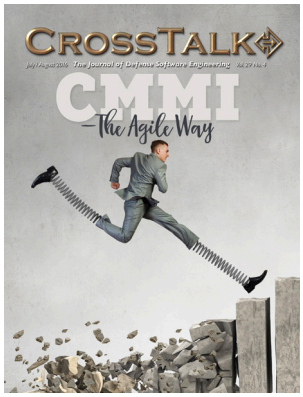
CrossTalk

July / August 2016 The Journal of Defense Software Engineering Vol. 29 No. 4

CMMI —The Agile Way

TABLE OF CONTENTS

Departments



Cover Design by Kent Bingham

CMMI The Agile Way

CrossTalk

NAVAIR Jeff Schwalb
DHS Peter Fonash
309 SMXG Karl Rogers
76 SMXG Mike Jennings

Publisher Justin T. Hill
Article Coordinator Heather Giacalone
Managing Director David Erickson
Technical Program Lead Thayne M. Hill
Managing Editor Brandon Ellis
Associate Editor Colin Kelly
Senior Art Director Kevin Kiernan
Art Director Mary Harper

Phone 801-777-9828
E-mail Crosstalk.Articles@hill.af.mil
CrossTalk Online www.crosstalkonline.org

CrossTalk, The Journal of Defense Software Engineering is co-sponsored by the U.S. Navy (USN); U.S. Air Force (USAF); and the U.S. Department of Homeland Security (DHS). USN co-sponsor: Naval Air Systems Command. USAF co-sponsors: Ogden-ALC 309 SMXG and Tinker-ALC 76 SMXG. DHS co-sponsor: Office of Cybersecurity and Communications in the National Protection and Programs Directorate.

The USAF Software Technology Support Center (STSC) is the publisher of **CrossTalk** providing both editorial oversight and technical review of the journal. **CrossTalk's** mission is to encourage the engineering development of software to improve the reliability, sustainability, and responsiveness of our warfighting capability.

Subscriptions: Visit <www.crosstalkonline.org/subscribe> to receive an e-mail notification when each new issue is published online or to subscribe to an RSS notification feed.

Article Submissions: We welcome articles of interest to the defense software community. Articles must be approved by the **CrossTalk** editorial board prior to publication. Please follow the Author Guidelines, available at <www.crosstalkonline.org/submission-guidelines>. **CrossTalk** does not pay for submissions. Published articles remain the property of the authors and may be submitted to other publications. Security agency releases, clearances, and public affairs office approvals are the sole responsibility of the authors and their organizations.

Reprints: Permission to reprint or post articles must be requested from the author or the copyright holder and coordinated with **CrossTalk.**

Trademarks and Endorsements: **CrossTalk** is an authorized publication for members of the DoD. Contents of **CrossTalk** are not necessarily the official views of, or endorsed by, the U.S. government, the DoD, the co-sponsors, or the STSC. All product names referenced in this issue are trademarks of their companies.

CrossTalk Online Services:
For questions or concerns about crosstalkonline.org web content or functionality contact the **CrossTalk** webmaster at 801-417-3000 or webmaster@luminpublishing.com.

Back Issues Available: Please phone or e-mail us to see if back issues are available free of charge.

CrossTalk is published six times a year by the U.S. Air Force STSC in concert with Lumin Publishing <luminpublishing.com>. ISSN 2160-1577 (print); ISSN 2160-1593 (online)

CrossTalk would like to thank **76 SMXG** for sponsoring this issue.

The Agile umbrella of software thought formally found its way into the discussions of the 76th Software Maintenance Group at Tinker Air Force Base nine years ago. Agile introduced a new level of consternation to our management team with its new terminology and a perception that the foundations of disciplined management practices were being threatened. Like many across the software development community, a second perception developed among the software engineers of our organization that the CMMI and Agile frameworks were at odds. It wasn't long until I begin to hear that "CMMI is too heavy a burden for this project, so we'll use Agile instead," or "Agile is too undisciplined, so we'll use CMMI instead." While both of these criticisms can be true about poor implementations of Agile or CMMI, neither characterizes the ways that CMMI or Agile were intended.

I am very pleased to say that after years of learning from ourselves and the larger software community, that Agile and CMMI are not mutually exclusive and that when used in a complimentary way can result in improved processes and products achieving world class results. Our organization's Agile path started as a grassroots effort when one of our projects discovered the Agile Scrum methodology, and approached management about the possibility of adopting it. Our organization's Engineering Process Group (EPG) invested heavily with our pilot teams and designed a new process set that embraced iterative software builds, increased customer collaboration, and responsiveness to requirements volatility. Organizational projects piloted these processes, while the EPG consulted with them to insure that all of the best practices identified in CMMI were still being performed. As the pilots wrapped up, the EPG took the work of the project and built a new lifecycle model based around Agile Scrum. That model is now available for any project in the organization to adopt and use.

One thing that I can say unequivocally is that CMMI and Agile can be used together successfully. CMMI provides the ideal framework for managing and continuously improving our organization's processes. Agile Scrum provides a new lifecycle development model that yields a better fit for many of our projects over the traditional waterfall model, driving greater customer interaction and employee investment. Both frameworks have provided value to the organization when we embraced them intelligently. When the best practices from both models are thoughtfully applied to the right project, it is possible to do CMMI the Agile Way.

I hope that this issue of CrossTalk continues to inform and assist a software community that continues to adapt to address the ever growing complexity of systems. Our project teams are being asked every day to achieve a greater level of user experience with greater capability in a rapidly changing environment at a lower cost. These types of objectives will not be achieved without a continued directed focus on collaboration on improvement. I would like to encourage all of our community partners to redouble our focus on building relationships focused on learning, sharing, and improving.

Michael Jennings
Group Director, 76 Software Maintenance Group

The Way Forward
A Strategy for Harmonizing Agile and CMMI

Don O'Neill

"The unquestioning acceptance and refusal to envision alternative explanations leads to a festering of inconsistencies that pile up until the tipping point is reached."
-Jeremy Rifkin, "The Zero Marginal Cost Society" [15]

Abstract. Such is the case with Agile and CMMI, two paradigms that appear to be diametrically opposite [1] ... or are they perhaps coexistent or even mutually reinforcing [8]? To what extent is Agile or CMMI at the tipping point and why? On the LinkedIn blogs, the advocates of Agile or CMMI so vigorously reject the alternative explanations of the other camp that sensible discussion of coexistence of the methods seems beyond the tipping point... until now. A new way of thinking put forth by Ivar Jacobson in the Software Engineering Methods and Technology (SEMAT) and the Essence Kernel [4] may supply the key to this puzzle.

The purpose in preparing this paper is to demonstrate in practice the way of thinking of SEMAT and its Essence Kernel and its utility in framing the harmonization issues between two disparate approaches, the Agile method and the CMMI framework.

Underlying the opportunity value proposition, expanding and accelerating the dissemination and adoption of SEMAT and its Essence Kernel will be accomplished by systematically demonstrating its application to the audience and user base of the leading frameworks and methods, thereby, engaging and involving these new audiences in SEMAT and its Essence Kernel, its way of thinking, and its way of working... and yielding new converts. The largest audiences among the leading frameworks and methods are Agile and the CMMI. And so we start there with A Strategy for Harmonizing Agile and CMMI Tensions. The issue is whether Agile and CMMI harmonization is a done deal or whether Agile and CMMI harmonization is a work in progress with the heavy lifting yet to be committed to and undertaken. Let's begin.

Capability Maturity Model Integration (CMMI)

The CMMI is a process maturity framework, and Agile is a software development method. Watts Humphrey viewed software process as the set of tools, methods, and practices used to produce a software product where the quality of the software process largely determines the quality of the software products that result [2]. With Agile, and to Agile's credit, the Agile developer and the customer are in the same silo inwardly and intently focusing on the essential needs of the project at hand no more no less. Everyone and everything else is outside the silo. This is in contrast with the CMMI with its top down, global reach, compliance driven, organizationally focused culture. Properly aligned and harmonized with the CMMI process maturity framework, an Agile implementation might be considered an instance of CMMI implementation. However, much of Agile practice is non-compliant with the CMMI even at its lowest measured level of process maturity, Level 2.

The Capability Maturity Model (CMM) for Software paved the way for software process improvement for software development [13]. The Capability Maturity Model Integration (CMMI) then extended the space to include systems and software engineering process improvement for acquisition, development, and sustainment [14].

With its capitalistic, vertically integrated tilt [15], the Capability Maturity Model Integration (CMMI) has been selected for adoption worldwide by government, military, and commercial organizations as the standard for process improvement. The CMMI is a framework of best practices that focus on assuring product quality through process performance. From the outset, the CMM eschewed practiced-based methods in favor of an overarching framework.

Beginning with the innovators and early adopters [16] of the CMM in the late 1980's and proceeding with the early majority in the 1990's, the CMMI is now left with ferreting out and harvesting the late majority and laggards while its infrastructure of Lead Appraisers, like members of a Guild, struggles to sustain the market and their place in it by continuously refining and tinkering with the mechanics of compliance underlying CMMI appraisals while ignoring the more essential underling practice-based methods of software engineering, software product engineering, and software project management.

Agile

Not so with upstart Agile and its free market, laterally integrated approach [15], as it continues its disruptive intrusion into the space like lava from a volcano. More popular with programmers themselves, Agile offers an actionable method for practitioners not simply a framework for middle managers. More than that, Agile empowers and assigns programmers the decision making for their way of working previously reserved for middle managers under the CMMI regime. Consequently, Agile values freedom of choice and delivers innovation while rejecting and disparaging notions of compliance. In short, Agile connects with programmers who do the work and customers who use the work while CMMI connects with middle managers who oversee the work.

Boosting the CMMI Value Proposition

Still not without value, the CMMI has its advocates, and deserves to have more, despite the abandonment of the U.S. Department of Defense, the original funding source and sponsor of the CMM and CMMI. Without committed, capitalistic sponsors, the CMMI is like a ship at sea with a valuable cargo that will never reach port until its opportunity value proposition is resurrected or renovated.

Such is the CMMI quandary, balanced precariously between the state of being undervalued and the state of yet to be fully valued. Beyond these struggles from the trenches, some are intent on extending the range of value of the CMMI [10] and committed to renovating the CMMI opportunity value proposition with a new way of thinking. It is now time to leap frog beyond a dwindling market, pesky Agile disruptions, and an overly compliant Lead Appraiser infrastructure to the new way of thinking put forth by Ivar Jacobson in the Software Engineering Methods and Technology (SEMAT) and the Essence Kernel [4] as the actionable and practical means to unlock the value of the CMMI in the large and strike the right balance between practice-based method and overarching framework. Few organizations are finding that "their existing governance strategy works well with Agile teams" [10]. Furthermore, the CMMI does "little to help empower development teams to solve the common challenges faced on each day on the job" [7].

The Route to Harmonization

An Agile implementation cannot be CMMI compliant by accident. Instead, an Agile organization needs to make an explicit commitment of intent and resources to CMMI implementation, must exhibit process-based confirmation through people, must demonstrate process execution-based verification, and must demonstrate outcome-based validation through measured results. This is a tall order, and the Agile community falls short beginning with its lack of commitment.

Similarly, the CMMI framework cannot encompass, adopt, or embed Agile by accident. Instead, the CMMI framework needs to make an explicit accommodation to Agile orientation and its inward looking, bottom up, local, need driven, team focused culture and its sensitivity and adversity to management interference. In addition, the Agile community needs to make an explicit accommodation to CMMI and its top down, global reach, compliance driven, organizationally focused culture. All this too is a tall order, and its likelihood depends on leadership so far not present.

Beyond just religious-like zealotry, perhaps there needs to be a litmus test for evaluating Agile and CMMI. For example, which approach deals with the challenges of Cyber Security more effectively and why? Also which approach deals with the challenges of austerity. Rather than simply insist that Cyber Security is paradigm-neutral with respect to Agile and CMMI, perhaps the answer could be framed around the expertise needed to meet the challenge of Cyber Security including Build Security In practices and behaviors, such as, software assurance, trustworthiness, and rigor. How do Agile and CMMI stack up in addressing the challenge of austerity? Agile with its inward looking, bottom up, local, need driven, team focused culture tilts towards austerity. How do they stack up on trustworthiness and software assurance?

- Whether companies or governments, the current economic climate is one of austerity. This austerity and affordability challenge has the effect of tying our hands just when the starter's gun signals the start of the race for the twenty-first century. In accordance with the austerity of the times, the immediate goal of practical Next Generation Software Engineering is to drive systems and software engineering to do more with less... fast using smart and trusted technologies [9]. Clearly austerity is Agile's long suite.
- Software Assurance only has meaning in the context of trustworthiness, that is, worthy of being trusted to fulfill the critical requirements needed for a particular software component, system, or system of systems. Software Assurance demands two capabilities associated with trustworthiness, the capability to produce trustworthy software products and the capability to verify that software products are trustworthy. Each depends on engineering and technology rigorously applied. The kernel of the layered defense approach to Software Assurance is Build Security In and Structured Programming with its rigorous and provably correct use of zero and one predicate prime programs along with proper programs composed of multiple prime programs limited to single entry and single exit.

Framework Versus Method

Prototyped in 1988 and now retired, the original CMM focused on software processes [13]. Introduced in 2000, the CMMI focused on software development and was expanded to include systems engineering, product acquisition, integrated team, and requirements development. The CMMI is now organized into three constellations and has become the basis for assuring an organization's capability to perform software development (CMMI-DEV 2006), acquisition (CMMI-ACQ 2007), and service (CMMI-SVC 2009). The current CMMI is labeled Version 1.3 and was released December 2010 [14].

Due to its origins, the CMMI lacks an explicit correlation to business alignment and strategic planning, sources of essential value to the enterprise. In addition, the CMMI may operate best in a closed system with top-down command and control decision-making [10]. In open organization environments with more diverse bottom-up consensus-based decision-making, other choices may be preferred. With pressure mounting on the value of the CMMI, the benefits of Agile and Iterative Development methods known since the 1970's [6] and the wide spread adoption of Six Sigma, the source and range of value of the CMMI are being questioned and tested. Even Watts Humphrey expressed concern.

Asked about the direction the CMMI was headed, Watts Humphrey conceded that the CMMI had a problem with performance for high maturity organizations and specifically cited the use of process performance baselines and models by Lead Appraisers [3]. He made a careful distinction between procedural (the what) and operational (the how) processes. Whereas, the procedural process depends on a bureaucracy to enforce it, the operational process depends on coaching a self-managing trusted workforce to apply its methods.

In accordance with the need to foster innovation, the bureaucratic top-down appraisal-driven compliance may be giving way to more diverse bottom-up self-directing team empowerment and self-determination. Just as the CMMI focuses on the what in assuring product quality through process performance, Agile deals with how to build software through well-defined methods that place an

emphasis on increasing customer satisfaction. Similarly, Six Sigma further supplies the how with an emphasis on the systematic use of artifact templates, measurement, and control graphics in data-driven decision-making and the reduction of waste.

CMMI Value

Carnegie Mellon University software engineering process (SEP) maturity framework is composed of five levels. The initial stage is characterized by no orderly process and an absence of expectation. The second level has defined processes for managing software cost, schedule and change, all necessary to sustain the commitment process on the project. Level three has defined processes for technology, such as, systematic design, and the technology transition mechanisms that assist its application throughout the organization, such as, software engineering process groups. Level four has initiated process and product measurements. Level five utilizes the process measures systematically to continuously improve the process and its products.

The value of the CMMI can be viewed comprehensively in a systems perspective and is ultimately determined by the increasing value of software to an enterprise and to a mission. This expansive vision of software value must take into account the essential role of systems engineering and its tight coupling with software engineering. In the large, the value of the CMMI lies in its role as an enabler of strategic software management. Strategic software management revolves around knowing what the customer needs most, aligning the best capability to provide it, understanding current practice, measuring its critical aspects, selecting the most promising changes, planning for lasting improvement, raising the ability to improve, and staying the course.

In framing the issue around strategic intent, means, and measured outcomes, the value of the CMMI can be leveraged in terms of strategic software management; and the statements of strategic intent can be cast directly in the context of the business, management, process, engineering, and operations cultural drivers of the organization and its industry sector. The CMMI with its local Software Engineering Process Group (SEPG) and its global dissemination infrastructure of Lead Appraisers promotes an organizational culture, professional environment, and process framework foundation designed to sustain its continued worldwide adoption and foster its expert use. The way of thinking put forth by Ivar Jacobson in the Software Engineering Methods and Technology (SEMAT) and the Essence Kernel [4] provides the means to unlock the hidden value of the CMMI in the large.

SEMAT and the Essence Kernel

The Software Engineering Method and Theory (SEMAT) formulation and its kernel are the essence and common ground of software engineering [4, 5]. This common ground of seven dimensions termed alphas and the sequential states of progression associated with each alpha is that basis. The alphas and the alpha states are intended to be independent of particular methods, practices, and tools and so possess the capability to guide progress and assess status of any software project regardless of method and practice selections. The result is manager-friendly and understandable (Figure 1).

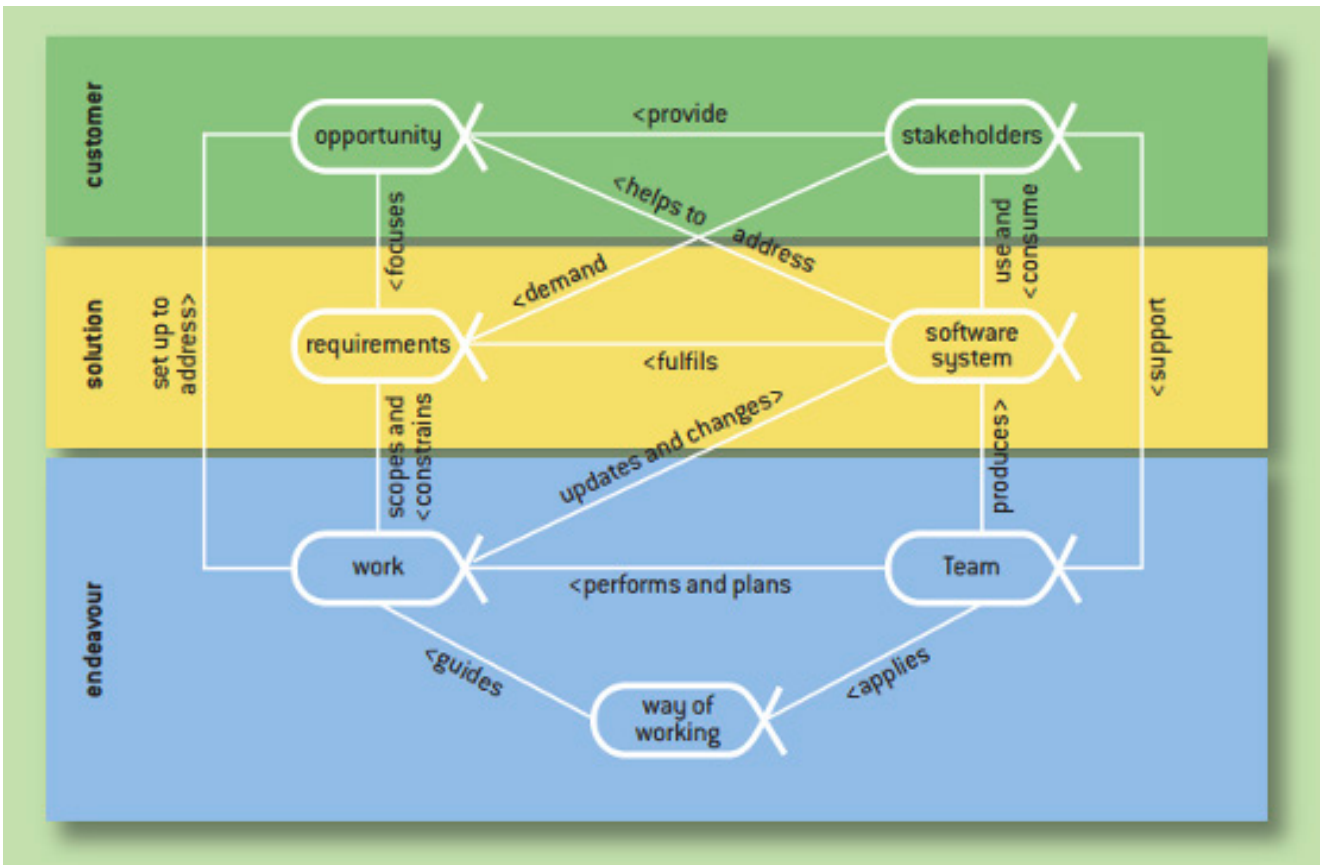


Figure 1. SEMAT Essence Kernel

Alphas	**Kickoff** *Commitment*	**Stage I** *Pointing the Way*	**Stage II** *Digging In*	**Stage III** *Major Milestones*	**Stage IV** *Locking In the Gains*	**Stage V** *Preparing to Stand Down*	**Project Termination** *Retirement*
Stakeholders				In Agreement	Satisfied with Deployment	Satisfied In Use	
Opportunity	Identified		Value Established			Benefit Accrued	
Requirements		Bounded	Coherent	Accepted			
Software System		Architecture Selected					
Team	Selected			Performing [Initial Release]	Performing [Incremental Releases]	Performing [Final]	
Way of Working			Foundation Established			Working Well	
Work				Started	Under Control		Closed

Table 1. Selected Milestone States

1. The customer space is framed by a stakeholder shared vision for a well-conceived value proposition for the opportunity with convincing and consequential outcomes.
2. The solution is bounded by stakeholder agreed to requirements and user stories and a software system architecture that facilitates a usable and operational software product.
3. The endeavor's work is performed by a well selected and ready team and a way of working based on established principles and foundations.

