NPS-AM-16-017

ACQUISITION RESEARCH PROGRAM SPONSORED REPORT SERIES

Reducing Risk in DoD Software-Intensive Systems Development

March 2016

Brad Naegle, Senior Lecturer

Graduate School of Business & Public Policy

Naval Postgraduate School

Approved for public release; distribution is unlimited.

Prepared for the Naval Postgraduate School, Monterey, CA 93943.

ACQUISITION RESEARCH PROGRAM
GRADUATE SCHOOL OF BUSINESS & PUBLIC POLICY
NAVAL POSTGRADUATE SCHOOL

The research presented in this report was supported by the Acquisition Research Program of the Graduate School of Business & Public Policy at the Naval Postgraduate School.

To request defense acquisition research, to become a research sponsor, or to print additional copies of reports, please contact any of the staff listed on the Acquisition Research Program website (www.acquisitionresearch.net).

ACQUISITION RESEARCH PROGRAM
GRADUATE SCHOOL OF BUSINESS & PUBLIC POLICY
NAVAL POSTGRADUATE SCHOOL

Abstract

The Department of Defense (DoD) continues to experience both successful and unsuccessful software-intensive systems development, despite all of the considerable qualifications of the workforce and the significant controls offered by the Defense Acquisition System (DAS). The unsuccessful software-intensive systems have suffered dramatic cost and schedule overruns, impacting modernization timetables and reducing funding for other developmental priorities.

This research is a continuation and consolidation of previous research projects conducted for the U.S. Navy Open Architecture Task Force. That previous research is identified and cited where appropriate.

The purpose of this research is to analyze why DoD risk management processes have not been more effective in reducing software-intensive systems development risk. This research addresses the use of the Technical Readiness Assessment (TRA) using the nine-level software Technology Readiness Levels (TRLs), use of the software developer maturity level assessments using the Software Engineering Institute's (SEI's) Capability Maturity Model-Integrated for Development (CMMI-DEV), and use of the software acquirer maturity level assessments using SEI's CMMI for Acquisition (CMMI-ACQ). The recommendations include the use of effective tools, techniques, and analyses presented in the author's previous research efforts.

The previously researched tools, techniques, and analyses include the SEI's Quality Attribution Workshop (QAW), the MUIRS (maintainability, upgradability, interoperability, reliability, and safety/security) analysis methodology, SEI's Architectural Tradeoff Analysis Methodology^{sm} (ATAM^{sm}), Logistics Supportability Analysis (LSA), and the Failure Modes and Effects Criticality Analysis (FMECA).

The major research findings include the following:

- The software TRLs are ineffective in reducing technical risk for the software component development.
- Without the software TRLs, there is no effective method to perform software TRA or reduce the technical development risk. The software component will behave as a new, untried technology in nearly every software-intensive weapon system development.
- Given that the software technical risk cannot be effectively managed, risk reduction is focused on reducing management risk for both the supplier (contractor) and the acquirer (government PM organization).

- Assessing organizational maturity as a risk-reducing approach, industry appears to be gaining excellent results through moving up the SEI CMMI-DEV maturity levels. There seems to be little DoD interest in using the SEI CMMI-ACQ for assessing or improving the PM organization maturity.

The major research recommendations include the following:

- Shift the software development risk management focus from technical to managerial risk reduction.
- A nine-level software Management Readiness Level (MgtRL) model is recommended. The model integrates the previously researched tools, techniques, and analyses with maturity assessments and goals for achieving CMMI maturity levels for both the software developer and the Government PM organization.
- Consistent with the use of other TRLs, achieving MgtRL level 6 is recommended to reduce the risk of developing the software component as a new technology.

Keywords: Software-intensive system acquisition, system acquisition risk management, software system management, Quality Attribute Workshop (QAW), Architecture Trade-off Analysis Methodology (ATAM), Failure Modes and Effects Criticality Analysis (FMECA), MUIRS, Software Engineering Institute (SEI) Capability Maturity Model-Integrated (CMMI) for Development (CMMI-DEV) and for Acquisition (CMMI-ACQ), DoD Acquisition System.

Acknowledgments

The author would like to thank Rear Admiral James Greene (Ret.) for his remarkable contribution to the DoD acquisition community as the Acquisition Chair for the Naval Postgraduate School. His efforts have enabled these research efforts that address current and future problems and vastly improve the education product that NPS delivers.

THIS PAGE INTENTIONALLY LEFT BLANK

About the Author

Brad R. Naegle, Lieutenant Colonel, U.S. Army (Ret.), is a senior lecturer and academic associate for Program Management Curricula at the Naval Postgraduate School, Monterey, CA. In addition to acquisition course development and delivery, he is the academic associate for the Master of Science in Program Management curriculum, which is a distance learning master's program, serving students around the world. He is currently serving on the Department of Navy software community of practice. While on active duty, LTC (Ret.) Naegle was assigned as the product manager for the 2½-ton Extended Service Program (ESP) and USMC Medium Tactical Vehicle Replacement (MTVR) from 1994 to 1996 and served as the deputy project manager for Light Tactical Vehicles from 1996 to 1997. He was the 7th Infantry Division (Light) Division materiel officer from 1990 to 1993 and the 34th Support Group Director of Security, Plans, and Operations from 1986 to 1987. Prior to that, LTC (Ret.) Naegle held positions in Test and Evaluations and Logistics fields. He earned a Master of Science degree in Systems Acquisition Management (with Distinction) from the Naval Postgraduate School and an undergraduate degree in economics from Weber State University. He is a graduate of the Command and General Staff College, Combined Arms and Services Staff School, and Ordnance Corps Advanced and Basic Courses.

Brad Naegle
Senior Lecturer
Graduate School of Business & Public Policy
Naval Postgraduate School
Monterey, CA 93943
Tel: (831) 656-3620
E-mail: bnaegle@nps.edu

THIS PAGE INTENTIONALLY LEFT BLANK

NPS-AM-16-017

ACQUISITION RESEARCH PROGRAM SPONSORED REPORT SERIES

Reducing Risk in DoD Software-Intensive Systems Development

March 2016

Brad Naegle, Senior Lecturer

Graduate School of Business & Public Policy

Naval Postgraduate School

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

THIS PAGE INTENTIONALLY LEFT BLANK

Executive Summary

The Department of Defense (DoD) continues to experience both successful and unsuccessful software-intensive systems development, despite all of the considerable qualifications of the workforce and the significant controls offered by the Defense Acquisition System (DAS). The unsuccessful software-intensive systems have suffered dramatic cost and schedule overruns, impacting modernization timetables and reducing funding for other developmental priorities.

This research is a continuation and consolidation of previous research projects conducted for the U.S. Navy Open Architecture Task Force. That previous research is identified and cited where appropriate.

The purpose of this research is to analyze why DoD risk management processes have not been more effective in reducing software-intensive systems development risk. This research addresses the use of the Technical Readiness Assessment (TRA) using the nine-level software Technology Readiness Levels (TRLs), use of the software developer maturity level assessments using the Software Engineering Institute's (SEI's) Capability Maturity Model-Integrated for Development (CMMI-DEV), and use of the software acquirer maturity level assessments using SEI's CMMI for Acquisition (CMMI-ACQ). The recommendations include the use of effective tools, techniques, and analyses presented in the author's previous research efforts.

The previously researched tools, techniques, and analyses include the SEI's Quality Attribution Workshop (QAW), the MUIRS (maintainability, upgradability, interoperability, reliability, and safety/security) analysis methodology, SEI's Architectural Tradeoff Analysis Methodology^{sm} (ATAM^{sm}), Logistics Supportability Analysis (LSA), and the Failure Modes and Effects Criticality Analysis (FMECA).

