INSIGHT

This Issue's Feature:

Agility in the future of systems engineering

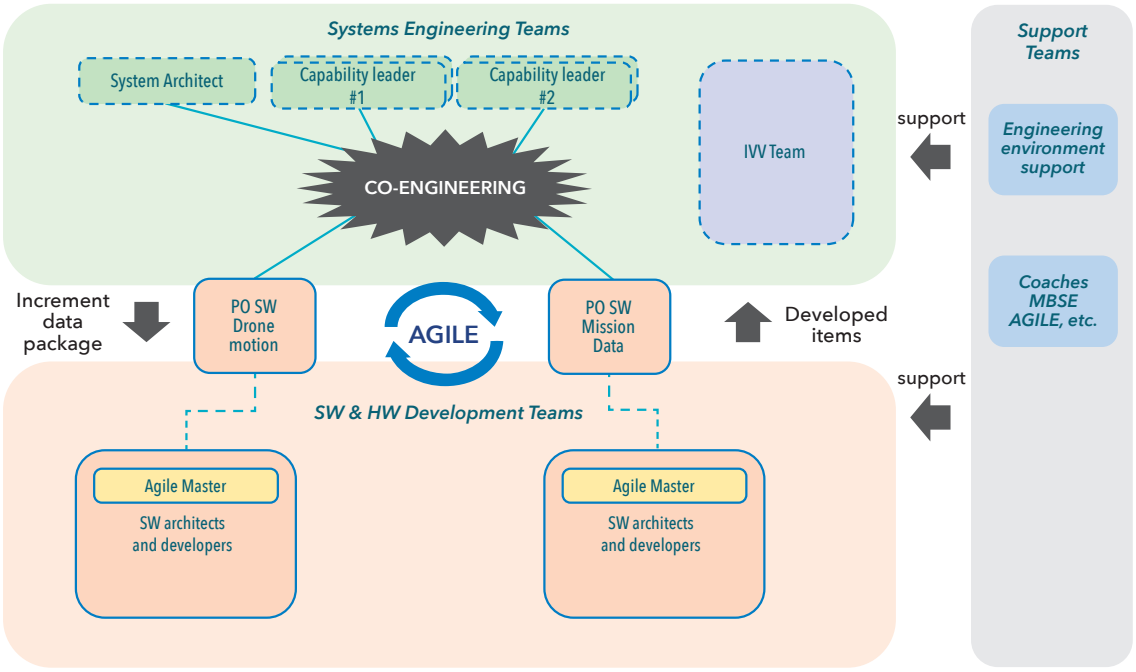


Illustration credit: from the article
Model-Based Systems Engineering as an Enabler of Agility
by Sophie Plazanet and Juan Navas (page 26)

JUNE 2023
VOLUME 26 / ISSUE 2

A PUBLICATION OF THE INTERNATIONAL COUNCIL ON SYSTEMS ENGINEERING

FuSE

Future of Systems Engineering

Sponsored by SPEC INNOVATIONS

OUR MISSION

FuSE refines and evolves the SE Vision 2035 across competencies, research, tools & environment, practices, and applications

FuSE identifies critical gaps towards the vision realization and initiates & supports relevant actions

FuSE fosters involvement and collaboration within and outside of INCOSE

FuSE educates, shares success, and expands

JOIN US

INCOSE.ORG/FUSE

INCOSE

International Council on Systems Engineering

A better world through a systems approach

INSIGHT

A PUBLICATION OF THE INTERNATIONAL COUNCIL ON SYSTEMS ENGINEERING

JUNE 2023 VOLUME 26 / ISSUE 2

Inside this issue

About This Publication

INFORMATION ABOUT INCOSE

INCOSE's membership extends to over 20,000 individual members and more than 200 corporations, government entities, and academic institutions. Its mission is to share, promote, and advance the best of systems engineering from across the globe for the benefit of humanity and the planet. INCOSE charters chapters worldwide, includes a corporate advisory board, and is led by elected officers and directors.

For more information, click here:
The International Council on Systems Engineering
(www.incose.org)
INSIGHT is the magazine of the International Council on Systems Engineering. It is published four times per year and

OVERVIEW

features informative articles dedicated to advancing the state of practice in systems engineering and to close the gap with the state of the art. ***INSIGHT*** delivers practical information on current hot topics, implementations, and best practices, written in applications-driven style. There is an emphasis on practical applications, tutorials, guides, and case studies that result in successful outcomes. Explicitly identified opinion pieces, book reviews, and technology roadmapping complement articles to stimulate advancing the state of practice. ***INSIGHT*** is dedicated to advancing the INCOSE objectives of impactful products and accelerating the transformation of systems engineering to a model-based discipline.
Topics to be covered include resilient systems, model-based systems engineering, commercial-driven transformational systems engineering, natural systems, agile security, systems of systems, and cyber-physical systems across disciplines and domains of interest to the constituent groups in the systems engineering community: industry, government, and academia. Advances in practice often come from lateral connections of information dissemination across disciplines and domains. ***INSIGHT*** will track advances in the state of the art with follow-up, practically written articles to more rapidly disseminate knowledge to stimulate practice throughout the community.

Editor-In-Chief William Miller
insight@incose.net +1 908-759-7110

Theme Editor
Rick Dove dove@parshift.com

Layout and Design Chuck Eng
chuck.eng@comcast.net

Member Services INCOSE Administrative Office
info@incose.net +1 858 541-1725

PERMISSIONS

ARTICLE SUBMISSION insight@incose.net

Publication Schedule. ***INSIGHT*** is published four times per year. Issue and article submission deadlines are as follows:

- September 2023 issue – 1 July 2023
- December 2023 issue – 1 October 2023
- February 2024 issue – 1 November 2023
- April 2024 issue – 2 January 2024
- June 2024 issue – 1 March 2024
- August 2024 issue – 1 May 2024
- October 2024 – 1 July 2024
- December 2024 – 1 September 2024

For further information on submissions and issue themes, visit the INCOSE website: www.incose.org

 ISSN 2156-485X; (print) ISSN 2156-4868 (online)

ADVERTISE

Readership
INSIGHT reaches over 20,000 individual members and uncounted employees and students of more than 100 CAB organizations worldwide. Readership includes engineers, manufacturers/purchasers, scientists, research and development professionals, presidents and chief executive officers, students, and other professionals in systems engineering.

