SYM-AM-17-046

Proceedings
of the
Fourteenth Annual Acquisition Research Symposium

Wednesday Sessions
Volume I

Acquisition Research:
Creating Synergy for Informed Change

April 26–27, 2017

Published March 31, 2017

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 Policies and Economics of Software Sustainment: DoD's Software Sustainment Ecosystem

Forrest Shull—is Assistant Director for Empirical Research at Carnegie Mellon University's Software Engineering Institute. His role is to lead work with the U.S. DoD, other government agencies, national labs, industry, and academic institutions to advance the use of empirically grounded information in software engineering, cybersecurity, and emerging technologies. Dr. Shull has been a lead researcher on projects for the U.S. DoD, NASA's Office of Safety and Mission Assurance, the Defense Advanced Research Projects Agency (DARPA), the National Science Foundation, and commercial companies. He serves on the IEEE Computer Society Board of Governors and Executive Committee. [fjshull@sei.cmu.edu]

Michael McLendon—is an Associate Director, Software Engineering Institute, Carnegie Mellon University. Previously, he served as Senior Advisor, Office of the Assistant Secretary of Defense for Systems Engineering. He also was a principal in the Office of the Assistant Secretary of Defense for Program Analysis & Evaluation and in the Office of the Under Secretary of Defense for Policy. Dr. McLendon was Professor at the Defense Systems Management College and a career Air Force officer in a range of leadership and management positions in system and technology development and acquisition, as well as the federal level and the private sector. [mmclendon@sei.cmu.edu]

Abstract

Software is the foundational building material for the engineering of systems, enabling almost 100% of the integrated functionality of cyber physical systems—especially mission- and safety-critical software reliant systems—to the extent that these systems cannot function without software. As a result, it is imperative that the DoD has the capability and capacity to affordably sustain software-reliant systems and to continually operate and achieve mission success in a dynamic threat, cybersecurity, and net-centric environment.

The Carnegie Mellon University (CMU) Software Engineering Institute (SEI) has been performing studies to inform Departmental decisions regarding software sustainment policies and programs regarding complex weapon systems. These studies were based on interviews and discussions with sustainment centers across all of the Services, case studies on selected programs, and a literature review.

In this paper we present an overview of our initial study regarding the DoD's organic software sustainment infrastructure and its key components related to complex weapon systems, and a selection of key themes from our analysis of sustainment practices. There are two key takeaway messages. First, software sustainment is not effectively described with a model based on hardware (where sustainment can be treated as a discrete series of activities intended to restore form, fit, and function). Secondly, software sustainment is really about continuous engineering in which the software undergoes a series of engineering activities intended to deliver the latest capability to the warfighter, a task which is never "done."

Motivation

Software is the foundational building material for the engineering of systems, enabling almost 100% of the integrated functionality of cyber-physical systems—*especially* mission- and safety-critical software-reliant systems—to the extent that these systems cannot function without software. There is no plateau in sight for the advancement of software technology and its use by the DoD in new systems, as well as to enhance the capabilities of legacy systems and extend their operational value far beyond their designed service life.

Software can also be a major a source of defects and potential security vulnerabilities with potentially fatal consequences, due to the increased complexity of interactions among embedded software, the hardware platform, and its associated subsystems. The dynamics of the cyber environment and the constantly changing nature of the cyber threat mean that with software we are never "done."

The issues surrounding sustainment become increasingly complex as the DoD's reliance on software increases. For example, the ever-expanding reliance on software means that an increasing portion of the acquisition cost as well as the sustainment cost of systems is driven by software (NRC, 2010a). There is evidence that it is three to 10 times more expensive to mitigate software defects/vulnerabilities in sustainment rather than early in acquisition and development. Successfully mitigating this software cost trend, while still enhancing warfighter capability, must be an essential element in the DoD's affordability strategy.

Therefore it is imperative that software sustainment be a priority in defining system requirements, design, and development. This means that the software sustainment community must be an active participant early in the requirements and engineering process and that the Product Support Manager in acquisition programs must be knowledgeable and proactive in representing software sustainment equities.

