Chapter 16

Sustainment and Product Improvement

CONTENTS

This page intentionally left blank.

Chapter 16

Sustainment and Product Improvement

"My name is Ozymandias, king of kings: Look on my works, ye Mighty, and despair!"
– Percy Bysshe Shelley, "Ozymandias"

16.1 Introduction

In Shelley's poem, the king of kings proclaims his greatness by warning all others to despair after beholding his unequaled excellence. Ironically, it becomes a warning to all who think lofty thoughts of themselves; no matter how great we think we are, or how well we have done our work, there will always be imperfections. And those imperfections lead to a finite lifetime. Anyone who has built or accomplished something will see his work superceded or his record broken if he lives long enough. The statue of Ozymandias had fallen into ruin until it was nothing but two legs of stone and a face half buried in the sand of the surrounding desert. And thus it is with software. There will come a time when its ability to serve will wane. But don't let thoughts of the end keep you from experiencing the excitement and joy of building and accomplishing something in our time.

Eventually, almost any software or system will be upgraded or replaced. From the moment users get their hands on the new system they will try to break it. Is it because of their evil natures? Yes. But we still need to learn to live with it. Besides, all of us have been users at some point. We realize the software would be better if it did this function or did that function better. It takes too long to start up, to find information, or requires more or less input from us than it should. Or there's a new generation of computers that will allow a whole new generation of functionality. Or, and I hesitate to mention it, there are just a few bugs that need to be fixed. It gets harder and harder to convince people that the bugs are "features."

While it is sometimes better to throw out the old system and build a completely new one, we usually find it more advantageous, time and money wise, to upgrade or fix the original software. During the great Year 2000 scare, companies spent billions of dollars upgrading software that was decades old because it was cheaper and took less time than designing, building, testing, and moving to a new system. This is what software sustainment is all about – fixing problems and adding functionality to existing software.

This chapter covers two approaches to building better software. The first, sustainment, has already been introduced. The second is to improve the process we use to build software, making it better and cheaper simply by the way we do things. One deals with improving an existing product. The other approach is to improve the way we build software in the first place. This is not an either or choice. Most software will go through sustainment cycles, and all software organizations need a program of constant process improvement.

16.2 Software Sustainment

Industry often refers to software maintenance. However, the term "software maintenance" is a misnomer. A more correct term would be software sustainment. Why? Because software is not maintained in the same way hardware is maintained. When hardware fails, the repairperson replaces the failed part with an identical but functioning part. When software fails, the software engineer does not replace the offending code with an identical piece of code. The code must be modified to function correctly. Tests must be performed to verify the revised code corrected the problem. This is why we hear quotes that sustainment will account for somewhere between 60 and 90 percent of the total lifetime costs of a software product. Exactly where that number lies today is unknown. It is sufficient to know that it is costly and that it is the longest phase of the software life cycle. The reasons for this costliness are that software will likely be upgraded several times, and it is often difficult to understand what the original developers were doing in any given part of the software so that it can be modified. Great strides have been made to improve the sustainability of software, including:

- Structured programming
- Object oriented programming
- Developing software with sustainment in mind
- Better programming languages
- Better-defined and better-followed development processes
- Computer Aided Software Engineering (CASE) tools
- Better and more consistent documentation

It should be noted that some of these reasons are based on better technology while others are dependent on the cognizance, consideration, and efforts of developers. Implementing these methods should be considered wherever possible to reduce the total cost and improve the quality of software.

Sustainment is often thought of in the context of fixing bugs, but it can be of four different types, depending on the reason or need. While a sustainment effort may be precipitated by a single type of sustainment need, most efforts include two or more sustainment types. The four types are summarized here. [2]

1. **Corrective Sustainment** – diagnosis and correction of program errors after its release.
2. **Perfective Sustainment** – the addition of new capabilities and functionality to existing software.
3. **Adaptive Sustainment** – modification of software to interface with a changing environment.
4. **Preventive Sustainment** – modification of software to improve future maintainability or reliability.