The major research findings include the following:

- The software TRLs are ineffective in reducing technical risk for the software component development.
- Without the software TRLs, there is no effective method to perform software TRA or reduce the technical development risk. The software component will behave as a new, untried technology in nearly every software-intensive weapon system development.
- Given that the software technical risk cannot be effectively managed, risk reduction is focused on reducing management risk for both the supplier (contractor) and the acquirer (government PM organization).

- Assessing organizational maturity as a risk-reducing approach, industry appears to be gaining excellent results through moving up the SEI CMMI-DEV maturity levels. There seems to be little DoD interest in using the SEI CMMI-ACQ for assessing or improving the PM organization maturity.

The major research recommendations include the following:

- Shift the software development risk management focus from technical to managerial risk reduction.
- A nine-level software Management Readiness Level (MgtRL) model is recommended. The model integrates the previously researched tools, techniques, and analyses with maturity assessments and goals for achieving CMMI maturity levels for both the software developer and the government PM organization.
- Consistent with the use of other TRLs, achieving MgtRL level 6 is recommended to reduce the risk of developing the software component as a new technology.

Keywords: Software-intensive system acquisition, system acquisition risk management, software system management, Quality Attribute Workshop (QAW), Architecture Trade-off Analysis Methodology (ATAM), Failure Modes and Effects Criticality Analysis (FMECA), MUIRS, Software Engineering Institute (SEI) Capability Maturity Model-Integrated (CMMI) for Development (CMMI-DEV) and for Acquisition (CMMI-ACQ), DoD Acquisition System.

Introduction

> Acquisition of needed capabilities is challenging because acquirers have overall accountability for satisfying the end user while allowing the supplier to perform the tasks necessary to develop and provide the solution. … Unfortunately, many organizations have not invested in the capabilities necessary to effectively manage projects in an acquisition environment. Too often acquirers disengage from the project once the supplier is hired. Too late they discover that the project is not on schedule, deadlines will not be met, the technology selected is not viable, and the project has failed. (SEI, 2010b, p. 3)

This research is the latest in a series of research products focused on improving the DoD software-intensive system acquisition. Previous research addressed numerous tools, techniques, and analyses designed to improve the system development process. All of the research follows a common theme: reducing risk in developing DoD software-intensive systems. This research follows that same theme.

As this is a continuation of previous research, there are concepts referenced here from those other efforts. The following concepts will not be completely repeated in this research but do play an important role in the recommendations:

- DoD Requirements Generation: "The DoD acquisition environment features a requirements flow-down process that involves user-stated capabilities-based requirements translated to performance-based requirements, then translated to the detailed design specifications" (Naegle, 2014, p. 10). The translation process between capbilities-based to performance-based, and finally to detailed specifications provides opportunities for misinterpreted, vaguely stated, weakly articulated requirements, and for completely missing requirements as well.
- The Defense Acquisition System (DAS): "The DoD acquisition environment appears to remain vulnerable to significant variability when developing software-intensive systems, similar to the problems currently plaguing the F-35 JSF program. Although the new phases and milestones models address the software component development, other critical management functions remain unchanged. Requirements generation, performance specification development, RFP, source selection, and contracting processes have yet to adapt to the unique challenges presented when managing software-intensive system development" (Naegle, 2014, p. 10).

- The Software Engineering Environment: There is significant evidence for software engineering immaturity, and it is nearly impossible to find widely accepted, industry-wide development standards, protocols, architectures, or formats. There are no dominant programming languages, design and development processes, standard architectures, or software engineering tools, which means that reusable modules and components rapidly become obsolete. All of these combine to make it nearly impossible to institute a widely accepted software reuse repository. Without significant software architecture and code reuse in developing software-intensive weapon systems, each development process essentially starts from scratch. This fact is one of the main reasons that the technology readiness assessment (TRA) and the software technology readiness levels (TRLs) are ineffective in predicting software development risk (Naegle & Petross, 2007). The software engineering state-of-the-practice currently is wholly dependent on the requirements that are passed to the software development team. From the requirements, a software architecture is designed, and the requirements "flow down" through that architecture to the individual modules and computer software units that are to be constructed. The software build focuses on the requirements that flowed down to that level and the integration required for functionality. The standards, protocols, formats, languages, and tools used for the build will likely be unique to the contractor developing the software and will most certainly not be universally accepted or recognized across the software industry (Naegle, 2014, pp. 11–12).
- Tools, Techniques, and Processes:
    - The Software Engineering Institute's Quality Attribute Workshop (QAW): "The QAW is primarily a method for more fully developing system software requirements and is intended to provide stakeholders input about their needs and expectations from the software As the system requirements are developed, software quality attributes are identified and become the basis for designing the software architecture." (Barbacci et al., 2003, p. 1). "The QAW process is primarily designed to more fully develop system software requirements so that the Government RFP is clearer to potential contractors. In turn, the resulting proposals should be more accurate and realistic, reducing requirements and project scope creep" (Naegle, 2014, pp. 23–24).

- The Maintainability, Upgradability, Interoperability, Reliability, & Safety and Security (MUIRS) analytic technique: “The MUIRS analytic technique is designed to provide a framework for better understanding of essential supportability and safety/security aspects that the warfighter needs and expects but often doesn’t communicate clearly with the capabilities-based JCIDS documents. This analytic technique helps compensate for the immature software engineering environment as the MUIRS analysis illuminates the derived and implied requirements that the immature environment cannot. With its capabilities and performance based requirements processes, the DoD significantly depends on mature engineering environments to fill the gaps left from the requirements generation and communication processes, but the software engineering environment is unable to do so. The MUIRS analysis is also an enabler for the QAW and Architectural Tradeoff Analysis Methodology (ATAM^{sm}).” (Naegle, 2014, pp. 24–25).
- The Software Engineering Institute’s (ATAM^{sm}): The SEI’s ATAM^{sm} is an architectural analysis tool designed to evaluate design decisions based on the quality attribute requirements of the system being developed. The methodology is a process for determining whether the quality attributes are achievable by the architecture, because it has been conceived before enormous resources have been committed to that design. One of the main goals is to gain insight into how the quality attributes trade-off against each other (Kazman, Klein, & Clements, 2000, p. 1). “The ATAM process addresses four primary problem areas:
    - The scenario development provides much more operational context than the typical OMS/MP provides. This level of detail helps to compensate for the immature software engineering environment and is critical for the proper design of the software architecture.
      The ATAM serves as a very effective software design metric function. With the software development team using 50% or more of the available resources for requirements analysis and software design before the Preliminary Design Review (PDR), it is critical to have an effective software design metrics function. Traditional software design metrics focus on the design complexity and do not address whether the design is adequate or not. ATAM directly links the user requirements to the system

architectural design.

- As the testing program is developed from the scenarios, it becomes difficult to omit any critical testing event. In addition, the software developer understands the tests or verification events that must be passed for user acceptance.
- By integrating the MUIRS analyses into the ATAM scenario development, sustainability and safety/security aspects cannot easily be omitted from the system design. As the testing plan flows from the scenarios, the MUIRS design elements will have corresponding test or verification events identified in the test plan." (Naegle, 2014, pp. 27–28)

o The Failure Modes and Effects Criticality Analysis (FMECA): "As the title indicates, this analysis methodology is designed to identify system failure modes and those failures' effects on the system, and ascertain the relative criticality of that type of failure. The primary problem areas addressed by FMECA include requirements clarification and prioritization, and helping to ensure a sound software architecture design. This analysis also ensures that the most critical software systems are designed with the requisite reliability and will continue to function in degraded modes. As previously stated, one of the main functions of performing FMECA is to identify those software functions that are not critical, and ensuring that failures or anomalies in those non-critical functions do not preclude or negatively affect system capabilities. Today's systems typically have numerous enhancing functions that improve performance but are not critical and the software developers have no way to discern the difference between a critical system and an enhancing one without employing FMECA." (Naegle, 2014, pp. 28–29)

This research focuses on risk-reducing techniques including software TRA, the resulting TRLs, software developer "maturity," and DoD Program Management Office (PMO) "maturity." TRLs and developer maturity have long been methodologies used to reduce the developmental risk in software-intensive systems.