Issuance	Circulation
2023, Vol 26, 4 Issues	100% Paid

Contact us for Advertising and Corporate Sales Services
We have a complete range of advertising and publishing solutions professionally managed within our global team. From traditional print-based solutions to cutting-edge online technology the Wiley-Blackwell corporate sales service is your connection to minds that matter. For an overview of all our services please browse our site which is located under the Resources section. Contact our corporate sales team today to discuss the range of services available:

- Print advertising for non-US journals
- Email Table of Contents Sponsorship
- Reprints
- Supplement and sponsorship opportunities
- Books
- Custom Projects
- Online advertising

Click on the option below to email your enquiry to your nearest office:

- Asia and Australia corporatesalesaustralia@wiley.com
- Europe, Middle East and Africa (EMEA) corporatesaleseurope@wiley.com
- Japan corporatesalesjapan@wiley.com
- Korea corporatesaleskorea@wiley.com

USA (also Canada, and South/Central America):

- Healthcare Advertising corporatesalesusa@wiley.com
- Science Advertising Ads_sciences@wiley.com
- Reprints Commercialreprints@wiley.com
- Supplements, Sponsorship, Books and Custom Projects busdev@wiley.com

Or please contact:
Marcom@incose.net

CONTACT

Questions or comments concerning:

Submissions, Editorial Policy, or Publication Management
Please contact: William Miller, Editor-in-Chief
insight@incose.net

Advertising—*please contact:*
Marcom@incose.net

Member Services – *please contact:* info@incose.org

ADVERTISER INDEX

June Volume 26-2

CORPORATE ADVISORY BOARD — MEMBER COMPANIES

Aerospace Corporation, The
Airbus
AM General LLC
Analog Devices, Inc.
ARAS Corp
Arcfield
Australian National University
AVIAGE SYSTEMS
Aviation Industry Corporation of China
BAE Systems
Ball Aerospace
Bechtel
Becton Dickinson
Belcan Engineering Group LLC
Boeing Company, The
Bombardier Transportation
Booz Allen Hamilton Inc.
C.S. Draper Laboratory, Inc.
CACI, Inc – Federal
California State University Dominguez Hills
Carnegie Mellon University Software Engineering Institute
Change Vision, Inc.
Colorado State University Systems Engineering Programs
Cornell University
Cranfield University
Cubic
Cummins Inc.
Cybernet MBSE Co, Ltd
Dassault Systèmes
Defense Acquisition University
Deloitte Consulting, LLC
Denso Create Inc
Drexel University
Eindhoven University of Technology
EMBRAER
Federal Aviation Administration (U.S.)
Ford Motor Company
Fundacao Ezute
GE Aerospace
General Dynamics
General Motors
George Mason University
Georgia Institute of Technology
IBM
Idaho National Laboratory
ISAE – Supaero
ISDEFE
ITID, Ltd
IVECO SPA
Jama Software
Jet Propulsion Laboratory
John Deere
Johns Hopkins University
KBR
KEIO University
L3Harris Technologies
Lawrence Livermore National Laboratory
Leidos
LEONARDO
Lockheed Martin Corporation
Los Alamos National Laboratory
Loyola Marymount University
Mahindra University
ManTech International Corporation
Marquette University
Massachusetts Institute of Technology
MBDA (UK) Ltd
MetaTech Consulting Inc.
Missouri University of Science & Technology
MITRE Corporation, The
Mitsubishi Heavy Industries, Ltd
Modern Technology Solutions, Inc.
National Aeronautics and Space Administration (NASA)
National Reconnaissance Office (NRO)
National Security Agency Enterprise Systems
Naval Postgraduate School
Nissan Motor Co, Ltd
Northrop Grumman Corporation
Pacific Northwest National Laboratory
Pennsylvania State University
Peraton
Petronas Nasional Berhad
Prime Solutions Group, Inc
Project Performance International (PPI)
Purdue University
QRA Corp
Raytheon Technologies
Rolls-Royce
Saab AB
SAIC
Sandia National Laboratories
Saudi Railway Company
Shell
Siemens
Sierra Nevada Corporation
Singapore Institute of Technology
SPEC Innovations
Stevens Institute of Technology
Strategic Technical Services LLC
Swedish Defence Materiel Administration (FMV)
Systems Planning and Analysis
Taiwan Space Agency
Tata Consultancy Services
Thales
The University of Arizona
Torch Technologies
TOSHIBA Corporation
Trane Technologies
Tsinghua University
UC San Diego
UK MoD
University of Alabama in Huntsville
University of Arkansas
University of Connecticut
University of Maryland
University of Maryland Global Campus
University of Maryland, Baltimore County
University of Michigan, Ann Arbor
University of New South Wales, The, Canberra
University of Southern California
University of Texas at El Paso (UTEP)
US Department of Defense
Veoneer
Virginia Tech
Vitech
Volvo Cars Corporation
Volvo Construction Equipment
Wabtec Corporation
Woodward Inc
Worcester Polytechnic Institute- WPI
Zuken Inc

FROM THE EDITOR-IN-CHIEF

William Miller, insight@incose.net

We are pleased to announce the June 2023 *INSIGHT* issue published cooperatively with John Wiley & Sons as the systems engineering practitioners' magazine. The *INSIGHT* mission is to provide informative articles on advancing the practice of systems engineering and to close the gap between practice and the state of the art as advanced by *Systems Engineering*, the Journal of INCOSE also published by Wiley. The issue theme is *agility in the future of systems engineering*.

