Software Sustainment Now and Future

Mary Ann Lapham, SEI

Abstract. Today's systems are increasingly reliant on software which must be sustained into the future. To sustain these systems organizations must define sustainment, meet criteria to enter sustainment, and overcome some classic sustainment challenges. This article discusses these tasks along with historical parallel development and sustainment and potential future trends in software sustainment.

Introduction

This article provides an overview of current software sustainment practices and challenges within the Department of Defense (DoD) and a look at the potential future of software sustainment within the federal government. It takes a broad view based on a specific study done in 2006 which was not meant to be all inclusive for every software sustainment topic. Thus there are areas not covered that may be relevant to an individual situation. Areas such as open source software, anti-tamper, sustainment cost estimation, and specific authority and responsibility for transition to sustainment should be explored if relevant to your situation.

As today's systems become increasingly reliant on software, the issues surrounding sustainment become increasingly complex. The risks of ignoring these issues can potentially undermine the stability, enhancement, and longevity of systems in the field.

At the center of this puzzle are disparate definitions. Developers and acquirers have a general understanding that sustainment involves modifying systems and deploying changes to meet customer needs, but does this understanding align with common practice and the DoD's definition of sustainment? DoD Instruction 5000.02 describes sustainment as follows:

Life-cycle sustainment considerations include supply; maintenance; transportation; sustaining engineering; data management; configuration management; Human Systems Integration (HSI); environment, safety (including explosives safety), and occupational health; protection of critical program information and anti-tamper provisions; supportability; and interoperability [1, section 8.c.1.b].

The terms software maintenance and software sustainment are often used interchangeably. It is important to make sure that all stakeholders use the same terminology when discussing software sustainment.

The IEEE Standard Glossary of Software Engineering Terminology defines "software maintenance" as follows:

The process of modifying a software system or component after delivery to correct faults, improve performance or other attributes, or adapt to a changed environment [2].

Software maintenance consists of correcting faults, improving performance or other attributes, and adapting to a changing organization and technical environment. To be complete, there is usually a fourth category of maintenance activities focused on anticipated problems, or preventive maintenance[1] [3].

While DoD Instruction 5000.02 describes sustainment in detail, no authoritative definition of "software sustainment" exists. The SEI's working definition is as follows:

The processes, procedures, people, material, and information required to support, maintain, and operate the software aspects of a system.

Given this definition, software sustainment addresses other issues not always an integral part of maintenance such as documentation, operations, deployment, security, configuration management, training (users and sustainment personnel), help desk, commercial off-the shelf (COTS) product and license management, and technology refresh. Successful software sustainment consists of more than modifying and updating source code. It also depends on the experience of the sustainment organization, the skills of the sustainment team, the adaptability of the customer, and the operational domain of the team. Thus, software maintenance as well as operations should be considered part of software sustainment.

Criteria to Enter Sustainment

The Operations and Support phase of the Defense Acquisition Management System has two major efforts, Life-Cycle Sustainment and Disposal. The entrance criteria include an approved Capabilities Production Document (CPD), an approved Life-Cycle Sustainment Plan (LCSP), and a successful Full-Rate Production (FRP) decision [1, section 8.a and b]. In addition, the following criteria among others should be considered:

- Stable software production baseline–Most sustainment organizations will not accept software into sustainment until the software is stable. Merriam-Webster Online defines stable as "a. firmly established: fixed, steadfast; b. not changing or fluctuating: unvarying; c. permanent, enduring" [4]. However, in the realm of software stable can mean different things.

If one were to apply Merriam-Webster's definition to software, he or she could infer that a single instance of loss of availability or a system failure would indicate that the software is not stable. In other words, software is stable only if it does not have problems that cause it to stop working. For software, unfortunately, the definition of stable can be a subjective one from several different perspectives. One organization may be willing to accept software as stable if it only fails once a week, while others would deem this rate of failure too high and would not accept the software. In other situations, software may be considered stable if no Category 1 or 2 Software Trouble Reports (STRs) exist.

Defining the stability of a system depends somewhat on its intended use, its mission criticality, and the potential consequences if the system fails. For instance, a system such as navigation software or command and control software whose failure could result in loss of life should have more stringent requirements for maintaining stability than one that is business software supporting actions that could be postponed for hours or even days.

The program office should define the criteria for accepting a system as stable in the Sustainment Transition Plan. These criteria should at the very least identify the types of STRs allowed to be active in a system that is entering sustainment.