Another critical challenge is the magnitude of the DoD's software sustainment inventory. The inventory is immense, but there is limited visibility and understanding at the enterprise level of the total size, complexity, and characteristics of the DoD's software inventory, which may be trending toward one billion lines of custom developed software code or more.[1] Additionally, the engineering of systems also relies on an extensive portfolio of commercial-off-the-shelf (COTS) software, government-off-the-shelf (GOTS) software, and increasingly on free and open-source software (FOSS). The use of non-custom development software is pervasive across all DoD system domains and its use comes with significant technical and management challenges. The DoD relies on comprehensive and complex information systems to provide almost real time visibility and management of its wholesale and retail inventory of parts and supplies, but it has no similar capability for software.

It is imperative that the DoD have the capability and capacity to affordably acquire and sustain software-reliant systems to continually operate and achieve mission success in a dynamic threat, cyber, and net-centric environment. However, the DoD's ability to produce high-quality software more affordably and efficiently across the system life cycle is a strategic challenge (NRC, 2010a). The acquisition and sustainment of software, particularly for distributed real-time and embedded systems, remains high risk and more problematic as system complexity continues to grow.

Research Goal, Scope, and Methodology

Our research goal was to *characterize* the factors that affect the effectiveness and cost of software sustainment in the DoD. For this initial step of our research, we did not intend to produce value judgments as to the efficacy or cost-effectiveness of different

[1] This is an estimate based on the limited data available and expert judgment. However, we note that several experts in the DoD software engineering community have expressed the opinion in discussions that the number may be even higher.

sustainment choices; rather, we intended to identify and describe the major factors that DoD organizations must manage, in order to impact software sustainment performance.

To provide a manageable focus for our work, we concentrated on describing the software sustainment ecosystem as it relates to complex weapon systems. Embedded software presents the most technically difficult and resource-intensive software engineering challenge because of tightly coupled interfaces, integration with unique hardware, real-time requirements, and very high reliability and assurance needs due to life-critical and mission-critical demands. Of course, the DoD's software sustainment challenge is broader than the software embedded in complex weapon systems. Mission critical non-embedded systems, mission support systems (e.g., test equipment, mission planning, engineering models, and simulations), and the range of business systems also present significant software sustainment challenges. While our initial results can be used to understand the software sustainment ecosystem for other types of software intensive systems, a more detailed description of how the factors apply to those domains will be a subject of future work.

Our team leveraged multiple streams of data and information for this study.

- Literature Search—The body of knowledge related to software engineering is extensive. The formal body of knowledge, academic research, and practitioner publications has evolved so that there are now various communities of interest and professional organizations that focus on software engineering.[2] However, there has been limited systematic study focused on DoD software sustainment, so there is no organized set of literature and ongoing study or research agenda to create and refresh a software sustainment body of knowledge.
- SEI DoD Engagements—The SEI has been actively engaged with the military services for three decades to provide technical expertise to enhance organizational capabilities (processes, practices, and competencies) for software engineering across the life cycle and to solve technical challenges for specific weapon system and information system programs. This has included continuous engagement with the principal Army, Navy, and Air Force software sustainment centers to include the provision of knowledge and practice to institutionalize CMMI.
- DoD IPT Report—The SEI had access to the data and results of the DoD UAS Software Sustainment Integrated Product Team (IPT) Report. This IPT effort, which concluded in 2015, was led by the office of the DASD (MP&P). The SEI was invited to serve as an ex officio member of the IPT, which made visits to a number of Army, Navy, and Air Force organic software sustainment organizations.
- Interviews with Key Leaders—It was critical that the SEI inform its analysis based on the views of decision-makers who influence a range of software

[2] For example, the Software Engineering Body of Knowledge (SWEBOK), a community-driven approach using an open consensus model to document generally accepted software engineering knowledge under the leadership of the IEEE Computer Society (https://www.computer.org/web/swebok). A DoD-specific example is the Software Assurance Community of Practice.

> sustainment policies, programs, and resource allocation. To that end, the SEI complemented its research with information from meetings with key leaders across all three Services, including those in the Senior Executive Service (SES), senior managers and staff in OSD, and from industry. This study was conducted at the unclassified level, and our interviews with DoD sustainment staff were conducted under the conditions of non-attribution to enable an open exchange of perspectives with senior leaders, managers, and staff engaged in software sustainment. These conversations provided context for understanding the evolution of the DoD’s current software sustainment posture and enabled the SEI to refine its model of the software sustainment ecosystem.