The future of systems engineering (FuSE) is a systems community initiative enabled and facilitated by INCOSE to realize the *Systems Engineering Vision 2035*, freely accessible at https://www.incose.org/about-systems-engineering/se-vision-2035. FuSE began in late 2017 leveraging the previous *Systems Engineering Vision 2025* and in anticipation of the latest vision announced at the 2022 International Workshop in January 2022. FuSE has identified four streams to drive implementation to realize the Vision 2035: systems engineering vision & roadmaps, systems engineering foundations, systems engineering methodologies, and systems engineering application extensions. See the FuSE webpage at https://www.incose.org/fuse.

We thank Keith Willett for inspiring and initiating the agile systems engineering work as an early FuSE project, theme editor Rick Dove, and the authors for their contributions. Agility has been a past *INSIGHT* theme with *agile system-security: sustainable systems evolve with their environment* (July 2016) and *enabling and practicing systems engineering agility* (June 2018).

ISO, IEC, and IEEE have released standard 15288:2023 *System life cycle processes* as this issue of *INSIGHT* was in the final stages of publication. The articles herein reference the 2015 version of the 15288 standard. The FuSE agile systems engineering project will be addressing the new standard and the INCOSE *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities* 5th edition as it continues its work.

Rick Dove leads off the June 2023 *INSIGHT* by "Setting the Current Context for Agility in the Future of Systems Engineering." A roadmap for near-term improvement, presented at the 2021 INCOSE International Symposium, introduced nine strategic concepts appropriate and ready for further movement toward standard practice. Initial work in that direction enticed several practitioners and researchers to address selected concepts in this special issue of the INCOSE *INSIGHT* publication.

"Systems Engineering Agility in a Nutshell" by Rick Dove, Kerry Lunney, Michael Orosz, and Mike Yokell succinctly describes eight strategic aspects with application discussions at the systems engineering level. Systems engineering must necessarily have the agility anticipate and effectively respond to a dynamic and uncertain environment. Agile systems engineering, agile software engineering, and agile any-kind-of engineering share common goals and leverage common agility-enabling strategies.

"The Supra—System Model" by Tom McDermott, Kelly Alexander, and Richard Wallace promotes systems engineering as a continuous process that is 1) iterative across the full life of a system and 2) managed through a digital transformation centered on data and models. This article also discusses the value of "shared and authoritatively managed data and models" in the lifecycle of future systems. These together present a modernized view of systems engineering where "seamless and efficient transfer of data and models" will support practices that are "more agile and responsive to changing stakeholder needs."

"How Large Scale Agile Can Operate Systems Engineering in the Future" by Laurent Alt and Mikaël Le Mouëlli describe why it is important to make agile and systems engineering work together, how to do it, and how this impacts how we see value, systems, digital twins, and leadership.

"Model-based Systems Engineering as an Enabler of Agility" by Sophie Plazanet and Juan Navas illustrate how model-based systems engineering (MBSE) may be an effective enabler of agility in systems engineering, focusing on dynamic learning and evolution per the FuSE roadmap, with a fictive testimony interview of a system engineer based on a fictive example of a drone-based product for inspection of electrical networks using a "warm-up/run/evaluation" process.

"Agile MBSE: Doing the Same Thing we Have Always Done, But in an Agile Way with Models" by Matthew Hause examines some of the aspects of MBSE, specifically the systems modeling language (SysML®), and shows how an agile approach to MBSE can help with the concepts of stakeholder engagement, continual integration, and dynamic learning and evolution.

"FuSE Agility as a Foundation for Sound MBSE Lifecycle Management" by

JUNE 2023
VOLUME 26/ ISSUE 2

Barry Papke, Matthew Hause, and David Hetherington describes how three FuSE agility foundation concepts (system of innovation, effective stakeholder engagement, and continuous integration) directly address some of the problems seen in adoption, deployment, and sustainment of the MBSE digital environment as a system of interest.

"An Agile Systems Engineering Process for Stakeholder Needs Identification and Solution Concept Design" by Lymari Castro presents a case study where an agile systems engineering process was used to identify stakeholder needs to design an improved cross-organizational proposal development process during the proposal formulation phase of a program.

"Applying Agility for Sustainable Security" by Larri Rosser describe how the broadly adopted technical processes from the ISO/IEEE/IEC 15288:2015 standard can be executed using agile methods to realize a large complex solution. Specific recommendations are provided for executing these processes in a manner that enables systems to be sustainably secure — that is, to retain the desired level of security throughout the life cycle.

"Agile Programs Need Agile Reviews" by Larri Rosser explores ways to provide insight and responsive forward looking actionable guidance for agile projects in the context of government and defense programs. It proposes a general oversight approach that produces minimal drag and disruption and keeps pace with agile product development.

We end this issue of *INSIGHT* with an application article by Michael A. Cabrera and Steve Simske titled "Project Lifecycle for a Next Generation Space Suit Project" that describes the modified agile concept (MAC) and its multi-disciplinary approach to a sampling of various lean and agile methods integrated alongside traditional, waterfall methods (such as a hybrid model) to support the hypothesized project lifecycle development. This approach was developed as part of a case study with a design and test team responsible for building test stations to qualify components of the life support system on the next generation space suit. This article outlines exclusively the scrum and lean methods in the MAC with a cursory overview on kanban development supporting the MAC.

We hope you find *INSIGHT*, the practitioners' magazine for systems engineers, informative and relevant. Feedback from readers is critical to *INSIGHT*'s quality. We encourage letters to the editor at insight@incose.net. Please include "letter to the editor" in the subject line. *INSIGHT* also continues to solicit special features, standalone articles, book reviews, and op-eds. For information about *INSIGHT*, including upcoming issues, see https://www.incose.org/products-and-publications/periodicals#INSIGHT. For information about sponsoring *INSIGHT*, please contact the INCOSE marketing and communications director at marcom@incose.net. ■

INSIGHT Special Feature

Setting Current Context for Agility in the Future of Systems Engineering

Rick Dove, dove@parshift.com

ABSTRACT
Agility in the future of systems engineering (FuSE) is one of the topic areas under the INCOSE FuSE initiative. A roadmap for near-term improvement, presented at the 2021 INCOSE International Symposium, offered nine strategic concepts appropriate and ready for further movement toward standard practice. Initial work in that direction enticed several practitioners and researchers to address selected concepts in this special issue of the INCOSE *INSIGHT* publication. The purpose of this lead-off article is to provide a contextual backdrop for the articles that follow.