Of course, there has been much researched and written regarding DoD system development risk reduction in general, and the DoD has a very good model for helping to manage risk (see Figure 1).

DoD Risk Management Process

Figure 1. DoD Risk Management Process (DAU, 2015, p. 1)

Some of the very latest DoD guidance reiterates the absolute need for effective risk management, and the Better Buying Power 3.0 implementation memorandum states that

> successful product development requires understanding and actively managing program risks. Risk management is an endeavor that begins with requirements formulation and assessment, includes the planning and conducting of a technical risk reduction phase if needed, and strongly influences the structure of the development and test activities. Active risk management requires investment based on identification of where to best deploy scarce resources. (Under Secretary of Defense for Acquisition, Technology, and Logistics [USD(AT&L)], 2015, p.2)

Part of the risk management for system development includes assessing the readiness of the key technologies for development through a TRA. The DoD's *Technology Readiness Assessment Guide* describes the basic concept of technology risk identification:

> Technology risk identification should start well before the formal TRA process. In fact, potential critical technology identification begins during

> the Materiel Solution Analysis (MSA) phase, which precedes MS A. An early evaluation of technology maturity, conducted shortly after MS A, may be helpful to refine further the potential critical technologies to be assessed. It may be appropriate to include high-leverage and/or high-impact manufacturing technologies and life-cycle-related technologies if there are questions of maturity and risk associated with those technologies. (DoD, 2011, p. 2-6)

A TRA is required for most Major DoD Acquisition Programs (MDAPs; DoD, 2011, p. 1-1). The purpose for conducting a TRA is to address the risk of attempting to develop a system with a key technology that is too immature to successfully deploy the system when needed by the warfighter. To benchmark the assessment, TRLs have been developed in a nine-level model.The point of the TRA is to identify a system's key technologies, assess the developmental risk associated with those technologies, and assign an appropriate TRL. TRLs are divided into nine levels, as depicted in Figure 2. The higher the TRL number achieved, the lower the risk in successfully developing a system using that technology. Typically, TRL 6 or 7 technologies are the highest risk that PMs are likely to accept. However, lower technolgy readiness levels have been included in Programs of Record (PORs). As detailed in Table 1,TRL 7 features a new technology that has been proven in an operational environment.

TRL Definitions, Descriptions, and Supporting Information

TRL	Definition	Description	Supporting Information
1	Basic principles observed and reported.	Lowest level of technology readiness. Scientific research begins to be translated into applied research and development (R&D). Examples might include paper studies of a technology's basic properties.	Published research that identifies the principles that underlie this technology. References to who, where, when.
2	Technology concept and/or application formulated.	Invention begins. Once basic principles are observed, practical applications can be invented. Applications are speculative, and there may be no proof or detailed analysis to support the assumptions. Examples are limited to analytic studies.	Publications or other references that outline the application being considered and that provide analysis to support the concept.
3	Analytical and experimental critical function and/or characteristic proof of concept.	Active R&D is initiated. This includes analytical studies and laboratory studies to physically validate the analytical predictions of separate elements of the technology. Examples include components that are not yet integrated or representative.	Results of laboratory tests performed to measure parameters of interest and comparison to analytical predictions for critical subsystems. References to who, where, and when these tests and comparisons were performed.
4	Component and/or breadboard validation in a laboratory environment.	Basic technological components are integrated to establish that they will work together. This is relatively "low fidelity" compared with the eventual system. Examples include integration of "ad hoc" hardware in the laboratory.	System concepts that have been considered and results from testing laboratory-scale breadboard(s). References to who did this work and when. Provide an estimate of how breadboard hardware and test results differ from the expected system goals.
5	Component and/or breadboard validation in a relevant environment.	Fidelity of breadboard technology increases significantly. The basic technological components are integrated with reasonably realistic supporting elements so they can be tested in a simulated environment. Examples include "high-fidelity" laboratory integration of components.	Results from testing laboratory breadboard system are integrated with other supporting elements in a simulated operational environment. How does the "relevant environment" differ from the expected operational environment? How do the test results compare with expectations? What problems, if any, were encountered? Was the breadboard system refined to more nearly match the expected system goals?
6	System/subsystem model or prototype demonstration in a relevant environment.	Representative model or prototype system, which is well beyond that of TRL 5, is tested in a relevant environment. Represents a major step up in a technology's demonstrated readiness. Examples include testing a prototype in a high-fidelity	Results from laboratory testing of a prototype system that is near the desired configuration in terms of performance, weight, and volume. How did the test environment differ from the operational environment? Who performed the tests? How did the test compare with expectations? What problems, if any, were encountered? What are/were the plans, options, or

TRL Definitions, Descriptions, and Supporting Information (Continued)

TRL	Definition	Description	Supporting Information
		laboratory environment or in a simulated operational environment.	actions to resolve problems before moving to the next level?
7	System prototype demonstration in an operational environment.	Prototype near or at planned operational system. Represents a major step up from TRL 6 by requiring demonstration of an actual system prototype in an operational environment (e.g., in an aircraft, in a vehicle, or in space).	Results from testing a prototype system in an operational environment. Who performed the tests? How did the test compare with expectations? What problems, if any, were encountered? What are/were the plans, options, or actions to resolve problems before moving to the next level?
8	Actual system completed and qualified through test and demonstration.	Technology has been proven to work in its final form and under expected conditions. In almost all cases, this TRL represents the end of true system development. Examples include developmental test and evaluation (DT&E) of the system in its intended weapon system to determine if it meets design specifications.	Results of testing the system in its final configuration under the expected range of environmental conditions in which it will be expected to operate. Assessment of whether it will meet its operational requirements. What problems, if any, were encountered? What are/were the plans, options, or actions to resolve problems before finalizing the design?
9	Actual system proven through successful mission operations.	Actual application of the technology in its final form and under mission conditions, such as those encountered in operational test and evaluation (OT&E). Examples include using the system under operational mission conditions.	OT&E reports.

Table 1. TRL Definitions, Descriptions, and Supporting Information (DoD, 2011, pp. 2-13–2-14)

DoD TRLs specifically developed for software technologies are presented in Table 2, alongside the general TRLs previously presented.

Hardware TRL Definitions, Descriptions, and Supporting Information			Software TRL Definitions, Descriptions, and Supporting Information		
TRL Definition	Description	Supporting Information	TRL Definition	Description	Supporting Information
1 *Basic principles observed and reported.*	Lowest level of technology readiness. Scientific research begins to be translated into applied research and development (R&D). Examples might include paper studies of a technology's basic properties.	Published research that identifies the principles that underlie this technology. References to who, where, when.	1 *Basic principles observed and reported.*	Lowest level of software technology readiness. A new software domain is being investigated by the basic research community. This level extends to the development of basic use, basic properties of software architecture, mathematical formulations, and general algorithms.	Basic research activities, research articles, peer-reviewed white papers, point papers, early lab model of basic concept may be useful for substantiating the TRL.
2 *Technology concept and/or application formulated.*	Invention begins. Once basic principles are observed, practical applications can be invented. Applications are speculative, and there may be no proof or detailed analysis to support the assumptions. Examples are limited to analytic studies.	Publications or other references that outline the application being considered and that provide analysis to support the concept.	2 *Technology concept and/or application formulated.*	Once basic principles are observed, practical applications can be invented. Applications are speculative, and there may be no proof or detailed analysis to support the assumptions. Examples are limited to analytic studies using synthetic data.	Applied research activities, analytic studies, small code units, and papers comparing competing technologies.
3 *Analytical and experimental critical function and/or characteristic proof of concept.*	Active R&D is initiated. This includes analytical studies and laboratory studies to physically validate the analytical predictions of separate elements of the technology. Examples include components that are not yet integrated or representative.	Results of laboratory tests performed to measure parameters of interest and comparison to analytical predictions for critical subsystems. References to who, where, and when these tests and comparisons were performed.	3 *Analytical and experimental critical function and/or characteristic proof of concept.*	Active R&D is initiated. The level at which scientific feasibility is demonstrated through analytical and laboratory studies. This level extends to the development of limited functionality environments to validate critical properties and analytical predictions using non-integrated software components and partially representative data.	Algorithms run on a surrogate processor in a laboratory environment, instrumented components operating in a laboratory environment, laboratory results showing validation of critical properties.

