SYM-AM-23-064

Excerpt from the Proceedings of the Twentieth Annual Acquisition Research Symposium

Acquisition Research:
Creating Synergy for Informed Change

May 10–11, 2023

Published: April 30, 2023

An Agile Model-Based Systems Engineering Method to Accelerate Value Delivery

Ronald Giachetti—is a Professor of Systems Engineering at the Naval Postgraduate School (NPS) in Monterey, CA. He has over 30 years' experience teaching and conducting research in model-based systems engineering, system architecting, and system of systems engineering. He has published extensively with over 50 technical papers, including a textbook on the design of enterprise systems. He has lectured internationally in Europe, Latin America, and Asia. He has held multiple leadership positions including Chair of the Systems Engineering Department and Dean of the Graduate School of Engineering at NPS. He is currently Chair of the Corporate Advisory Board for the International Council on Systems Engineering (INCOSE). He holds engineering degrees from Rensselaer Polytechnic Institute, New York University—Polytechnic, and North Carolina State University. [regiache@nps.edu]

Abstract

Agile methods have shown their value in the software domain and are now the dominant approach to software development. All programs would like to experience similar benefits of customer satisfaction while being on time and within budget. Yet, adopting agile methods to larger scale programs as well as programs involving both hardware and software remains fraught with difficulty, and programs lack guidance on how to tailor these methods. Moreover, defense programs need to adopt model-based systems engineering and digital engineering, which is often seen as counter to agile methods. To overcome this challenge, we cast agility as a mindset to be adopted rather than a set of practices. This paper presents a method that is plan-based at the macro level but implements agility at the micro level. The paper demonstrates the approach for the development of a microgrid for a military base. We discuss the merits of combining the approach and why it is suitable for many defense acquisition programs.

Introduction

We all want to be faster and more agile so as to more quickly respond to changes in what has become a very dynamic strategic and operational environment. Unfortunately, this is not the experience of many Department of Defense (DoD) program managers, some of which spend 2 years or more just gathering needed information to meet acquisition requirements (GAO, 2015). These programs—developing large-scale, complex systems—are under a lot of pressure to more quickly design, develop, and deploy systems (GAO, 2022). The military wants capability delivered now. However, many systems experience long and costly development times. For instance, see the latest news about the Air Force's vision system for aiding mid-air refueling of planes (Losey, 2023). What these organizations want is to be more agile. They want to be able to get capability into the hands of their customers more quickly. These organizations, as well as their customers, see that many software companies in Silicon Valley are able to quickly release and constantly update apps using agile software engineering methods. There is a tremendous push to adopt these agile methods to acquisition programs.

The defense acquisition community is also in the midst of a major transformation with the adoption of digital engineering and its subset of model-based systems engineering (Zimmermann, 2019). Digital engineering promises greater efficiency and effectiveness of the system development process through an integrated tool set connecting all the system designers and other stakeholders who can seamlessly share information.

Engineering is a goal-oriented activity, and traditional engineering is bottom-up, in which an engineer identifies a problem and designs a product to address the problem (Pahl

& Beitz, 2013). However, as systems became large in the 1940s and 1950s, the prevailing engineering approaches feel short. The systems being envisioned were too complex, too large, involved many disciplines, and had many requirements, which overwhelmed traditional approaches. Systems engineering grew out of this environment as a top-down approach to organize and control the technical development of the system. Now, there is growing evidence that the plan-driven systems engineering processes, such as the systems engineering vee model adhered to by most defense acquisition programs, are inadequate for some of the challenges facing programs today. Programs face what is termed a volatile, uncertain, complex, and ambiguous (VUCA) environment. It is near impossible to adequately plan long-term projects in this highly dynamic environment. Moreover, the sequential engineering vee process is too cumbersome and slow for incorporating changes due to emerging technology and requirements changes.

The DoD seeks the speed, agility, and innovation seen in many of the technology companies found in places such as Silicon Valley. Towards this end, the DoD has open Defense Innovation Units (DIU) in multiple cities. One of the practices that enable the observed speed to market and innovation is agile development. Agile practices have been very successful in the development of software products and services, often on smaller scales, such as websites and apps. The agile practices are now being scaled and extended to systems development involving both hardware and software and for much larger, more complex systems.

This paper contributes to the literature thoughts on how to adopt and adapt agile methods to the large-scale, complex projects involving both hardware and software typical of defense acquisition programs. Straight-forward taking of agile practices from software engineering is not possible for these types of systems. Instead, we propose a hybrid approach preserving plan-driven aspects that remain appropriate for large-scale, complex programs with hardware and mingle in principles of agile practices.