INTRODUCTION

The future of systems engineering (FuSE) is an INCOSE facilitated systems community collaborative initiative that identified several specific project areas to pursue. For the FuSE agility area, a collaborative team was formed with representation from INCOSE's Agile Systems & Systems Engineering Working Group, Lockheed Martin, NASA, Northrop Grumman, Raytheon, and US Department of Defense. Team workshops held biweekly from June to November in 2020 deliberated on objectives and appropriate strategic foundation concepts for near-term systems engineering agility improvement; and assembled the concepts as a synergistic roadmap (Figure 1) suitable for immediate development and deployment attention (Willett et al. 2021).

Figure 1 links the foundation concepts to the objectives in a strategic activity web of non-dependent synergistic relationships. Linkage lines have no arrowheads as objectives give purpose to concepts and concepts give means to objective accomplishment. The purpose of the linkage display is to show principal relationships among concepts and objectives; encouraging developers and implementers to emphasis and strengthen these relationships. As concepts get developed and implemented, additional links will emerge.

Figure 1 is not intended to depict a comprehensive agility strategy, but rather a set of foundation concepts for agility improvement appropriate for the near term.

More recent work socializes the roadmap concepts and attempts to instigate strategy and practice development. One activity toward those ends is this issue of *INSIGHT* magazine, with a series of articles exploring one or more of the foundation concepts in

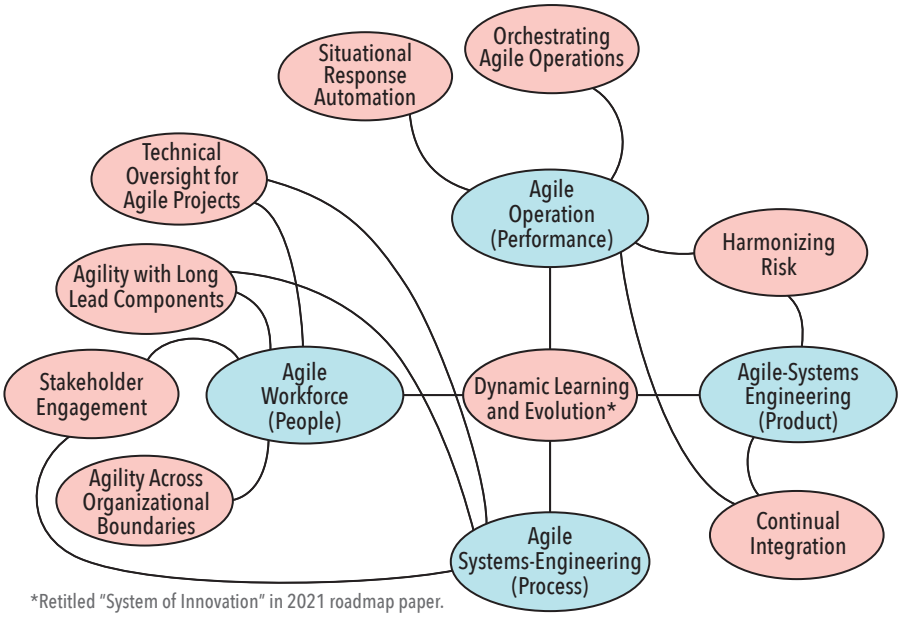


Figure 1. Synergistic linkage among nine strategic foundation concepts and four objectives

Table 1. Brief synopsis of FuSE agility 2021 roadmap concepts

Concept	General Problems to Address	General Needs to Fill	General Barriers to Overcome
Dynamic Learning and Evolution	Insufficient learning and knowledge management processes; barriers to learned-knowledge application.	Situational awareness and learning embedded in lifecycle processes; timely/affordable learning-application; knowledge management.	Unclear what to do or where to do it beyond learning ceremonies and contract obligation satisfaction.
Technical Oversight	Traditional technical oversight methods are counterproductive in agile programs.	An interactive approach that reveals relevant knowledge for guidance and decision making.	Oversight traditions; standard contract wording; disrespect for oversight.
Stakeholder Engagement	Timeliness and depth of stakeholder collaborative engagement.	Discovery of true requirements and integration conflicts.	Time involved; travel cost; inconvenient scheduling; lack of motivation.
Agility Across Organizational Boundaries	Incompatible siloed cultures and languages.	Common language; less handoffs; product-based teams; common metrics.	Functional organizational silos.
Agility with Long Lead Components and Dependencies	Components and external dependencies with long lead times complicate schedule coordination and disrupt technical performance.	Scheduling and acquisition techniques that better align with agile-SE principles.	[False] justification that long-lead items prohibit the use of agile-SE.
Continual Integration	Late discovery of integration and requirements issues.	Minimize risk and rework with fast learning; maximize stakeholder engagement.	Development effort and expense; technologies for integrating/testing software before hardware is ready.
Orchestrating Agile Operations	Coherence among loosely coupled multi-actor outcomes.	Dynamic operational coordination in real-time.	Ability to encode self-learning; adaptive logic as decision-support for people and for autonomous decision making.
Situational Response Automation	Decision and action too slow.	Continual dynamic adaptation within cyber-relevant time.	Complicatedness of encoding autonomous governance and adjudication logic and rules; situational awareness that provides necessary inputs.
Harmonizing Risk in Agile Operations	Agility focus is principally loss avoidance	Expand awareness and operational realization of both the negative side of risk (loss) and the positive side of risk (opportunity, seek gain, optimize).	Silo-thinking and predominance of looking at risk only in terms of loss.

a variety of systems engineering contexts. The purpose of this lead-off article is to provide a contextual backdrop for the articles that follow.

OBJECTIVES AND CONCEPTS

FuSE agility objectives and strategies will continuously evolve. The initial team identified four objectives as timely and appropriate:

1. Agile systems-engineering (adaptable processes).
2. Agile-systems engineering (adaptable products)
3. Agile operations (adaptable performance)
4. Agile workforce (adaptable people).