Corrective sustainment requires examination of the existing program code to determine the cause of the error, analysis to determine the best way to correct the error without introducing new errors, and regression testing to validate that the original error has been eliminated without introducing new errors. Perfective and Adaptive sustainment usually involve a complete development effort with requirements, design, coding, and integration and test phases. Preventive sustainment is performed by reverse engineering the existing software and re-engineering (redeveloping) it.

16.2.1 Sustainment Process

The software sustainment process can be very similar to a regular new product development process, depending on the software being upgraded. If the software is relatively modern, developed with current languages and methodologies, and properly documented, it becomes a development project where other programmers have already completed their designs and current programmers are walking through the designs and adding to or correcting them. It's almost as if they are all working on the same effort, just separated in time.

At the other extreme, there are programs which are still being used for which there exists no development documentation. The implementation language has not been used for years and there are few, if any, programmers with working knowledge of the language or system. Why not just build a new system? Because the technology of the entire system is old enough that no one working with it knows how everything works or what is going on behind the scenes. Remember, when we fix things we often introduce new errors into the system. If we're working with a weapon system, we need to know exactly what we're doing. Additionally, some older systems are resource constrained, and newer languages may require more resources (memory, processor speed, etc.) than are available.

Figure 16-1 depicts a generalized software sustainment process. As you see, it's very similar to the standard waterfall development effort. Just as the waterfall life cycle is not used for many current development efforts, the sustainment process may also be modified to better fit the type of project.

Any sustainment effort begins with identifying and gathering requirements. What needs to be fixed? What bugs have been found? What additional features need to be added? This is an exercise that involves both the developers and the users. It will also involve meetings where requirements are presented to make sure everyone agrees on them. They will also likely need to be prioritized, since sustainment funds and time are probably limited.

During the analysis phase, the requirements are validated and feasibility studies are performed. What requirements can reasonably be met within the cost and time of the sustainment effort? This requires familiarity with the existing system. The results of the analysis are reviewed with the users and likely result in modification and reprioritization of requirements. If you can't have everything, what is most important to you?

With the requirements agreed upon, software design begins. Existing software modules are redesigned and new modules are introduced. A new software architecture is produced and documented. Design is usually performed in phases as discussed Chapter 15. It includes a functional and a system design, followed by a programming phase. The difference between sustainment and a new development is that some modules already exist and will need to be modified to operate properly in the new system architecture. The programming phase also includes unit testing.

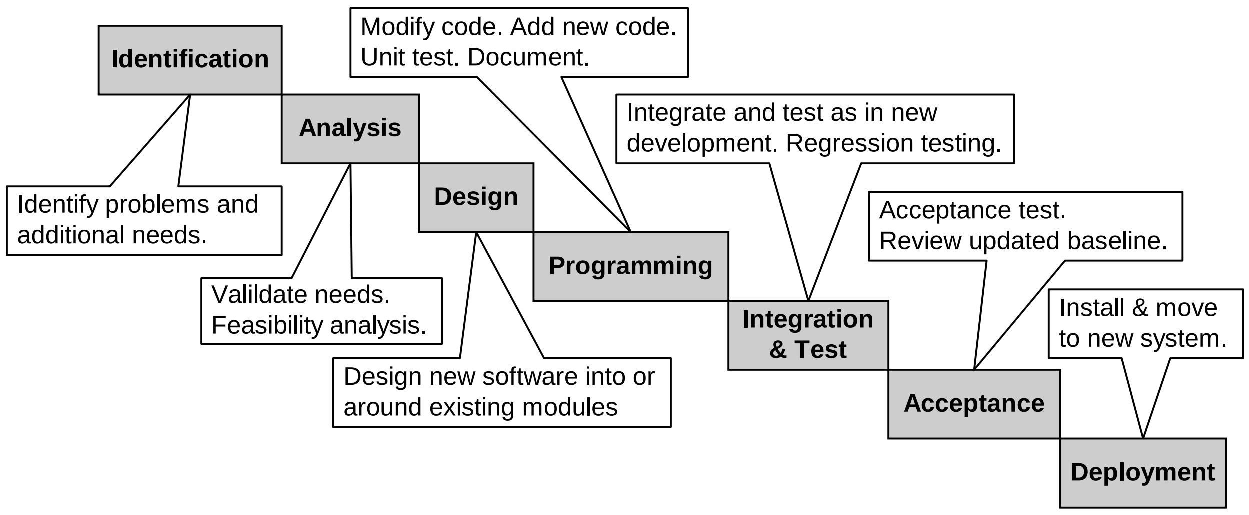
Identification
Analysis
Design
Programming
Integration & Test
Acceptance
Deployment
Identify problems and additional needs.
Valildate needs. Feasibility analysis.
Design new software into or around existing modules
Modify code. Add new code. Unit test. Document.
Integrate and test as in new development. Regression testing.
Acceptance test. Review updated baseline.
Install & move to new system.