The alpha state checkpoints are the leading indicators that suggest consequential and satisfactory outcomes for these alpha states. A checkpoint on product work has been added to integrate with trustworthy software engineering expectations. For best results, think globally and act locally by adhering to the following expectations which strike a better balance between practice-based method and overarching framework:

1. Stakeholders are in agreement and share a vision for the project.
2. An opportunity value proposition has been established, and there is stakeholder shared vision for achieving it.
3. Requirements or user stories are coherent and acceptable, and there is stakeholder shared vision for them.
4. The software system architecture is selected and comprises a domain specific architecture to guide software system implementation, and the software system implementation is made ready and operational with no technical debt [11].
5. The team operates in collaboration, shares a vision for the project, and is ready to perform with respect to shared vision, software engineering process, software project management, software product engineering, operations support, and domain specific architecture processes, methods, and tools.
6. The way of working by the team has established foundations for software engineering process, software project management, software product engineering, and operations support.
7. The work is started only when all is prepared including coherent requirements and acceptable user stories, stakeholders in agreement, and an established foundation for the way of working.
8. All work products are prepared and inspected in accordance with a defined standard of excellence assuring completeness, correctness, and consistency.

Alphas are Abstract Level Progress Health Attributes. Simple yet powerful, these sensible alphas and their natural states of progression are actually very useful in guiding a project on its way and in guiding a software industry that has lost its way. More specifically, the alphas and the sequence of their state transitions (Table 1) include:

1. Stakeholder- recognized, represented, involved, in agreement, satisfied with deployment, satisfied in use
2. Opportunity- identified, software needed, value established, viable, addressed, benefit accrued
3. Requirements- conceived, bounded, coherent, acceptable, addressed, fulfilled
4. Software System- architecture selected, demonstrable, usable, ready, operational, retired
5. Team- selected, formed, collaborating, performing, adjourned
6. Way of Working- principles established, foundations established, in use, in place, working well, retired
7. Work- initiated, prepared, started, under control, concluded, closed

Certain selected milestone states serve as indicators of project success [12]. When completion of these states is neglected or postponed, the outcome of the project is placed at risk. Table 1 presents the Selected Milestone States critical to success on a project.

CMMI and the Essence Kernel

Since CMMI connects with middle managers who oversee the work, the challenge is to provide a better way of thinking in overseeing the work. SEMAT and its Essence Kernel and alpha states and their alpha checkpoints provide such an improved way of thinking with application to both the CMMI and Agile as well as a wide range of methods in use in the industry. For openers, consider the following alpha state checkpoints:

1. Stakeholders of the CMMI can be found in the ranks of Carnegie Mellon University and its CMMI Institute, the Lead Appraiser community, and the using industry sectors of Telecommunications, Financial Services, Manufacturing, Transportation, Medical, Utilities and Energy, E-Commerce, and Defense and those enterprises seeking consequential outcomes in business, management, process, engineering, and operations. Owing to such diversity, stakeholder agreement and satisfaction may be hard to come by.

- Stakeholders- recognized, represented, involved, in agreement, satisfied with deployment, satisfied in use

2. Opportunity value propositions vary in accordance with type of stakeholder and the forces that drive them, such as, reputation, economics, mission, competitiveness, outsourcing, and high assurance. Consequently, each stakeholder comes with a unique opportunity value proposition without expectation of alignment.

- Opportunity- identified, software needed, value established, viable, addressed, benefit accrued

3. User stories revolve around the strategic intent, means, and consequential outcomes of the diverse stakeholders including the leading indicators of Capability Control, Capacity Control, Change Control, Complexity Control, Defect Free, Innovation, Predictability Control, Quality Control, Release Frequency, Repeatability, Resiliency, Schedule Control, Span of Responsibility, Time to Market, and Traceability. Consequently, user stories may lack alignment in the configuration of consequential outcomes sought and found.

- Requirements- conceived, bounded, coherent, acceptable, addressed, fulfilled

4. The Architecture of the CMMI is framed in terms of the five Process Maturity Levels and the Process Areas that populate each level. It is at the Process Area that stakeholder and user story alignment can be expanded to include the provision for Agile and Cyber Security.

- Software System- architecture selected, demonstrable, usable, ready, operational, retired

5. The CMMI is the Way of Working for its adopters and spans management, process, and engineering. Software project management (SPM) is based on the commitment management paradigm: planning, controlling, and measuring. Planning includes activities and products, tasks and responsibilities, and cost and schedule estimation and earned value management. Software product engineering (SPE) is based on the life cycle activities and the methods and tools used in each activity [6] whether waterfall, incremental or iterative. Operations support (OPS) is based on creating software products that produce the right answers on time every time, using processes that are dependable with respect to cost and schedule, and sustaining the software product and the processes used to create it. The maturity of the domain specific architecture (DSA) is determined by the breadth and depth of recorded experience on the models, methods, and paradigms used in the application domain.

- Way of Working- principles established, foundations established, in use, in place, working well, retired

6. The Team operates in collaboration, shares a vision for the project, and is ready to perform with respect to shared vision, software engineering process, software project management, software product engineering, operations support, and domain specific architecture processes, methods, and tools.

- Team- Lead Appraiser selected, target organization appraisal team selected, appraisal team formed, appraisal team trained, appraisal conducted, appraisal report completed, adjourned

7. Work focuses on obtaining consequential outcomes associated with user stories and way of working beyond simple conformance with process areas in anticipation of a CMMI future appraisal.

- Work- initiated, prepared, started, under control, concluded, closed

8. Work Products focus on perfection and expectations for completeness, correctness, and consistency as shown here:

- The work product is identified as part of the way of working.
- The work product is produced, shared with the team, and inspected.
- The work product is complete and its parts are traceable to predecessor work products.
- The work product is correct and its parts are verified and provably correct.
- The work product is consistent in style and form of recording and with the software system architecture and its rules of construction.
- The work product is value add, traceable to user stories and the "done" criteria for the way of working.

Conclusion

Harmonizing Agile use and CMMI adoption requires stepping away from the points of irreconcilable tension, viewing Agile as a method useful in establishing the foundations for the way of working on a project, viewing the CMMI as a framework useful in garnering senior management commitment to software engineering and management capability of an organization, and adopting the new way of thinking in SEMAT and its Essence Kernel for assessing the state of project progress and choosing next steps.

In this way, what is done on a project can be decoupled from a predetermined pattern of sequence and dependency. Interpreting project progress through alpha state transitions and similarly selecting next steps is left up to the project team and informed only by the contribution to delivering on the opportunity value proposition.

It is useful to think globally and act locally in adhering to the alpha state expectations. This is best illustrated in the resolution of the source of tension between Agile and CMMI associated with premature commitment to requirements in the context of a waterfall life cycle, all of which are thought to be built into the CMMI way of thinking. By viewing this issue and decision as a local action and a project choice determined by the way of working chosen by the project team, the project team can choose to initiate work without requirements, with some requirements, or with all requirements in place before work is initiated.

Nevertheless, in whatever way the project team chooses to address requirements, the alpha states are useful in assessing the current state of completion and in setting expectation for next steps based upon the current state which include the states of conceived, bounded, coherent, acceptable, addressed, and fulfilled. As a result, a trace of alpha state transitions on a project will reveal how close to the edge of chaos the project is and even whether unconditional trust in the project team is warranted and wise.

REFERENCES

1. Ambler, S. and Lines, M. (2012) Disciplined Agile Delivery, IBM Press, 2012
2. Humphrey, W.S. (1989) Managing the Software Process, Addison-Wesley Publishing Company, Inc., 1989, 494 pages, ISBN 0-201-18095-2
3. Humphrey, W.S. (2010) An Interview with Watts S. Humphrey, CrossTalk: The Journal of Defense Software Engineering, Vol. 23 No. 4, September/October 2010, Hill AFB, Salt lake City, Utah
4. Jacobson, I., Pan-Wei Ng, P.E. McMahon, I. Spence, S. Lidman (2013) Essence of Software Engineering: Applying the SEMAT Kernel, Addison-Wesley Professional, January 16, 2013, 352 pages ISBN-10: 0-321-88595-3
5. Jacobson, I., et.al. (2015) SEMAT and the Essence Kernel, appearing in Software Engineering in the Systems Context, Systems Series, Volume 7, College Publications, Kings College, UK
6. Larman, C. (2004) Agile & Iterative Development: A Manager's Guide, Pearson Education, Inc., 2004, ISBN 0-13-111155-8, pages 82-85
7. McMahon, P.E. (2015) Essence: A Thinking Framework to Power Software Team Performance, Software Engineering in the Systems Context, Part I, 2015
8. McMahon, P.E. (2012) Taking an Agile Organization to Higher CMMI Maturity, CrossTalk, The Journal of Defense Software Engineering, January/February 2012 http://www.crosstalkonline.org/storage/issue-archives/2012/201201/201201-McMahon.pdf
9. O'Neill, D. (2009) Preparing the Ground for Next Generation Software Engineering, IEEE Reliability Society, Annual Technology Report 2008, pp. 148-151, June 2009
10. O'Neill, D. (2012) Extending the Value of the CMMI to a New Normal, CrossTalk, The Journal of Defense Software Engineering, January/February 2012 http://www.crosstalkonline.org/storage/issue-archives/2012/201201/201201-ONeill.pdf
11. O'Neill, D. (2013) "Technical Debt in the Code: Cost to Software Planning", Defense AT&L Magazine, March-April 2013 http://www.dau.mil/pubscats/ATL%20Docs/Mar_Apr_2013/O%27Neill.pdf
12. O'Neill, D. (2015) A Constructive Approach to the Effectiveness Analysis of Essence Alpha State Sequencing, appearing in Software Engineering in the Systems Context, Systems Series, Volume 7, College Publications, Kings College, UK
13. Paulk, M.C. (1995) The Capability Maturity Model: Guidelines for Improving the Software Process, Addison-Wesley Publishing Company, 1995, 441 pages, ISBN 0-201-54664-7
14. Phillips, M. and S. Schrum (2010) Process Improvement for All: What to Expect from CMM Version 1.3, CrossTalk: The Journal of Defense Software Engineering, Vol. 23 No. 1, January/February 2010, Hill AFB, Salt lake City, Utah
15. Rifkin, J. (2014) The Zero Marginal Cost Society: The Internet of Things, the Collaborative Commons, and the Eclipse of Capitalism, Palgrave MacMillan, 2014, 356 pages, ISBN: 978-1-137-27846-3
16. Rogers, E.M. (1962) Diffusion of Innovations, 5th Edition. Simon and Schuster. 16 August 2003, ISBN 978-0-7432-5823-4

ABOUT THE AUTHOR

Don O'Neill served as the President of the Center for National Software Studies (CNSS) from 2005 to 2008. Following twenty-seven years with IBM's Federal Systems Division (FSD), he completed a three-year residency at Carnegie Mellon University's Software Engineering Institute (SEI) under IBM's Technical Academic Career Program and has served as an SEI Visiting Scientist. A seasoned software engineering manager, technologist, independent consultant, and expert witness, he has a Bachelor of Science degree in mathematics from Dickinson College in Carlisle, Pennsylvania. His current research is directed at public policy strategies for deploying resiliency in the nation's critical infrastructure; disruptive game changing fixed price contracting tactics to achieve DOD austerity; smart and trusted tactics and practices in Supply Chain Risk Management Assurance; a defined Software Clean Room Method for transforming a proprietary system into a Clean System devoid of proprietary information, copyrighted material, and trade secrets and confirming, verifying, and validating the results; and a constructive approach to sequencing the transition of SEMAT Essence Kernel Alpha states with an eye to pinpointing the risk triggers that threaten success and lead to the accumulation of technical debt.

CMMI the Agile Way in Constrained and Regulated Environments

Paul E. McMahon, PEM Systems

Abstract. Today organizations are questioning if the CMMI is still relevant as they move to more popular agile approaches, such as Scrum. But many of these same organizations are also discovering that agile approaches alone are failing to provide the product quality their customers are demanding especially in constrained and regulated environments. This article employs a case study of an organization that recognized they had gone too far in abandoning their CMMI-based heavy-weight processes in favor of an agile approach. The article describes how the organization rapidly put critical lite-weight practices in place-- complementing their agile approach-- that measurably improved performance in just a few months by using a combination of three frameworks; CMMI, Scrum and Essence.

Background

Is the CMMI still relevant given today's new agile approaches? [1, 2, 3] When the agile manifesto [4] was signed close to fifteen years ago in 2001 many believed agile approaches to software development were just a passing fad that would not be around in 2016. However, today agile approaches are not only being employed by small teams, but are now a focus of improvement efforts in many large enterprises [5]. As further evidence of the staying power of agile one needs to look no further than recent changes in the way the US Department of Defense (DoD) is acquiring new weapon systems.

The DoD acquisition process is governed by Directive 5000.01, The Defense Acquisition System [6], and Instruction 5000.02 which was recently released in January, 2015 [7]. The DoD's Defense Acquisition System is not intended to be a rigid, one-size-fits-all process. As stated in Instruction 5000.02:

"The structure of a DOD acquisition program and the procedures used should be **tailored** *as much as possible to the characteristics of the product being acquired, and to the totality of circumstances associated with the program including operational urgency and risk factors."*

While the word **"tailored"** appears in the instruction, nowhere in this instruction will you see the word **"agile.**" This is because the DoD doesn't want to endorse or dictate any specific method or approach. However, the Defense Agile Acquisition Guidebook [8] states:

"Agile has emerged as the leading industry software development methodology, and has seen growing adoption across the DoD and other federal agencies. Agile practices enable the DoD to achieve reforms directed by Congress and DoD Acquisition Executives. DoD Instruction 5000.02 heavily emphasizes **tailoring** program structures *and acquisition processes to the program characteristics.* **Agile** *development can achieve these objectives through:*

- Focus on small, frequent capability releases
- Valuing working software over comprehensive documentation
- Responding rapidly to changes in operations, technology, and budgets
- Actively involving users throughout development to ensure high operational value"

Why "Being Agile" is Critical to Future Success, but isn't Easily Achieved

While you don't have to be agile to comply with instruction 5000.02, according the DoD's Agile Acquisition Guidebook being agile can make it easier. Nevertheless, if you google "hate agile" you will find hundreds of stories of failed agile efforts [9, 10, 11]. So while there is great motivation to become agile, it isn't as easy as it might sound, and at least part of the reason relates to the word **"tailoring."** While tailoring is encouraged by instruction 5000.02, there is little guidance in how to conduct tailoring, what acceptable tailoring looks like, and what tailoring pitfalls exist.

What Makes Agile Attractive to the DoD?

The world today is changing fast both due to new threats and new technology. The old idea that you could create requirements and hold them constant for years as you developed new weapon systems doesn't hold so well today. This old way of thinking could be referred to as "backward-looking", or always looking back at the fixed requirements as we try to develop new capability that keeps moving further and further away from what we actually need today [12].

What makes agile attractive to the DoD today is that the problems they are facing keep changing and the priorities keep changing. Agile practices are more continually forward-looking which are more conducive to the continual changing world around us. But at the same time while looking forward at continual changes, the challenge we face is how to get the speed of agile while not jeopardizing the assurances of traditional proven engineering practices and frameworks, such as the CMMI. [13] Now let's look at a recent case study that demonstrates how one organization handled this challenge.

Background for the NORO Case Study

NORO is a small organization (less than 100 people) with a DoD contractor heritage that began with an agile-vision in order to respond rapidly to changing customer needs. This vision was in part a reaction of the founders who sought to escape the bureaucratic-slow-moving world of their previous organization's heavyweight CMMI-based processes. However, in 2015 as NORO experienced growth they realized they had gone too far in dropping traditional processes and needed to add back a level of process discipline. I was engaged by NORO to help them in their improvement journey in October 2015.

First Step to Improvement at NORO

As I do with all new clients, we started by conducting a discussion with the leadership team to gain a baseline understanding of what was working well and what wasn't working well

at NORO. My goal in helping any organization improve is to first understand where they are so we don't break what is already working well. The intent is to figure out where the greatest pain is, and rapidly put small improvements in place that can payback quickly. The next step is to keep working to adjust the process in short iterations making further small improvements a normal part of the client's way of working [14, 15].

In NORO's case I observed a pattern I had seen in other organizations that had attempted to "*become agile*" without fully understand what true agility required [16]. This caused the organization to adopt a reactionary interrupt-driven style of work that led to frequent defect-ridden releases and customer dissatisfaction.

As it turned out we were able to put an incremental improvement plan in place by first teaching the team the basics of Scrum [17], along with a number of "extended-Scrum" best practices to address key pain points. The team chose two-week sprints, and in just the first three sprints were able to measurably improve performance. As part of this effort we employed a combination of three frameworks; CMMI [14], Scrum and Essence [18, 19].

Why Use Three Frameworks?

The reason we chose three distinct frameworks is because each has strengths that NORO required. [15]

About the CMMI

The CMMI is a process improvement framework that can help process professionals identify gaps in their organizational processes, but it does not provide sufficient help in "how-to" fill the gaps.

About Scrum

Scrum is a project management framework. Its strength is its simplicity and wide appeal. It is easy to learn enough to get started quickly, but very difficult to master because it lacks "how-to" specifics for many essential practices.

About Essence

Essence is a software engineering framework that is intended to be used by software practitioners on an endeavor to help them assess their current status, risks, and gaps, and help them decide what is most important to focus on next. Essence helps teams discover the "**how to**" specifics they need without dictating "how to" specific practices.

How we used each of these frameworks at NORO is described in the remainder of the article.

Improvement Approach at NORO

In the improvement effort kickoff meeting at NORO we discussed pain points, success criteria, organizational culture, policies, and constraints. During initial discussions it became clear that fundamental work management practices were a priority. After I gave the NORO leadership team a "7-minute Scrum chalk-talk", it was agreed we would start with a two-hour team training session including five basic Scrum practices [17] with some recommended tailoring to address NORO constraints followed by three two-week sprints where the tailored Scrum practices, along with a few Essence-based extended practices would be piloted. The agreed approach was one where the team would learn by using the new practices on a real project. I coached them through the sprints by taking on the role of their Scrum Master while also training an internal Scrum Master and Product Owner. The agreed approach included a planned release at the conclusion of the third sprint.