All of these objectives have some limited or narrow-domain practice; but none are in standard practice.

Criteria for foundation concepts was established as follows:

- Concept has relevance to systems engineering considerations.
- Concept can provide new and useful value to the state of practice.
- Concept value proposition articulation is in systems engineering terms.
- Concept has notional support in a referenceable knowledge base.
- Concept does not yet have sufficient published exposure for broad-based actionable systems engineering consideration.
- Concept implementation could be now.
- Concept is principally about what to achieve and why (strategic intent), rather than how (prescriptive tactics), though notional examples of how can augment understanding.

A brief synopsis of the concepts is in Table 1. The team developed the entries in Table 1 as general notions to help orient the nature of each concept. The team did not and does not intend to limit or constrain concept-development thinking, rather, to

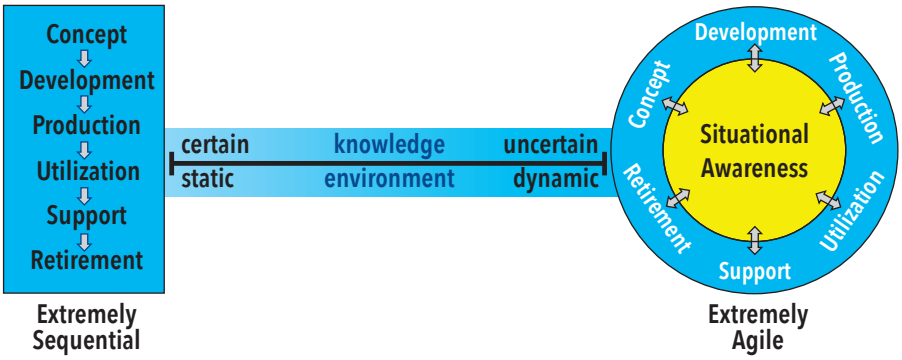


Figure 2: Systems engineering lifecycle spectrum – sequential to agile

point the thinking in the intended direction.

IN CONCLUSION

The roadmap concepts address four objectives. One objective is called out as *agile systems-engineering*, the others are there, from our FuSE perspective, to enable and support agile systems-engineering (hyphens to distinguish the process from the product objective).

The roadmap is about agility in the future of systems engineering – it was created by people who have already started down that road, people with experience in agile systems engineering who have discovered where the pavement ends and the going gets rough.

Agile systems engineering is a principle-based method for designing, building, sustaining, and evolving systems when knowledge is uncertain and/or environments are dynamic.

Agile systems engineering is best understood in contrast to sequential systems engineering in how the two relate to the system life cycle spectrum. Figure 2 shows pure forms of these two life cycle models in terms of their activity phases and data flows. All systems engineering life cycle models fall somewhere between the two ends of the spectrum, depending upon the process-encoded degree of attentiveness and responsiveness to dynamics in knowledge and environment. ■

REFERENCE

- Willett, K. D., R. Dove, A. Chudnow, R. Eckman, L. Rosser, J. S. Stevens, R. Yeman, and M. Yokell. 2021. "Agility in the Future of Systems Engineering (FuSE) – A Roadmap of Foundational Concepts." Paper Presented at the 31st Annual INCOSE International Symposium, Virtual: 17-22 July.

ABOUT THE AUTHOR

Rick Dove is an unaffiliated independent operator. He chairs the INCOSE working groups for Agile Systems and Systems Engineering, and for Systems Security Engineering; and leads both the security and agility projects for INCOSE's initiative on the future of systems engineering (FuSE). He is an INCOSE Fellow, and author of *Response Ability, the Language, Structure, and Culture of the Agile Enterprise.*

Systems Engineering Agility in a Nutshell

Rick Dove, dove@parshift.com; **Kerry Lunney,** kerry.lunney@thalesgroup.com.au; **Michael Orosz,** mdorosz@isi.edu; and **Mike Yokell,** mike.r.yokell@gmail.com

■ **ABSTRACT**
Systems engineering must necessarily have the agility to anticipate and effectively respond to an increasingly dynamic and uncertain environment. Agile systems engineering, agile software engineering, and agile any-kind-of engineering share common goals and leverage common agility-enabling strategies. This article succinctly describes eight strategic aspects with application discussions at the systems engineering level.

CRACKING THE SHELL

Agile software development has pioneered and proliferated methods for managing software projects (for example, Scrum etc.) and engineering software products (for example, XP etc.) when knowledge is uncertain, and environments are dynamic. The success of these approaches is challenging other engineering disciplines to find better ways to navigate their development activities through similar uncertainties and dynamics.

Agile software development methods (process tactics) necessarily leverage the nature of software engineering. A software product is created by engineers who are supported by an integrated hierarchy of many tools (computers, code compilers, user interfaces, development platforms, etc.) that gives them fast turn-around control over design, fabrication, and verification. Piecewise functional prototypes can be created and tested in minutes and deployed into evolving user product in hours and days.

Contrast that with electronic printed circuit board (PCB) development — procured parts, separate design and fabrication engineers, custom mechanical enclosure designs, procurement interaction, and supply chain issues. Oversimplified, but the nature of engineering activity and concerns is clearly very different. Making a PCB engineering process more agile would necessarily use different methods than software development. Nevertheless, those methods would have the same fundamental goals: reduce the adverse effects of uncertain knowledge and dynamic environments.

While tactical methods necessarily vary among different engineering domains (the how part), strategies for achieving common goals (the what and why parts) are domain independent. This article offers eight strategic aspects (Figure 1) that individually can improve the agility of engineering in any domain as well as at the systems engineering level. The next two pages present all eight strategic aspects, each in terms of needs (why) and behaviors (what), with a discussion of application from the systems engineering point of view.