Figure 16-1 Generalized Software Sustainment Process [1]

When the software modules are complete and tested, they are integrated and tested as subsystems and eventually as a complete system. A very critical part of integration and testing is regression testing. Just as the first rule of medicine is to do no harm, the first rule of sustainment is to not destroy existing functionality. Regression testing ensures that original functions are still intact, in addition to the improved functionality sought through the sustainment effort.

When the new system is functioning as required, it is presented to the customer for acceptance testing. When the system is formally accepted, the system baseline is updated to include the new system.

Following acceptance the new system is deployed according to plan. Deployment must follow or be accompanied by adequate training and documentation for operating and maintaining the new system. It must also be performed in such a way as to not interfere unduly with necessary normal operations.

16.2.2 Other Sustainment Considerations

A sustainment plan should be developed for each software system. It should include recommendations for identifying and collecting problems and candidates for improved functionality. It should also include instructions for maintaining proper documentation, source code libraries, and development history for future sustainment efforts. Thought should also be given to maintaining development software (compilers, etc.) for languages that are likely to change or fall into disuse.

In addition to modifying the software, a key sustainment activity is the development of an installation and transition plan to move from the old system to the new. This is a critical activity and should include fallback contingencies if something goes wrong. By the way, few transitions go completely right.

16.3 Product Improvement

Product improvement is the result of consistent, deliberate effort to move the developing organization to a higher level of capability. It requires recognizing and admitting the current state of the organization. It also requires a planned and guided path to excellence, reaching various levels of improvement as intermediate goals along the way. Improvement goes far beyond product quality. It sees product quality as a byproduct of the total development capability of the developers. In other words, if we improve our process, the product will be improved as a natural consequence.

Different groups and professional organizations have from time to time endeavored to establish guidelines and standards for excellence in organizations and business. One attempt to set standards for quality is the International Standards Organization (ISO) 9000 series of standards. Another is the software Capability Maturity Model Integration (CMMI) developed by the Software Engineering Institute at Carnegie Mellon University. While there are other strategies, these two are discussed in this work because of their widespread acceptance and success in the industry.

16.3.1 ISO 9001

The ISO 9000:2000 series seeks to establish standards for business operations, manufacturing, development, and other activities. Of particular interest to us is ISO 9001, the standard in the series that pertains to software development and sustainment. It identifies the minimal requirements for a quality system needed to develop and supply a product. The standard includes various sections pertaining to most aspects of a developing organization. They are:

- General Requirements
- Documentation and Records
- Management commitment
- Customer focus
- Quality policy
- Quality system planning
- Organization and communication
- Management review
- Provision of resources
- Human resources and training
- Infrastructure
- Work environment
- Planning of product realization
- Customer-related processes
- Design control
- Purchasing
- Operations
- Measuring and monitoring equipment
- Planning of monitoring and measurement
- Monitoring and measurement
- Control of nonconforming product
- Analysis of data
- Continual improvement

If an organization meets the minimum standards for these sections it can be certified as complying. ISO 9001 allows organizations to measure their processes against an industry gauge to see where they have strengths and weaknesses. If properly used, the standard shows companies what they need to work on to improve their efficiency, quality, and capacity. It provides a set of goals to which can be tailored for their particular industry and organization. Ideally, it will improve their standing in the market and improve the overall capability of the industry to produce better products, in shorter time, and at better prices.

TickIT is the British and Swedish extension to ISO 9001. It focuses on applying ISO 9001 to software development. More can be found regarding TickIT at www.tickit.org.