Tailoring at NORO

I have never seen two organizations implement Scrum the same way. And even when an organization tries to roll-out a common repeatable organizational "agile/Scrum" process, as soon as the individual teams start to implement retrospective improvements they immediately begin to diverge.

In NORO's case we also recognized the need for specific tailoring to the standard Scrum approach to help address pain points identified. Before we discuss the specific tailoring and extended practices we put in place at NORO, let's talk more about tailoring in general.

The Agile Way to Conduct Tailoring

Tailoring is a best practice encouraged by the CMMI framework. However, the way tailoring has been conducted in the past-- especially in many highly regulated/constrained environments-- represents the antithesis of effective agile/lean practices.

This is because it is often conducted in a "tailoring-down" manner which means you start with a long list of products/practices and you identify the ones you don't need. The problem with this approach is first that it runs the risk of someone inexperienced deleting an essential product/practice.

The second problem is that it requires an oftentimes lengthy effort to explain why you don't need to do something. Lean/agile approaches are about eliminating waste, and one of the best ways to eliminate waste is by not requiring someone to explain why they don't need to do something that isn't essential.

A better approach is to start with a small list of **essentials** and then "tailor-up" thus eliminating inefficiencies and risk of tailoring out an essential. So where should you look for such an agreed-to-minimal-essential starting point?

A Common Ground Starting Point for Lean/Agile Tailoring-Up

An effort began in 2010-- referred to as Software Engineering Method and Theory (SEMAT) [20] -- to define such a minimal essential set, or common ground, for all software engineering endeavors. The result of that effort led to an Object Management Group (OMG) standard in 2014 referred to as Essence [18]. The Essence standard meets three important goals that any common ground in a high regulatory environment should have:

Goal 1: Widely agreed upon

Goal 2: Independent of any specific practices

Goal 3: Extensible

What made the development of Essence challenging was that it had to be independent of any specific practices thereby supporting any approach to software development. The framework

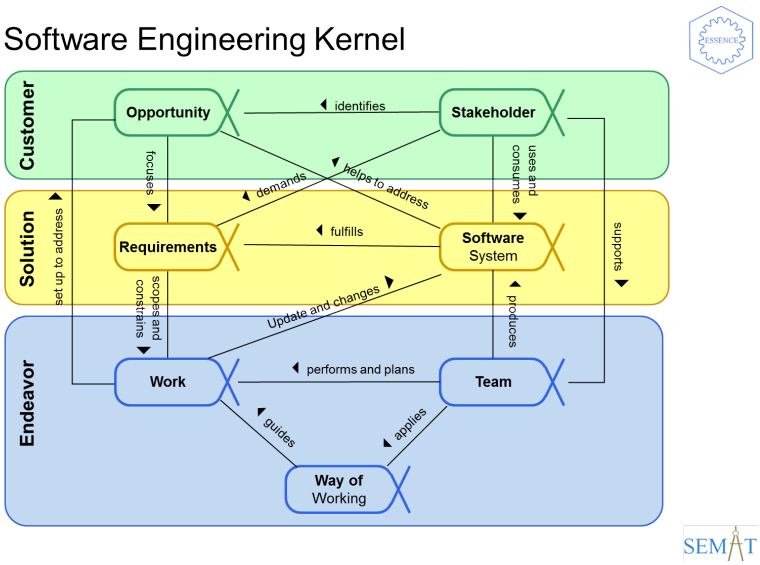


Figure 1–The Essence Kernel

is organized around a kernel containing seven essential things we work with, referred to as alphas, on all software engineering endeavors. Each alpha has a set of states and checklists. See Figure 1. For more information on Essence refer to [19].

What makes Essence particularly attractive to the challenge of high regulatory software endeavors, such as those faced by the DoD, is the fact that it does not limit, or dictate any specific software method.

Rationale for Scrum Tailoring at NORO

While some "purists" argue that you aren't using Scrum if you aren't following the Scrum rules precisely as prescribed by the official Scrum guide [17], the fact is many organizations must live with constraints beyond their control. As an example, at NORO I heard about contracts that required reported bugs to be fixed within 24 hours. Partly because of such customer constraints, NORO had a culture of being reactionary and interrupt-driven. When a key customer reported a problem, everyone dropped whatever they were doing to solve the problem immediately.

Using Essence to Help Team's Discover the "How-To" Specifics They Need

One way Essence can help teams discover the "how-to" specifics they need is by stimulating risk discussions leading to practical mitigation activities. As an example, during a pre-sprint planning session at NORO I conducted an independent risk assessment using an Essence-based risk practice. By "Essence-based risk practice" I mean a practice that has been developed using the Essence kernel. The Essence-based risk practice uses the seven essential alphas, along with their states and checklists to stimulate risk discussions. At NORO this activity led to the identification of the following three high risks and agreed to mitigation activities.

1. Stakeholder Risk

NORO has many customers that use their core product. To address varying customer needs the product is configurable. A common problem NORO faces is changes made to address one customer's reported defects, too often cause unintended negative consequences in the way another customer uses the product.

The Essence Stakeholder alpha state "*In Agreement*" contains the following checklist:

The stakeholder representatives have agreed on their minimal expectations for the next deployment of the new system. See Figure 2.

While it is understood that multiple stakeholders often have competing needs and often reaching complete agreement is not realistic, this checklist highlights the importance of getting your key stakeholders to at least agree on their minimal expectations for the next deployment of the new system.

In NOROs case discussions around this checklist item led to actions accepted by the product owner to conduct meetings with two critical stakeholders most likely to be in disagreement based on past product releases. The goal was to get key representatives from each stakeholder organization to attend an internal sprint review to gain early feedback before the next formal release of the product. This risk discussion uncovered the fact that one critical stakeholder would not be able to attend the sprint review due to a conflict, and therefore a risk mitigation plan was put in place to deliver the early product version on site to this customer to proactively gain their early feedback prior to formal release.

2. Software System Test Risk

When I was contracted to help NORO they were very open in telling me they knew they needed help with their testing approach. They did not have formal written test procedures, although they did have an independent test group, and they did have an independent quality control department that was required to approve each product release before shipment to any customer.

The Useable state of the Software System alpha defined by Essence contains the following checklist item:

Defect levels are acceptable to the stakeholders. See Figure 3

Initially we planned to defer improving test practices at NORO until a later sprint. However, because the reactionary interrupt-driven culture was causing serious pain in the organization, I recommended that we raise the priority and start putting some small test improvements in place immediately on the very first sprint.

Our first improvement to address this pain point was to initiate a three-sprint cycle where every third sprint would be a formal release. Previously NORO did not have a well-defined product release process.

This release process didn't require all customers to necessarily install the new release every third sprint, but it did require the NORO team to work to that possibility. With the three-sprint cycle we instituted a plan where in the third sprint of each cycle only 50% capacity of developers would be used for new functionality, and the other 50% would be used for increased testing-- focusing specifically on regression testing. Previous to this recommendation NORO had no regression test suite.

Figure 2
Stakeholder Alpha State "In Agreement"

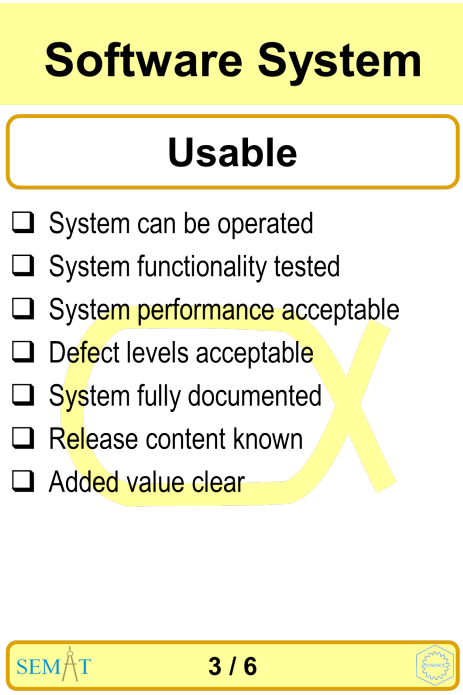


Figure 3
Software System Alpha State "Useable"

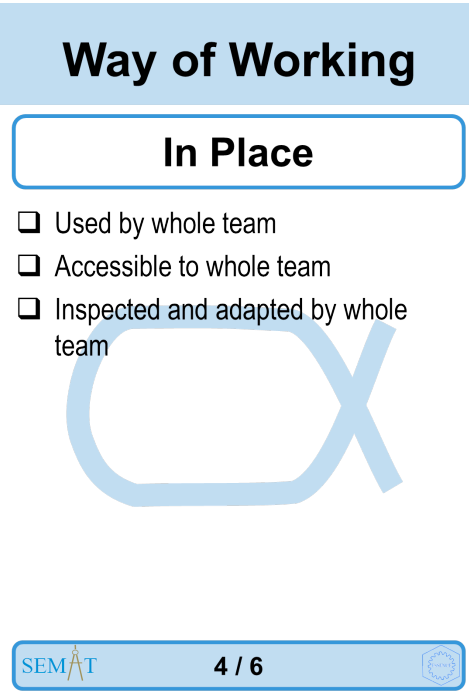


Figure 4
Way of Working Alpha state "In Place"

One of the developers mentioned during a daily standup meeting that "bugs often come back". This led the team to agree that they should capture the tests they run to fix bugs and run these tests again in an automated way rather than keep re-inventing them every sprint and running them manually which had become common practice. We started NORO on the path to improve their testing in the first sprint with a small improvement related to a developer's suggestion. This is an example of how NORO began to continuously improve their practices in small steps.

Agile improvements start small, are continuous and are discovered and implemented by the team

I want to highlight how improvements began to happen at NORO. Most practical and useful improvements start out small and continue incrementally and are discovered and implemented by the team, rather than a separate "process group" as is often found in organizations that implement the CMMI in traditional non-agile ways.

Too often when I go into large organizations that are supposedly "CMMI mature" I find, for example, that they don't have regression test suites. When I challenge them to start improving in this area they say they want to do this, but they don't have the time or budget now-- so nothing happens!

Continuous improvement isn't about increasing cost

The view in too many organizations is that change is always costly. This is a mistaken belief. This is not to say that change isn't hard. But if you change your perspective to recognize that the best way to change is in small steps every sprint[1] then you can start a new culture that says we can get better every day. When you empower your teams to improve their own processes on a regular basis, process improvement becomes a natural part of their normal way of working.

3. Way of Working Risk

During the risk assessment session I asked the leaders at NORO what they were most worried about related to the improvement effort. Another way I often phrase this question is:

"What keeps you up at night?"

The candidate Scrum Master at NORO who I was training replied that his biggest concern was the way we were just training the software team, and that the rest of the organization would continue to operate as they always did. This meant they would be running to the software developers expecting them to drop everything to solve their current problem.

The "*In Place*" state of the Way of Working alpha defined by Essence contains the following checklist item:

The practices and tools are being used by the whole team to perform their work. Refer to Figure 4.

When I heard this concern, I turned to the President of NORO and said,

"We need you to handle this risk. You need to let the whole organization know that the product owner is the one and only person who owns the product backlog, and if they want work done by the team it has to go on the backlog, and be prioritized."

He nodded, accepting the action, and in the very first sprint the resulting improved organizational performance was observed when a developer commented that an Operations Vice-President came to him with what normally would have been an emergency interrupt, and said:

"I know you are in the middle of a sprint, but I wanted to get this high priority issue onto your backlog so it can be addressed as soon as possible."

Key CMMI Process Areas and Generic Practices	Scrum Feature	Essence Feature
Project Planning	Product Backlog, Sprint Planning	Early Alpha state checklists for all 7 Kernel Alphas
Project Monitor and Control	Daily Stand-Up	Work Alpha
Risk Management	Daily Stand-up	Common ground, and Essence-based Risk Practice
Requirements Management, Requirements Development	Product Backlog	Requirements Alpha, Stakeholder Alpha
Verification and Validations	Sprint Review	Software System Alpha checklists
Supplier Agreement Management	Common Scrum base practices	Agreed to common ground
Organizational Process Focus and Definition Generic Practice 3.1 Establish a Defined Process	Common Scrum base practices	Common ground supporting tailoring up approach
Generic Practice 2.7 Ident & Involve Stakeholders	Product Owner role	Stakeholder Alpha
Generic Practice 3.2, Continuous Improvement	Scrum Retrospective	Essence state checklists

Table 1 Scrum and Essence Features Helping the CMMI

Subcontract Management at NORO

NORO, at times, uses a subcontractor that has specialized knowledge about a specific area of their core product. We had decided not to involve any subcontractors in the initial improvement effort so as not to add risk by taking on too much change at once. However, the day after we kicked off the first pilot sprint two subcontractor developers showed up on site to work a number of high priority customer reported defects.

NORO management was trying to figure out how they would monitor the subcontractors work when we realized how easy it would be pull them into the pilot effort. This was because the two developers had been trained in a similar process previously by myself on an improvement effort for a different client.

Often managing subcontractors can present large challenges due to variances in processes across organizations. But because we had an *"agreed-to set of essentials",* the subcontract's work was easily integrated into NORO's pilot improvement effort.

Scrum and Essence Features Helping the CMMI

The CMMI is a process improvement framework that is still relevant today, but it doesn't provide everything an organization needs to continuously improve their way of working. Table 1 shows Scrum and Essence features that can help organizations effectively implement key CMMI Process Areas and Generic Practices [21].

Measurable Improvement at NORO

Survey's conducted with NORO leadership, development team members and NORO stakeholders confirmed measurable improved performance through reduced interrupts, improved reuse of tests, improved tracking of work tasks and increased team velocity and morale in the very first sprint. By the end of the second sprint we were able to validate the survey results with quantifiable data indicating a doubling of the workload being completed, or, in other words, a 100% improvement in team productivity.

Conclusion

The CMMI is a process improvement framework that can help process professionals identify gaps in their organizational processes, but it does not provide sufficient help in "how-to" fill the gaps. The NORO case study demonstrated that Scrum and Essence used together can help organizations implement key CMMI process areas and generic practices in an agile and effective way. Agile approaches, such as Scrum, are not just a passing fad, but they are insufficient to ensure software intensive products exhibit the quality customers are demanding today-- especially in constrained and regulated environments. Essence is a software engineering framework that teams can rapidly start using along with whatever practices or improvement frameworks they are currently using to assess current status, risks, root causes of problems, practice gaps, and put timely actions in place to continuously improve.[2] Essence helps teams discover the "how to" specifics they need without dictating "how to" specific practices. It can help teams transition from a static way of working to a more dynamic way that embraces continual improvement through continual small changes to the way they are working today.[3]

ABOUT THE AUTHOR

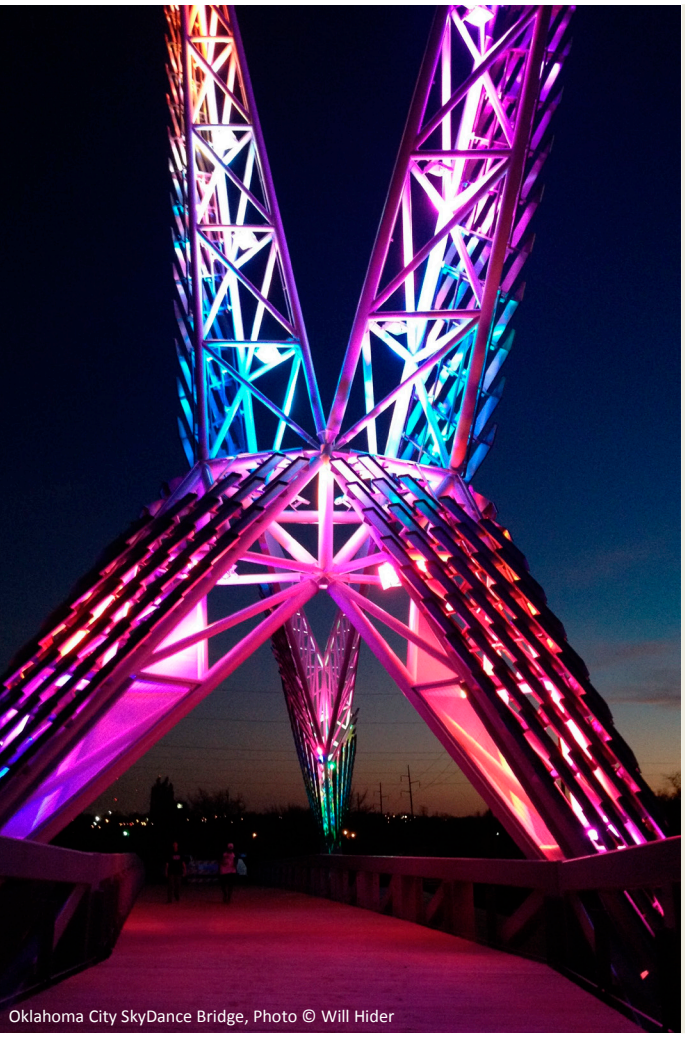
Paul E. McMahon, Principal, PEM Systems (www.pemsystems.com) has been an independent consultant since 1997. He has published more than 45 articles and multiple books including "15 Fundamentals for Higher Performance in Software Development." Paul is a Certified Scrum Master and a Certified Lean Six Sigma Black Belt. His insights reflect 24 years of industry experience, and 17 years of consulting/coaching experience. Paul has been a leader in the SEMAT initiative since 2010.
E-mail: pemcmahon@acm.org

NOTES

1. The focus of improvement when using Scrum is on each individual team. A strength of the CMMI is bringing attention to the need to propagate improvements across the organization.
2. Essence state checklists help teams with continuous improvement by stimulating conversation related to where weaknesses exist leading to team-agreed improvements.
3. I thank Bob Epps, Winifred Menezes, and Barry Myburgh for reviewing and providing improvement suggestions for this paper. The Essence figures are courtesy of SEMAT, Inc.

REFERENCES

1. Is CMMI Still Relevant? http://vip-vatsa.blogspot.com/2012/08/is-cmmi-still-relevant.html
2. CMMI Still Relevant? SEPG 2012. http://www.davidconsultinggroup.com/insights/blog/posts/2012/march/05/cmmi-still-relevant-sepg-2012/
3. Is the CMMI Dead? http://www.cmmifaq.info/#200
4. Agile Manifesto, http://agilemanifesto.org/
5. Disciplined Agile Delivery; Agility at Scale http://www.disciplinedagiledelivery.com/agility-at-scale/.
6. Department of Defense Directive 5000.01, May 12, 2003, http://www.dtic.mil/whs/directives/corres/pdf/500001p.pdf
7. Department of Defense Instruction 5000.02, http://www.acq.osd.mil/fo/docs/500002p.pdf
8. Defense Agile Acquisition Guide, March, 2014, http://www.mitre.org/publications/technical-papers/defense-agile-acquisition-guide-tailoring-dod-it-acquisition-program
9. Why Do Managers Hate Agile?, http://www.forbes.com/sites/stevedenning/2015/01/26/why-do-managers-hate-agile/
10. Seven Things I hate about Agile, http://blog.assembla.com/AssemblaBlog/tabid/12618/bid/87899/Seven-Things-I-Hate-About-Agile.aspx
11. I can't take this agile crap any longer, https://news.ycombinator.com/item?id=5406384
12. Miller, Suzy, Latham, Mary Ann et al, Parallel Worlds: Agile and Waterfall Differences and Similarities (CMU/SEI-2013-TN-021), October, 2013, http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=62901
13. Myburgh, Barry, Effective Governance Enables Success, Chapter 11, Software Engineering in the Systems Context, College Publications, 2015
14. McMahon, Paul E., Integrating CMMI and Agile Development: Case Studies and Proven Techniques for Faster Performance Improvement, Addison-Wesley, 2011
15. McMahon, Paul E., 15 Fundamentals for Higher Performance in Software Development, Appendix D, PEM Systems, 2014
16. McMahon, Paul E., Hybrid-Agile Software Development: Anti-Patterns, Risks, and Recommendations, Crosstalk, The Journal of Defense Software Engineering, July/August 2015
17. Scrum Guide, The Definitive Rules of Scrum, http://www.scrumguides.org/docs/scrumguide/v1/scrum-guide-us.pdf
18. OMG Essence Specification, http://www.omg.org/spec/Essence/Current
19. McMahon, Paul E., "A Thinking Framework to Power Software Development Team Performance, Crosstalk, The Journal of Defense Software Engineering, Jan/Feb 2015
20. www.semat.org
21. CMMI for Development, V1.3, http://www.sei.cmu.edu/reports/10tr033.pdf



Oklahoma City SkyDance Bridge, Photo © Will Hider

How L-3 Leveraged the CMMI for Services to Drive Cultural Change and Business Growth

John Ryskowski, JFR Consulting
Dianne Tolliver, Engineering Solutions & Products, LLC (ESP)
Jeremy Williams, L-3 National Security Solutions (NSS)

Abstract. This is the story of a group that used the CMMI for services (CMMI-SVC) to help create a small and powerful business capture organization that demonstrated impressive results in their first three years (from $87M to $1.5B). They maximized the value of the CMMI-SVC by choosing only those process areas that immediately supported their business needs and vision. Once spun up, this organization proved if one puts the business needs of the organization first, one is able to apply the CMMI with surgical precision that affords pure benefit with no waste, and paves the way for increase revenue.