Agile Development

Agile development is a software development approach that emphasizes iterative and incremental development, continuous delivery, and customer collaboration. Agile methods value individuals and interactions over processes and tools, working software over comprehensive documentation, and responding to change over following a plan. Agile is beneficial in dynamic environments when the requirements may change or new requirements discovered during development; in the face of complexity of the problem, system, or organization and resulting in learning as you go with the inevitable course corrections.

Agile development executes iterative and incremental development through the use of sprints, which are short periods, usually 2-weeks in length, during which small teams scope, analyze, design, build, and test small increments of working code. This form of time-boxing flips the triple constraint of project management on its head (see Figure 1). Plan-driven methods usually hold the scope or requirements invariant, and then budget and schedules adjust (usually slip) to accommodate the scope. Agile methods hold schedule invariant as well as budget through a constant team size. As a result, the scope must change to accommodate both schedule and budget. Other differences with traditional plan-driven approaches are summarized in Table 1.

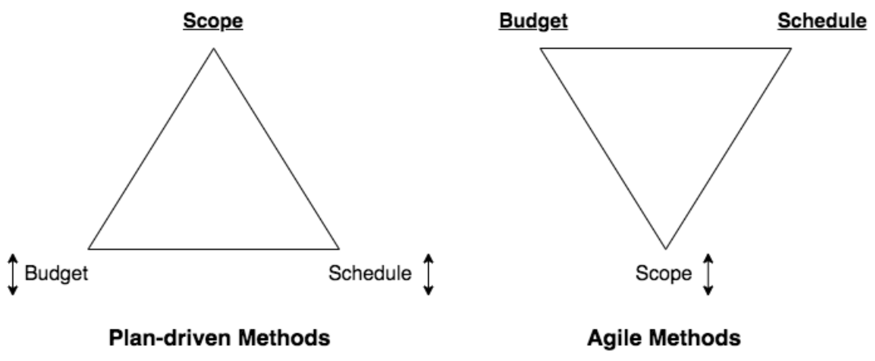

Figure 1. The Triple Constraint in Plan-Driven Vice Agile Methods

The Global Hawk program used iterative and incremental development (Henning & Walter, 2005). Consequently, the iterative and incremental approach is not completely foreign to the DoD. The Global Hawk program had increments of 1-year duration and allowed the operational users to change requirements based on evolving operational needs and what they learned from the previous increments. The increments were as follows:

1. Operationalized the existing system to provide a worldwide operating capability and established sustainable support system.
2. Expanded imagery intelligence (IMINT) and introduced initial signal intelligence (SIGINT).
3. Added full-spectrum SIGINT and defensive threat awareness.
4. Improved the radar to track moving ground targets. Added airborne surveillance and enhanced airspace operations and survivability.
5. Completed full-spectrum operations, expanded communications, and hardened for extreme environments including nuclear, biological, and chemical.

Table 1. Comparison of Plan-Driven and Agile Programs

Plan-Driven Program	Agile Program
Align budget, schedule, and resources (e.g., test range)	Iterative and incremental development of work products
Top-down design	Close and frequent engagement with stakeholders
Long lead-time items	Continuous verification
Interconnections with other systems	Self-managed teams
Identify and design for needed quality attributes (-ilities such as reliability, maintainability, cybersecurity, etc.)	Continuous integration
Comply with regulations and policies; ensure traceability	Rapid learning and risk reduction
Safety critical issues	

Digital Engineering

Digital engineering describes the use of models as the primary means of reasoning, analyzing, designing, documenting, and communicating about the system-of-interest (SoI). Specifically, the DoD, a major proponent of digital engineering, defines *digital engineering* as “an integrated digital approach that uses authoritative sources of systems’ data and models as a continuum across disciplines to support life-cycle activities from concept through disposal" (Office of the Deputy Assistant Secretary of Defense for Systems Engineering, 2018). A system development project uses a myriad of models of the system, including descriptive models of the systems requirements and architecture, geometric models of all the system's parts using computer-aided design (CAD), physics-based models of the system for analysis (e.g., finite element analysis, computational fluid dynamics, circuit design), operational models such as captured in discrete-event simulation, reliability models, and many other models of various aspects of the system. All these models would be part of the digital thread, in which changes to data in one model are propagated to all the other interrelated models.

Organizations need to create a digital engineering infrastructure in order to implement the digital engineering vision. Given no single tool exists, organizations must pursue a best of breed approach in which they select and integrate software tools for each of the domains and tasks in the system life cycle. Figure 2 shows the digital thread that emerges when you integrate together the entire tool set for a program. The models in the digital thread become the Authoritative Source of Truth (ASoT) for the program, meaning they are used in design, contracts, and so on.