While ISO 9001 can be an indicator of competence, it does not guarantee it. If an organization focuses primarily on achieving the standards as a paper exercise, but does not incorporate the intent into the mindset and activity of the organization, the true benefit of the standard will not materialize. This problem can be alleviated by following an improvement strategy that emphasizes continuous improvement rather than meeting a standard. This is where the CMM comes in.

16.3.2 Capability Maturity Model Integration (CMMI)

Originally, there were several different versions of capability maturity models: one for software, one for system engineering, and one for software acquisition. Recently, these separate models have been integrated into a single model, the CMMI. Two different representations are available for the CMMI, a continuous representation (previously used by the System Engineering CMM) and a staged representation previously used by both the Software and Software Acquisition CMMs). The staged representation shows progress as a series of five levels. Each of these levels is described by certain attributes characterizing its level of competency. Each level is associated with process areas, and each process area is described in terms of common practices that support that level's goals. These levels, descriptions, and process areas are shown in Table 16-1.

Table 16-1 Key Process Areas in the CMM

Level	Description	Key Process Areas
5 – Optimizing	The focus is on continually improving process performance through both incremental and innovative technological improvements.	Organizational Innovation and Deployment Causal Analysis and Resolution
4 – Quantitatively Managed	Quantitative objectives for quality and process performance are established and used as criteria in managing processes. Quantitative objectives are based on the needs of the customer, end users, organization, and process implementers. Quality and process performance are understood in statistical terms and are managed throughout the life of the processes.	Organizational Process Performance Quantitative Project Management
3 – Defined	Processes are well characterized and understood, and are described in standards, procedures, tools, and methods. Projects tailor organizational standards, procedures, and methods for use by the project.	Requirements Development Technical Solution Product Integration Verification Validation Organizational Process Focus Organizational Process Definition Organizational Training Integrated Project Management for IPPD Risk Management Integrated Teaming Integrated Supplier Management Decision Analysis and Resolution Organizational Environment for Integration
2 – Managed	The projects of the organization have ensured that requirements are managed and that processes are planned, performed, measured, and controlled.	Requirements Management Project planning Project Monitoring and Control Supplier Agreement Management Measurement and Analysis Process and Product Quality Assurance Configuration Management
1 – Initial	Processes are characterized as ad hoc, occasionally even chaotic. Few processes are defined, and success depends on individual effort and heroics.	–

The level of capability is shown in the first column. It begins at the bottom of the table with the initial level of capability. This is characterized by ad hoc processes. Each new effort is treated differently and processes are seldom repeated. Capability matures through several levels until it reaches the top, or optimizing, level. Rather than being characterized by achievement of a static state, an organization at this level is in a dynamic state of continuous process improvement. This is where CMM differs from ISO 9000. Instead of reaching a plateau or mountain top and planting a flag, the CMM achiever attains the capability to climb mountains of specific heights and difficulty, and continues to climb that type of mountain while continuously striving to reach higher levels.

An organization begins process improvement by undergoing an evaluation of its current capability. There will usually be a disparity among the capabilities of different areas. Once the level of capability has been ascertained, planning and effort are directed toward improving processes in those deficient areas to move the organization to the next level. Studies have shown that it takes two to three years to advance to a higher level. Since levels are determined by an organization's ongoing activity, those who do not continuously strive for improvement will not only stop advancing, they will regress in their capability level.

Both ISO 9001and CMMI are far too complex to present more than the briefest of overviews here. Study some of the recommended resources at the end of the chapter to gain a fuller understanding of what they are and, more importantly, what they can do for you. If your organization is not implementing an improvement strategy, such as CMMI, it is highly recommended that the advantages and increased capacity of such an effort be considered and evaluated. In today's constantly changing world, no one has the luxury of staying on one place very long.

16.4 Sustainment and Product Improvement Checklist

This checklist is provided to assist you in understanding the sustainment and product improvement issues of your project. If you cannot answer a question affirmatively, you should carefully examine the situation and take appropriate action.