Introduction

Faced with increasing competition, L-3 National Security Solutions (NSS) established the Multiple-Award Resource Center (MRC) to enable new growth from the win of task orders through L-3's Government-wide Acquisition Contracts (GWACs) and Indefinite Delivery/Indefinite Quantity Contracts (IDIQs) with the US government. The MRC had three goals to achieve:

- Create a positive and enabling business culture
- Quickly streamline operations
- Significantly improve sales through the GWAC & IDIQ contracting vehicles

To reach these goals, the MRC partnered with the L-3 NSS Process & Quality Management team to support the implementation of the CMMI for Services Model (CMMI-SVC) [1] across the MRC. Over eighteen months, the two organizations worked hand-in-hand to innovatively implement the CMMI-SVC to drive positive cultural changes, improve operational efficiencies, and increase sales via the L-3 MRC GWAC/IDIQ contracts.

This paper discusses the MRC's unique implementation of CMMI-SVC to support the achievement of its goals. It addresses how the selection of CMMI-SVC process areas yielded streamlined processes designed to fit the fast-paced business needs of the MRC. It highlights the MRC's approach to social change that garnered staff buy-in and the institutionalization of processes. This paper also includes pertinent data demonstrating impressive business results. Finally, it reviews the lessons learned and salient takeaways from the MRC CMMI-SVC implementation.

Audience

- Senior managers interested in the right amount of infrastructure to stand up a lean organization and increase profit.
- Senior leaders that must quickly implement major change management initiatives
- Any organizational manager desiring a precise application of improvement methodology that is form fitted to his or her business needs.
- CMMI practitioners who wish to entertain a new paradigm for client success and put business needs and profit ahead of "being maturity level 3."

Breaking Free from Maturity Level 3

The idea of a maturity model was motivated by the US government, particularly the Department of Defense. Procurement agencies had little or no visibility into the software development portion of programs that seemed to be 90% finished 90% of the time. The Software Engineering Institute was formed to take on the task of solving this problem. Their solution was predicated on the idea that if one uses a quality process, one will produce a quality product. The key was to identify the correct process requirements, and a constructive methodology to determine their achievement. The Capability Maturity Model (CMM), which was the precursor to the current CMMI (I for Integration), captured the process requirements and was published in 1994. The methodology for determining achievement is an appraisal method, which is done "with" an organization, and differentiates itself from an audit which is done "to" an organization.

The CMM was a "staged" process model, which means that a non-overlapping group of requirements are needed to achieve each of maturity levels 2, 3, 4, and 5 (all organizations start at maturity level 1). An organization's project management sophistication increases as it moves up the maturity levels. This maturity level rating furnished the ability to clearly articulate a very complex process achievement with a single digit number that was easily recognizable and comparable across the industry. The drawback was an early industry assumption that all the process requirements of maturity levels 2 and 3 were a good fit for all organizations.

The initial users of the CMM(I) were military contractors and their subs. In the early 2000's however, commercial users began to outpace the military contractors, a trend that has continued. After years of application to organizations all over the world, it became clear: putting the principles of the CMMI into practice was indeed a benefit to adopting organizations. The CMMI has proven itself to be effective in a myriad of development, services, and acquisition industries.

The CMMI, unlike its precursor the CMM, is not staged and can be applied in what is known as a "continuous" representation. An adopting organization no longer has to take on all the model requirements that comprise maturity levels 2 and 3 to be considered successful. It can take on only those process requirements that best suit its business needs. This approach, though a possibility since 2001, has rarely been applied for fear of acceptance. The notion that echoes throughout the CMMI is that one should never lose sight of the true business needs of the organization. When putting the business needs of the organization first, one is

able to apply the CMMI with surgical precision, a level of greater granularity that affords pure benefit to the organization with no waste. The organization is ultimately more capable.

The CMMI has three "constellations," the CMMI for Development (CMMI-DEV), the CMMI for Acquisition (CMMI-ACQ), and the CMMI for Services (CMMI-SVC). The purpose of the CMMI-SVC is to guide service providers as they improve the way they do their work. The MRC was clearly a service provider to L-3 NSS and therefore chose the CMMI-SVC. The MRC chose the continuous representation which gave it the flexibility to pursue only those Process Areas that best support its business needs. The MRC chose 10 of the possible 24 Process Areas, and they are:

1. Measurement & Analysis - to provide meaningful insight into the MRC's vitality and a critical (objective) view of performance,
2. Process and Produce Quality Assurance- to ensure process adherence,
3. Configuration Management- to supply a beneficial balance between MRC agility and CM rigor,
4. Decision Analysis and Resolution- to utilize a one-page DAR template for prime/subcontractor selection,
5. Service Delivery- to define Service Level Agreements for each of Task Order proposal, Master Contract Management, and Task Order Identification and Development,
6. Incident Resolution and Prevention- to ensure trends and systemic observations are detected, formally tracked, and resolved,
7. Service System Transition- to create a New or Changed Services Implementation Plan (aka Transition Plan) template that assists the planner and ensures verification of the transition,
8. Strategic Service Management- to understand competitive positioning of the MRC and capacity that is carefully managed,
9. Risk Management- to create a risk system that is immediately useful, and
10. Service Continuity- to create simple and common sense solutions that are rigorously practiced to ensure continuity of services.

Project Background

In the summer of 2011, L-3 Communications realized the need to quickly change and adapt to the fast-paced and growing environment of Indefinite Delivery Indefinite Quality (IDIQ) contract vehicles. IDIQs are a key enabler for company growth and an outstanding mechanism for supporting the missions of Federal Government customers. L-3 leadership was interested in creating the right amount of infrastructure to grow a lean organization capable of expanding sales. Key elements of their vision were:

- A high energy growth engine
- A streamlined proposal engine with quick turnaround and high quality deliverables on the proposal side
- An efficient, strong, lean, and compliant program management structure
- Lean but highly effective sales structure

To meet this vision, L-3 established the MRC in the summer of 2011 to centrally manage IDIQs, GWACs and General Services Administration (GSA) Schedules.

After completing its initial detailed planning, the L-3 MRC "went live" in February 2012 providing services supporting IDIQ program management and compliance, task order identification and sales, and quick-turn proposal development. Working alongside various internal L-3 business development organizations from opportunity identification through task order (TO) completion, the L-3 MRC team ensured that the L-3 contract vehicles and associated processes met their customer needs and that contract compliance requirements were achieved.

Defining the Solution

Prior to the establishment of the MRC, L-3 maintained a loosely affiliated group supporting the organization's IDIQs and GWAC vehicles. The original group was understaffed and primarily focused on the basic transactional processing of work through the contract vehicles. There was little to no focus on key customers, new work identification, value added contract administration or task order proposal development. The group lacked focus, structure, cohesiveness, and energy. Concurrent with any process changes required to evolve the organization, a significant cultural change would be required to meet the MRC vision and goals. The unstructured staff needed to become a tightly knit team with the correct subject matter expertise and skills to thrive in the new fast-paced and dynamic MRC environment that was coming.

Due to time pressures and pressing business requirements, the newly appointed Vice President for the MRC, Dianne Tolliver, engaged the services of L-3's NSS Vice President for Business Transformation, Process & Quality Management, Jeremy Williams. Together with their teams, they engaged in an eighteen-month transformation process to evolve the organization. The team started with a goal to drive the business objectives into the process solutions to create a business model that was sustainable, scalable, affordable, and that enabled movement at the speed of business. Moreover, the team wanted to use an accepted industry framework to leverage any existing best practices. After evaluating the available industry models, the CMMI-SVC was selected.

For the MRC to quickly and successfully implement CMMI-SVC in a way that met its business goals, a "common sense approach" to the model was critical. The team selected, used, and leveraged the process areas that supported the critical business requirements and left those that did not "fit" the MRC business case. For example, Service Delivery was selected but Work Planning and Work Monitoring & Control were not. Highest value was placed on creating simple and streamlined processes that incorporated both the spirit of the CMMI-SVC (using the "continuous" representation) and the "on-the-ground practicality" of the business needs.

Implementing the Change

A change management approach, with a thorough and lengthy understanding of the current processes and the graceful transition from it, would simply not work. Time would not permit it. Once "spun up" to the energy level expected, the L-3 NSS executive team and MRC members simply would not, and could not, tolerate a slow change process.

The classic design constraints would apply:

- Feasibility - What could be achieved in the required timeframe (by December 2012).
- Viability - Must match the speed of the business and consider only activities that added immediate value.
- Desirability - People must quickly embrace the new business model/processes and execute.

Key roles that were in place to leverage success:

- VP of MRC (Dianne Tolliver) - "Fire in the belly" leader who came with IDIQ Center and change management expertise. She embraced her role as change agent for the required cultural transformation.
- VP of Process & Quality (Jeremy Williams) - Seasoned process modeler and quality leader, who sees quality and infrastructure as profit-making assets. MRC quality and process strategist.
- Quality Practitioner (Lynn Iffland) - Tactical, boots on the ground quality support.
- MRC Practitioners (Tomma Bersie, Laura Strafer) - Quality's counterpart within the MRC with significant process modeling experience.
- MRC Proposal Center Director (Christian Schoener) - Key staff level champion.
- MRC Team Members - All but one program manager was PMI certified: high level of applied experience across the team.
- Lead Appraiser (John Ryskowski)- Understands business beneficial applications of CMMI.
- Sponsorship - Les Rose, President of L-3 NSS, and Hans Hollister, Sr VP of L-3 NSS.

The posture going in was to be creative, then move hard and fast. Ms. Tolliver challenged the MRC team, significantly raised the bar on performance expectations, and provided tools and infrastructure to achieve the transformation. There would be no process whose only purpose was to meet the CMMI-SVC. New processes would be organically formed and instantly put into practice, with formal definition following close behind. Incremental changes to existing processes would take too long, and were seen as the death of innovation.

The Activities of Change

The team started with naming the required conversations; what would be valued added and what would not. Once this was established, it went well. The conversation determining which CMMI-SVC process areas would serve the business needs only took forty-five minutes. This was due to the QA sponsor putting the business needs first then identifying logical value-added assets from the CMMI-SVC.

The culture had to quickly transform. People were used to managing one GWAC/IDIQ at a time and they now needed to support five to seven contracts, while also focusing on increasing growth. The MRC/QA leadership helped the team envision their potential, and get them excited about the possibilities. A conscious decision was made to move as quickly as possible. The idea was to move so fast that it was impossible to get in front to obstruct progress, to keep moving toward the vision before anything had time to lose momentum. As the solutions were formulated, they were immediately used.

Ms. Tolliver began the transformation process by creating a safe environment. Every member of the MRC team was engaged to leverage their knowledge, ideas, and recommendations to build streamlined, focused, and documented processes. In addition, key internal representatives from every functional area were also brought into the transformation (e.g. contracts, subcontracts, finance, quality, human resources, training). In parallel, Ms. Tolliver worked with Mr. Williams and the Quality Team to clearly understand the CMMI-SVC and existing QA practices, in order to establish processes that would be immediately leveraged. The themes throughout the fast-paced and flexible implementation of the CMMI-SVC were simplified, streamlined, and institutionalized processes that supported the MRC. Complexity was rejected and only those portions of the CMMI-SVC that added value for the MRC business were implemented.

The CMMI-SVC inspired the MRC to have a great solution by supplying topics to address. Together the team would create the solution and put it into operation. Formal process definition and mapping to the CMMI-SVC, sometimes seen as institutionalization, trailed closely behind.

Generous time was spent training the team on the goals of the MRC and the CMMI-SVC model. This was necessary since every member of the MRC was a process creator, critic, and practitioner. Buy-in from all team members at every key step became a "way of life." The team was part of the design process attending weekly creative sessions with all the disciplines, which was part of the battle rhythm.

A technique, similar to Hoshin Kanri "catchball," [2] was used where the "ball" of process design and solution is tossed back and forth from management to team (and back) until a final consensus is reached. A snippet of the process framework design would be passed to the team. The team would write up a solution snippet and send it back. This worked well in an agile sprint fashion. The team could only take on digestible chunks each week. Part of this was due to personnel availability, but it naturally worked well. An example of a snippet would be service continuity (one of the CMMI-SVC process areas). This approach helped the team to achieve a high change velocity of both speed and direction.

The team celebrated successes frequently, and mourned proposal losses as a group. People were proud to be part of the MRC team. Simply said, the team worked together!

It is important to note that the MRC took on the CMMI-SVC while spinning up this new organization. The MRC used the CMMI-SVC model to help develop the organization and all of its processes. Initially this added a layer of difficulty, but was ultimately rewarding. In addition, the team was responsible for accomplishing key performance targets while the transformation was underway.

Results - "We have blown our numbers out of the park since we implemented CMMI-SVC"

When the MRC was chartered it received three specific strategic objectives from the Division President, the Business Unit General Managers, and the Senior VP of Business Development:

- Ensure Master Contract Compliance

- Centralize and leverage L-3's major GWAC/IDIQ contracts to increase awards and revenue
- Develop high quality, quick turn-around task order proposals

In 2011, a baseline metric was established for the business value of the IDIQ assets to see if the MRC investment brought value to L-3 NSS. The MRC went "live" one year later.

Within three years, the MRC partnering with L-3 Business Units achieved the following:

- More than doubled proposal submittals
- Increased awards over seven fold
- Achieved over $1.5B in cumulative awards during the 3-year timeframe (compared to $87M in 2011)
- More than doubled revenue

The team developed and maintained a $30B funnel of task order opportunities. Previous to the MRC's existence, the Business Units were constantly surprised by new business opportunities. Without the ability to write quick-turn proposals, many opportunities were discarded. Now, the Business Units are positioned for key proposals and are able to easily complete quality proposals that are submitted on schedule.

Even though the submittal rate has only increased by 255%, the team is winning more often as demonstrated by the 717% growth. The MRC moved the revenue needle by leveraging their focus on the opportunity funnel: customer relationship management, task order sales, personnel and business unit partnership, and a proposal team that can handle over ten simultaneous proposals a month. The MRC is a "well-oiled machine."

In 2013 the MRC was challenged to broaden its horizons from supporting one business segment to supporting multiple business segments towards a "One L-3" solution. Based upon that initiative, the MRC GSA Schedules' Project Manager significantly expanded the GSA Schedules so that the Divisions had another way to market their services and products to the Federal Government. In addition, the MRC added more IDIQ vehicles to the Center without increasing personnel. Originally the MRC was executing five to six IDIQ contracts, and they are now running over forty IDIQs in a given year. This was due in part to the infrastructure afforded by the CMMI-SVC.

In Conclusion

The MRC took a non-traditional approach to applying the CMMI-SVC. The target was not a maturity level; it was supplying the right amount of infrastructure to support the MRC and significantly increase awards and revenue. Although the MRC reached Capability Level 3 for ten process areas of the CMMI-SVC, success was principally recognized as it supported financial results. Another non-traditional aspect is the use of the CMMI-SVC as a tool to help spin-up a new organization.

What the MRC realized:

- The recipe for success: Cultural Change + Operational Efficiencies= Improved Sales and Morale.
- Accomplish more with less, and the CMMI-SVC allowed the team to do that.
- The solution must be embraced by the entire team and come before institutionalization.

	Submittals	Awards	Revenue
% Growth Increase	255%	717%	240%

Table 1: Growth Compared to 2011

- In the midst of change, a focus on process removes the emotion from the change experience. The CMMI-SVC furnished the MRC with this. It was shiny, different, and fun.
- The CMMI-SVC gave the MRC a clean and crisp framework to hang the processes on.
- Drive the process, but do not let it be the driver.
- The CMMI-SVC gave the team ideas for what processes to pursue.
- Experience with other models is a big help.
- Practice the 3 S's- simple, strategic, and spirit of the CMMI-SVC model.
- Manage chaos with a confident drive fueled by a solid vision, and without attachment to any single solution.

If there was no CMMI-SVC, the MRC would have developed another way to establish process creation and improvement. They would have developed their own framework, but it would have slowed the cultural change and growth process.

Lessons Learned

If you have a significant change management initiative in a short timeframe, you need:

- Senior executive sponsorship and support.
- A strong leader/change agent with a vision and a strong constitution.
- A safe environment that encourages creativity, risk taking, and working together.
- Buy-in from all participants; If people don't "buy-in", they need to "get off the bus."
- A framework that fits nicely, that is very tailor-able.
- An experienced CMMI-SVC guide to help; the model can be confusing at first blush.
- To not stop with your initial success. The MRC now has two off-site sessions per year to identify changes/improvements, and meet monthly to discuss improvement actions. All MRC members and key internal enablers and customers attend the open sessions.
- To always celebrate your successes and remember to have fun along the way!

The MRC identified ten process areas (containing 42% of the total model practices) from the CMMI-SVC that supported their business needs spot on, and brought them to capability level 3. As it turned out, some these process areas were from maturity level 2 and some were from maturity level 3. They used the CMMI-SVC to inspire great solutions and never once uttered the term "maturity level."

Disclaimers

ABOUT THE AUTHORS

John Ryskowski has been helping organizations transform themselves and increase their capability since 1989. He is particularly interested in the social entanglements that preclude progress as organizations work to create their vision, identify actions, and achieve goals. John holds a MA in Education from California Lutheran College, a BA in Mathematics from California State University Northridge, is a Problem Solving Leadership graduate, and a CMMI High Maturity Lead Appraiser for services, development, and acquisition.
E-mail: jfryskowski@yahoo.com
Phone: 310-937-6077

Jeremy Williams is vice president of process and quality management at L-3 National Security Solutions (NSS). He is responsible for improving NSS service quality, innovation/automation for internal corporate business systems, providing consultation to NSS executive management and external customers, managing core staff & budgets, and overseeing ISO/CMMI activities. Williams is PMP certified, a trained lead ISO auditor, is ITIL foundations-certified, and holds MA and BA degrees in geography from the University of Tennessee, Knoxville.
E-mail: jcwilliams@l-3com.com
Phone: 703-434-5074

Dianne Tolliver is now Senior Vice President for Growth and Innovation at Engineering Solutions & Products, LLC (ESP). While at L-3, Ms. Tolliver led the creation of the Multiple-Award Resource Center (MRC) whose Indefinite Delivery/Indefinite Quantity (IDIQ) task-order awards increased to $1.7B from $87M in less than four years. Other capacities for which she has served include CIO and VP of Program Operations. Ms. Tolliver is PMP-certified, holds a BS in Business Administration from California State University, Northridge, and a MS in Public Administration from California State University, Bakersfield. She received her Executive Coaching Certificate from George Mason University, Fairfax, VA.
E-mail: Dianne.Tolliver@espus.com
Phone: 571-499-7787

REFERENCES

1. Forrester, Eileen C., Brandon L. Buteau, and Sandy Shrum. CMMI for Services: Guidelines for Superior Service. Upper Saddle River, NJ: Addison-Wesley, 2010. Print. ® Capability Maturity Model Integration and CMMI are registered in the U.S. Patent & Trademark Office.
2. Akao, Yōji. Hoshin Kanri, Policy Deployment for Successful TQM. Cambridge, MA: Productivity, 1991. Print.