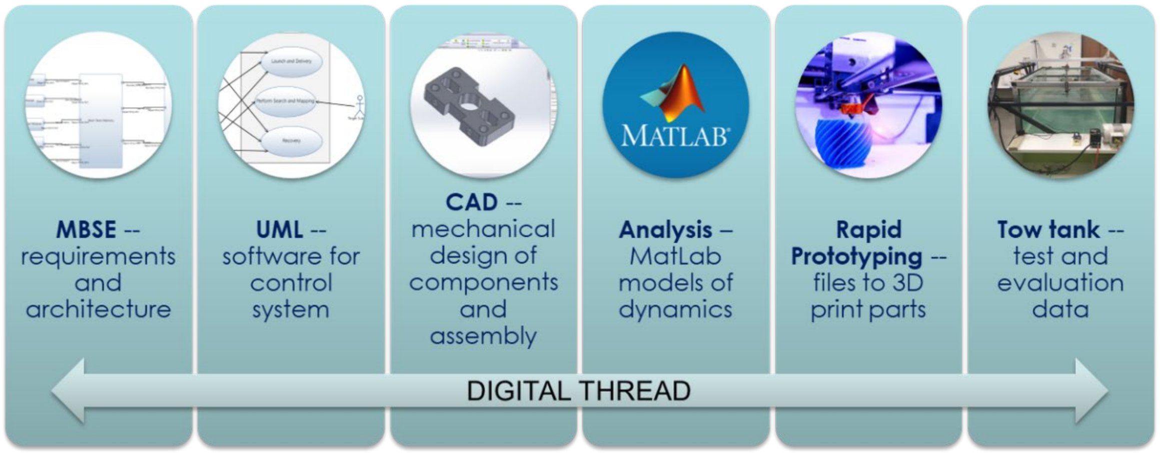
MBSE -- requirements and architecture
UML -- software for control system
CAD -- mechanical design of components and assembly
MATLAB
Analysis – MatLab models of dynamics
Rapid Prototyping -- files to 3D print parts
Tow tank -- test and evaluation data
DIGITAL THREAD

Figure 2. Digital Thread for a Program

Digital engineering can be an important enabler of agile practices on defense acquisition programs because it allows for short iterations of scoping, designing, building models, and testing models in the early phases of system development. This would be a deviation from agile method's emphasis on working code as output of each sprint. Instead, we would have verified models as the output of a sprint.

Related Work

Agile methods have been widely adopted in software development projects to improve team collaboration, communication, and flexibility in responding to changes in customer requirements. However, the application of agile principles to systems engineering processes, particularly in the development of complex systems with hardware components, is still an area of active research and discussion. The International Council on Systems Engineering (INCOSE) has a team that has been looking at how agility can be infused or adopted by systems engineering organizations (Willett et al., 2021). Their work is part of the Future of Systems Engineering (FuSE) initiative of INCOSE.

Several studies have shown the benefits of applying agile principles to systems engineering processes. For example, Berczuk et al. (2012) showed that the use of agile methods in the development of a safety-critical system led to improved communication, increased customer satisfaction, and better project management. Similarly, Dove (2023) suggested eight principles for agile systems engineering. While the authors do not group the principles, we view four of them as emerging from self-organizing teams (Dove's common-mission teaming, attentive decision-making, shared knowledge management, and being agile), iterative and incremental development, attentive situational awareness, continual integration and test, and feature-based product line architectures.

Scaling agile to larger systems is another area of active research (Dingsøyr et al., 2019). Several frameworks have been proposed to scale agile methods to larger systems, such as the Scaled Agile Framework (SAFe) and the Large Scale Scrum (LeSS) framework (Knaster & Leffingwell, 2017). These frameworks provide guidance on how to coordinate multiple agile teams working on different parts of a larger system. Several studies have explored the effectiveness of scaling agile methods to larger systems. For example, McCaffery et al. (2017) showed that the use of SAFe in the development of a complex software-intensive system led to improved project planning, better coordination between teams, and increased stakeholder satisfaction. Similarly, Elssamadisy et al. (2018) showed that the use of LeSS in the development of a large-scale hardware and software system led to improved team collaboration, reduced project risk, and increased customer satisfaction.

Agile methods have been primarily applied to software development projects, but their application to hardware development is also an area of active research. Hardware development involves longer lead times, more complex dependencies, and greater risk than software development, making it more challenging to apply agile principles. Several studies have explored the application of agile principles to hardware development. For example, Yang et al. (2019) showed that the use of agile principles in the development of a hardware product led to improved communication, reduced project risk, and increased customer satisfaction. Similarly, Thakurta et al. (2020) showed that the use of agile principles in the development of an embedded system led to improved team collaboration, faster development cycles, and reduced project risk. Paasivaara and Lassenius (2019) described Ericsson's long journey of adopting agile to the design and development of their products.