DoD’s Organic Software Sustainment Organizational Infrastructure

The DoD’s organic software sustainment organizations successfully respond to a range of customer needs and deliver critical software updates and enhancements, often under the intense schedule pressure of wartime operations, to deliver critical warfighter capability. This organic infrastructure is composed of a number of principal organizations and a myriad of other smaller organizations and offices that have not been fully identified and characterized. To a large degree, in our view, the critical role and functions of these organizations are not well understood or visible. These organic organizations are structured and resourced in different ways by each Service, each performing software sustainment utilizing a variety of government and contract staffing strategies. The Services employ a variety of business model strategies in making decisions about allocating sustainment workload across their organic software development capabilities and the defense industrial base (DIB), as well as structuring public–private sector partnerships. These decisions are made within the context of a number of statutory requirements and DoD policies, such as determination of core requirements and the 50% ceiling, measured in dollars, on the amount of depot maintenance workload that may be performed by a contract with industry for a military department or defense agency during a fiscal year.

The Nature of Software in Systems

One of the keys to addressing the software sustainment challenge is to understand the nature of software in DoD systems. The characteristics of software relative to hardware are generally not well appreciated, especially in relation to how the DoD traditionally uses the term maintenance.

The critically important and growing role of software in defense systems has been noted in many prior studies (DSB, 2000; NRC, 2010a, 2010b). This growth is due in many ways to the unique characteristics of software, as summarized eloquently in a study by the National Academy of Sciences:

> This growth is a natural outcome of the special engineering characteristics of software: Software is uniquely unbounded and flexible, having relatively few intrinsic limits on the degree to which it can be scaled in complexity and capability. Software is an abstract and purely synthetic medium that, for the most part, lacks fundamental physical limits and natural constraints. For example, unlike physical hardware, software can be delivered and upgraded electronically and remotely, greatly facilitating rapid adaptation to changes in adversary threats, mission priorities, technology, and other aspects of the operating environment. The principal constraint is the human intellectual

> capacity to understand systems, to build tools to manage them, and to provide assurance—all at ever-greater levels of complexity. (NRC, 2010a)

These aspects of software are not always well understood or at least addressed in practice. For example, much of DoD depot policy (and industrial base policy) remains hardware-centric, despite software enabling an increasingly large percentage of system functionality. Due to its “uniquely unbounded and flexible” nature, the sustainment of software operates very differently from that of other building materials of contemporary systems.

Software is not a “physics of failure” domain, which is to say that software itself does not wear out or degrade over time. Maintenance at any organizational level for hardware typically focuses on returning components categorized as repairable items, such as avionics line-replaceable units (LRUs), to their original functional condition and configuration by replacing parts, using smaller electronic components, or treating corrosion. This typically involves applying standardized processes and procedures for diagnostics and repair. In the case of software, sustainment takes the form of making intentional changes to the software source code and related work products for many different reasons, not exclusively (or in many cases even primarily) driven by correction of failures. These changes are driven by a number of goals, such as to correct a flaw, to mitigate a security vulnerability, to make fact-of-life changes due to systems and system-of-system interface and interoperability impacts, and to incorporate system enhancements that deliver greater warfighter capability.

Demand and funding requirements for software sustainment do not scale with operational tempo or the size of the force structure. From a hardware or weapon system platform perspective, depot maintenance or sustainment demands and funding are routinely forecasted on the basis of the number of reparable units anticipated, taking into account certain factors such as reliability, flying hours, miles driven, engine hours, number of landings, or calendar time since last overhaul. From a software perspective, sustainment is about applying the disciplines of systems and software engineering (knowledge, processes, practices, and skills) each time the software is touched.

Due to the complexity of software, the great majority of the software sustainment effort is spent on the analysis of the specific need for a change and then designing, implementing, and testing a unique change. Once implemented, it is trivial to make additional copies of the new configuration version of the software system, and generally it is inexpensive to push out updates for all the instances of the weapon system in the force structure inventory. Further, the number of a particular type of weapon system that is in the force structure is not the driver of software sustainment. In other words, relying on a “cost per asset” analysis can be hugely misleading. Since costs are independent of the quantity of a given system in the inventory or force structure, dividing over a fairly small fleet like the B-2s is a misleading comparison with other systems. Another factor to consider is the differences in complexity of systems and the associated complexity of sustainment from system to system. Addressing cybersecurity issues is another distinction.