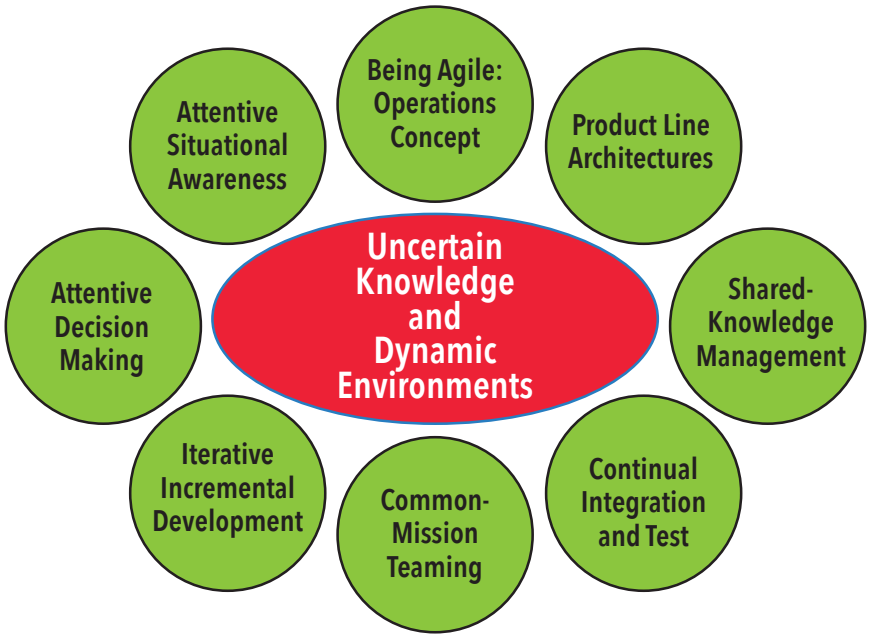


Figure 1. Eight strategic aspects of agile engineering

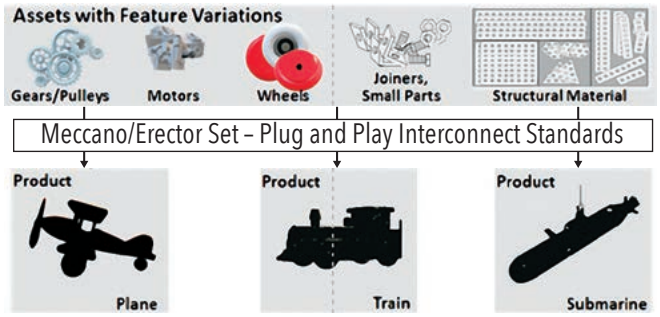
Product Line Architectures

Needs: Facilitated product and process experimentation, modification, and evolution.

Behaviors: Composable and reconfigurable product and process designs from variations of reusable assets.

Discussion: One fixed process approach won't fit all projects, so an appropriate process should be easy to compose and evolve according to context and usage experience. Variations of reusable assets are built over time as features are modified for different contextual usage.

A hallmark of agile systems engineering is iterative incremental development, which modifies work in process as suitability is repetitively evaluated. The agility of the process depends upon the agility of the product – so both process and product can be easily changed.



Notational Agile Architecture Pattern

Attentive Situational Awareness

Needs: Timely knowledge of emergent risks and opportunities.

Behaviors: Active monitoring and evaluation of relevant internal and external operational-environment factors.

Discussion: Are things being done right (internal awareness) and are the right things being done (external awareness)? Having the agile capability for timely and cost-effective change does little good if you don't know when that ability should be exercised. Situational awareness can be enhanced with systemic methods and mechanisms.

Alert in-the-moment attention

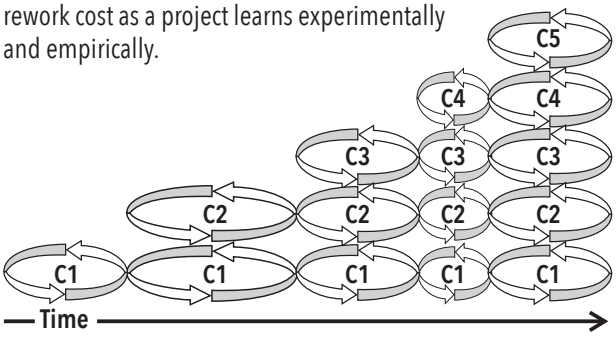
Iterative Incremental Development

Needs: Minimize unexpected rework and maximize quality.

Behaviors: Incremental loops of building, evaluating, correcting, and improving capabilities.

Discussion: Generally increments *create* capabilities and iterations add and augment features to *improve* capabilities.

- Increment cycles are beneficially timed to coordinate events such as integrated testing and evaluation, capability deployment, experimental deployment, or release to production.
- Increments may have constant or variable cadence to accommodate management standards or operational dynamics.
- Iteration cycles are beneficially timed to minimize rework cost as a project learns experimentally and empirically.



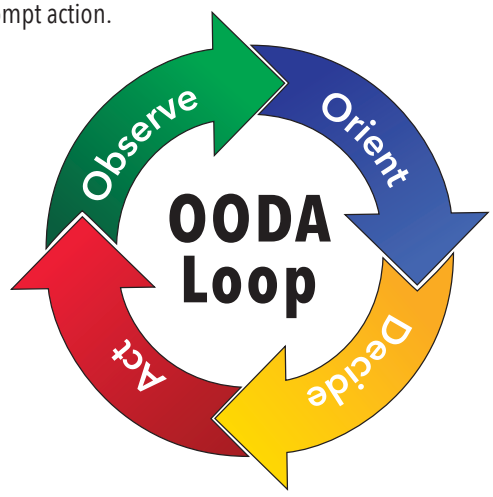
Iterative capability improvements (looping) and incremental capability additions (successive development periods)

Attentive Decision Making

Needs: Timely corrective and improvement actions.

Behaviors: Systemic linkage of situational awareness to decisive action.

Discussion: Empower decision making at the point of most knowledge. As a counter example, technical debt (a term for knowing something needs correction or improvement but postponing action) is situational awareness without a causal link to prompt action.



John Boyd's OODA loop

Common-Mission Teaming

Needs: Coherent collective pursuit of a common mission.

Behaviors: Engaged collaboration, cooperation, and teaming among all relevant stakeholders.