Tailoring a Hybrid Plan-Driven and Agile Development Method

Viewing whether to adopt agile as an all-or-none proposition is the wrong way to be viewing the issue. Defining agile through the methods and practices in software engineering is probably not the best way to think about it either. Rather, a systems engineering organization needs to consider how agile they need to be. Stelzmann (2012) surveyed companies and came up with the suggestion that companies ask themselves two questions. First, to what degree is agility demanded by the market, technology, and other environmental factors? Second, to what degree can the organization be agile? The title of Barry Boehm and Rich Turner's book captures what we are saying in that they see it as balancing disciplined methods—that is, plan-driven methods with agile methods (Boehm & Turner, 2004).

Organization science has long recognized the most effective management style is often contingent upon various internal and external factors (Galbraith, 1973). This is called contingency theory, and it emphasizes the importance of situational analysis, flexibility, and decision-making based on the specific circumstances of each situation. Consequently, we view the best system development model as the fit between the organization, its people, and its culture; the system, how complex it is, how connected to other systems, new technologies; and the business environment, how dynamic it is, and the degree of uncertainty. Figure 3 shows multiple factors as continuums, with those on the left suggesting plan-driven methods are more appropriate, and those factors on the right suggesting agile methods are more appropriate. The first four factors fit the acronym VUCA—an apt description of the business environment facing many organizations. How large the system is comes into play because agile methods have been most successfully applied to smaller projects (although there are counter examples of large project successes). Organizations working in regulated environments and/or dealing with safety critical systems will need more planning, documentation, and traceability of requirements. When systems are part of systems of systems and must interoperate with other systems, then identifying those interfaces and ensuring interoperability is critical, which requires greater levels of planning. Large teams, multiple organizations, and geographically dispersed teams all suggest a need for more planning—certainly a lot more coordination.

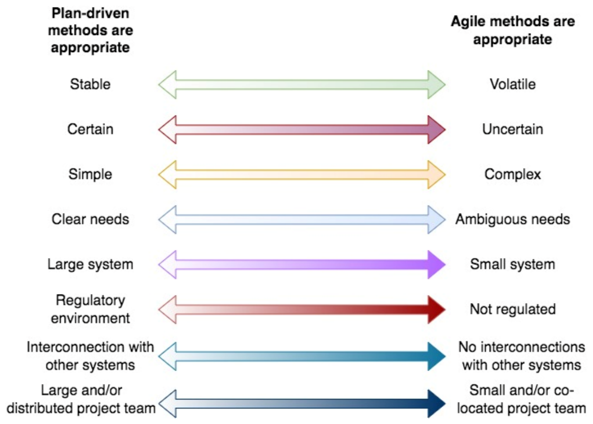
Plan-driven methods are appropriate
Agile methods are appropriate
Stable
Volatile
Certain
Uncertain
Simple
Complex
Clear needs
Ambiguous needs
Large system
Small system
Regulatory environment
Not regulated
Interconnection with other systems
No interconnections with other systems
Large and/or distributed project team
Small and/or co-located project team

Figure 3. Factors for Deciding on Balance Between Plan-Driven and Agile Methods

Most companies doing large, complex systems development characteristic of the aerospace, defense, and even automotive sectors will likely find they fall to the right on some factors and to the left on others. Such an organization could benefit from greater agility for performing quick iterations to understand the requirements, early discovery of risk, and frequent customer feedback—all in order to deal with the market volatility, uncertainty, and ambiguous customer needs. However, those same organizations still need traditional planning and discipline because they operate in a regulated environment, the system has to interoperate with other systems, and the team is spread out between multiple organizations and time zones.

Planned and Agile System Development

Having established with contingency theory that many defense acquisition programs require both planning and agility, we embark upon how such a hybrid approach can be executed. The method adopts and adapts important agile concepts at multiple levels of development. The method iteratively and incrementally evolves the system models from which the necessary systems engineering artifacts can be generated. The method uses self-organizing teams who determine the best way to handle their work. The teams follow agile practices to learn fast and early through iterations of digital modeling, prototyping, and testing. Lastly, and importantly, is the adoption of an agile mindset by all the people on the program—meaning they trust the highly self-directed teams to do quality work, are committed to continuous delivery of work products and capabilities, measure progress based on work completed, and maintain close interaction and engagement of stakeholders.

