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).

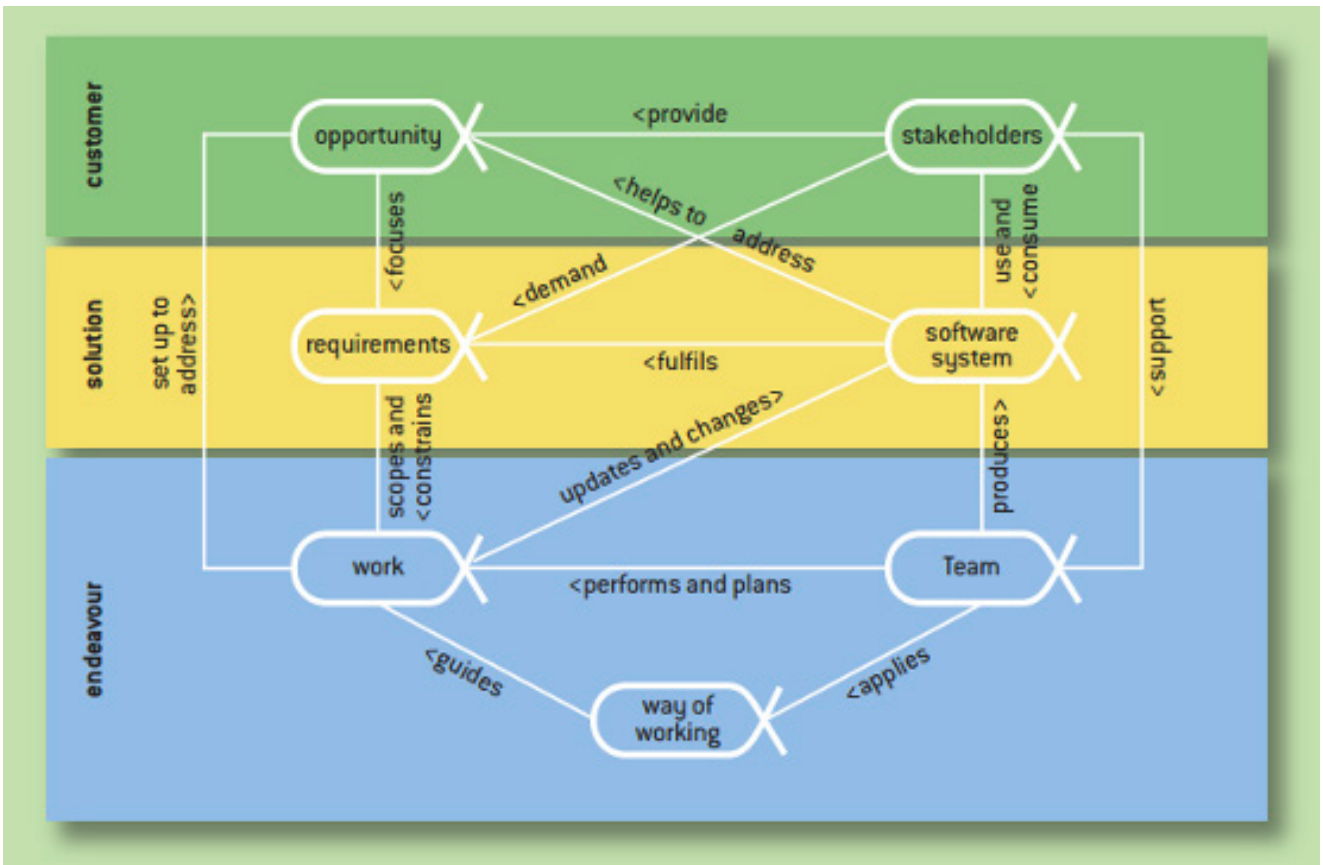

Figure 1. SEMAT Essence Kernel

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.