• Complete and current software documentation–Complete, current software documentation is paramount for the software sustainment organization. Without it, the sustainment organization has limited insight into how the software was designed and implemented. Incompleteness or omissions increase software maintenance costs because software engineers have to reverse-engineer the code to determine how it works. In addition, this process increases the risk of inadvertently introducing errors into the code. Well documented code is a plus and–for those using incremental and iterative methods–expected.

The program office should determine what constitutes complete documentation for its system. At a minimum the documentation set should include the "why, how, what, and where" of the system as built. That is, documents should allow the sustainment organization to understand why the system was designed, how the system was developed, what the system consists of, and where functions were allocated to different subsystems. The overall architecture or blueprint for the system needs to be provided. Plans on how the program office intended to handle COTS and configuration management issues are essential for sustainment and continued implementation. Interface definitions need to be documented. Database designs and their documentation are essential to understanding their purpose within the system. Lastly, the development environment needs to be defined so the sustainment organization knows what tools were used to develop and support the system.

• Authority to Operate (ATO) for an operational software system–Before a system can be considered operational in the field and thus meet the criteria to enter sustainment, an Authority to Operate must be issued. The ATO issuance depends on approval of security requirements by the Designated Approval Authority (DAA). Issuing an ATO means that a DAA has accepted that operation of the system represents a low security risk. An ATO is issued for a fixed period of time (typically three years) and must be renewed. Delay in obtaining ATO approval or renewal could cause the system to be deemed non-operational.

• Current and negotiated Sustainment Transition Plan–In many instances, a program has been developed, tested, and declared operational but there is no funding set aside to address creation and subsequent negotiation of the Sustainment Transition Plan. Unfortunately, in an era where budgets are becoming increasingly tight, sustainment planning is postponed and in some instances forgotten.

Both the development organization and the sustainment organization need to be involved in creating the Sustainment Transition Plan. If a contractor is involved in development, that organization also needs to participate in the development and subsequent negotiation of the Sustainment Transition Plan. In addition, the contract should include tasks that address the contractor's role in the sustainment planning and transition process.

The program office should ensure that while the program is being developed, sustainment tasks are not forgotten or removed from the development contractor's tasking. While the development contractor may not necessarily become the sustainment organization, the development contractor is responsible for developing and maintaining documentation that the sustainment organization will need. It is the program office's responsibility to ensure that the contractor does not create documentation that is proprietary or undeliverable. Even though it was cancelled in 1998, the MIL-STD-498, Section 5.13, "Preparing for Software Transition," contains good background and reference material in this area [5].

• Sustainment staffing and training plan–Staffing the sustainment organization is critical. The staff needs to be trained software professionals that can work with the development organization to transfer the necessary system knowledge. One should not assume that any of the development organization staff will transition to the sustainment organization; rather, adopt a plan to transfer the knowledge from one organization to the other as part of the staffing plan. The staffing and training plan are related to and should be coordinated with the Sustainment Transition Plan.

As with many other areas associated with sustainment, training for the sustainment organization is often treated as an afterthought and is usually an under-funded activity. Even though the sustainment staff is composed of trained software professionals, they still need training on the specifics of the system entering sustainment. This is especially true for the increasingly complex systems that contain a mixture of COTS, government off-the-shelf (GOTS), and organic (government-developed) software code. "On-the-job" training is not sufficient for personnel sustaining these types of complex systems. The system's specific architecture, design decisions, and other nuances need to be communicated in some depth.

Sustainment Challenges

Our research in the 2006 timeframe identified a variety of issues or challenges prevalent with software sustainment at that time. These were grouped into six categories. This is not to say that one would not find other issues that must be addressed when a system is entering sustainment. In addition, no priority is implied by the order in which these topics are discussed.

The following categories of sustainment challenges were identified:

• Sustainment with COTS software–requires consideration of system obsolescence, technology refresh, source code escrow, and vendor license management among related topics.

• Programmatic considerations–discusses issues with relegating the sustainment requirement to the category of "minor requirements."

• System transition to sustainment–considers topics of support database transition, development and software support environment infrastructure (software test lab, hardware spares, release processes and procedures), staffing, operations training, and transition planning

• User support–discusses help desk, user documentation, and user training.

• Information assurance–discusses the unique challenges of IA and COTS software products and testing for IA.

• Development versus sustainment.