Software quality is related not only to operational failures but also to technical debt—that is, the reflection of inadequate attention to the design of the software architecture coupled with developers optimizing short-term goals (like the ability to deliver code on time) over longer term impacts (such as the need to create clean, well-organized code that is easy to maintain). Technical debt, as generally understood, affects the internal quality of the code and its extensibility to more easily accommodate change. However, technical debt does not necessarily impact behaviors of the software that would be visible to the end user. Technical debt's impact on architecture and internal software quality directly affects the scope, magnitude, and complexity of software sustainment.

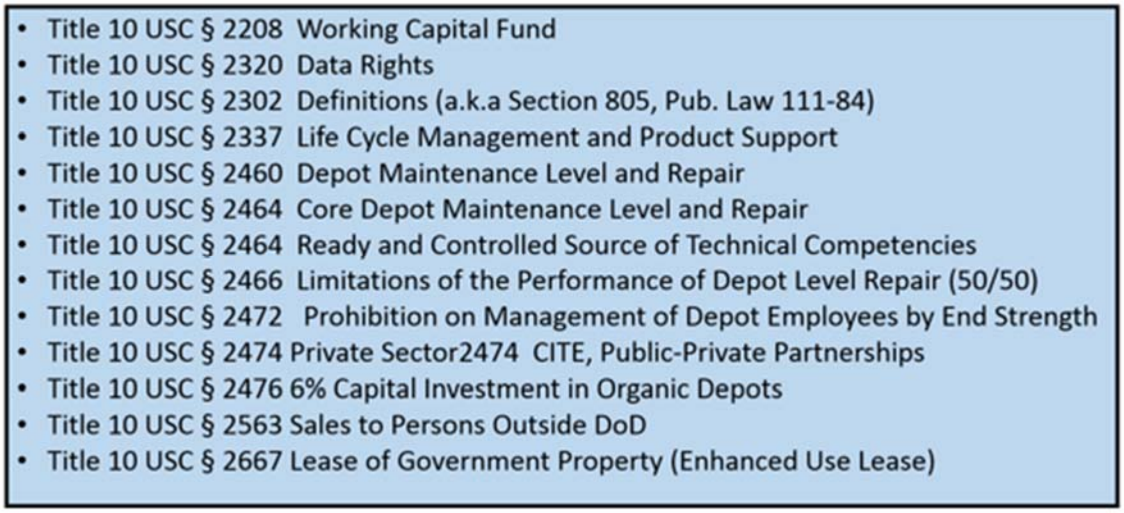
The scope and complexity of the technical debt in an individual program is also driven by the complexity of the software supply chain. The number of different suppliers in the software supply chain for a weapon system program can be extensive. A program often has limited visibility and understanding of the architecture considerations and software practices (not only for development but for assurance as well) that each vendor employs and the degree to which there is a consistent approach to software development for the entire program. From a software sustainment perspective, organic sustainment organizations inherit the cumulative technical debt generated from the multiplicity of software development efforts on one program.

An implication of the points highlighted above is that software sustainment is more usefully viewed as *continuous engineering* rather than a set of discrete maintenance activities. Software sustainment enables an ongoing evolution of system capability to address the changing environments in which DoD systems are deployed, especially related to ongoing changes in cyber threats.

Policy Context

Software sustainment organizations plan and execute their missions within the context of the existing depot maintenance and associated governance environment. As highlighted in Figure 1, the DoD, and in turn the Services, promulgate depot maintenance policy and guidance based on a number of statutory requirements. These mandates then drive decisions relative to planning and executing software sustainment.

The overall direction and guidance for software sustainment are based on statutory requirements in Title 10 USC and DoD policies for depot-level maintenance. Figure 1 summarizes the relevant Title 10 USC statutes that influence product support and depot maintenance decisions. As a result, the legacy of the DoD's depot maintenance paradigms and policies is rooted in a hardware-centric paradigm. In turn, each Service has developed its own guidance for implementing the DoD's policy to address Service software sustainment needs within the depot maintenance framework.