Hardware TRL Definitions, Descriptions, and Supporting Information			Software TRL Definitions, Descriptions, and Supporting Information		
TRL Definition	Description	Supporting Information	TRL Definition	Description	Supporting Information
4 *Component and/or breadboard validation in a laboratory environment.*	Basic technological components are integrated to establish that they will work together. This is relatively "low fidelity" compared with the eventual system. Examples include integration of "ad hoc" hardware in the laboratory.	System concepts that have been considered and results from testing laboratory-scale breadboard(s). References to who did this work and when. Provide an estimate of how breadboard hardware and test results differ from the expected system goals.	4 *Module and/or subsystem validation in a laboratory environment (i.e., software prototype development environment).*	Basic software components are integrated to establish that they will work together. They are relatively primitive with regard to efficiency and robustness compared with the eventual system. Architecture development initiated to include interoperability, reliability, maintainability, extensibility, scalability, and security issues. Emulation with current/legacy elements as appropriate. Prototypes developed to demonstrate different aspects of eventual system.	Advanced technology development, stand-alone prototype solving a synthetic full-scale problem, or standalone prototype processing fully representative data sets.
5 *Component and/or breadboard validation in a relevant environment.*	Fidelity of breadboard technology increases significantly. The basic technological components are integrated with reasonably realistic supporting elements so they can be tested in a simulated environment. Examples include "high-fidelity" laboratory integration of components.	Results from testing a laboratory breadboard system are integrated with other supporting elements in a simulated operational environment. How does the "relevant environment" differ from the expected operational environment? How do the test results compare with expectations? What problems, if any, were encountered? Was the breadboard system refined to more nearly match the expected system goals?	5 *Module and/or subsystem validation in a relevant environment.*	Level at which software technology is ready to start integration with existing systems. The prototype implementations conform to target environment/interfaces. Experiments with realistic problems. Simulated interfaces to existing systems. System software architecture established. Algorithms run on a processor(s) with characteristics expected in the operational environment.	System architecture diagram around technology element with critical performance requirements defined. Processor selection analysis, Simulation/Stimulation (Sim/Stim) Laboratory buildup plan. Software placed under configuration management. Commercial-of-the-shelf/government-off-the-shelf (COTS/GOTS) components in the system software architecture are identified.

Hardware TRL Definitions, Descriptions, and Supporting Information			Software TRL Definitions, Descriptions, and Supporting Information		
TRL Definition	Description	Supporting Information	TRL Definition	Description	Supporting Information
6 *System/subsystem model or prototype demonstration in a relevant environment.*	Representative model or prototype system, which is well beyond that of TRL 5, is tested in a relevant environment. Represents a major step up in a technology's demonstrated readiness. Examples include testing a prototype in a high-fidelity laboratory environment or in a simulated operational environment.	Results from laboratory testing of a prototype system that is near the desired configuration in terms of performance, weight, and volume. How did the test environment differ from the operational environment? Who performed the tests? How did the test compare with expectations? What problems, if any, were encountered? What are/were the plans, options, or actions to resolve problems before moving to the next level?	6 *Module and/or subsystem validation in a relevant end-to-end environment.*	Level at which the engineering feasibility of a software technology is demonstrated. This level extends to laboratory prototype implementations on full-scale realistic problems in which the software technology is partially integrated with existing hardware/software systems.	Results from laboratory testing of a prototype package that is near the desired configuration in terms of performance, including physical, logical, data, and security interfaces. Comparisons between tested environment and operational environment analytically understood. Analysis and test measurements quantifying contribution to system-wide requirements such as throughput, scalability, and reliability. Analysis of human-computer (user environment) begun.
7 *System prototype demonstration in an operational environment.*	Prototype near or at planned operational system. Represents a major step up from TRL 6 by requiring demonstration of an actual system prototype in an operational environment (e.g., in an aircraft, in a vehicle, or in space).	Results from testing a prototype system in an operational environment. Who performed the tests? How did the test compare with expectations? What problems, if any, were encountered? What are/were the plans, options, or actions to resolve problems before moving to the next level?	7 *System prototype demonstration in an operational high-fidelity environment.*	Level at which the program feasibility of a software technology is demonstrated. This level extends to operational environment prototype implementations, where critical technical risk functionality is available for demonstration and a test in which the software technology is well integrated with operational hardware/software systems.	Critical technological properties are measured against requirements in an operational environment.
8 *Actual system completed and qualified through test and demonstration*	Technology has been proven to work in its final form and under expected conditions. In almost all cases, this TRL represents the end of true system development. Examples include developmental test and evaluation (DT&E) of the system in its intended weapon system to determine if it meets design specifications.	Results of testing the system in its final configuration under the expected range of environmental conditions in which it will be expected to operate. Assessment of whether it will meet its operational requirements. What problems, if any, were encountered? What are/were the plans, options, or actions to resolve problems before finalizing the design?	8 *Actual system completed and mission qualified through test and demonstration in an operational environment.*	Level at which a software technology is fully integrated with operational hardware and software systems. Software development documentation is complete. All functionality tested in simulated and operational scenarios.	Published documentation and product technology refresh build schedule. Software resource reserve measured and tracked.

Hardware TRL Definitions, Descriptions, and Supporting Information			Software TRL Definitions, Descriptions, and Supporting Information		
TRL Definition	Description	Supporting Information	TRL Definition	Description	Supporting Information
9 *Actual system proven through successful mission operations.*	Actual application of the technology in its final form and under mission conditions, such as those encountered in operational test and evaluation (OT&E). Examples include using the system under operational mission conditions.	OT&E reports.	9 *Actual system proven through successful mission-proven operational capabilities.*	Level at which a software technology is readily repeatable and reusable. The software based on the technology is fully integrated with operational hardware/software systems. All software documentation verified. Successful operational experience. Sustaining software engineering support in place. Actual system.	Production configuration management reports. Technology integrated into a reuse "wizard."

Table 2. DoD Definitions of Technology Readiness Levels for Hardware and Software (Blanchette, Albert, & Garcia-Miller, 2010, pp. 33–36)

Following the same logic of reducing risk to acceptable levels, the system software would have to achieve level 6 or more preferrably, level 7 before integrating the software technology into a system under development. As Table 2 states, level 6 is the “level at which the engineering feasibility of a software technology is demonstrated. This level extends to laboratory prototype implementations on full-scale realistic problems in which the software technology is partially integrated with existing hardware/software systems” (Blanchette, Albert, & Garcia-Miller, 2010, p.9). At level 7, the software readiness would be at the “level at which the program feasibility of a software technology is demonstrated. This level extends to operational environment prototype implementations, where critical technical risk functionality is available for demonstration and a test in which the software technology is well integrated with operational hardware/software” (Blanchette, Albert, & Garcia-Miller, 2010, p. 36).