Applying Agile Methods to Software Sustainment

Harold Lowery, Robins Air Force Base
Carl Carhuff, Robins Air Force Base
Phillip Rowan, Robins Air Force Base

Abstract. In this article we report our experiences applying Agile practices and Art of the Possible methods in a software sustainment organization, specifically the F-15 Avionics Integration Support Facility (AISF) at Robins Air Force Base.

Overview

Modern weapons are complex, software intensive systems. In Air Force terminology, the Operational Flight Program (OFP) software embedded in a weapon system allows for corrective, perfective, adaptive, and preventive changes to the system's capabilities. [1] OFP sustainment ensures the embedded software continues to meet evolving requirements throughout its lifecycle. This article describes our experiences introducing some Agile practices along with Art of the Possible methods in the F-15 Avionics Integration Support Facility (AISF) at Robins Air Force Base.

Background

F-15 OFP

Both the F-15C/D air superiority fighter and F-15 Strike Eagle have federated system architectures in which multiple computers execute embedded software (OFP) and communicate via dedicated busses to accomplish the aircraft's mission. The F-15 program office manages the development and release of OFPs as a "suite", i.e. all OFPs are developed in parallel and release to the fleet together. In addition, aircraft capability upgrades and the integration of new weapons are coordinated with the suite cycle since such modifications nearly always require OFP updates. As a result, the OFP suite cycle is a high visibility, tightly managed set of inter-related projects that typically spans 3 – 5 years.

For the past 30 years the sustainment of most F-15 OFPs has been accomplished in the AISF at Robins AFB. Organizationally the AISF comprises a flight of three elements, which the authors have the privilege of supervising.

Avionics Element

The Avionics Element currently supports a large base of Ada, C, and assembly language code for various avionics sub-systems including two armament control computers that manage all weapons on the aircraft.

AISF Support Element

The Support Element performs all actions required to keep the AISF operational for OFP development, including the modification, maintenance, and repair of avionics and radar benches and their associated networks.

Radar Element

The Radar Element sustains the APG-63(v)1, APG-70, and APQ-180 radar systems. The latter is a APG-70 variant that flies on the AC-130U gunship. These systems were designed in the late 1980s to early 1990s, running JOVIAL and assembly language on proprietary hardware.

The Need for Change

The AISF sustains OFPs for six different avionics systems across multiple aircraft configurations with usually three suite cycles overlapping in different phases. Our relatively small workforce of some fifty engineers, computer scientists, and support staff are generally working on a dozen or more projects at any given moment. By the fall of 2014, as budget cuts constrained our ability to bring on additional staff, we began exploring ways to more efficiently manage the dozen or more projects we had in work at any given moment. We needed not only to ensure the most efficient utilization of our people, but also that we did not commit to more work than we could accomplish.

Agile

We found that the spirit of the Agile Manifesto, and the principles behind it [2], resonated with the "can do", mission focused ethos of the AISF. While the requirements of the suite cycle process would not allow us to adopt Agile entirely, we felt we could advantageously adopt certain Agile/Scrum practices. In particular, we liked the emphasis on

1) continuously producing working code within time boxed sprints
2) frequent face to face communications among stakeholders
3) the use of burn-down charts for simple, focused metrics
4) addressing constraints in a timely manner

Art of the Possible

Also in 2014, the Air Force Sustainment Center (AFSC) commander, Lt Gen Bruce Litchfield, challenged the center to improve performance in every area by adopting best practices to achieve "Art of the Possible" (AoP) results. "The AFSC Way is a standard, repeatable tool set for achieving results. Disciplined and enlightened application of the AFSC Way enables the progression of improved performance from meeting expectation to achieving 'Art of the Possible'". [3] The "Art of the Possible" grew out of efforts to streamline aircraft Programmed Depot Maintenance (PDM) through incorporating lessons learned from Lean Production, Theory of Constraints, Six Sigma, etc.

Among the tenets of AoP is the reduction of work in progress through an emphasis on "flow". On the PDM line this means continuously monitoring the progress of aircraft from one gate to the next. Any issue that threatens to reduce flow is identified and addressed as early as possible. Work in progress is strictly managed to prevent bottlenecks that impede flow.

First Steps

F-15 aircraft undergo PDM on a recurring 6-year cycle. Each year dozens of jets fly into Robins for an extensive schedule of inspection and maintenance actions performed by the 561st Aircraft Maintenance Squadron. The 561st has created an AoP "Production System" comprised of stages and gates representing the flow of aircraft through the PDM process. They have made significant progress in improving the flow of aircraft and returning them to service on schedule. As we considered how to integrate AoP concepts into our OFP sustainment process, we wanted to leverage their lessons learned. Late in 2014 we visited their weekly "Walk the Wall" meeting, so named because their entire gated process is mapped out on the conference room walls, showing detailed statistics on flow days, work in progress, etc, along with commentary – roughly 100 pieces of paper showing the status of every aircraft. The production system we saw that day was the squadron's third revision to their implementation of AoP.

That visit sparked serious thought on how we could design and implement an AoP production system that delivered products within the F-15 block cycle constraints, all while incorporating Agile practices as integral building blocks.

Designing the Production System

To begin designing our production system, we held several sessions to map out our existing processes. One of the first questions we wrestled with was "What exactly should flow through our process?" This proved to be a surprisingly difficult question to answer, but after much discussion we settled on "tasks". We defined a task to be something that had a deliverable and that a person could accomplish within a matter of days or at most a few weeks. From an Agile perspective this definition is unremarkable, but it stretched the boundaries of AoP, which had been mostly applied to physical items such as aircraft or their sub-systems.

In accordance with the basic principles of AoP, we decided to carefully track work in progress since "Controlling the amount of WIP into and within the process will ensure the process resources are not overwhelmed within the gates or within the system as a whole." [4]

Since we had an existing requirement to release OFPs to flight test at the beginning of each month, it was an easy decision to time box our sprints to that cycle. We divided each monthly sprint into either 4 or 5 weeks as necessary to stay synchronized with the OFP release dates. With the AoP emphasis on "flow" we recognized the need for frequent feedback and decided to hold our Walk the Wall meetings weekly. Previously, status meetings had been monthly, but we knew quite well that the further into the future we asked our project leads to look, the hazier their crystal balls would become. This scheme of weekly review leading to a monthly release fits well with a key principle of Agile: "Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale." [5]

Tools

As we prepared to roll out our new process to our workforce, one issue that we struggled with was how to plan tasks and track actual performance. For years our parent organization, the 402d Software Maintenance Group (402 SMXG), had used a pair of home grown tools (SITE and EVTAT) in tandem to accomplish project planning, earned value accounting, and time and attendance accounting, however, the group was then in the process of transitioning to major upgrades of both tools. The uncertainty of when the new tool capabilities would be available and how they would be deployed made it difficult to integrate them with our new and evolving production system. Thus we elected to begin with a simple MS Excel spreadsheet to plan and track assigned tasks in a burn down chart.

From Theory to Practice

We launched the new production system and conducted our first sprint in February 2015.

Sprint Planning

In our production system, sprint planning occurs the week before the sprint begins. Project leads work with their team members to determine what tasks are to be worked, then use our standard Excel spreadsheet to create the product backlog. For each task they estimate in which week it will be completed. At the end of each week the project lead updates the burn down chart by simply entering the actual week a task starts and/or finishes.

An important point here is that every task is either not started, in work, or done. We never speak in terms of a task being "X % complete". Our initial design did not account for unplanned tasks, but we soon learned we needed more flexibility. Now when unplanned tasks arise that must be worked in the current sprint, they are entered in the spread sheet as they occur, along with the estimate week of completion. Any tasks not completed in the current sprint are rolled over to the next sprint, along with unplanned tasks that are not urgent.

Walk the Wall (WtW)

The Walk the Wall meeting is held early the first day of the work week. Each team lead briefs 4 charts:

1. Burn down chart
2. What we did last week
3. What will we do this week
4. Constraints / Issues

The burndown chart gives the complete status of the team's work. At a glance all participants can see the actual progress versus plan, the impact of unplanned tasks, and the amount of work in progress.

Figure 1 is a sample burn down chart showing the results of a full 4-week sprint. In this case, the team began the sprint committed to 16 tasks. They fell behind in week one and never quite recovered. Note the four unplanned tasks that occurred in weeks 1, 3, and 4. Work in progress spiked at five in week 3 and declined to the two unfinished tasks at the end of week 4, which will be carried over to the next sprint.

The second chart that is briefed lists the tasks that were accomplished during the previous week, while the third chart charts shows what tasks are to be worked in the current week. There is a strong emphasis on getting tasks done. A

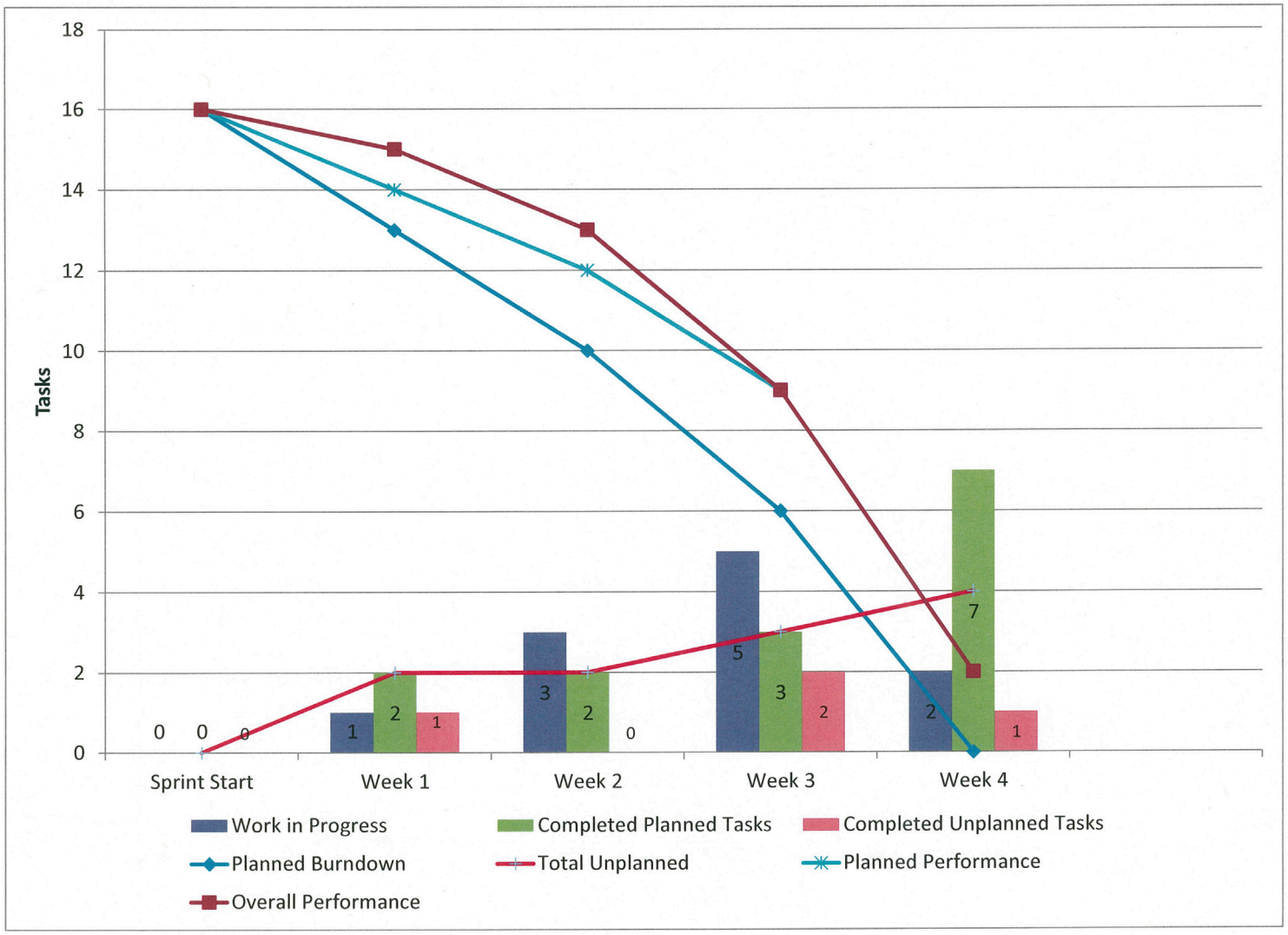


Figure 1: Example Burndown Chart

final chart lists any constraints that threaten to impede flow, as well as other supplementary information specific to the given project under consideration. Timely identification and resolution of constraints is an AoP method to prevent bottlenecks that reduce flow. "Any resource that is not available at the time and location necessary represents a constraint that must be identified, understood, communicated, elevated, and resolved appropriately." [6]

Typically, about a dozen projects are briefed during the meeting, which usually lasts about 45 minutes. Often an issue will be addressed through an immediate decision. If not then the general rule is not to discuss it at length, but rather to assign the appropriate people to work it.

In the early sprints we emulated the aircraft WtW format by posting the charts on the conference room walls and having each team leader "walk" the room through his or her charts. The value of posting the charts on the conference room is that every project's status is immediately available to anyone who walks into the room. However, to facilitate the presentation, we now project them on a large screen during our meetings, in addition to posting slides on the walls.

Results

We have found the chief benefit of a weekly Walk the Wall review is that it allows us to surface and address issues quickly. Communication between individual teams that support each other is enhanced when the leads are in the same room and working off the same metrics.

Each project's briefing supports that communication by a laser focus on the results. The burn down chart provides a clear, visual summary of progress being made, work in progress, and unplanned tasks.

What's Next?

At this point we have developed practices and metrics that allow us, as first line supervisors, to manage projects more effectively than before, with an emphasis on identifying and resolving issues that threaten flow. After reviewing these results, our senior leadership said "Okay, what you've done works at the 'tactical', day by day level. What metrics can you collect that will allow your boss and his boss to manage at a 'strategic' level?"

So our next action is to look for practices and metrics that support reporting to and decision making by higher levels of leadership.

ABOUT THE AUTHORS

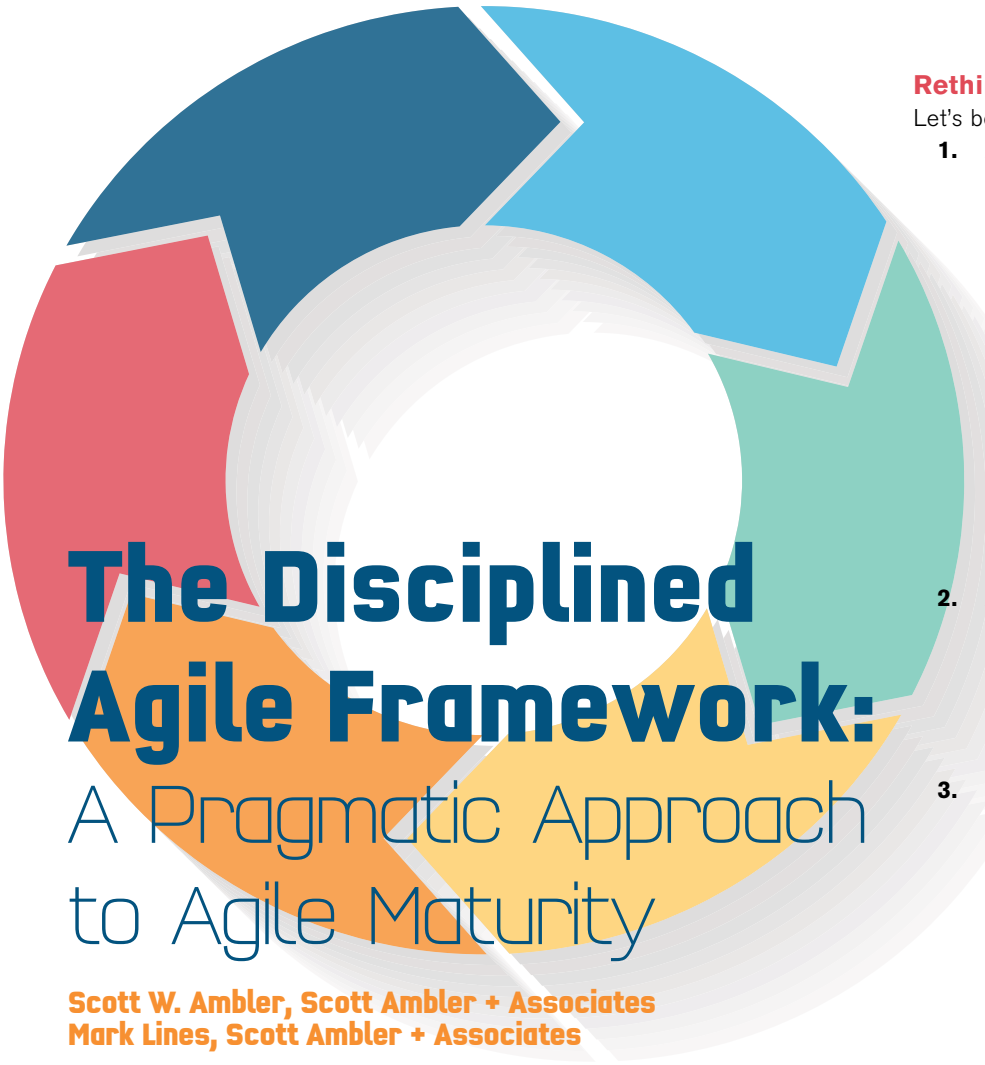
Harold Lowery is the Director of Radar OFP element, 578th Software Maintenance Support Squadron. Most of his 34-year career with the U. S. Air Force has been devoted to the development and support of the F-15 air superiority fighter, with occasional excursions into other DoD weapon systems. He holds a BS and MS from the University of Georgia – Go Dawgs.

Carl "Pnut" Carhuff is the Director of the 578 SMXS, Flight B, Element A, responsible for F-15 Avionics OFP Development and OFP Test and Integration. He is a retired F-15C Pilot and has been in Civil Service for 6 years working in F-15 Software development the entire time as the OFP Suite Lead and now as Element director. He received a BS in Electrical Engineering from the USAF Academy, and an MS in Aviation management from Embry Riddle University. He is a diehard New York Yankees fan and an avid golfer.

Phillip Rowan is the Director of the Avionics Integration Support Facility, 578th Software Maintenance Squadron. During his 11-year career with the U.S Air Force he has supported Test Program Set development, continuous process improvement initiatives, and software support for the F-15. He holds a MS from Mercer University, however he considers himself a Tennessee Volunteer from his undergraduate work obtaining a BS in Computer Engineering.