16.4.1 Sustainment

- ❑ 1. Is all your software developed with a goal to facilitate its future sustainment?
- ❑ 2. Do you understand the four types of sustainment and their purposes?
- ❑ 3. Do you understand the place and purpose of the sustainment phase in the software life cycle?
- ❑ 4. Do you understand your sustainment process?
- ❑ 5. Is there a sustainment plan?
- ❑ 6. Is there a process in place to gather problem reports and upgrade requests for the software?
- ❑ 7. Does the plan provide for reviewing, evaluating, and prioritizing upgrade requests?
- ❑ 8. Are all sustainment activity steps included in the plan?
- ❑ 9. Is there a transition plan to move to the upgraded system?
- ❑ 10. Have all activities been planned and organized to keep interference and downtime to the operating system to a minimum?
- ❑ 11. Does the plan call for running critical systems redundantly during testing and installation?
- ❑ 12. Do the deliveries include source code, documentation, and all else that is needed in addition to the software itself to continue maintaining the software?
- ❑ 13. Are all products under configuration control?

16.4.2 Product Improvement

- ❑ 14. Is your organization following a product improvement strategy?
- ❑ 15. Do you know where your organization is at, capability wise, relative to ISO 9001, CMMI or your chosen improvement standard?
- ❑ 16. Do you have a plan for achieving higher levels of capability?
- ❑ 17. Do your product improvement efforts emphasize improving processes to achieve product improvement?
- ❑ 18. Are your development processes documented?

❑ 19. Are your development processes consistent and repeated?

❑ 20. Does the leadership of your organization support continuous process improvement?

❑ 21. Is your organization committed to achieving a state of continuous improvement vs. a certificate of compliance with standards?

16.5 References

[1] Department of Energy (DOE) *Software Engineering Methodology*, Chapter 10: http://cio.doe.gov/sqse/sem_toc.htm

[2] *Program Manager's Guide for Managing Software*, 0.6, 29 June 2001, Chapter 12: www.geia.org/sstc/G47/SWMgmtGuide%20Rev%200.4.doc

16.6 Resources

Caputo, Kim, *CMM Implementation Guide*, Addison Wesley Longman, Inc., 1998.

Crosstalk Magazine: www.stsc.hill.af.mil/crosstalk/

- "Confusing Process and Product: Why the Quality is not There Yet": www.stsc.hill.af.mil/crosstalk/1999/07/cook.asp
- "It's Time for ISO 9000": www.stsc.hill.af.mil/crosstalk/1994/03/xt94d03i.asp
- "Coding Cowboys and Software Processes": www.stsc.hill.af.mil/crosstalk/1997/08/processes.asp
- "Improvement Stages": www.stsc.hill.af.mil/crosstalk/1998/10/cusick.asp
- "The Process King vs. the Cowboys": www.stsc.hill.af.mil/crosstalk/1997/08/backtalk.asp
- "Using the CMM Effectively": www.stsc.hill.af.mil/crosstalk/1995/10/usingcmm.asp
- "If You Get Straight A's, You Must Be Intelligent - Respecting the Intent of the Capability Maturity Model": www.stsc.hill.af.mil/crosstalk/1998/02/respecting.asp
- "Software Process Proverbs": www.stsc.hill.af.mil/crosstalk/1997/01/proverbs.asp
- "Preparing for a CMM Appraisal": www.stsc.hill.af.mil/crosstalk/1996/08/preparin.asp
- "Ten Things Your Mother Never Told You About the Capability Maturity Model": www.stsc.hill.af.mil/crosstalk/1998/09/kulpa.asp

International Standards Organization, ISO 90001, www.iso.ch (also available from http://qualitypress.asq.org)

Jeffries, Ron, "Extreme Programming and the Capability Maturity Model": www.xprogramming.com/xpmag/xp_and_cmm.htm

Paulk, Mark C., "Effective CMM-Based Process Improvement": www.sei.cmu.edu/publications/articles/effective -spi.html

Randall's Practical Resources Online, ISO 9000 and Quality Related Topics: http://home.earthlink.net/~rpr-online/Contents.htm#ISO9000

Software Engineering Institute, CMMI and resources: www.sei.cmu.edu/cmmi/

Software Insight Tool, Checklists for acquisition and risk mitigation, etc.: www.sed.monmouth.army.mil/sit/sitstart.htm

TickIT information, www.tickit.org

University of Massachusetts Dartmouth, Software Process Resource Collection: http://www2.umassd.edu/SWPI/

This page intentionally left blank.