Plan-Driven at Macro Level

Systems with hardware require some planning, because hardware often requires parts with long lead times, hardware cannot be refactored, and customers want to know when the system will be first deployed. Additionally, larger systems often have many interfaces and interactions with other systems that must be planned for and controlled. These issues are addressed by macro-planning of the overall system development process and by using a top-down approach starting with a system architecture. Figure 4 shows a high-level view of the system development activities, and Figure 5 shows the more detailed planning prior to the next milestone. The figures show there is extensive parallelism of the activities, but what is missing is the intensity of effort changes. Figure 4 shows verification and validation (i.e., testing) occurs continuously throughout the process. The frequent testing enables continuous design maturation and risk reduction. Within each phase are iterations in the spirit of design thinking of understanding, designing, building, and testing ideas through the use of models. Additionally, there are feedback loops and iterations between phases. For instance, as capabilities are analyzed and defined, the team might rethink how they framed the problem and revise their problem analysis. As a result, the design method progressively analyzes, designs, and evaluates the stakeholder needs, requirements, and mission to build the architectural products.

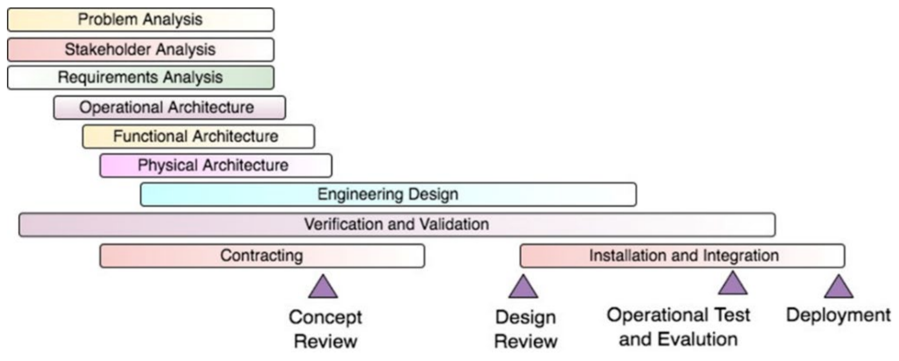

Figure 4. Macro-Plan for a Program Fitted to Acquisition Milestones

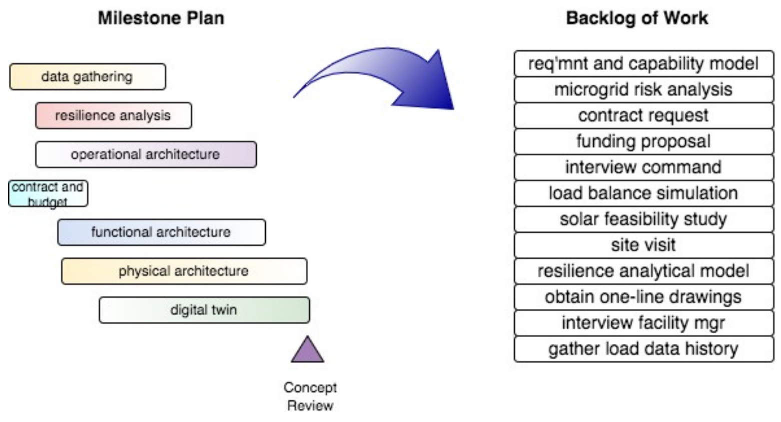

Figure 5. Milestone Planning and Generation of Work Backlog

An important planning document for defense programs would be a roadmap showing the incremental delivery of capability to the forces. Figure 6 shows an example roadmap for a microgrid project to achieve energy security at a military base.

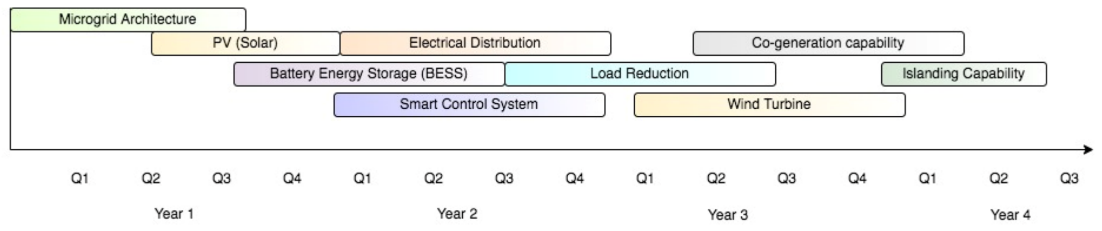

Figure 6. Capability Roadmap

Top-Down Development

The method is top-down, meaning we start with the high-level, holistic design of the entire system and then move on to the subsystems, assemblies, and lower levels. Top-down design is a hallmark of the systems engineering process. Top-down development in systems engineering is important because it provides a structured and organized approach to designing and building complex systems, ensures that the resulting system meets the required functionality and quality standards, helps to manage complexity, and facilitates the integration of subsystems and components. Top-down design allows for a systematic and organized process of designing and building a system that meets the needs of the users and stakeholders. Traditionally, by breaking down the system requirements into smaller subsystems and components, top-down development also makes it easier to manage the complexity of the system. It allows for the identification of potential problems early in the development process, which can then be addressed before they become major issues. Additionally, top-down development facilitates the integration of the subsystems and components, which is critical to ensuring that the system operates as a cohesive whole.