THIS PAGE INTENTIONALLY LEFT BLANK

Assessing Maturity and Risk Reduction

> The SEI [Software Engineering Institute] has taken the process management premise, "the quality of a system or product is highly influenced by the quality of the process used to develop and maintain it," and defined CMMs [Capability Maturity Models] that embody this premise. (SEI, 2010b, p. 5)

The Software Engineering Institute (SEI) has been a leader in defining organization and process maturity for developing successful software components. The go on to explain: "CMMs focus on improving processes in an organization. They contain the essential elements of effective processes for one or more disciplines and describe an evolutionary improvement path from ad hoc, immature processes to disciplined, mature processes with improved quality and effectiveness" (SEI, 2010b, p. 5).

One of the primary purposes for assessing the software developer's maturity is to reduce the development risk. The relationship between the CMMI maturity levels to system development risk is fairly straight forward and is reflected in Figure 2.

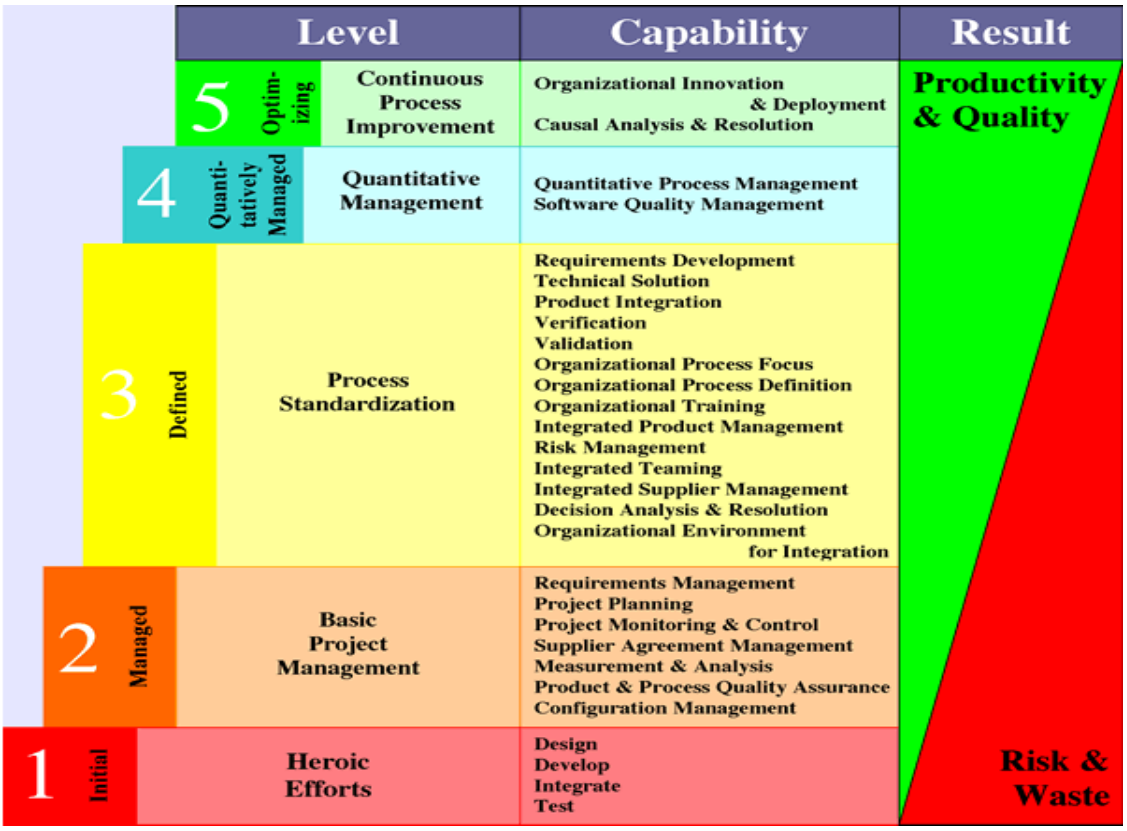

Figure 2. CMMI Five-Level Model and Risk

As depicted in Figure 2, as each higher maturity level is achieved, overall risk and waste is reduced, and for the DoD, acceptable levels of risk reduction have occurred at level 3.

Considering maturity and risk from the software-intensive system acquirer's functions, the SEI has also developed the CMMI for Acquisition (CMMI-ACQ; CMMI, 2010).

> Now, more than ever, organizations are increasingly becoming acquirers of needed capabilities by obtaining products and services from suppliers and developing less and less of these capabilities in-house. This widely adopted business strategy is designed to improve an organization's operational efficiencies by leveraging suppliers' capabilities to deliver quality solutions rapidly, at lower cost, and with the most appropriate technology. (CMMI, 2010, p. 3)

This model accurately describes how the DoD overwhelmingly acquires software-intensive weapon systems.

Similar to the other Capability Maturity Models, the CMMI-ACQ uses a five-level scale as depicted in Table 3.

Level	Focus	Key Process Areas
5	Optimizing	Continuous process improvement, Quantitative Acquisition Management, Continuous Process Improvement, Organizational Performance Improvement
4	Quantitative	Quantitative management, Quantitative Acquisition Management, Quantitative Process Management, Statistical Process Control
3	*Defined*	*Process standardization, Integrated Project Management, Acquisition Technical Management, User Requirements Satisfaction is Defined and Verified, Standard Processes, Proactive Process Management and Detailed Process Measures*
2	Managed	Institutionalized Project Management and Acquisition Engineering, Transition to Support Evaluation, Requirements Development and Management, Configuration Management, Product and Process Quality Assurance, Process Metrics, Acquisition Team Training
1	Initial	Competent people and heroics, Over Commitment, Abandoned Processes

Table 3. CMMI-ACQ Five Levels (CMMI, 2010, pp. 29–31)

The same concept of reducing risk as the maturity level increases applies to the CMMI-ACQ. Currently, there is no DoD requirement for any program management office to achieve a specified level of CMMI-ACQ maturity, leaving that decision to the individual organization. There are certain requirements to achieve the "competent" portion of Level 1, including educational and training certifications for key positions within the PM team, including the PM.

THIS PAGE INTENTIONALLY LEFT BLANK

The DoD Software-Intensive Risk Management Problem and Research Technique

Problem

As this research is the latest in a series of research on the subject, the overarching problem statement remains the same. From a systems management perspective, the problem is that the DoD Acquisition Management System produces both successful and unsuccessful software-intensive systems. The management oversight, structure, and discipline offered do not produce repeatable success in complex, software-intensive systems development.

This research focuses primarily on risk management approaches in software intensive systems. Software Technology Readiness Levels (SwTRL), software producer maturity levels, and PMO maturity levels are addressed.

Primary Research Question

The problem identified above drives this primary research question: Why do the DoD Acquisition Management System risk management processes produce both successful and unsuccessful software-intensive systems?

Secondary Research Questions

I analyze the DoD software-intensive system development challenge by addressing these secondary research questions:

- How effective are the DoD software TRLs in reducing the software development risks?
- What are the anticipated benefits of the software contractor's achieving higher levels of maturity through the CMMI?
- Could the DoD benefit from requiring its own software-intensive system program management offices to achieve higher levels of maturity through the CMMI-ACQ?

THIS PAGE INTENTIONALLY LEFT BLANK

Software Technology Readiness Level Analysis

> To date, efforts by the Services to interpret the DoD software TRLs have been restrained by the requirement to retain the basic definitions when creating guidance. Though marginally useful, these efforts have only confirmed for the participants the futility of continuing to base readiness decisions for software aspects of systems on the DoD software TRLs. (Blanchette, Albert, & Garcia-Miller, 2010, p. 2)

As the quote above suggests, there is significant frustration with using the software TRLs in the same fashion that hardware-related TRLs are used to reduce the developmental risks. This section of the research provides analyses regarding the software TRLs.