- Title 10 USC § 2208 Working Capital Fund
- Title 10 USC § 2320 Data Rights
- Title 10 USC § 2302 Definitions (a.k.a Section 805, Pub. Law 111-84)
- Title 10 USC § 2337 Life Cycle Management and Product Support
- Title 10 USC § 2460 Depot Maintenance Level and Repair
- Title 10 USC § 2464 Core Depot Maintenance Level and Repair
- Title 10 USC § 2464 Ready and Controlled Source of Technical Competencies
- Title 10 USC § 2466 Limitations of the Performance of Depot Level Repair (50/50)
- Title 10 USC § 2472 Prohibition on Management of Depot Employees by End Strength
- Title 10 USC § 2474 Private Sector2474 CITE, Public-Private Partnerships
- Title 10 USC § 2476 6% Capital Investment in Organic Depots
- Title 10 USC § 2563 Sales to Persons Outside DoD
- Title 10 USC § 2667 Lease of Government Property (Enhanced Use Lease)

Figure 1. Title 10 USC Laws Influencing Sustainment

The DoD's Depot Source of Repair (DSOR) practices also drive software sustainment decisions. In practice, the Program Manager conducts a level of repair analysis (LORA) to determine if there is a depot level requirement (DAU, 2017). The PM also conducts a core logistics assessment (CLA) to determine in accordance with Title 10 USC § 2464 if there is a requirement to establish an organic (core) depot maintenance capability (i.e., government owned and government operated [GOGO]). This practice evolved from a focus on hardware and is now applied to software.

A key factor that drives software sustainment is the program manager-centric nature of decisions about sourcing strategies for the sustainment of specific weapon system programs. These program manager decisions ripple through and impact virtually every component of the software sustainment ecosystem. These program-specific policy decisions, in our view, may not be balanced with considerations for optimizing the DoD software sustainment enterprise to contribute to greater enterprise affordability and productivity.

The Software Sustainment Ecosystem Factors

Based on our findings, we believe that the software sustainment infrastructure is best described and understood as an *ecosystem* composed of interrelated elements. We found over and over that the factors that drive software sustainment are highly interrelated. For example, it is difficult to discuss the workforce needed to perform necessary sustainment activities without first understanding the business model in terms of public-private partnerships, which activities can be done by contractors, and which activities need to remain in the organic DoD workforce. Decisions about the nature and types of these business models may also be influenced by the degree to which the government has provisioned for and exercised its technical data rights for a given program at the time of developing an acquisition strategy and contract. These decisions have implications for the scope of the software sustainment system. Because of the high degree of connectivity that exists among the drivers and factors, we use the metaphor of an "ecosystem" to describe the interdependencies among these elements; decisions made at any point are affected by and affect whole series of other decisions.

There are many variables that are inherent in this ecosystem, not the least of which is time. The time variable is one of the key factors that makes this ecosystem dynamic. There is a time dependency among and between certain software sustainment demand drivers and the critical elements. For example, demands for software changes are frequent, and the underlying technology of software changes rapidly. Failure to invest in software quality up front during initial system development creates a bow wave of risk and technical debt that may continue for decades. Similarly, inadequate investments early in software workforce capital, tools, and engineering processes will increase the cost of sustainment. *In operation, the software sustainment ecosystem is dynamic.*

Based on our research, we created a framework that describes the software sustainment ecosystem, depicted in Figure 2. We abstracted the issues raised in our discussions with DoD sustainment stakeholders into six *demand drivers* and 10 *ecosystem elements.*