Agile at Micro Level and In Mindset

Agility is foremost a mindset of how to organize projects and conduct work. The agile mindset is shaped by the core values and principles underlining agile system development. These principles are put into practice through various agile methods for how to organize the project team, how to plan the project, and how to execute the work activities in the project. We now proceed to discuss the aspects of the method that are agile. Essential to the development method is working in an iterative and incremental fashion.

Iterative and Incremental Development of Work Products

Iterative development involves building a product through a series of cycles or iterations, with each iteration building on the previous one (see Figure 7). Each iteration includes planning, design, implementation, and testing activities. The goal of each iteration is to add new features, improve existing ones, and fix any issues or bugs that were discovered in the previous iteration. This approach allows developers to receive feedback from users and stakeholders early on, and to adjust the product accordingly.

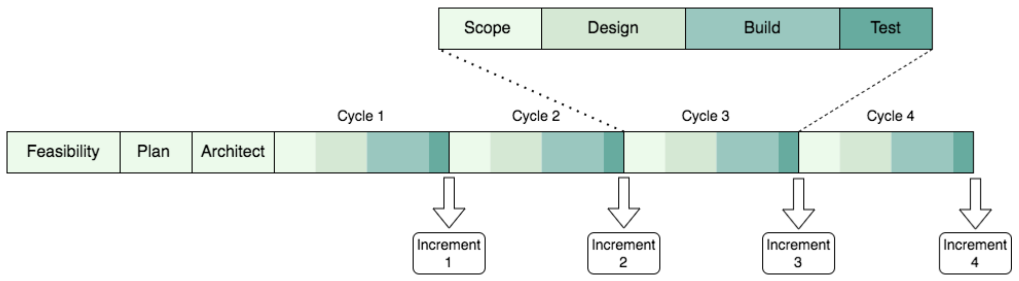

Figure 7. Iterative and Incremental Development Sprints

Empowered Teams

Empowered teams in agile development are teams that have been given the autonomy and authority to make decisions and take ownership of their work. This means that team members are trusted to self-organize and collaborate to deliver high-quality products (Ibarra & Scoular, 2019). In an empowered team, each member is responsible for their role and is accountable for the outcome of their work. They are encouraged to share their ideas and opinions and to challenge the status quo when necessary. They have a sense of ownership over their work and are motivated to deliver value to their customers.

Figure 8 shows the program organization for a microgrid consisting of self-organizing teams. Agile development emphasizes the importance of communication, collaboration, and feedback. Empowered teams are able to communicate freely, share information, and collaborate effectively to solve problems and achieve their goals. They are also able to receive feedback from stakeholders and customers, and use it to improve their work and deliver better results.

Empowered teams in agile development are an essential component of the agile methodology. They help to create a culture of continuous improvement and innovation, and enable organizations to respond quickly and effectively to changing customer needs and market demands.

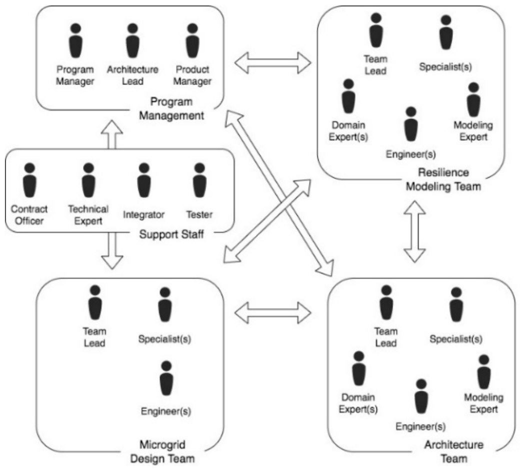

Figure 8. Agile Teams for a Microgrid Project

The agile teams will have a backlog of work identified during the planning phase that needs to be completed. The backlog is a prioritized list of capabilities, system features, intermediate artifacts (e.g., a model), or any task the team needs to work on. In large systems, the tasks sometimes might be just for risk reduction such as a feasibility study. The product owner, who is responsible for representing the stakeholders and defining the product vision, leads this effort.

The backlog is continuously refined and updated as new information becomes available or the team gains a better understanding of user needs. Items in the backlog are prioritized based on their business value, user impact, and technical feasibility, among other factors.

During sprint planning, which is a time-boxed meeting held at the beginning of each sprint, the team selects a subset of items from the backlog that they can commit to completing within the sprint (see Figure 9). The team also discusses the technical details of how they will implement each item and identifies any dependencies or risks that need to be addressed. The team then estimates the effort required to complete each item using a relative sizing technique, such as story points, and creates a sprint backlog, which is a plan of the work that they will undertake during the sprint. The sprint backlog serves as a guide for the team's work during the sprint, and progress is tracked daily during the daily standup meeting. At the end of the sprint, the team reviews the work completed and identifies areas for improvement in the next sprint.