REFERENCES

1. Guidelines for Successful Acquisition and Management of Software-Intensive Systems: Weapon Systems, Command and Control Systems, Management Information Systems - Condensed Version, Chapter 16, 4.0 February 2003
2. http://www.agilemanifesto.org/principles.html
3. Art of the Possible (AFD-140911-029), Air Force Sustainment Center, 2014
4. Ibid, page 48
5. Beck, Kent; et al. (2001). "Principles behind the Agile Manifesto". Agile Alliance
6. Art of the Possible (AFD-140911-029), Air Force Sustainment Center, 2014, page 49



The Disciplined Agile Framework: A Pragmatic Approach to Agile Maturity

Scott W. Ambler, Scott Ambler + Associates
Mark Lines, Scott Ambler + Associates

Abstract. It is possible to combine Agile and Capability Maturity Model Integrated (CMMI), but few organizations are doing this in practice. There are many challenges to be overcome, not the least of which is the very different mindsets of adherents of each approach. More importantly, we have a better option available to us in the form of the Disciplined Agile (DA) Framework. The DA process decision framework provides light-weight guidance to help organizations streamline their software processes in a context-sensitive manner. It does this by showing how various capabilities such as solution delivery, operations, enterprise architecture, portfolio management, and many others work together in a cohesive whole. The framework also provides a range of options for addressing these capabilities, describing the tradeoffs associated with each option. The DA Framework provides organizations with an easier and better described path to agile capability maturity than the strategy of force-fitting Agile and CMMI together.

The Capability Maturity Model Integrated (CMMI) model purports to provide guidance to help organizations improve their software processes. This has in fact happened in some organizations, albeit in manners that are not as streamlined as many people would like. Agile strategies – such as agile development techniques, agile architecture techniques, agile management techniques, agile modeling techniques, and more – offer the potential to streamline and improve your software processes. This has occurred in practice in a multitude of environments. Agile has become the standard approach in many organizations and continues to grow in the majority of companies around the world. It is possible to combine Agile strategies and CMMI together: CMMI describes what should be done and Agile describes how it should be done, so combining them should be a good idea. That's the theory – Practice shows there is a better way.

Rethinking Agile Capability Maturity

Let's begin with several important observations:

1. **Agile and CMMI promote different paths to capability**. In CMMI [1], capability is defined as "the inherent ability of a process to produce the planned results. As the capability of a process improves, the process becomes predictable and measurable, the quality of the result product augments and the productivity of the teams too." The Agile Manifesto [2], on the other hand, is very clear that the path to Agile capability is to build teams of motivated people and provide them with an environment where they can succeed. These teams should reflect on a regular basis so that they can identify, and then adopt, potential improvements to the way that they work. Agile teams own their own process and evolve it over time.
2. **Agile seems to help CMMI**. Tadros [3] found that there were fewer gaps found by SCAMPI reviews in organizations that had taken an agile approach compared with non-agile. The study also found that quality metrics were noticeably better in the organizations combining agile with CMMI strategies.
3. **CMMI is attractive to government organizations and those that serve them**. CMMI has "succeeded" in US government agencies where competition is non-existent and in companies focused on working for government organizations where questionable procurement practices virtually assure an onerous, documentation-heavy process. This is a harsh observation, but CMMI tends to thrive when productivity isn't of paramount importance.
4. **CMMI enables offshoring**. CMMI is common with Indian service providers (ISPs) who use CMMI as a marketing strategy to lure customers, as well as Chinese service providers hoping to compete against ISPs. CMMI is also adopted by groups within private-sector companies that hope CMMI will help them to manage the work that they offshore to CMMI-compliant service providers.
5. **Few agilists are interested in CMMI**. Although McMahon [4] argues that CMMI can be applied to enhance agile techniques the reality is that agilists consider CMMI to be an anathema. Without a coach of McMahon's caliber it is highly unlikely that you'd succeed at inflicting CMMI on an existing agile team – most likely your agile developers would find ways around that mandate or choose to seek employment elsewhere.
6. **Leading-edge software engineering organizations rarely consider CMMI**. Organizations in highly competitive environments – Apple, Google, and Etsy to name a few – would laugh you out of the room were you to suggest that they adopt CMMI. Time to market and quality are paramount for these organizations, so that forces them to be incredibly effective at software development. If CMMI truly offered the potential for them to be more effective they would adopt it as swiftly as possible, yet that isn't happening.

7. **CMMI is shrinking in the private sector**. Most large financial institutions, retailers, telecommunications companies, and automotive companies have tried to adopt CMMI at some point. Yet it is rare to find one that applies CMMI-based strategies on a regular basis, and if they're still doing CMMI at all it is on a handful of legacy teams.

The real question isn't whether we can apply CMMI and Agile together, but instead does CMMI realistically bring anything of value to the table for Agile teams? Although McMahon [4] makes very good arguments for how Agile can help CMMI, his arguments for the reverse seem a bit stretched at times. Yes, CMMI does provide guidance for what software organizations need to do, but our experience is that CMMI guidance is both incomplete and often inflexible in practice. Organizations that are serious about improving their agile capability can do a whole lot better than trying to force fit Agile and CMMI together.

Disciplined Agile to the Rescue

The Disciplined Agile (DA) process decision framework is a people-first, learning-oriented hybrid agile approach to IT solution delivery [5]. It has a risk-value delivery lifecycle, is goal-driven, is enterprise aware, and is scalable. DA is a hybrid approach which extends Scrum with proven strategies from Agile Modeling (AM), Extreme Programming (XP), Unified Process (UP), Kanban, Lean Software Development, Scaled Agile Framework (SAFe), and many other methods. The framework provides contextual advice showing how agile strategies work together in a light-weight, streamlined manner from beginning to end. It describes the tradeoffs associated with hundreds of agile strategies, enabling you to tailor your software process to effectively address the situation in which you find yourself. By describing what works, what doesn't work, and more importantly why, DA helps agile teams increase their chance of adopting strategies that will work for them in the situation that they face. In short, the DA framework does the heavy lifting when it comes to showing how Agile works in practice.

Earlier we described the challenges associated with combining Agile and CMMI in practice. Now let's explore strategies for how the DA framework can be applied to help organizations move towards greater maturity in their Agile capability.

Strategy #1: Adopt a flexible, context sensitive approach.

The DA framework originally came about from empirical observations of dozens of teams apply agile and lean strategies in dozens of organizations working in different domains around the world. Although there were similarities between how these teams worked every team worked in a unique manner, and every team had spent considerable time and effort determining how to do so. And every team still had improvements to make, and many were struggling with doing so because they didn't have the process background to identify candidate options. It was clear to us that agile/lean teams could benefit from light-

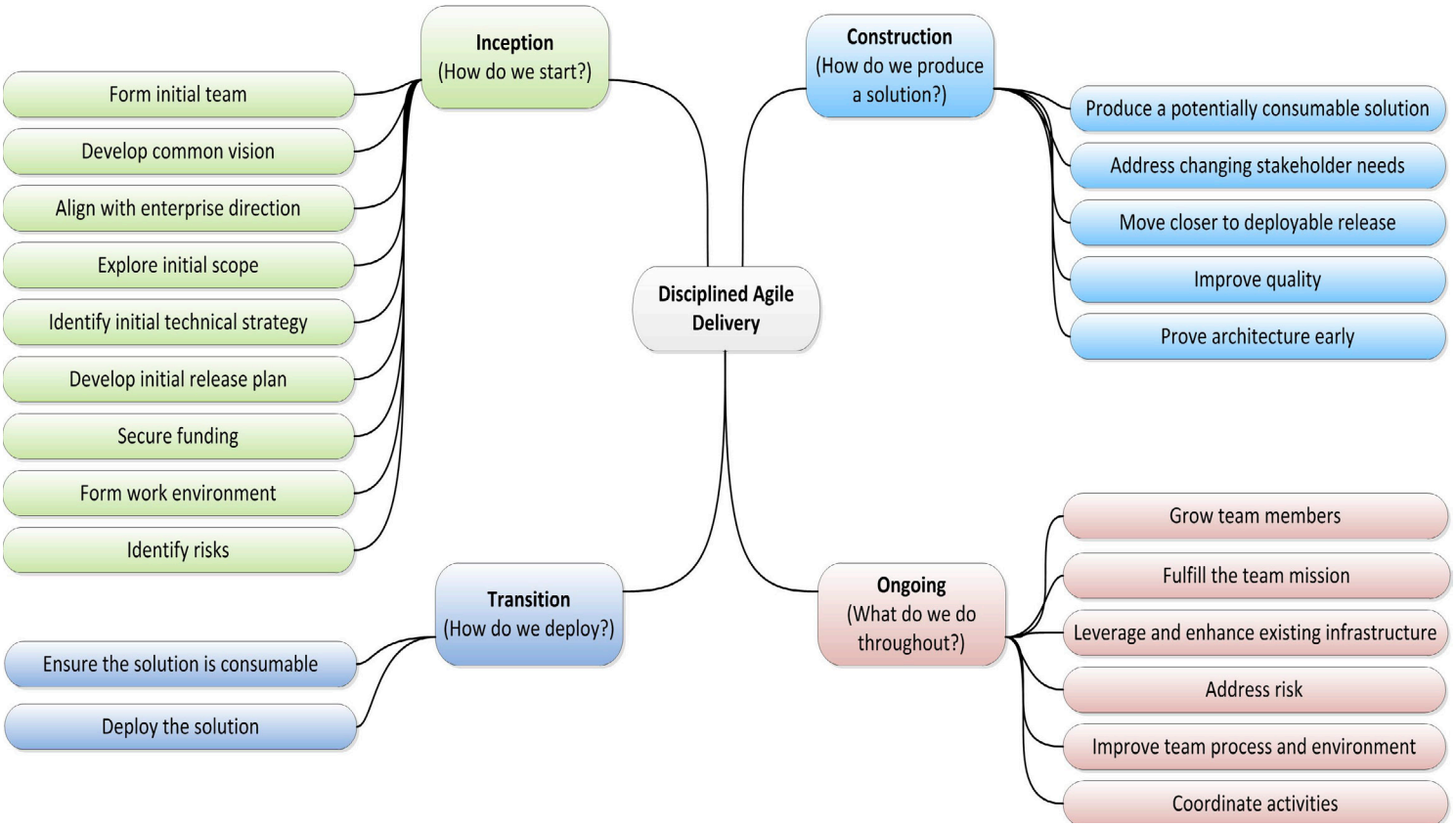


Figure. 1. Process goals (capabilities) for agile delivery teams.

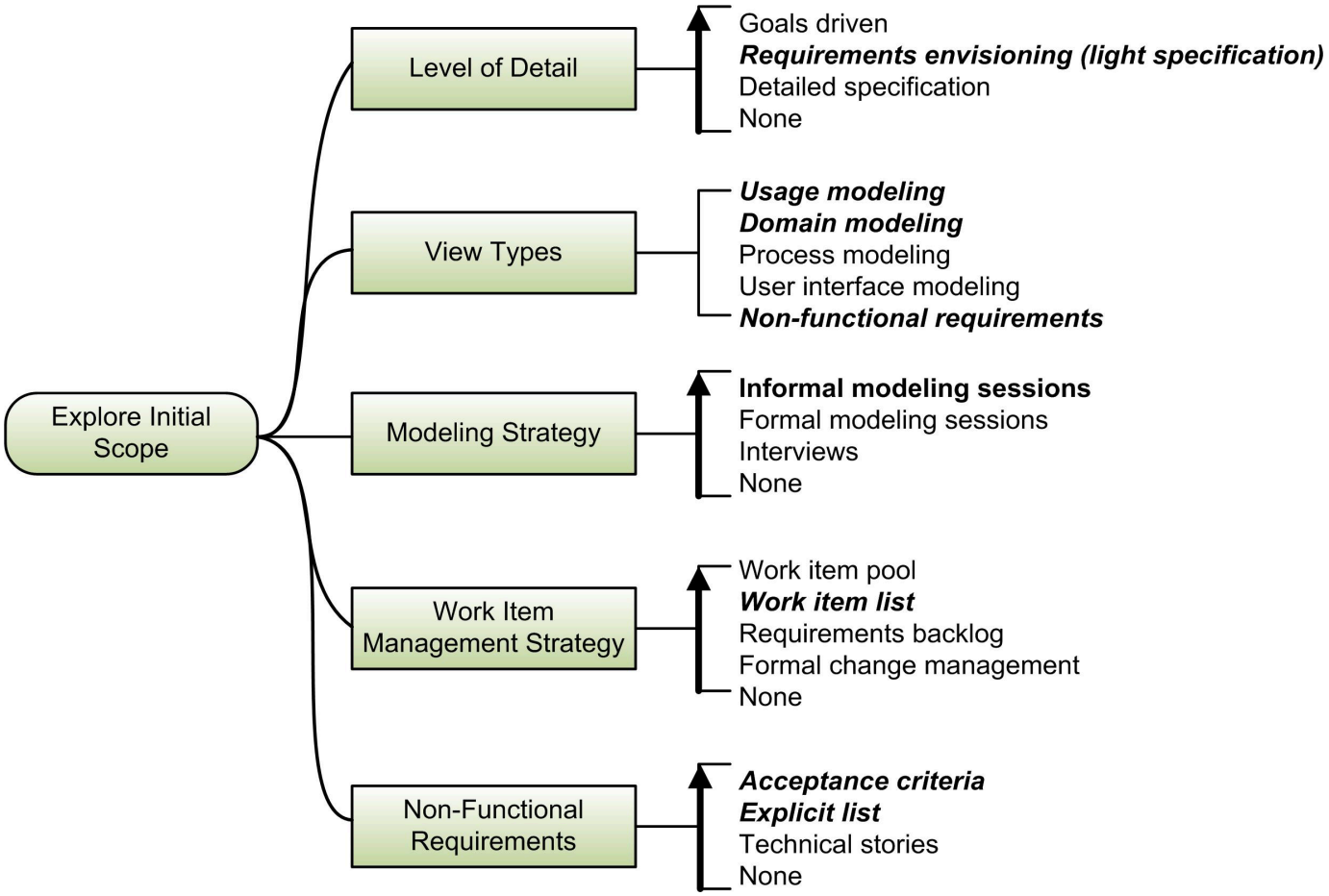


Figure. 2. The process goal diagram for Explore Initial Scope.

weight process guidance that described the range of options available to them.

To do this DA adopted a goal-driven, or capability-based, approach. In many ways these process goals are analogous to CMMI Process Areas (PAs), albeit with a time-oriented focus rather than a subject-oriented focus. The delivery goals are summarized in Figure 1. The diagram indicates that when a team is in the Inception phase (sometimes called Sprint 0 or Initiation) that they must address goals such as forming the initial team, aligning with the enterprise direction (e.g. follow your corporate roadmaps and guidelines), and explore the initial scope amongst other things. There are also process goals for the Construction and Transition phases and ongoing goals that apply throughout the entire lifecycle. Every team addresses these goals in some way, but does so in a different manner and will evolve their approach over time as they learn from experience.

The DA framework makes your process choices explicit and provides guidance for selecting the options most appropriate for your team. This guidance helps teams to get going in the right direction and provides options when they realize

that they need to improve. Because we observed that teams find themselves in a range of situations the framework supports multiple lifecycles [6]– a Scrum-based lifecycle, a lean lifecycle, a continuous delivery lifecycle, and an exploratory ("Lean Startup") lifecycle.

Strategy #2: Marry the what with the how

To provide teams detailed process guidance the DA framework introduced the concept of process goal diagrams [7]. A process goal diagram depicts the process factors that should be considered when addressing the goal and then a representative list of options for those goals. Because people around the world are constantly improving upon and identifying new practices and strategies the list of options presented in the goal diagrams cannot possibly be definitive. Instead they represent the range of options available, letting people know that choices exist (regardless of what prescriptive methodologies like Scrum imply) and that sometimes some choices are distinctly better than others (again, regardless of what prescriptive methodologies imply).

Figure 2 depicts the goal diagram for *Explore Initial Scope*, a goal that you should address at the beginning of a project during the Inception phase. Where some agile methods will simply advise you to populate your product backlog with some initial user stories, the goal diagram makes it clear that you might want to be a bit more sophisticated in your approach. What level of detail should you capture, if any (a light specification approach of writing up some index cards and a few whiteboard sketches is just one option you should consider)? What view types should you consider (user stories are one approach to usage modeling, but shouldn't you consider other views to explore the data or the UI)? Suggested starting points, are shown in bold italics. The framework makes it clear that agile teams do more than just implement new requirements, hence our recommendation to default to a work item list over Scrum's simplistic Requirements Backlog strategy. Work items may include new requirements to be implemented, defects to be fixed, training workshops, reviews of other teams' work, and so on. Finally, the goal diagram makes it clear that when you're exploring the initial scope of your effort that you should capture non-functional requirements – such as reliability, availability, and security requirements (among many) – in some manner.

There are several fundamental advantages to taking a goal-driven approach to agile solution delivery:

1. **Process decisions and choices are explicit**. This approach makes your process options very clear and thereby makes it easier to tailor your process for the situation you find yourself in.
2. **Tactical agility at scale is enabled**. A goal-driven approach enables effective scaling by guiding you through process tailoring to reflect the realities of the scaling factors that you face. These scaling factors include team size, geographic distribution, regulatory compliance, tech-

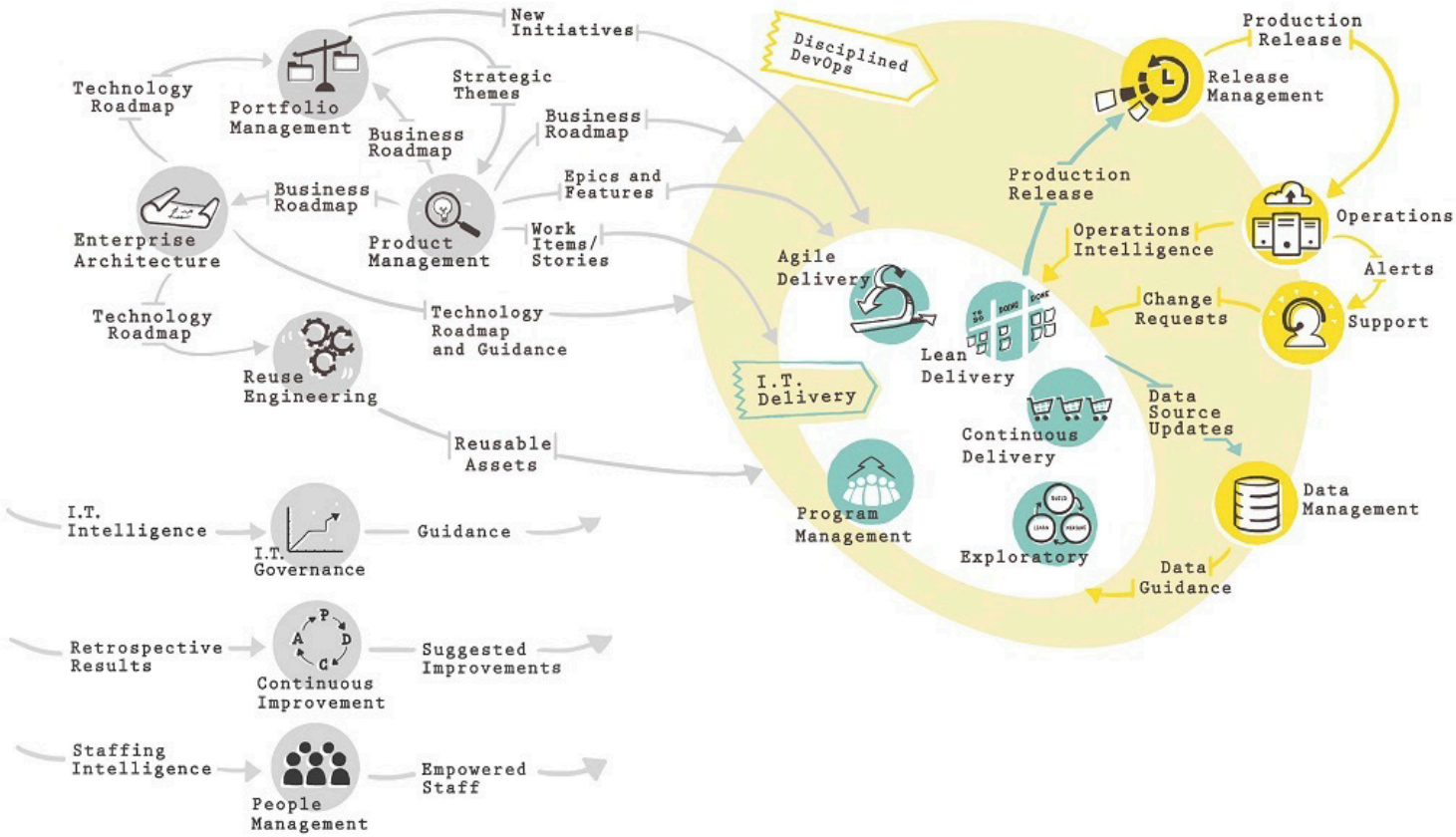


Figure 3. Workflow of a Disciplined Agile IT department.

nical complexity, domain complexity, and organizational distribution [8].