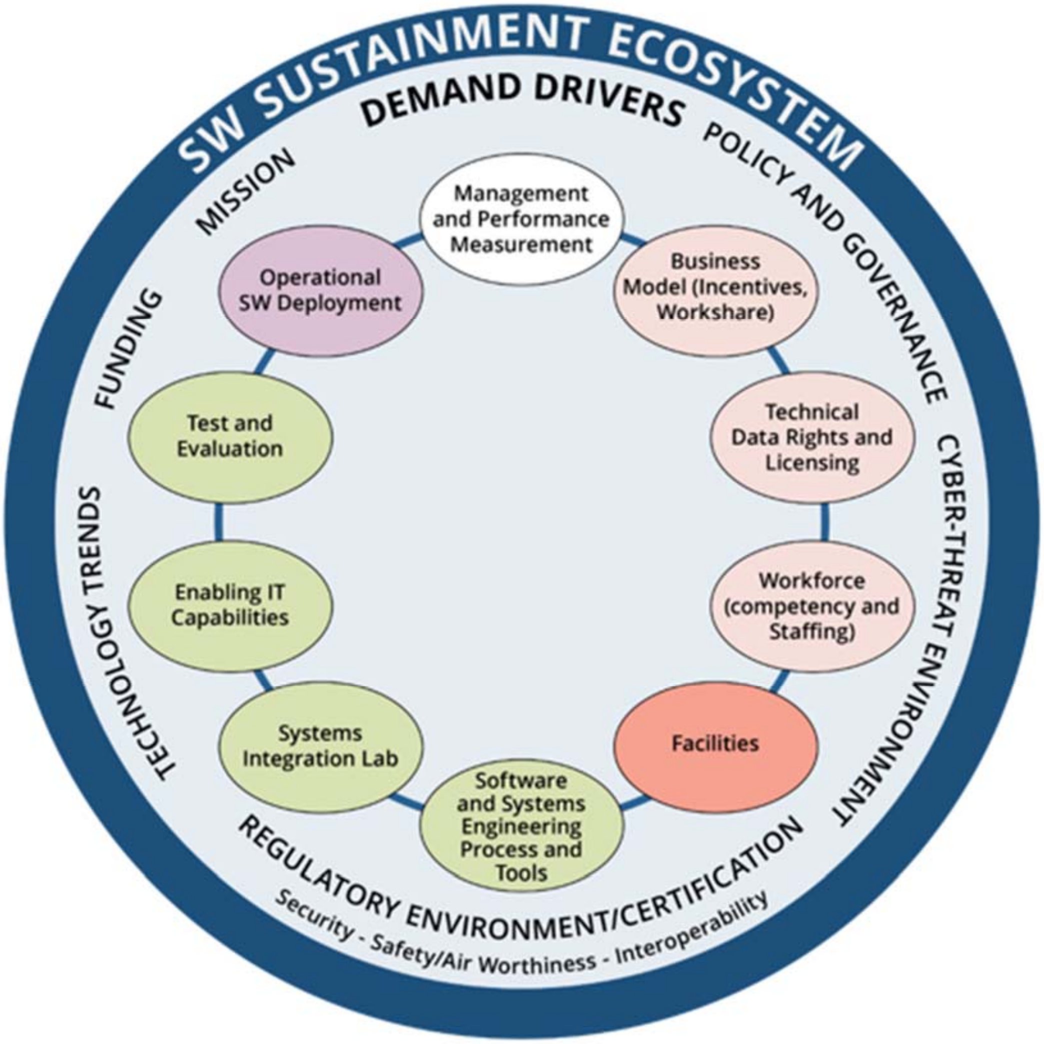

SW SUSTAINMENT ECOSYSTEM
DEMAND DRIVERS
MISSION
POLICY AND GOVERNANCE
FUNDING
CYBER-THREAT ENVIRONMENT
TECHNOLOGY TRENDS
REGULATORY ENVIRONMENT/CERTIFICATION
Security - Safety/Air Worthiness - Interoperability
Management and Performance Measurement
Business Model (Incentives, Workshare)
Technical Data Rights and Licensing
Workforce (competency and Staffing)
Facilities
Software and Systems Engineering Process and Tools
Systems Integration Lab
Enabling IT Capabilities
Test and Evaluation
Operational SW Deployment

Figure 2. The DoD Software Sustainment Ecosystem Framework

These six **demand drivers** capture the fact that DoD systems exist in an environment that is highly dynamic, where there is a need to respond to constantly changing threats and mission needs. This dynamism drives many of the system changes that need to be made during software sustainment. For many of these changes, the most cost-effective way of implementing the new capability relies on the unique flexibility of software.

Our work with the DoD software sustainment community continually highlighted an array of what some called constraints, factors, or “outside the organization influences” that directly impact software sustainment planning and execution. We mapped these considerations into the six higher level demand drivers. These demand drivers include policy (which may include formal guidance such as the Defense Acquisition Guidance and standards) and governance, the nature of the mission, the cyber and mission threat environment, funding, technology trends, and regulatory/certification requirements.

The 10 **ecosystem elements**, shown as interconnected “bubbles” within Figure 2, are the tightly interconnected factors that sustainment organizations need to manage in order to effectively and continuously engineer the software. The drivers and elements of this ecosystem represent a virtual spider web of linkages and relationships. The ecosystem elements shown in the figure are as follows:

The four **infrastructure** elements are the basic, fundamental resources that are necessary for the sustainment activities to occur.