Reviewing the SwTRLs presented earlier, achieving level 7 would be the desired software state before initiating a Program Of Record (POR). At the SwTRL level 7, the description reads,

> Level at which the program feasibility of a software technology is demonstrated. This level extends to operational environment prototype implementations, where critical technical risk functionality is available for demonstration and a test in which the software technology is well integrated with operational hardware/software. (Blanchette, Albert, & Garcia-Miller, 2010, pp. 33–36)

This clearly suggests that the actual software that would be used on a proposed system already exists and has been tested in the relevant environments.

In the overwhelming majority of cases, there simply is no reusable software in existence that could satisfy the TRL objectives. As I have documented in my previous research, the software engineering environment is not mature. There are no widely accepted, industry-wide standards for software languages, formats, architectures, tools or protocols (Naegle,2014). Without these, there cannot be useful repositories of reusable software, ready to be inserted into the architecture of proposed weapon systems. This fact is one of the sources for the frustration echoed in the quotation provided at the beginning of this section.

So what can be used as a SwTRL surrogate for the nonexistent software? Most times, a recent, similar system is used to estimate the technolgy readiness of the software in the proposed system. The premise is that the existing system's architecture, complexity, and functions are similar enough to fairly accurately predict the software development resources required for the new system. Unfortunately, this technique has proven to be ineffective as evidenced by the F-22 Raptor development and the follow-on F-35 Joint Strike Fighter (JSF) effort. The two high-performance, supersonic aircraft, have overlapping missions, are significantly

similar, and are both developed by the same contractor. The F-22 would seem to be a very good predictor of the F-35 software development effort with the SwTRL model, but it clearly was not:

> The lines of code necessary for the JSF's capabilities have now grown to over 24 million—9.5 million on board the aircraft. By comparison, JSF has about 3 times more on-board software lines of code than the F-22A Raptor and 6 times more than the F/A-18 E/F Super Hornet. This has added work and increased the overall complexity of the effort. The software on-board the aircraft and needed for operations has grown 37 percent since the critical design review in 2005. … Almost half of the on-board software has yet to complete integration and test – typically the most challenging phase of software development. (GAO, 2012, p. 11)

The report goes on to state that typical software size growth in DoD systems development ranges from 30% to 100%.

JSF design changes were originally supposed to taper off and be completed by January 2014. Actual design changes through September 2011 failed to taper off and continue at a significantly high rate. The projections in the GAO (2012) report indicated that the revised design change projections would continue and actually grow in number, until January 2019 (p. 16). Given this level of redesign, the software and system complexity growth are likely to continue.

In addition to the software growth problems experienced in the F-35, software architecture and performance problems indicate that the use of F-22 SwTRLs was inneffective. The Director of Operational Test and Evaluation (DOT&E), Michael Gilmore, has assessed the software demonstrations on the F-35 Program:

"DOT&E Gilmore's Verdict on F-35 Software Demonstrations

- Block 2F for Marine Corps F-35B IOC in 2015 delivered with "hundreds of deficiencies"
- Block 3i for Air Force F-35-A IOC in 2016 "problematic," and performing poorly in development test
- Block 3F to complete F-35 system development in 2017 "demonstrating poor performance"
- Block 4, first post-service-entry upgrade, too aggressive and under-resourced" (Sweetman, 2016, p. 1)

The DOT&E findings and the amount of software growth in the lines of code experienced by the F-35 program are certainly indicators that using the F-22 as a surrogate for assessing software technology readiness was not effective in this case. Considering how similar these two systems are and the fact that they are manufactured by the same contractor, the validity of using this approach is questionable at best.

Software Technology Readiness Conclusion

Addressing the research question, how effective are the DoD software TRLs in reducing the software development risks?

Software TRLs are significantly ineffective in reducing software developmental risk in DoD weapons system development. The actual system software is overwhelmingly unavailable to perform the software TRL analysis, since it has not been built when the assessment needs to be made. Using a like system as a surrogate for the assessment appears to be highly unreliable, as well.

Considering how much system functionality is dependent on software, a system TRA is significantly impacted by not having a reliable software TRL as part of the assessment. This increases the developmental risk exponentially.

With no viable alternative for software TRLs, DoD software-intensive weapon system development must proceed with a key technology, software, which must be managed as a new, immature technology in nearly every case.

Capability Maturity Model Risk Reduction Analysis

> **CMMI: Major Benefits to DoD**. "Does CMMI work?" We asked our nation's defense contractors, as well as governement agencies, to share results from their performance improvement efforts using CMMI. The results spoke for themselves: "***Yes, CMI works!***" (SEI, 2010a, p. 5)

The above quote is from a Software Engineering Institute report titled *Benefits of CMMI Within the Defense Industry*. This report documents both quantitative and qualitative improvements by defense contractors that have achieved higher levels of CMMI Development (CMMI-DEV) maturity. The results reported are dramatic improvements in nearly every category examined including

- 6.35 times less defect discovery and repair hours after start of system testing; potential savings of 5–6.5 months in schedule delay after systems tests begin for average sized project (p. 7)
- 105.3 fewer hours per defect. 88.6 fewer hours during testing alone (p. 9)
- Percent of defects removed prior to system test: > 85%
- Cost savings to customer in a cost-plus contract: Rose from 5.7 million to 7.1 million (25%; p. 15)
- Organization 2a reported their quantified ROI from CMMI Maturity Level 5 activity to be 24:1 (p. 17)

- Many qualitative benefits reported, including products with lower levels of defects and lower risk; improved program insight, control, and tracking; and personnel retention and job satisfaction. (Benefits of SEI, 2010a)

It is abundantly clear that moving up the CMMI maturity scale has positive, tangible results for defense contractors and by extension, the DoD entities contracting with them. If the return on investment published is realizable, there is ample incentive for the DoD industry partners to achieve and maintain higher levels of CMMI maturity, as the return on investment is extremely significant.

SEI has developed a model for assessing the maturity of the acquiring organizations called Capability Maturity Model Integrated for Acquisition (CMMI-ACQ). The basic concept is that the software-intensive system acquiring organizations are as important to the success of the development as the developing organizations. The system requirements development and communication is the purview of the acquiring organization and is obviously critical to the development process. If you had the best developer in the world and provided that developer with incomplete or wrong requirements, you could end up with a perfectly developed,wrong system, which is still a system failure as far as the user is concerned. The SEI CMMI-ACQ used the depiction shown in Figure 3 to illustrate the concept.