Discussion: Collaboration, cooperation, and teaming are not synonymous, and need individual support attention. Collaboration is an act of relevant information exchange among individuals, cooperation is an act of optimal give and take among individuals, and teaming is an act of collective endeavor toward a common purpose.

Tightly integrated coherent operation

Continual Integration & Test

Needs: Early revelation of system integration issues.

Behaviors: Integrated demonstration and test of work-in-process.

Discussion: Discovering integration issues late in development activities can impact cost and schedule with major rework. Synchronizing multiple domain engineering activities via continual integration and test provides faster and clearer insight into potential system integration issues.

SpaWar iteratively evolving unmanned technology integration platform

Shared-Knowledge Management

Needs: Accelerated mutual learning and single source of truth for internal and external stakeholders.

Behaviors: Facilitated communication, collaboration, and knowledge curation.

Discussion: There are two kinds of knowledge to consider. Short time frame operational knowledge: what happened, what's happening, what's planned to happen. Long time frame curated knowledge: what do we know of reusable relevance, e.g., digital artifacts, lessons learned, and proven practices.

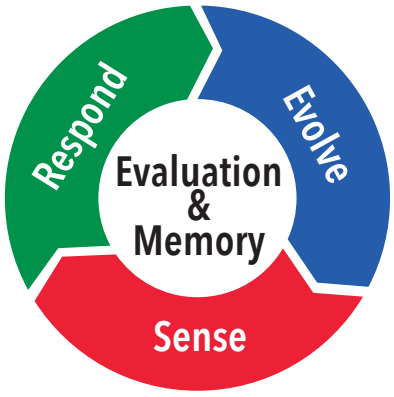
Depicted books represent informational containers of any kind; but typically digital

Being Agile: Operations Concept

Needs: Attentive operational response to evolving knowledge and dynamic environments.

Behaviors: Sensing, responding, evolving.

Discussion: Agile systems engineering is not about doing Agile, it is about being agile. Being agile is a behavior, not a procedure – a behavior sensitive to threats and opportunities in the operational environment, decisive when faced with threat or opportunity, and driven to improve these capabilities. Deciding how to implement any of the core aspects, even this one, should be done with sense-respond-evolve principles in mind as aspect objectives.



Three principles that operationalize agility

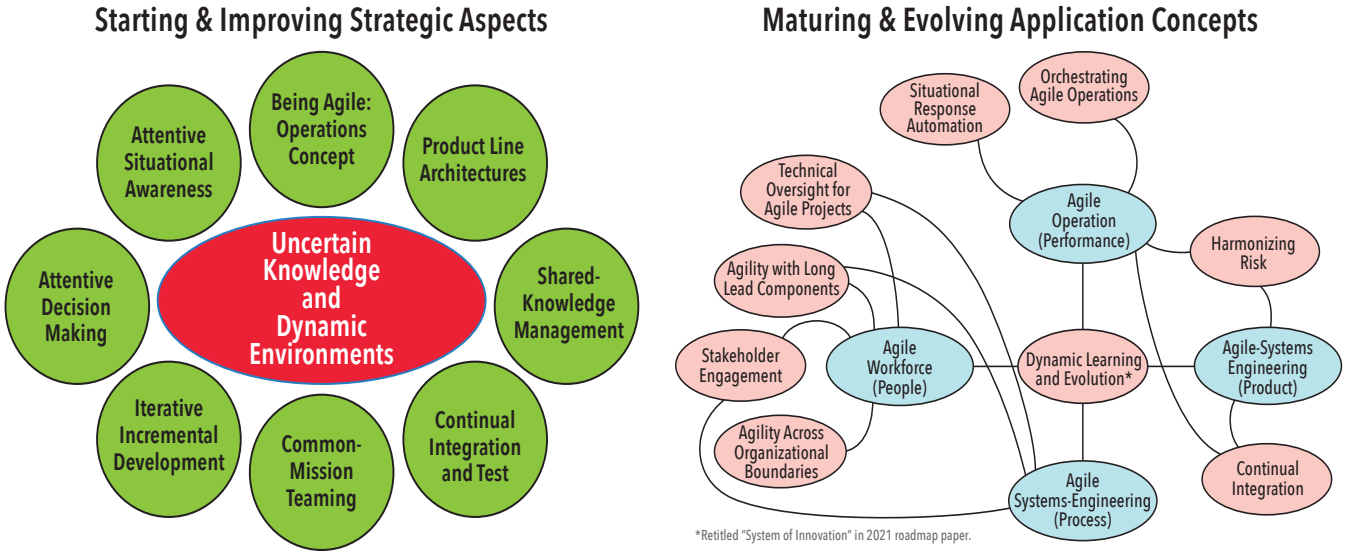


Figure 2. Large organizations likely have units working in both early and advanced stages

The succinctness of the descriptions and the display on two pages is done with purpose. Descriptive content attempts to be sufficient to inform and direct application intent without overly constraining approaches compatible with culture, organizational readiness, and possible contract constraints. This two-page brief can function as a personal things-to-consider scope reminder or as a whole-picture guide for collaborative discussion or improvement.

Each of the aspects can individually improve capability to deal with uncertain knowledge and dynamic environments in any engineering process; but to have something intended as an agile engineering process at either domain or system level requires multiple aspects operating in concert. Individual aspects are strategic concepts that can tactically manifest over a range of intensity. Thus, the degree of agility is a product of how many of these aspects are operational as well as how effectively each one contributes to the agility required by the operating environment.

These eight aspects in their current form have emerged from the pooled knowledge of the authors of this article – knowledge gained from their experiences in case study work, university research work, and responsibilities for organizational systems engineering processes and practices. None of these aspects are new concepts. What is new is the amalgamation organized as domain independent fundamental strategies for engineering when knowledge is uncertain and operating environments are dynamic.

Figure 2 depicts the relationship between the eight strategic aspects presented here and the nine foundational concepts in the roadmap developed for agility in the future of systems engineering (Willett et al. 2021). Maturing and evolving the concepts on the right side will leverage the aspects on the left side.

