

Although projects are never the same, the process should be consistent because projects require discipline and predictability. Before planning how something will be accomplished, it is useful to understand what has to be done. Techniques must be defined before they can be integrated into a project; and while innovative technology is often required to meet project goals, their impact must be realistic and have broad project support. Esoteric solutions and the use of leading edge technology not tailored to project objectives are counterproductive.

Essential to construction are tools that sustain project requirements, not vice versa. Automated tools, supported by configuration management, solve project problems more efficiently than manual techniques. However, automated aids are useful only if they satisfy an identified need and are defined and selected in a top-down sequence. All too often construction starts with the tool, forcing the tool to fit the problem. Improper tool selection and application result in data rework at the low end and wasted work/scrap at the high end.

The discipline of configuration management (CM) is vital to the success of any software development effort. Two questions which give a CM "quick look" are:
- Can you access the earliest/most recent version of a software system?
- Can you produce the change documentation for the approved last change to the current system?

A CM process is probably not effective if any of the following alarms occur:
- The Configuration Control Board (CCB) merely rubber stamps requests; requests are submitted "after the fact." There is not a mixture of accepted, rejected, and held for further investigation actions.
- The CM process is considered level-of-effort and not tied to specific tasks/products.

Some metrics which measure the effectiveness of CM are:
- number of days since last change to library documents
- turn-around time for CM products

When uncertainty arises concerning the CM process, one should ask detailed questions such as:
- Have several people described the CM approach and process? Are these descriptions consistent? Do they match the documented process?
- When under heavy schedule pressure, are changes made to code without going through a controlled change process managed by CM?
- Are CCBs fully assessing the impacts of each proposed change or the risk and cost of making the proposed change prior to authorizing that change? Are all impacted configuration items identified?

For CM to be effective it must be empowered. Corrective actions to empower the CM team include:

• Charter the CM organization. Give it a clear mission, responsibilities, and authority.
• Staff it adequately with experienced developers.
• Train the team thoroughly in the tools that are to be used.

Product Integrity

Software development is a continuum of events, one building on the next. If one is done poorly, subsequent activities that build on the work suffer. Project success and acceptability criteria depend on managing the project to ensure quality. Generally, there will never be time to clean it up. When problems occur, options are limited. It is better to manage quality from the beginning.

"Compile and smoke-test frequently" helps ensure that the product is growing in a controlled manner. These questions provide a "quick look" assessment of how well an organization is following this practice.
• Is the build of the current system baseline more than five days old?
• Can the CM group build the current system baseline unaided?

Management needs to scrutinize the compile and smoke-test process if any of the following alarms occur:
• gradual increases in the number of changes included in builds or in the time between builds
• use of binary patches

Management can use the following measures to monitor the compile and smoke-test practice.
• days since last build
• number of problems identified during smoke-tests

When the compile and smoke-test practice needs redirection, detailed questions, such as the following, can help pinpoint problem areas.
• Can the customer explain the regression and smoke-test philosophy to an outside organization?
• Do the regression and smoke-test suites address all capabilities in the current configuration?

Implementing an effective smoke-test strategy requires:
• building systems and executing tests at least twice a week.
• smoke-testing systems built only from the central CM library. Test files, stubs, drivers, or other components not held by the CM system must not be used.
• smoke-tests based on a pre-approved, traceable procedure

run by an independent organization — not the engineers who produced the change package.

Conclusion

The 16-Point Plan integrates the critical software practices into a road map that can help program managers navigate around the hazards and obstacles that often block the path to success. It is a set of high-leverage practices that distill the experience of successful program managers into an executable strategy that can be applied to virtually any development effort. It is a starting point for structuring and deploying an effective process for managing large-scale software development and maintenance, but must be tailored to the particular culture, environment, and phases of a program.

The 16-Point Plan incorporates proven commercial best practices and focuses on the essential details of each practice necessary to achieve high return on investment, bottom-line improvements. Together the practices constitute a powerful set of technical and management disciplines that can be put in place quickly to achieve rapid bottom-line results. Successful implementation of these essential details should ensure big savings. Of course, these practices cannot save "death march" programs that are expected to deliver under impossible schedule deadlines with inadequate funding and without the required skilled staff. ◆

About the Author

Jane T. Lochner is a 1984 U.S. Naval Academy graduate. She served aboard USS Norton Sound (AVM-1) and USS Cape Cod (AD-43). She was selected to the Engineering Duty community in 1988. She has extensive experience with developing and fielding complex, real-time combat systems on aircraft carriers and large-deck amphibious ships. Currently, she is assigned to the Office of the Assistant Secretary of the Navy for Research, Development, and Acquisition working command, control, communications, computers, intelligence, surveillance, and reconnaissance and interoperability issues. She holds a bachelor's degree in marine engineering, master's degrees in logistics, applied physics, and computer science, and is a graduate of the Defense Systems Management College Program Manager's course.

U.S. Navy
1000 Navy Pentagon
Washington, D.C. 20350-1000
Voice: 703-602-6887
Fax: 703-601-0346
E-mail: lochner.jane@hq.navy.mil