- **Systems and Software Engineering Process and Tools—**The engineering practices to be applied to plan and execute the work.
- **Enabling IT Infrastructure—**The information technology environment and assets upon which the work must be conducted.
- **Test and Evaluation (T&E)—**The mechanisms by which changes made during software sustainment are verified as ready to be rolled out to users. For DoD weapons systems, significant investments in program-specific hardware may be required.
- **Systems Integration Laboratory (SIL)—**The SIL is a specific type of T&E equipment, providing accurate analysis of the impact of changes, and is increasingly important to DoD sustainment practice.

The three **knowledge and expertise** elements include the factors that describe how the necessary skill sets are brought to bear for sustainment activities and how the government grows its organic workforce and gets access to necessary technical information—perhaps with some level of interaction with the private sector—in order to deliver and deploy the capabilities that need to go to the warfighter.

- **Workforce (Competency and Staffing—**The means of accessing a sufficient organic workforce with appropriate skill sets.
- **Business Model (Incentives, Workshare)—**The strategic decision regarding which parts of the work will be done by the organic workforce and which by contractors, and how the overall work is managed both technically and contractually.
- **Technical Data Rights and Licensing—**The tactical decisions governing what technical information is necessary to be accessed by the organic workforce, and the mechanisms by which they have access.

Three ungrouped elements complete the ecosystem.

- **Facilities—**The physical location that meets the needs of the work (providing sufficient space, security levels, etc.).
- **Operational Software Deployment—**The mechanisms and strategy by which new versions of the software under sustainment are delivered to users.
- **Management and Performance Measurement—**The management function necessary to organize and monitor the work being conducted to ensure that it is executing as planned, and to identify any problems that need to be resolved.

Conclusion

The DoD's ability to continually evolve warfighter capability to address the dynamics of the vulnerability and thread environment is driven more and more by the affordable and timely continuous engineering of a system's software. However, there has been limited enterprise visibility and management of the DoD's critical organic software sustainment infrastructure. This paper provides insights into this complex issue, and we expect to provide more detailed information describing the software sustainment ecosystem when our report is cleared for broader distribution.

We also hope that the current summary can be useful for multiple stakeholders in the DoD, as a way to understand the unique issues related to the sustainment of software. Software is continuing to provide a greater percentage of the capabilities to be found in DoD weapons systems—and providing an increasing percentage of system cost as well. For both of these trends, no plateau is in sight. The unique flexibility and usefulness of software will make it central to sustainment strategies for adapting systems to the ever-changing mission needs and cyber-threat environment for the foreseeable future. We hope that the brief synopsis in this paper and the discrete ecosystem factors that we identified help to articulate many of the software-specific issues that are needed to do that effectively.

References

Defense Acquisition University (DAU). (2017.) Level of repair analysis (LORA). In *Acquipedia*. Retrieved from https://shortcut.dau.mil/acq/lora

Defense Science Board (DSB). (2000, November.) *Report of the Defense Science Board Task Force on defense software*. Washington, DC: Office of the Under Secretary of Defense for Acquisition, Technology, and Logistics. Retrieved from http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=ADA385923

National Research Council (NCR). (2010a.). *Critical code: Software producibility for defense.* Washington, DC: National Academies Press. doi: https://doiorg/10.17226/12979

National Research Council (NCR). (2010b.) *Achieving effective acquisition of information technology in the Department of Defense.* Washington, DC: National Academies Press. doi: https://doi.org/10.17226/12823

Acknowledgments

During the course of this study, the SEI interacted with a range of senior leaders, key managers, and staff across the DoD's software sustainment organizations. Universally, the Services and those we engaged were generally responsive to our data and information requests and candidly shared their experiences and perspectives on this critical and complex issue. We are exceedingly grateful for the many thoughtful conversations and insights we received in the course of this work.

This material is based upon work funded and supported by the DoD under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

Disclaimer & Distribution Statement

The view, opinions, and/or findings contained in this material are those of the author(s) and should not be construed as an official Government position, policy, or decision, unless designated by other documentation.

No warranty. This Carnegie Mellon University and Software Engineering Institute material is furnished on an "as-is" basis. Carnegie Mellon University makes no warranties of any kind, either expressed or implied, as to any matter including, but not limited to, warranty of fitness for purpose or merchantability, exclusivity, or results obtained from use of the material. Carnegie Mellon University does not make any warranty of any kind with respect to freedom from patent, trademark, or copyright infringement.

DM17-0081

Acquisition Research Program
Graduate School of Business & Public Policy
Naval Postgraduate School
555 Dyer Road, Ingersoll Hall
Monterey, CA 93943

www.acquisitionresearch.net