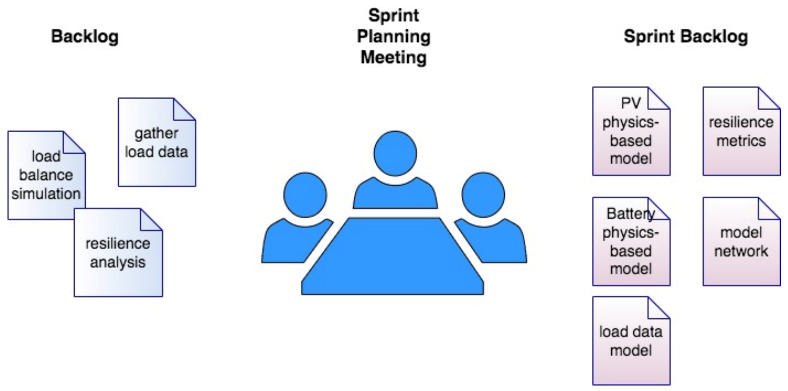

Figure 9. Sprint Planning Meeting Reviews Backlog and Determines Work Tasks to Complete

Continuous Integration

Continuous integration means the project teams are determining in each sprint how the multiple components of the system come together and function as a system. Early on in a program, the integration is of the various models and artifacts being developed by the teams. The digital engineer tools facilitate continuous integration because they can enforce consistency between the models. The benefit of continuous integration is it will reduce risk by identifying issues that occur only when components are integrated into subsystems, and subsystems are integrated into systems.

A program must do some planning in order for continuous integration to be feasible. Figure 10 shows the synchronization of a hardware component with its embedded software. Because hardware takes longer to develop, the sprint length is 4 weeks vice the 2 weeks common to software development. Different teams work on developing each component, and they conduct continuous integration by identifying and managing the dependencies between their components. This ends with an integration test of the functional prototype.

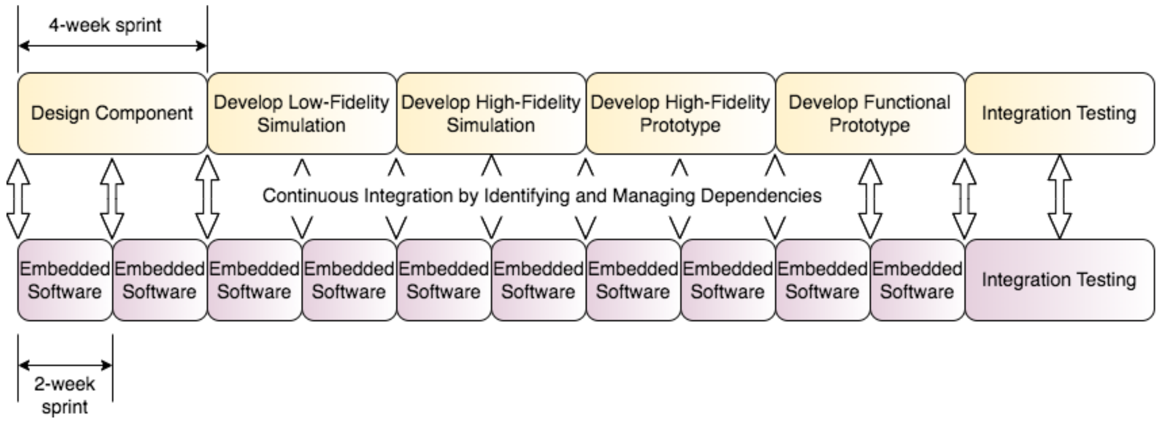
4-week sprint
Design Component
Develop Low-Fidelity Simulation
Develop High-Fidelity Simulation
Develop High-Fidelity Prototype
Develop Functional Prototype
Integration Testing
Continuous Integration by Identifying and Managing Dependencies
Embedded Software
Integration Testing
2-week sprint

Figure 10. Synchronization and Continuous Integration of Software With Hardware

Continuous Verification and Validation

An important concept in agile development, enabled by iterative development and digital engineering tools, is the continuous verification and validation (V&V) of all work products. The earlier a problem or issue is identified, then the lower the cost to fix the problem. The continuous verification of products—including models, code, prototypes, and so forth—ensures the project teams are always working with quality, functional products. Digital engineering tools enable teams to do the verification on models very early in the system development method. For instance, a team can verify a concept of operations (CONOPs) using simulation, thus lending greater confidence in the overall system design concept. Validation ensures what is being developed is useful and valued by the end users. For this reason, agile methods call for close and continuous interaction with users. Multiple means support the close interaction, including having such stakeholders involved with the program directly, having product owners to represent end user needs, and also through DevOps, which creates a feedback loop from operations to the development team (Miller et al., 2021).