3. **Agile capability maturation is actionable**. Disciplined Agile takes the guesswork out of applying agile strategies and thereby enables agile to focus on their actual job, which is to provide value to their stakeholders. In the case of some process factors a clear improvement or maturation path is indicated. Many of the process factors have ordered lists of options, as indicated by the arrows. The strategies towards the top of the list are generally more effective than the strategies towards the bottom. Disciplined Agile still supports all of the strategies, but recommends that agile teams strive to adopt the most effective strategy that they can. For example, in Figure 2 consider the *Work Item Management Strategy* process factor of the *Explore Initial Scope* process goal (which implements one aspect of CMMI's *Requirements Management (RM)* process area). Perhaps the team is new to agile and they've only received Scrum training to date. This team will likely adopt Scrum's prescribed *Requirements Backlog* approach, which is far better than the *Formal Change Management* strategy of yesteryear. During retrospectives the team may identify the need to improve their work item management strategy, or perhaps their Certified Disciplined Agile Coach (CDAC) will suggest they improve, which then motivates them to revisit this process goal (or more likely the *Address Changing Stakeholder Needs* process goal which also includes this process factor) to identify a better option. The better option might be to make a minor improvement and adopt a *Work Item Backlog* strategy (from the Unified Process) or a major improvement and adopt a *Work Item Pool* approach (from Kanban).
4. **Process risk is explicit**. A goal-driven approach makes it clear what risks you're taking on and thus enables you to increase the likelihood of success. Each of the strategies, such as *Requirements Backlog* or *Work Item Pool*, has advantages and disadvantages identified for them. There is no such thing as a best practice, instead every practice is contextual in nature – in some situations a practice works incredibly well, in other situations it makes things even worse.

History of Disciplined Agile

To date there have been three major release tiers of the Disciplined Agile framework:

1. Disciplined Agile Delivery 0.x.

The framework was originally developed at IBM Rational from early 2009 to June 2012. The IBM team worked closely with business partners, including Mark Lines, and was led by Scott Ambler. IBM Rational Method Composer (RMC) currently supports an early, 0.5 version of the DA framework.

2. Disciplined Agile Delivery 1.x.

The DA 1.0 release occurred in June 2012 with publication of the first DA book, Disciplined Agile Delivery [2]. Evolution and publication of the DA framework continued at the Disciplined Agile site starting in August 2012. Ownership of the DA framework intellectual property effectively passed over to the Disciplined Agile Consortium [3] in October 2012, a fact that was legally recognized by IBM in June 2014. The focus was on the software delivery process.

3. Disciplined Agile 2.x.

This is the current version of the framework, initially released in August 2015. The focus is on describing a flexible, context-sensitive approach to the entire IT process.

If your agile teams are to be capable then they need to know what process factors they need to address, what their options are to do so, and what the tradeoffs are of these options.

Strategy #3: Show how to apply proven strategies together

Many teams new to agile will adopt a method like Scrum or SAFe as if it's a recipe, ignoring advice from other sources and thereby getting into trouble. This is exacerbated when you weave in CMMI's process area approach into mainstream agile strategies. Many SEPGs will organize their improvement efforts around the process areas, which has the effect of local optimization of each process area at the expense of the overall flow and efficiency within your process. The fact that CMMI-Level 5, which few organizations achieve in practice, focuses on process optimization belies this point.

The DA framework adopts practices and strategies from a range of existing sources and provides advice for when and how to apply them together. The framework effectively weaves the CMMI process areas together. For example, the *Produce a Potentially Consumable Solution* process goal includes process factors such as *Needs Exploration*, *Solution Exploration*, and *Planning* which map to aspects of *Requirements Development (RD)*, *Technical Solution (TS)*, and *Integrated Project Management (IPM)* CMMI process areas respectively. Via the combination of lifecycles and process goals the DA framework is able to present time-oriented advice, e.g. at this point in time here's what you need to do and here's how it fits together, that is actionable by agile teams. In short, the framework takes the mystery out of how all this agile stuff actually works in practice.

Strategy #4: Optimize the whole

A Disciplined Agile IT department is a flexible learning organization that is responsive to the needs of the organization(s) that it supports in a financially effective manner. DA 2.x extends disciplined agile delivery strategies to the entire IT workflow [9]. We began the development of DA 2.x in the Autumn of 2014 and the work continues to this day. DA 2.x is based on several important observations. First, every organization is unique, and every IT department within each organization is also unique. Second, IT departments are dynamic complex adaptive systems that evolve over time. Third, the components of IT departments – teams and sub-departments – also evolve over time. Fourth, these components, when left to their own devices, are often not well aligned with each other. Worse yet, these groups work under their own locally optimized "improvement strategies." This misalignment is caused by competing leadership visions (or less delicately, by "politics") and exacerbated by disparate bodies of knowledge (BoKs) within our industry: The Agile Manifesto; The Project Management Institute's BoK; The Data Management BoK; The Business Analysis BoK; The Open Group Architecture Framework (TOGAF); The Information Technology Infrastructure Library (ITIL); and many more. Although all of these bodies of work provide valuable insight, they each provide their own locally optimized view of how things should work. These views overlap, provide inconsistent advice, and are often focused on a single specialty. For example, the BABoK provides a business analyst-centric view, TOGAF provides an architecture centric view, the DMBoK provides a data management centric view, and so on. All great views, but when combined with one another, which is a common approach in most organizations today looking for "best practices", they prove to be an ineffective mishmash. DA 2.x provides a coherent, integrated, high-level view of how an IT department may address all of these key areas in a consistent, flexible, and evolutionary manner. Wherever possible DA 2.x references the effective ideas in these BoKs and supplements them with strategies that are more consistent with modern agile approaches. Figure 3 overviews the workflow for Disciplined Agile IT.

DA 2.x has been arranged into components called process blades – such as *Enterprise Architecture*, *Data Management*, or *Reuse Engineering* – that in effect are process areas focused on major IT activities. Just like each delivery process goal encompasses a collection of potential strategies or practices, similarly so does each process blade. The point is that for true software process improvement organizations must look at the entire IT process, not just their software development processes. Focusing on software development alone is like focusing on building a race car engine. It's interesting and you can create a very fast and efficient engine, but if you take that engine and put it into the body of a tractor you're not going to win any races. What you really need is to create an IT race car and a racing team to race it, then you've got a shot at winning the race.

We Can Do Better Than Agile CMMI

The Disciplined Agile framework can be instantiated in a CMMI-compliant manner if you need to [10]. We've even mapped CMMI to Disciplined Agile, showing that DA fully supports the process areas of CMMI-Dev [11]. We definitely see the value off applying agile strategies to improve existing CMMI implementations, but we can't honestly recommend applying CMMI to existing agile teams. There are many challenges with combining agile and CMMI – although it's possible to overcome these challenges, in most situations it likely isn't worth the effort.

Most organizations that have embraced agile ways of working have moved beyond the false promise of repeatable processes and instead focus on being able to produce repeatable results. To do this they adopt a flexible, context-sensitive process; they marry the what with the how by making process-oriented decisions, and the options associated with them, explicit to their teams; and they focus on evolutionary learning and improvement over the false consistency of process conformance.

ABOUT THE AUTHORS

Scott W. Ambler is a Senior Consulting Partner of Scott Ambler + Associates, working with organizations around the world to help them to improve their software processes. He provides training, coaching, and mentoring in disciplined agile and lean strategies at both the project and organizational level. Scott is the (co)-creator of the Agile Modeling (AM), Agile Data (AD), Disciplined Agile Delivery (DAD), and Enterprise Unified Process (EUP) methodologies. He is the (co-)author of several books, including *Disciplined Agile Delivery*, *Refactoring Databases*, *Agile Modeling*, *Agile Database Techniques*, *The Object Primer 3rd Edition*, and *The Enterprise Unified Process*. Scott blogs regularly at DisciplinedAgileDelivery.com and he can be reached at scott [at] scottambler.com and tweets @scottwambler.

Mark Lines is Enterprise Coach and Managing Partner at Scott Ambler + Associates. He is co-creator of the Disciplined Agile Delivery (DAD) framework and co-author with Scott Ambler of Disciplined Agile Delivery: A Practitioner's Guide to Agile Software Delivery in the Enterprise. Mark helps organizations all over the world transform from traditional to agile methods. He writes for many publications including the Cutter Consortium and is a frequent speaker at industry conferences. Mark blogs about DAD at DisciplinedAgileDelivery.com. He is also a Founding Member of the Disciplined Agile Consortium (DAC), the certification body for disciplined agile. He can be reached at mark [at] scottambler.com and tweets @mark_lines

REFERENCES

1. CMMI Institute (2016). CMMI Models. http://cmmiinstitute.com/cmmi-models
2. Beck, Kent et. al. (2001). The Agile Manifesto for Software Development. http://agilemanifesto.org/
3. Tadros, Marian. (2014). CMMI with Agile: Industry Success Stories in Process Improvement. http://CMMIinstitute.com/sites/default/files/resource_asset/Agile%20CMMI%20Success%20Stories_Tadros_SEPGNA14.pdf
4. McMahon, Paul E. (2010). Integrating CMMI and Agile Development: Case Studies and Proven Techniques for Faster Performance Improvement. Addison-Wesley Professional.
5. Ambler, Scott W. and Lines, M.J. (2012). Disciplined Agile Delivery: A Practitioner's Guide to Agile Solution Delivery in the Enterprise. IBM Press.
6. Disciplined Agile Consortium (2012). Full Agile Delivery Lifecycles. http://DisciplinedAgileDelivery/lifecycle/
7. Disciplined Agile Consortium (2012). Process Goals. http://DisciplinedAgileDelivery/process-goals/
8. Disciplined Agile Consortium (2012). Scaling Agile: The Software Development Context Framework. http://DisciplinedAgileDelivery.com/sdcf/
9. Disciplined Agile Consortium (2014). Strategic Agility at Scale. http://DisciplinedAgileDelivery.com/agility-at-scale/strategic-agility-at-scale/
10. Ambler, Scott W. (2012). Disciplined Agile Delivery Meets CMMI. Cutter IT Journal, November 2012. http://DisciplinedAgileConsortium.org/Resources/Documents/itj1211_DADandCMMI.pdf
11. Disciplined Agile Consortium (2016). Mapping CMMI to Disciplined Agile. http://DisciplinedAgileDelivery.com/mapping/cmmi/

Agile 5

Using High Maturity CMMI Practices to Improve Agile Processes and Achieve Predictable Results

Deepti Sharma
Nishi Narula
Djindo Lee
Theron R. Leishman

Abstract. In this article we will share a case-study of our experience combining Scrum and CMMI Level 5, what we refer to as "Agile 5", to significantly improve the delivery rate of product capabilities while maintaining a high level of quality and employee satisfaction. We will also share the pain experienced during our journey of blending these industry best practices and the lessons we learned.

Introduction

The idea of agility has significantly different connotations depending on personal background and frame of reference. When considering what represents agility, ideas seem to reflect personal interests and perspectives. To an avid hunter, agility may be reflected in the behavior of a deer – quick, nimble, sneaky, allusive, able to quickly change direction. To a cat lover, agility may be reflected in their cat – sneaky, quiet, able to get into tight spaces with relative ease, ability to jump, twist, turn, and hang-on. To a dancer – flexible, flowing, smooth, ability to jump, spin, and twirl. To an NBA fan – a basketball star with speed, strength, the ability to cut, twist, jump, and fake to avoid defenders.

In the world of software development, the discussion of agility can conjure up similar thoughts. We have all likely experienced projects that profess to be following an agile approach when in reality they are allusively sneaky, quick to change direction, jump through hoops to report progress, claw, scratch, and hang-on to deliver something half-baked, not documented and with no responsibility for the final product.

By contrast, mentioning CMMI often brings to mind extremely large projects burdened by detailed processes, plans, procedures, and volumes of documentation and measures that drag on forever resulting in high risk of failure or a product that is not usable.

In reality, both agile and traditional CMMI based approaches have proven effective in delivering high quality products to customers. The question may be asked, is it possible for an organization to use the concepts and principles of the CMMI high maturity process areas to proactively improve agile processes? Is it possible to have agile processes that provide predictable results?

In this article we will share a case-study of our experience combining Scrum and CMMI Level 5, what we refer to as "Agile 5", to significantly improve the delivery rate of product capabilities while maintaining a high level of quality and employee satisfaction. We will also share the pain experienced during our journey of blending these industry best practices and the lessons we learned.

The Situation

During a period of tight and reducing budgets, sequestration, and threat of government shutdown, one of our large projects was faced with the challenge of continuing to provide essential aviation systems in a safe and efficient manner. The project was following an agile development lifecycle based on a Scrum process. As a CMMI Level 5 organization, Optimal Solutions and Technologies Incorporated (OST Inc.) has extensive experience using CMMI high maturity practices and principles to improve the effectiveness of processes and achieve business goals and objectives. With the intent of proactively improving our agile process and providing greater value to our client, we embarked upon an effort to improve our agile process using the practices of CMMI Level 5. The goal, based on these objectives, was to increase our feature delivery rate to better meet the demand for system updates while maintaining a high level of quality as measured by production defect density, and promote a high level of team satisfaction.

The Pain

However worthy our objectives were, our journey to improve our agile process using CMMI Level 5 was not without struggle, challenge, and pain. Listed below are the major areas of pain we experienced as we attempted to use CMMI to improve our agile process.

How should we proceed? Although we had significant experience using the high maturity processes of CMMI to gain improvement in our traditional processes, methods, and lifecycle, applying them to an agile process was a new experience. We had established process performance baselines and performance models based on traditional methods to provide insight into our supporting processes. The usefulness of these baselines and models in an agile environment was a question that we had to consider.

Agile – does it lend itself to data collection? We had a significant amount of data that had been collected, analyzed, and used to manage our traditional processes. We were uncertain if our agile process provided the opportunity to gather significant and useful data. Confusion as to what measurement data to collect, where to gather it from, what analysis should be performed on the data, what story the data was telling us, and how to use the data in an effective manner were also challenging.

Sacred Cows: The dictionary defines a "Sacred Cow" as an individual, organization, institution, etc., considered to be exempt from criticism or questioning. Figuratively, anything that is beyond criticism may be considered a sacred cow. In our organization we had many Sacred Cows. Existing processes, measures, and culture were so engrained into the organization that no one questioned their value. Our peer review process is an example of a sacred cow. Our traditional lifecycle plans and processes dictated what a peer review was, how it was conducted, who should attend, what a defect was, and what data about

a peer review should be gathered, analyzed, and used. The Scrum process did not lend itself well to our traditional definition and approach. We had to be open to thinking outside of our traditional way of doing things; to think of principle/intent, not existing practice/implementation, and consider different ways of accomplishing things.

CMMI –vs- Agile: A question that we continually faced was, do CMMI and Agile even work together? Our effort to actually blend the two caused us to come back to this question time and again. We had to ask serious questions such as: Do we still need CMMI Level 5? Will this add value? What models will we use? Do we throw away our existing models? Will we be able to blend seemingly opposing methodologies of Agile and CMMI Level 5?

The Approach

In an agile environment, clear and distinct separation between lifecycle phases does not exist as in traditional methodologies. This made modelling individual lifecycle phases challenging or impossible. We determined our best approach was to use Discrete Event Simulation (DES) to model our entire sprint process because it provided a holistic approach to capturing and depicting the sprint lifecycle. Our DES model allowed us to account for the entire lifecycle from user stories (requirements), design, development, and test within individual sprints. Baselines of the functional activities of business analysts, developers, and testers were used to populate the model with actual data from project sprints.

Based on these performance baselines, the model predicted the number of story points that could be completed in a given release and/or sprint. With each sprint, additional data was gathered and incorporated into our performance baselines which allowed us to further calibrate and refine the model. The Causal Analysis and Resolution (CAR) process was incorporated into the sprint retrospective to help refine and improve our process. The model allows various factors to be entered, which are used to statistically predict sprint outcomes. The factors include: development time, test time, test case development time, defect density, number of user stories, number of story points, and resource availability. Using this model during release and sprint planning, we were able to predict the number of story points likely to be completed during each sprint and release. By adjusting the factors, we were able to conduct a hypothetical/what-if analysis to determine the optimal team make up and dynamic to achieve results consistent with business goals and objectives. Our DES model allowed us to model wait times and bottlenecks and to change the controllable factors to evaluate what-if scenarios for downstream impacts. Figure 1 below depicts how the model was used.

The model also supported project planning, project monitoring, and control and risk management activities by guiding resource planning, determining if the velocity was improving over time, and adjusting resource levels based on the constraints.

The SEI has defined a valuable list of "Heathy Ingredients" of CMMI-based process performance models [1]. We use this list to ensure that our models adhere to modelling best practices. Table 1 maps our DES model to these ingredients.