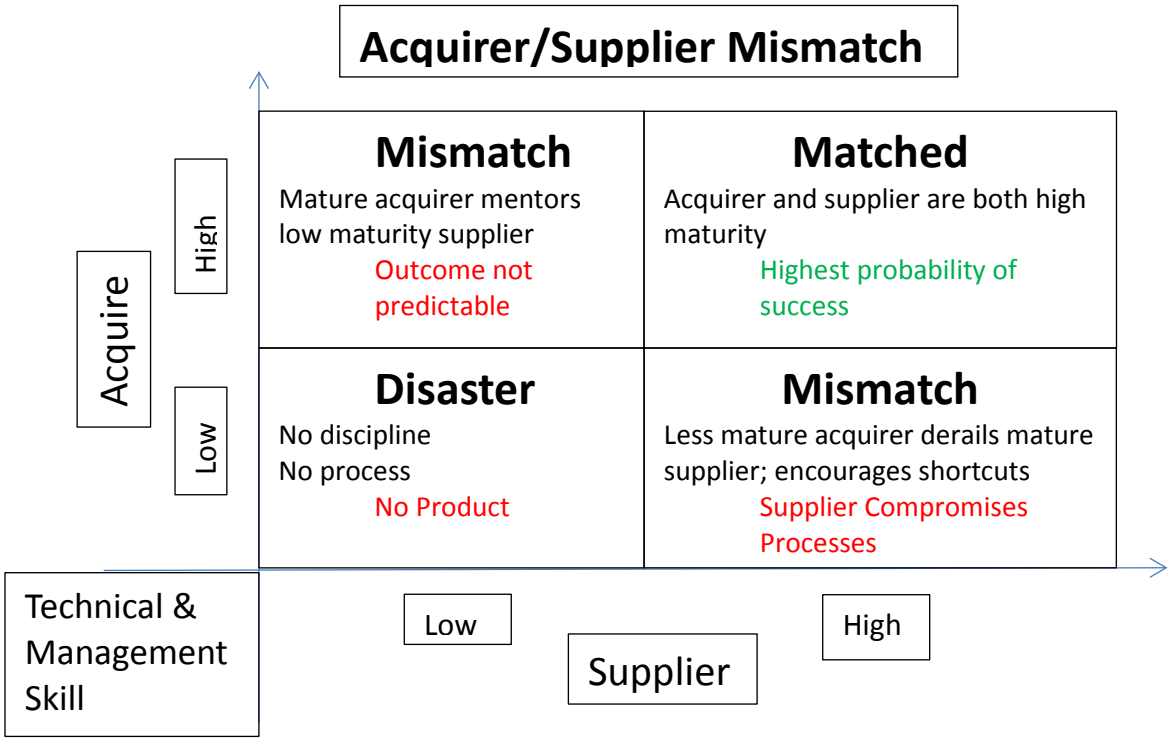

Figure 3. Acquirer/Supplier CMMI Mismatch (Gross & Fisher, 2008, p.21)

So what are the benefits to the DoD for moving up the CMMI-ACQ maturity levels? Brettle (2012) wrote,

> I recently worked with a client on their implementation of CMMI-ACQ. After much hard work and courage, they were rated as a maturity level 2 organisation late last year. Along the way, they accumulated many better practices, benefits and a better understanding of how their own organisation needs to work with partners and suppliers. So after this success, I was curious to know which other organisations had gone down the CMMI-ACQ path and published their appraisal results. I was well and truly shocked to see the numbers from the SEI's PARS [published appraisal report system] displayed on my laptop. In the past 3 years, only 10 organisations globally have invested in CMMI-ACQ to improve their acquisisition of products and services to achieve a maturity level rating of 2 or greater, whilst CMMI-DEV [CMMI for Development] is into the thousands." (p. 1)

As of the writing of this research, the CMMI PARS reports a total of 11 organizations globally with published appraisal reports during the period 2013 through 2015. There was only one U.S.-based contractor, Northrup Grumman Technical Services, in the report, and no U.S. government offices were listed. Of course, this does not mean that none participated or used the CMMI-ACQ for internal evaluation, just that none appear in the published record (SEI, 2016).

It appears that the use of the CMMI-ACQ is not widespread within the DoD, either. Searching for DoD organizations that had undergone and recorded any type of CMMI-ACQ evaluation drew scant results. There was one SEI presentation that indicated that at least one DoD entity, which remained unnamed, had been an early adopter of the CMMI-ACQ in 2008 (Gross & Fisher, 2008, p. 30).

One of the reasons that the DoD has not embraced the use of CMMI-ACQ may have to do with the extensive training and education requirements of the acquisition workforce under the direction of the Defense Acquisition Workforce Improvement Act (DAWIA). The DoD established the Defense Acquisition University (DAU) to set the training and certification process for the workforce, and the DAU has prescribed the training requirements for individuals and designated positions within the acquisition community.

Another reason that the DoD may not embrace the CMMI-ACQ methodologies is that the Defense Acquisition System (DAS) certainly appears to provide ample control mechanisms for the developmental process. The numerous technical reviews, audits, and baselines used to ensure the Systems Engineering Process (SEP), combined with the mandatory reporting requirements for the phases and milestones would at least appear to provide an effective environment for adequate control. The overarching problem is that the DAS produces both

successful and unsuccessful software-intensive systems, indicating that the DAS is not always effective with software-intensive systems development.

Capability Maturity Model Risk Reduction Conclusion

Addressing the research question, what are the anticipated benefits of the software contractor's achieving higher levels of maturity through the CMMI?

On the industry side, the advantages of achieving higher CMMI-DEV maturity levels at least appear to be significant and the positive impacts to the development cycle were dramatic, which provides ample incentive for the DoD's industry partners to invest in the CMMI process improvements. In short, the CMMI-DEV maturity helps them achieve what they must, profitability. DoD entities contracting with those that have achieved higher maturity levels would also benefit from that maturity.

Addressing the research question, could the DoD benefit from requiring its own software-intensive system program management offices to achieve higher levels of maturity through the CMMI-ACQ?

On the DoD side, there does not appear to be much interest in moving up the CMMI-ACQ maturity scale, or in even doing those type of assessments. There, of course, are significant individual qualifications that must be met for the members of the DoD PM team and others that are specific to acquisition positions, as prescribed by the DAU.

The CMMI-ACQ would provide an assessment of the maturity of the PM organization instead of reliance on the individual expertise of each individual within it. Even at CMMI-ACQ level 1, competent people are recognized, but the other processes tend to be ad hoc. This enables the acquirer/supplier mismatch presented earlier, setting up the DoD as the less mature acquirer that derails the mature supplier, compromising the entire process.

Overall Research Conclusion

Addressing the primary research question, why do the DoD Acquisition Management System risk management processes produce both successful and unsuccessful software-intensive systems?

There appear to be several root causes for why DoD risk management is producing both successful and unsuccessful software-intensive systems.

- Most weapon system software is engineered and built from scratch. This happens because the DoD is interested in developing cutting-edge technology systems, and must understand all of the software in-depth to know what cyber security vulnerabilities might exist. The immature software engineering environment makes it nearly

impossible to reuse software components that have been successful in past applications.

- "Due in large part to the immature software engineering environment, each major DoD software design and build tends to be unique. That means that the software development in complex systems will act the same way as integrating a new technology would, and the resulting program risk is very high. The software TRLs have little meaning in this type of environment, so risk management is highly dependent on the Government and software development teams' abilities to manage the system software development as a new technology with a low TRL." (Naegle, 2014, p. 31)
- The TRA using software TRLs is not effective. This leaves software-intensive system technical risk very high for the software component. Essentially, this means each software-intensive system is forced to manage the software development as a new, untried, unproven technology build. In other words, an extremely low software TRL is where they must begin.
- The difference between success and failure in the software-intensive system development is likely to be how effectively the PM organization can apply the DAS process controls, and augment them with other tools, techniques and analyses that improve effectiveness. In this situation, where there is little opportunity to manage risk through technology maturation, the success of the project is dependent on the effectiveness of the processes and the skill of the PM organizations for both the contractor and the government. The DoD's focus on the individual competence and the existing DAS controls available appears to overlook the impact of the organizational effectiveness.

THIS PAGE INTENTIONALLY LEFT BLANK

Recommendations

General

The recommendations are designed to address improving software-intensive system development risk reduction in an environment where the software technical risk cannot practically be assessed or managed. Under these circumstances, the software development must be managed as a new, immature technology by the PM organization.

The following recommendations represent an integrated approach to address the problems presented in all of the associated research efforts. Addressing problem areas within the complex DoD Acquisition System and adding the dynamics of the human component means that recommendations are necessarily complex. Adding to that complexity is the fact that the DAS produces successful software-intensive programs in addition to the unsuccessful efforts. This means that the existing system with its control measures, organizations, and professional workforce can produce a successful program, but does not always do so.

The integrated approach includes the other research conclusions and recommendations. The research conducted earlier was not wholly included in this paper, for brevity.