Summary

The research examined the characteristics and principles of agile methods through the lens of how they could be adopted within the defense acquisition community. A comparison with plan-driven approaches was also conducted. Together, through the lens of contingency theory, the paper proposed a hybrid approach combining the plan-based perspective at the macro level and adopting agile practices at the micro level. The method is enabled by digital engineering, which allows for iterations of model development and test during the early phases. The paper also shows how important aspects of systems engineering such as top-down refinement can be preserved in the hybrid development environment.

If agile is a mindset, then adopting agile involves a transformation of the DoD organizational culture, and such transformations take a lot of time and dedicated leadership. However, adopting agile principles into defense acquisition promises to increase the ability of the DoD to better respond to quickly changing requirements and other environmental uncertainties. The result can be getting some capability into the hands of the warfighter sooner.

References

Berczuk, S., Appleton, B., & Konieczka, S. (2012). Agile in a safety-critical project: An experience report. *IEEE Software*, *29*(1), 77–85.

Boehm, B., & Turner, R. (2004). *Balancing agility and discipline: A guide for the perplexed*. Addison–Wesley Professional.

Dingsøyr, T., Falessi, D., & Power, K. (2019). Agile development at scale: The next frontier. *IEEE Software*, *36*(2), 30–38.

Elssamadisy, A., Abu-Tayeh, G., & Brown, T. (2018). *Scaling agile and lean development in the enterprise*. Pearson Education.

Galbraith, J. R. (1973). *Designing complex organizations*. Addison–Wesley.

GAO. (2015, February). *DOD should streamline its decision-making process for weapon systems to reduce inefficiencies* (GAO-15-192).

GAO. (2022, June). *Weapon systems annual assessment: Challenges to fielding capabilities faster persist* (GAO-22-105230).

Hawker, N., & McBride, T. (2016). The use of agile development methods in a large-scale software-intensive system. *Journal of Systems and Software*, *122*, 1–16.

Henning, W. A., & Walter, D. T., (2005). *Spiral development in action: A case study of spiral development in the Global Hawk Unmanned Aerial Vehicle program* [Master’s thesis, Naval Postgraduate School].

Ibarra H., & Scoular, A. (2019, November–December). The leader as coach. *Harvard Business Review*.

Knaster, R., & Leffingwell, D. (2017). *SAFe 4.0 distilled: Applying the Scaled Agile Framework for lean software and systems engineering*. Addison–Wesley Professional.

Losey, S. (2023, February 22). After long, costly road, Air Force happy with new KC-46 vision system. *Defense News*.

McCaffery, F., Phalp, K., & Jeary, S. (2017). Scaling agile: An empirical study of large complex software systems. *Journal of Systems and Software*, *131*, 218–230.

Miller, A. W., Giachetti, R. E., & Van Bossuyt, D. L. (2022). Challenges of adopting DevOps for the combat systems development environment. *Defense Acquisition Research Journal*, *29*(1).

Office of the Deputy Assistant Secretary of Defense for Systems Engineering. (2018, June). *Digital engineering strategy* [Technical report]. DoD.

Paasivaara, M., & Lassenius, C. (2019). Empower your agile organization: Community-based decision making in large-scale agile development at Ericsson. *IEEE Software*, *36*(2), 64–69.

Pahl, G., & Beitz, W. (2013). *Pahl/Beitz Konstruktionslehre: Methoden und Anwendung erfolgreicher Produktentwicklung*. Springer–Verlag.

Stelzmann, E. (2012). Contextualizing agile systems engineering. *IEEE Aerospace and Electronic Systems Magazine*, *27*(5), 17–22.

Thakurta, R., Mukhopadhyay, D., & Das, D. (2020). Applying agile practices in embedded system development: An industrial case study. *Journal of Systems and Software*, *168*, 110660.

Willett, K. D., Dove, R., Chudnow, A., Eckman, R., Rosser, L., Stevens, J. S., Yeman, R., & Yokell, M. (2021, July). Agility in the future of systems engineering (FuSE): A roadmap of foundational concepts. *INCOSE International Symposium*, *31*(1), 158–174.

Yang, J., Lappas, T., Egan, M., & Bagaria, N. (2019). Applying Agile methods to hardware product development. *Journal of Systems and Software*, *154*, 1–14.

Zimmerman, P., Gilbert, T., & Salvatore, F. (2019). Digital engineering transformation across the Department of Defense. *The Journal of Defense Modeling and Simulation*, *16*(4), 325–338.