The Result

Although the journey was not a simple one, we are pleased to report that our results were well worth the effort. By applying CMMI high maturity principles to our sprint process we were able to achieve the following:

- Refined and improved our sprint process and exceeded our sprint velocity improvement goal

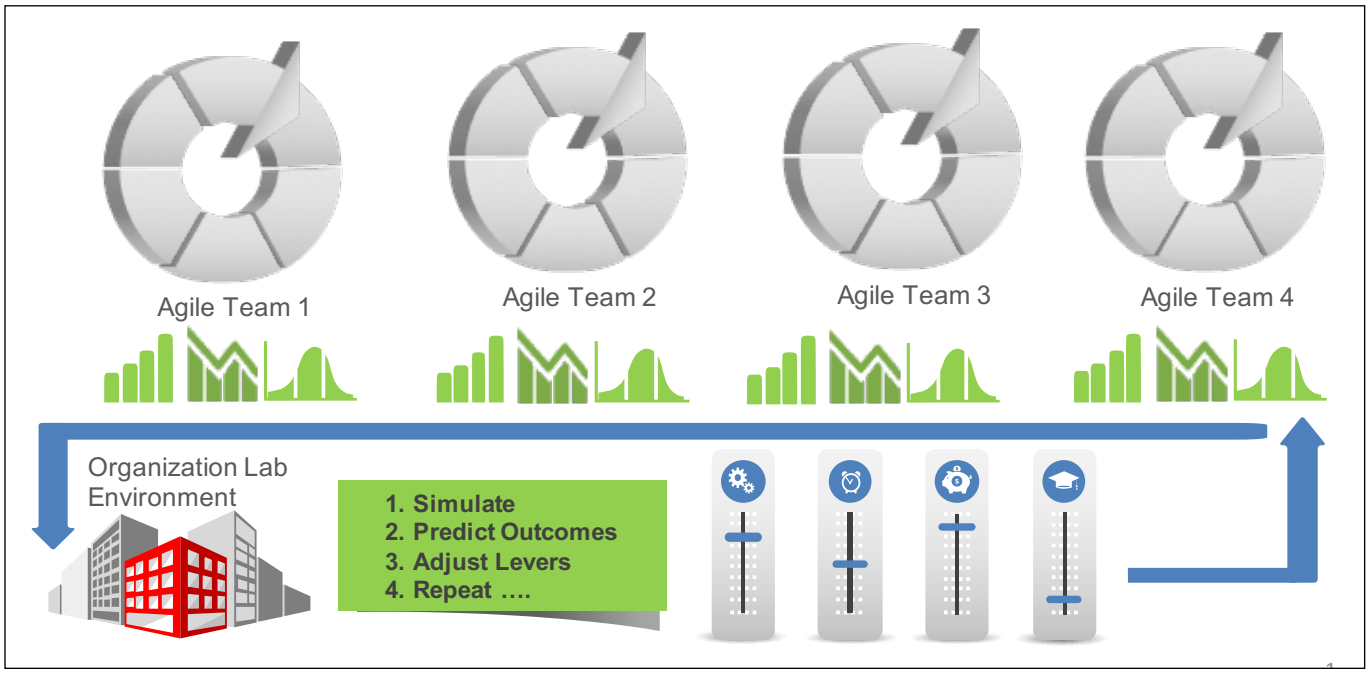


Figure 1. Sprint/Release Planning Model

SEI Healthy Model Ingredients	Release Planning & Monitoring Model
Statistical, probabilistic, or simulation in nature	Simulation model in Arena
Predict interim and/or final project outcomes	Predicts final and interim outcomes for release: Number of story points that will be delivered in the sprint/release
Use controllable factors tied to sub-processes to conduct the prediction	Uses controllable factors: development time, test time, test case development time, defect density, user stories, story points, resources available as factors tied to predictable outcomes
Model the variation of factors and understand the predicted range or variation of the outcomes	Models the variation in each factor and contributes to understanding of the prediction and confidence intervals
Enable "what-if" analysis for project planning, dynamic re-planning and problem resolution during project execution	Enables the running of "what-if" scenarios to plan/re-plan Sprints/releases and to resolve execution problems
Connect "upstream" activity with "downstream" activity	Connects upstream activities of development and test with defect density
Enable projects to achieve mid-course corrections to ensure project success	Predicts sprint and release results throughout the project lifecycle based on daily sprint realities supporting mid-course corrections and ultimate project success

Table 1

- Developed a predictive model that allowed us to more effectively plan and manage Sprint and Release activities
- Satisfied 20 percent more requirements per release than originally expected
- Maintained a high level of quality as measured by production defect density of only three percent
- Realized productivity gain of $600k over three releases
- Fostered a high level of team satisfaction due to increased sense of accomplishment

Lessons Learned

In addition to our improvement effort yielding tremendous results to us and our client, we also learned many lessons throughout the process. Those deemed most valuable are shared below.

95 percent Analysis Vs 5 percent Modelling: Accurate data is essential to quantitatively understanding process performance. The effort to gather, understand, analyze, and scrub data to ensure it has integrity can be labor intensive, challenging, and time consuming. It is also imperative that the data being used to provide quantitative insight into processes that support the accomplishment of identified business and quality goals and objects is the right data. We found it useful to ask, why are we measuring the things we measure? There needs to be solid correlation between the project goals and objectives, the critical processes that support the goal, and the things being measured to provide insight into the process.

Consistency in data recording and accounting is also key to the ongoing integrity of performance baselines and models. Attention to detail in this area helped to ensure proper calibration and effectiveness of models.

Giving attention to and planning for the 95 percent effort required for good analysis resulted in significant value in the 5 percent modeling (the fun part) and gaining value from our predictive models.

Have a Plan: We treated this improvement effort like a project. We established a plan to guide our efforts, monitored progress consistently, and involved the right people in a forum of open and honest communication. Appropriate guidance was also key to our plan. We involved an experienced and trusted lead appraiser in our effort to provide insight throughout the journey.

Capitalize on the "Power of Failure": The power of failure arises when we are not afraid of failure. This power comes as we make decisions based on the best information we have at the time and moving forward. Sometimes these decisions will not have the desired result, but do provide valuable data that can add to our knowledge base and help us make better decisions going forward. At OST, Inc., this principle has proven very valuable. As we began our initiative to use measures to guide business decisions we found in some cases, we were either measuring the wrong thing or using the analysis of the data inappropriately. As we began to develop process performance baselines and models, we found that sometimes they did not provide valuable insight

into our processes and toward goal achievement. Rather than give up, we learned from these experiences and continued to move forward. The adage, "If at first you do not succeed, try, try again" should be engrained in a good process improvement culture. As we developed a culture where employees were not afraid to move forward and make decisions due to fear of failure and retribution, we have been able to proactively learn from mistakes and have seen great value from these lessons.

Real value in blending Agile and CMMI Level 5: The practices of CMMI Level 5 added more value to our agile implementation than we saw with our traditional methods. There are those who say that CMMI and agile do not work and play well together. We found the opposite to be the case. A major premise of agile methods like Scrum is to have a process that cycles quickly, providing working features and functionality to the customer on regular intervals. These quick cycles provide more data, which can in turn provide greater insight into the processes that support accomplishment of business objectives.

Using this premise, we were able to achieve significant improvement in our agile process by using predictive models, based on good baseline data. By incorporating the practices of CAR, we were able drive improvements into the process and/or capitalize on good things happening within the process, to significantly improve productivity, customer satisfaction, and goal achievement.

Conclusion

Just as an experienced hunter, cat owner, dancer, or basketball team are able to predict to some degree of probability the behavior of their agile subject; so agile development activities via CMMI Level 5 practices can be analyzed, modeled, and improved to provide predictable outcomes. Although agile methods and CMMI are often perceived to be opposite ends of the spectrum, we found that our agile process actually worked more effectively and matured faster when coupled with the high maturity CMMI practices. The frequency with which agile processes cycle results in the generation of data, actually lends itself to the establishment of performance baselines, better modelling, and predictive results better than traditional methods. The bottom line is, OST Inc.'s Agile5 leverages the discipline, comprehensiveness, and sustainment focus of CMMI with the build, speed, and lean focus of agile to bring the best of both worlds to our clients!

REFERENCES

1. R. W. Stoddard and D. Goldenson, "Approaches to Process Performance Modeling: A Summary from the SEI Series of Workshops on CMMI High Maturity Measurement and Analysis," Carnegie Mellon Software Engineering Institute CMU/SEI-2009-TR-021, 2010. [Online]. http://www.sei.mcu.edu/library/abstracts/reports/09tr021.cfm

ABOUT THE AUTHORS

Ms. Deepti Sharma leads the Strategy and Business Performance Group at OST. This includes Corporate Strategy, Certifications, Data Analytics and Business Performance functions. She has led OST's CMMI L5 and ISO certifications. Her client portfolio includes FAA, DHS, HUD, DoD and Treasury. She has been a speaker at several conferences including SEPG, NDIA and SEI's High-Maturity Measurement & Analysis Workshops. She has a Master's Degree in Engineering from Cornell University.

Ms. Nishi Narula has over 20 years of experience supporting government agencies in the areas of software development, strategy formulation, change management, governance and process improvement. As OST's Director of Human Capital Management, Strategy and Business Performance, she blends best practices, quality and performance metrics to create innovative improvements. She has presented at multiple conferences including SEPG, NDIA and SEI's High-Maturity Measurement & Analysis Workshops. She holds Master's degrees in Operations Research and Computer Science.

Mr. Djindo Lee is an Account Manager currently overseeing OST's DoD portfolio. He leads the delivery of high end systems and fosters an innovation-oriented, high performance work environment that marries emphasis on creativity, collaboration and end results. Mr. Lee puts into practice solution based on industry best practices such as CMMI, Agile, ITIL and ISO to achieve stellar product quality and development efficiencies.

Theron R. Leishman is OST's Quality Assurance/Process Improvement Manager supporting the F-16 Program Office at Hill Air Force Base, Utah. Mr. Leishman has significant experience coaching and mentoring clients to leverage industry best practices to ensure quality and improve processes for the achievement of optimal results. He has a master's degree from the University of Phoenix, is a certified Scrum Master, an As9100 Lead Auditor and an authorized CMMI-DEV/CMMI-SVC instructor.

Upcoming Events

Visit <http://www.crosstalkonline.org/events> for an up-to-date list of events.

2016 31st Annual ACM/IEEE Symposium on Logic in Computer Science
New York, NY
5-8 July, 2016
http://www.ieee.org/conferences_events/conferences/conferencedetails/index.html?Conf_ID=38877

20th International Database Engineering & Applications Symposium
Montreal, QC, Canada
July 11-13, 2016
http://confsys.encs.concordia.ca/IDEAS/ideas16/ideas16.php

2016 IEEE International Conference on Automation Science and Engineering (CASE)
Fort Worth, TX
21-25 August 2016
http://www.ieee.org/conferences_events/conferences/conferencedetails/index.html?Conf_ID=35762

ICSEA 2016: The Eleventh International Conference on Software Engineering Advances
Rome, Italy
21-25 August, 2016
http://www.iaria.org/conferences2016/ICSEA16.html

The 14th International Workshop on Java Technologies for Real-Time and Embedded Systems- JTRES 2016
29 August – 2 September 2016
Lugano, Switzerland
http://jtres2016.compute.dtu.dk

31st IEEE/ACM International Conference
3-7 September 2016
Singapore, Singapore
http://www.ase2016.org

TAPIA '16: Richard Tapia Celebration of Diversity in Computing Conference
6-10 September 2016
Austin, TX
http://www.tapiaconference.org

10th ACM Conference on Recommender Systems
15-19 September 2016
Boston, MA
https://recsys.acm.org/recsys16

SIGDOC 2016
21-23 September 2016
Arlington, VA
http://sigdoc.acm.org/conference/2016

Structure Security 2016
27-28 September 2016
Menlo Park, CA
http://www.structuresecurity.com/security-2016/about

IWBDAC 2016 : The IEEE International Workshop on Big Data Analytics for Cybersecurity Computing
27-30 September 2016
Tucson, AZ
http://www.isi-conf-org

34th IEEE international Conference on Computer Design
3-5 October 2016
Phoenix, AZ
http://www.iccd-conf.com/Home.html

HCOMP 2016
30 October – 3 November 2016
Austin, TX
http://www.humancomputation.com/2016

Holy Utility Belt, Batman!

Nowadays, when you mention The Caped Crusader, folks think of Christian Bale, and his slick techo-toys. [NOTE: Why is Batman the "Caped Crusader"? Superman also wore a cape. As a matter of fact, so did Robin, the Boy Wonder. This always bugged me]. In fact, younger folks think of Ben Affleck as Batman.

Not me – I think of the REAL Batman (as played by Adam West, ably supported by Bert Ward as Robin, the Boy Wonder). Although the show was only on TV three seasons (120 episodes, 1966 – 1968), it left an indelible mark on me. It had the most far-fetched plots, the best guest stars (Esther Merman and Vincent Price, just to name a few), playing the best arch-villains – the Joker, the Penguin, the Riddler, and Catwoman (played by Eartha Kitt and Julie Newman in the two different seasons. Yes – I know, you're remembering Lee Meriwether – but she only played Catwoman in "Batman: The Movie" in 1966.) It even had (after season one) Batgirl (aptly played by Yvonne Craig) – and again, she made QUITE an impression on me, too - I think it was the costume. Amazing how Chief O'Hara and Commissioner Gordon never saw through Bruce Wayne's AMAZING disguise. But then, in the DC comic universe, a cheap set of glasses kept Superman's secret identify as Clark Kent hidden, too.

One VERY cool thing about Batman was his famous Bat Utility Belt. I kid you not – it was amazing! It had EVERYTHING on it: Bat Rope, the Bat Laser, the Bat Lock pick (usually kept in Batman's glove, however), three types of Batarangs (explosive, remote controlled, and shock-generating electrical), Bat shark repellant, and…. Well, you get the picture. As a matter of fact, Wikipedia, in their scholarly article regarding the Batman Utility Belt ((https://en.wikipedia.org/wiki/Batman%27s_utility_belt) lists about 30 items. I always wondered how Batman could rappel up the side of a building (using the Bat Grappler – also known at the Batclaw or Batline) when he must be hauling 200+ pounds of hardware along with him.

But – thank goodness for the Batman's utility belt. No matter what Batman needed to escape from the nefarious clutches of the week's arch villain – Batman had it with him. Locked in a safe? Thank goodness for the Bat Acetylene Torch. And – if the safe was small – well, there's the Bat Rebreather.

Mobile Radio Integration, Interoperability, and Frustration

For a long time – I had my utility belt, too. Rather than lock picks, my utility belt contained software development tools. My utility belt first contained flow charts. Flowcharts worked when the code I was writing was difficult to understand. In the 70s, flow charts were the only tools I kept in my utility belt for quite a while. I learned Nassi–Shneiderman diagrams (NSD) – a different way to do flowcharting. It helped me grow as a developer and designer.

As my skills matured and I tackled larger and harder problems, I learned a few other tools that I decided to keep in my utility belt as well. Structured Analysis and Structured Design (and Structured Analysis and Design Technique) filled a void in my arsenal that needed filling. TDSP (Top Down Structured Programming) also became a tool, as did Data Flow Diagrams (DFDs) and various other tools and techniques. I became an expert on the Waterfall Model. And, of course, the Waterfall Model begat 2167 and 2167A (Department of Defense Standard 2167A) titled "Defense Systems Software Development." Added to utility belt – check!

In the 1990s, my tool belt really grew – between Object-Oriented Analysis, Design and Programming, my utility belt became heavier. I also added various Unified Modeling Language (UML) techniques. I had already added the Booch Method – so I drooped Booch, and added UML.

And – starting in the 1990s – I started hearing about a new set of tool for my utility belt – a process model rather than a coding/analysis/design technique) to help an organization produce better software. It took a while to convince me – but the Capability Maturity Model (CMM) became an indispensable part of my utility belt. It worked so well, in fact, that there were soon lots and lots of different capability maturity models (The Software CMM, the People CMM, the Systems Engineering CMM, etc.) that I removed the CMM from my utility belt, and added the Capability Maturity Model Integration (CMMI) in its place. I also hung the Personal Software Process (PSP) and the Team Software Process (TSP) on the belt.

There – my utility belt was FINALLY full. I could relax, use the tools I had accumulated, and develop quality process which produced quality code which met my customers' needs.

Or could I? Holy Heartbreak, Batman. I often found myself in a position where none of the tools I had accumulated seemed to work. Sometimes, on some projects, traditional techniques just did not seem to be applicable. But there were rumblings in Gotham City – a new type of tool was ready for use! A new cast of heroes had developed "The Agile Movement." To quote from http://agilemethodology.org/: "The Agile movement seeks alternatives to traditional project management. Agile approaches help teams respond to unpredictability through incremental, iterative work cadences, known as sprints. Agile methodologies are an alternative to waterfall, or traditional sequential development."

You know what folks? Agile techniques are tools that hang very neatly on my utility belt. I don't use them for everything (but then, the Bat Concussion Mine is not really useful if you're already locked inside of the safe, is it?)

We need to value all of the tools at our disposal – each gives us a different (and in some cases, unique) viewpoint to solving a problem. Different people, different approaches, different techniques.

My utility belt is still heavy, even though I've thrown away a lot over the years. I don't use SASD and SADT a lot anymore – and I really don't flowchart much anymore either (although I do create UML Activity Diagrams – which really look like the same thing – but that's another column.)

I have available a variety of tools for a variety of problems – and some only work in certain places. If I'm developing a quick web app – CMMI is too big, and Agile works. If I'm writing nuclear reactor software, then Agile is not enough, and CMMI help produce a quality product. Applications between the two extremes? Then I use whatever tool or tools that help you to the job done! Hold Tool Choice, Batman!

See you next issue – same Bat time, same Bat channel.

David A. Cook, Ph.D.
Professor of Computer Science
Stephen F. Austin State University
cookda@sfasu.edu

By the way - https://en.wikipedia.org/wiki/List_of_exclamations_by_Robin lists over 300 expressions of the form "Holy xxxx, Batman" that Robin uttered. Batman ran three seasons over two years, and had 120 episodes. That's about three "Holy Somethings" per show. Holy Exclamation!

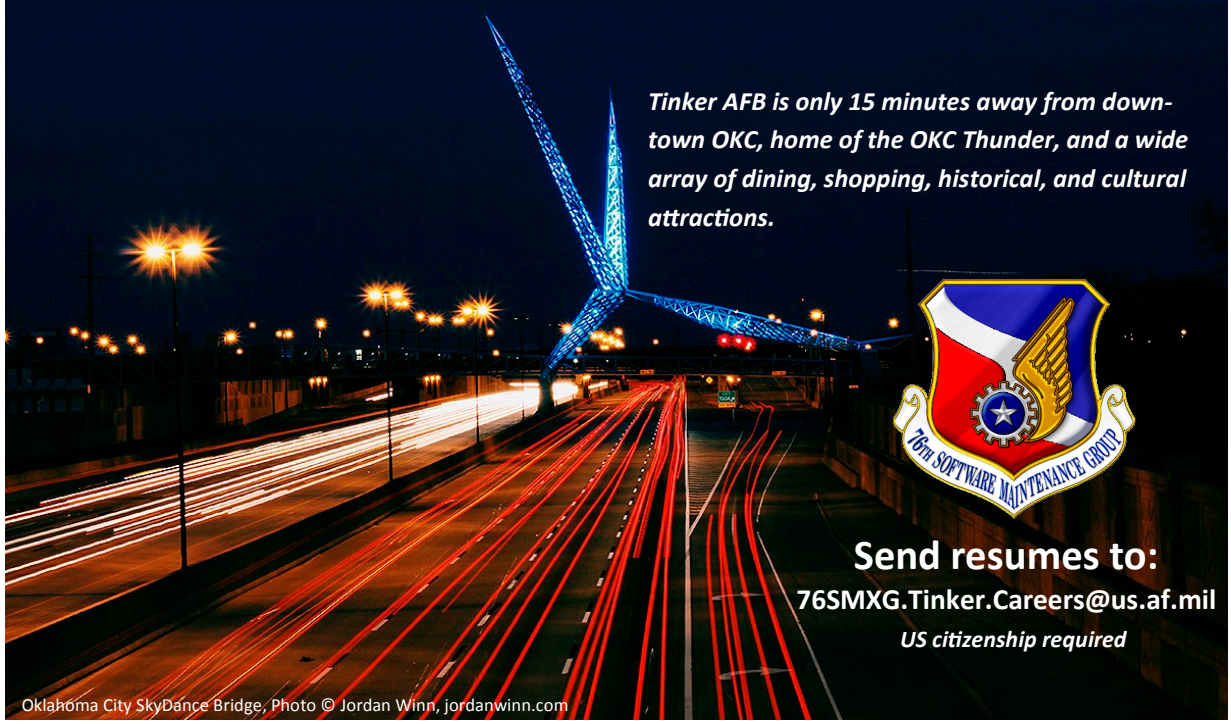
WE ARE HIRING

ELECTRICAL ENGINEERS AND COMPUTER SCIENTISTS

As the largest engineering organization on Tinker Air Force Base, the 76th Software Maintenance Group provides software, hardware, and engineering support solutions on a variety of Air Force platforms and weapon systems. Join our growing team of engineers and scientists!

BENEFITS INCLUDE:

- Job security
- Potential for career growth
- Paid leave including federal holidays
- Competitive health care plans
- Matching retirement fund (401K)
- Life insurance plans
- Tuition assistance
- Paid time for fitness activities



Tinker AFB is only 15 minutes away from down-town OKC, home of the OKC Thunder, and a wide array of dining, shopping, historical, and cultural attractions.

Send resumes to:

76SMXG.Tinker.Careers@us.af.mil

US citizenship required

Oklahoma City SkyDance Bridge, Photo © Jordan Winn, jordanwinn.com

NAVAIR

CROSSTALK thanks the above organizations for providing their support.