While these challenges are most likely still valid, the only one discussed in depth within this article is the last one, development versus sustainment.[2]

Parallel Development and Sustainment–History

As I found in 2006 (and continuing to the present), many systems are fielded in an incremental manner. Incremental

means that an increment or version of the system that provides partial capabilities is developed and fielded. The remaining capabilities are developed later depending on budget, requirements definition, and technology advancements. For the sustainment organization, this means that it will be sustaining a system in parallel with another version of the system that is still under development.

Development in parallel with sustainment is not a new concept; however, many sustainment organizations may not have experience with this mode of operation. In some instances, upgrades (development) are considered a sustainment activity. This makes the "line" between development and sustainment very hazy. To ensure continued operation of the system, the sustainment and development organizations need to develop processes and procedures, coordinate them with all parties, and obtain concurrence on their use. This should include an understanding of who is responsible for any upgrades.

Historically, in organizations that are successful in performing development and sustainment concurrently, groups within the organizations report to the same person. Given the organizational structure of the development and sustainment organizations, this can be problematic. In many instances, the person who has enough experience to oversee both the development and sustainment groups does not have the desire or the time to be involved in this level of oversight.

To better align parallel development and sustainment efforts, the program office needs to consider the current sustainment structure. With that in mind, it should then determine how the system being developed is evolving and how it can fit into the sustainment structure. Sustainment organizations should plan to adapt their processes to handle an evolving system, especially if it implements COTS hardware or software products.

In addition, a joint (development and sustainment) Configuration Control Board (CCB) needs to be created and given the authority to act. All decisions for changes to the baseline must go through the CCB **without exception**. The operational software **must** be driven from the CCB approved baseline. Last, a clear, documented path of escalation up to senior-level personnel **must** be created to address issues. It is not a question of if there will be issues, but when they will occur. Being prepared to handle issues reduces the impact problems have on the overall development and sustainment of the system. Emphasis in bold is added to point out the criticality of following a strict CCB process when there are two organizations (one development focused and one sustainment focused). Otherwise, keeping the two systems in alignment will be problematic at best.

Future of Software Sustainment

In 2006, when I authored the Sustaining Software Intensive Systems technical note, many commercial organizations were starting to use incremental and iterative methods known as Agile methods. These methods have evolved over time; today the commercial environment is using something referred to as DevOps. What is DevOps?

What it is. A way of working that encourages the Development and Operations teams to work together in a highly collaborative way towards the same goal.

What it is not. A way to get developers to take on operational tasks and vice versa [6].

Strangely, I find the definition very similar to what was described in the Parallel Development and Sustainment section. However, there are some major differences. DevOps seems to be the Agile community's term for doing sustainment and operations in parallel. The methods used are based on the Agile Manifesto four tenets and 12 principles but applied in a sustainment environment. Adopting these tenets and principles within DoD requires a major change in the paradigm for doing business [7].

The SEI currently has a team researching the use of Agile methods in sustainment within the federal government. This research is how I came upon the term DevOps. In addition, Gene Kim provided a keynote speech on DevOps at the 2013 Software Technology Conference. The question is whether this type of methodology will be useful and adopted within the federal government. We're still trying to determine this.[3]

However, we have learned that several maintenance organizations within the federal government are trending toward using more Agile-like methods for conducting sustainment. While the "jury is still out" on whether Agile methods are indeed in use, there seems to be a movement to try more incremental and iterative methods using empowered teams. This movement toward incremental and iterative methods does seem to make sense for a sustainment environment where defects and/or enhancements are prioritized and worked on in that order based on the amount of capacity the sustainment team possesses. This approach sounds eerily like the product backlog maintained by an Agile team [8].

In fact, one of the early conclusions by SEI in our Agile work included the following thoughts on using Agile for sustainment:

Operations and Support is where sustainment of the software is conducted. It is assumed that the software previously developed (during the Engineering and Manufacturing Development phase) is mature and stable, so the anticipated software effort expended during this phase is low and should follow a sustainment model, driven by the need to correct errors observed during qualification testing, or providing enhancements as requested by program stakeholders. It is quite possible for a software development team working in these life cycle phases to follow an Agile approach. Quite often the features requested during this phase are modifications that are only relevant within the context of the system that had been previously developed. The aspect of user involvement that naturally occurs at this point of the life cycle makes it easier for the use of a collaborative approach.