Software-Intensive System Development Risk Reduction

With no viable means of reducing the software development technical risk, the focus turns to the acquiring organization's ability to effectively lead and manage the development of the system software as a new, critical technology. To focus on the organization as a whole, I recommend management readiness assessments using Management Readiness Levels (MgtRLs) similar to the TRLs used for key technologies. To be consistent, I offer nine MgtRL levels with a goal of achieving level 6 or 7 before risk is sufficiently reduced to proceed with the program.

I introduced a very basic version of the MgtRLs in my previous research effort and expand on those here (Naegle, 2014). The MgtRL approach I use is to combine CMMI maturity with the use of tools, techniques, and analyses previously researched to improve the processes within the DAS. As both the acquirer (government PM) and the supplier (contractor) are critical to the success of a software-intensive system development, supplier maturity is included.

- Level 1: PM team and software developers (contractors) meet all professional certifications and adhere to DoD policy for achieving maturity levels. PM team uses the Defense Acquisition System (DAS) adequately.

- Level 2: PM team has fully developed derived and implied requirements using a QAW or similar approach, and RFP performance specification reflects the total inventory of discoverable requirements. The evaluation includes a MUIRS analysis for sustainability, safety, and cyber security requirements assessment. PM team conducts internal CMMI-ACQ maturity evaluation.
- Level 3: PM team has conducted a self-evaluation and meets the CMMI-ACQ level 2 criteria. Software developer has achieved CMMI-DEV level 2 in an external evaluation.
- Level 4: User representatives agree to use ATAM[sm]; the PM team has conducted at least one iteration of ATAM before RFP release. ATAM development includes MUIRS and FMECA scenarios.
- Level 5: PM team conducts pre and post contract award ATAM iterations, and continues ATAM through the initial program design reviews.
- Level 6: PM team has achieved CMMI-ACQ level 2 in an externally conducted evaluation. Software developer has achieved level 3 (or higher) in an externally conducted CMMI-DEV evaluation.
- Level 7: PM team has achieved CMMI-ACQ level 3 in an externally conducted SA-CMM evaluation. Software developer has achieved level 4 in an externally conducted CMMI-DEV evaluation.
- Level 8: PM team has achieved CMMI-ACQ level 4 in an externally conducted evaluation. Software developer has achieved level 5 in an externally conducted CMMI-DEV evaluation.
- Level 9: PM team has achieved CMMI-ACQ level 5 in an externally conducted evaluation.

Preparing and executing CMMI-ACQ evaluations, especially those performed by external entities, certainly adds to the burden on a PM organization, but the return on investment for achieving higher levels of acquirer maturity would likely justify the resources invested. The tools, techniques, and analyses suggested augment the DAS technical development control mechanisms, focusing on those elements most critical to the software development.

Moving up these suggested MgtRLs, using the proven tools, techniques, analyses, and maturity levels, provides a risk-reducing approach to developing the software as a new, critical technology. The MgtRLs focus on the organizations and include both government and contractor, recognizing the importance of both for successfully developing software-intensive systems within predictable cost and schedule parameters.

References

Barbacci, M., Ellison, R., Lattanze, A., Stafford, J., Weinstock, C., & Wood, W. (2003, August). *Quality attribute workshops (QAWs)* (3rd ed.) (CMU/SEI-2003-TR-016). Pittsburgh, PA: Carnegie Mellon University, Software Engineering Institute.

Blanchette, S., Albert, C., & Garcia-Miller, S. (2010, December*). Beyond technology readiness levels for software: U.S. Army workshop report* (CMU/SEI-2010-TR-044). Pittsburgh, PA: Carnegie Mellon University, Software Engineering Institute.

Brettle, A. (2012, January). CMMI for acquisition—The lost world. *Process Improvement—News and Views* (blog). Retrieved from http://cmmi.net/2012/01/20/cmmi-for-acquisition-the-lost-world-2/#more-1216

Department of Defense (DoD). (2015, June). *Department of Defense risk, issue, and opportunity management guide for defense acquisition programs*. Washington, DC: Author.

Department of Defense (DoD). (2011, April). *Technology Readiness Assessment (TRA) guidance.* Washington, DC: Assistant Secretary of Defense for Research and Engineering (ASD[R&E]). Retrieved from http://www.acq.osd.mil/chieftechnologist/publications/docs/TRA2011.pdf

Defense Acquisition University (DAU). ACQuipedia. Retrieved from https://dap.dau.mil/acquipedia/Pages/ArticleDetails.aspx?aid=df96b20d-4e88-4a61-bb35-898893867250

Government Accountability Office (GAO). (2012, March 20*). Joint Strike Fighter: Restructuring added resources and reduced risk, but concurrency is still a major concern* (GAO-12-525T). Retrieved from http://www.gao.gov

Gross, J., & Fisher, M. (2008, August). *Early adoption experiences with CMMI for acquisition* (CMMI-ACQ). Pittsburgh, PA: Carnegie Mellon University, Software Engineering Institute.

Hagan, C., Hurt, S., & Sorenson, J. (2013, November/December). Effective approaches for delivering affordable military software. *Crosstalk Magazine*, 26–32.

Humphrey, W. (1990, August). *Managing the software process.* Reading, MA: Addison-Wesley.

Kazman, R., Klein, M., & Clements, P. (2000, August). *ATAM[sm]: Method for architecture evaluation* (CMU/SEI-2000-TR-004). Pittsburgh, PA: Carnegie Mellon University, Software Engineering Institute.

Naegle, B. R. (2006, September). *Developing software requirements supporting open architecture performance goals in critical DoD system-of-systems* (NPS-AM-06-035). Monterey, CA: Naval Postgraduate School, Acquisition Research Program.

Naegle, B. R. (2014, December). *Gaining control and predictability of software-intensive systems development and sustainment* (NPS-AM-14-194). Monterey, CA: Naval Postgraduate School, Acquisition Research Program.

Naegle, B. R., & Petross, D. (2007, September). *Software architecture: Managing design for achieving warfighter capability* (NPS-AM-07-104). Monterey, CA: Naval Postgraduate School, Acquisition Research Program.

Software Engineering Institute/Carnegie Mellon. (2007). *The importance of software architecture.* Pittsburgh, PA: Carnegie Mellon University, Software Engineering Institute. Retrieved from http://www.sei.cmu.edu/architecture/index.html

Software Engineering Institute/Carnegie Mellon. (2010a, May). *Benefits of CMMI within the defense industry.* Pittsburgh, PA: Carnegie Mellon University, Software Engineering Institute.

Software Engineering Institute/Carnegie Mellon. (2010b, November). *CMMI for Acquisition, version 1.3.* (CMU/SEI-2010-TR-032) Pittsburgh, PA: Carnegie Mellon University, Software Engineering Institute.

Software Engineering Institute/Carnegie Mellon. (2016). *CMMI Institute Published Appraisal Results (PARS).* Pittsburgh, PA: Carnegie Mellon University, Software Engineering Institute. Retrieved from http://seir.sei.cmu.edu/pml/

Under Secretary of Defense for Acquisition, Technology, and Logistics (USD[AT&L]). (2015, April 9). *Implementation directive for Better Buying Power 3.0—Achieving dominant capabilities through technical excellence and innovation* [Memorandum]. Washington, DC: Author.

Sweetman, Bill. *Testing Chief Warns of JSF Software Delays*, Aviation Week and Space Technology, New york, NY (January 2016).

ACQUISITION RESEARCH PROGRAM
GRADUATE SCHOOL OF BUSINESS & PUBLIC POLICY
NAVAL POSTGRADUATE SCHOOL
555 DYER ROAD, INGERSOLL HALL
MONTEREY, CA 93943

www.acquisitionresearch.net