It should be noted that some of the Agile methods might not be as practical as others[4] during the Operations and Support phase. For example, it is quite likely that the capability provided during sustainment is planned to be provided over a significant period of time, typically on the order of two years. While the involvement of the user might be beneficial, the frequent releases may not be useful because of limitations with the verification and validation environments required for deployed systems. On the other hand, this constraint should not preclude the use of Agile during this stage of development [8].

In addition, many issues need to be explored including but not limited to documentation required, CCB interaction, release of updated software to the field, quality of code, and cost of code.

Our ongoing Agile and sustainment research is looking at these and other issues. The results of our Agile and sustainment study should be available in early 2014.

Summary

There are multiple issues associated with software sustainment. They start with agreeing on a standard definition for the term software sustainment. This is followed by knowing the criteria for entering sustainment which include stable software production baseline; complete and current software documentation; Authority to Operate; current and negotiated Sustainment Transition Plan; and sustainment staffing and training plan. Finally, specific known challenges need to be considered. These include but are not limited to sustainment with COTS software; programmatic considerations; system transition to sustainment; user support; information assurance; and development versus sustainment.

Parallel development and sustainment have historically been done which may lead to a move towards the more current DevOps approach. DevOps is becoming popularized by the Agile movement. Many issues need to be resolved and the jury is still out on the effectiveness of this approach in the federal government.

Disclaimers:

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

DM-0000536

ABOUT THE AUTHOR

Mary Ann Lapham, a Principal Engineer at the Software Engineering Institute (SEI) of Carnegie Mellon University, is the technical lead for SEI's agile in acquisition research, focused on identifying and addressing barriers to adopting Agile practices in DoD and other government settings. She is also the Space Sector lead within the Software Solutions Division, Client Technical Solutions Directorate. Prior to her coming to the SEI in 2004, Ms. Lapham spent 30 years in technical and program management roles on programs of variable size and complexity. She also is a PMP and CSM.

E-mail: mlapham@sei.cmu.edu
Phone: 412-268-5498

REFERENCES

1. Department of Defense. DoD Instruction Operation of the Defense Acquisition System (DoDI 5000.02). December 2008. Print.
2. Institute of Electrical and Electronics Engineers. IEEE Standard Glossary of Software Engineering Terminology (IEEE Std. 610.12-1990). New York, NY: IEEE, 1990 (ISBN: 155937067X). Print.
3. Lapham, M.A.; Woody, C. Sustaining Software Intensive Systems (CMU/SEI-2006-TN-007). Software Engineering Institute, Carnegie Mellon University. Web. 2006. <http://www.sei.cmu.edu/library/abstracts/reports/06tn007.cfm>
4. "Stable." Merriam-Webster Online Dictionary, 10th Edition. Web. <http://www.merriam-webster.com/dictionary/stable>
5. Department of Defense. MIL-STD-498 Software Development and Documentation. December 1994. (Cancelled June 1998). Print.
6. Swartout, Paul. Continuous Delivery and DevOps: A Quickstart Guide, Continuous Delivery and DevOps Explained. Packt Publishing, 2012. Print.
7. Lapham, M.A.; Miller, S; Adams, L; Brown, N; Hackemack, B; Hammons, C; Levine, L; and Schenker, A. Agile Methods: Selected DoD Management and Acquisition Concerns (CMU/SEI-2011-TN-002). Software Engineering Institute, Carnegie Mellon University. Web. 2011. <http://www.sei.cmu.edu/library/abstracts/reports/11tn002.cfm>
8. Lapham, M.A.; Williams, R.; Hammons, C.; Burton, D.; & Schenker, A. Considerations for Using Agile in DoD Acquisition (CMU/SEI-2010-TN-002). Software Engineering Institute, Carnegie Mellon University. Web. 2010. <http://www.sei.cmu.edu/library/abstracts/reports/10tn002.cfm>

NOTES

1. Information for sustainment is based on the SEI Technical Note Sustaining Software-Intensive Systems, CMU/SEI-2006-TN-007 and updated to reflect the DoDI 5000.02 released in 2008
2. For discussion on the first five challenges see Sustaining Software-Intensive Systems, CMU/SEI-2006-TN-007, http://www.sei.cmu.edu/library/abstracts/reports/06tn007.cfm
3. A future technical note is expected to be published in early 2014 addressing agile and sustainment topics.
4. Kanban/lean style of Agile might be the most relevant for this phase.