Whether your organization is down the road already or just thinking about the values of being more agile, each of the aspects likely has some form of practice in place already. One way to inspire actionable awareness of the collective view beyond theory is to develop and share a short case study — one that shows each aspect in real practice instances somewhere in your organization. ■

REFERENCES

- Dove, R., K. Lunney, M. Orosz, and M. Yokell. 2023. "Agile Systems Engineering – Eight Core Aspects." Proceedings International Symposium. International Council on Systems Engineering. Honolulu, US-HI, 15-20 July. www.parshift.com/s/230715IS23-AgileSE-EightCoreAspects.pdf.
- Willett, K. D., R. Dove, A. Chudnow, R. Eckman, L. Rosser, J. S. Stevens, R. Yeman, and M. Yokell. 2021. "Agility in the Future of Systems Engineering (FuSE) – A Roadmap of Foundational Concepts." Paper Presented at the 31st Annual INCOSE International Symposium, Virtual: 17-22 July.

ABOUT THE AUTHORS

Rick Dove is an independent researcher, systems engineer, and project manager generally focused on the systems agility and systems security areas. He chairs the INCOSE working groups for Agile Systems and Systems Engineering and for Systems Security Engineering. He leads both the agility and security project areas for INCOSE's future of systems engineering (FuSE) initiative. He is an INCOSE fellow, and author of *Response Ability — the Language, Structure, and Culture of the Agile Enterprise.*

Kerry Lunney is the country engineering director and chief engineer in Thales Australia. She has extensive experience developing and delivering large system solutions, working in various industries including ICT, gaming, financial, transport, aerospace and defence, in Australia, Asia, and US. She also participates in several global working groups and research projects. Kerry is a past president INCOSE, and holds the expert systems engineering professional (ESEP) qualification. She is also an INCOSE fellow, and a fellow of engineers Australia with the status of engineering executive and chartered professional engineer.

Dr. Mike Orosz directs the Decision Systems Group at the University of Southern California's Information Sciences Institute (USC/ISI) and is a research associate professor in USC's Sonny Astani Department of Civil and Environmental Engineering. Dr Orosz has over 30 years' experience in government and commercial software development, systems engineering and acquisition, applied research and development, and project management and has developed several successful products in both the government and commercial sectors.

Dr. Mike Yokell is a leader in systems engineering in the US aerospace and defense industry. He has been the US representative to international standards-setting bodies for systems and software engineering and was the project editor for two new international standards on systems of systems engineering. Mike is certified as an expert systems engineering professional by INCOSE. He holds multiple US and European patents for model-based systems engineering and large-scale immersive virtual reality.

JUNE 2023
VOLUME 26/ ISSUE 2

The Supra-System Model

Tom McDermott, tmcdermo@stevens.edu; **Kelly Alexander,** kelly.d.alexander12.ctr@mail.mil; and **Richard Wallace,** richard.wallace@gtri.gatech.edu

■ **ABSTRACT**

This article presents an initial set of concepts resulting from research by the Office of the Undersecretary of Defense for Research and Engineering (OUSD/RE) and the Systems Engineering Research Center (SERC) under an initiative called "systems engineering modernization" (SEMOD). This article discusses the "supra-system model," which evolved as a different view of systems engineering lifecycle activities across the entire life of an engineered system. This view promotes systems engineering as a continuous process that is 1) iterative across the full life of a system and 2) managed through a digital transformation centered on data and models. This article also discusses the value of "shared and authoritatively managed data and models" in the lifecycle of future systems. These together present a modernized view of systems engineering where "seamless and efficient transfer of data and models" will support practices that are "more agile and responsive to changing stakeholder needs."

INTRODUCTION

In this article, we propose a systems engineering lifecycle model called the "supra-system model" as a visual means to adjust how we think about how systems of any type are developed and modified across their entire lifetimes. This supra-system model is not a replacement or counter to other established lifecycle models, it is intended to augment other models. Since the 1970s, systems engineering practice, particularly its use in defense acquisition programs, has followed a mental model that the system's lifecycle is fully aligned with the engineering and program management activities that define, realize, and deploy it. This article promotes a more holistic view that the actual lifecycle of the system is composed of many engineering and program lifecycles to recognize that data and models need to be considered as part of the actual lifecycle of the system, not just a program or engineering cycle. Finally, this model aims to formalize the external lifecycle associated with the supra-system, which must be considered a primary contributor to system data and models.

This article is also not an argument for or against digital model-based systems engineering (MBSE) but recognizes that data is independent of models, and models have value independent of systems. These must be managed across entire system lifecycles. SEMOD conceptualizes a view of systems engineering that has at its core "shared and authoritatively managed data" that can be transformed through various models and tools to create digital decision artifacts and enduring virtual system representations. This fundamental change driver is well described in the DoD Digital Engineering Strategy (Office of the Deputy Assistant Secretary of Defense for Systems Engineering 2018) and the *Systems Engineering Vision 2035* (INCOSE 2035), but these documents do not clearly articulate the value of "model-based" and "digital data" well enough. The SEMOD project found:

> The **value** of systems engineering modernization will be realized through a more seamless and efficient transfer of data and models, starting from underlying performance drivers through models to decisions and ease of drilling back down from decisions to data.

In the early years, systems engineering artifacts were largely paper documents or drawings, and now they are mostly based on digital technologies but far from "seamlessly integrated and efficient." There also remained the question of where these data and models come from and how they live their life. This was the question that led to the supra-system model.

The SEMOD research team realizes that existing systems engineering mostly linear lifecycle depictions like the "Vee" model and the DoD's "Defense Acquisition Wall Chart" do not promote the future vision of data and models at the core of systems engineering and acquisition. The lifecycles of the data and models are associated with but not necessarily the same as the lifecycle of the realized system. We found:

> New systems engineering lifecycle processes must evolve that address shared and authoritatively managed sets of digital data and models associated with the system's entire lifecycle, not just a single engineering or program lifecycle.

In addition, newer systems engineering subdisciplines like software systems engineering, information technology, enterprise architecture, distributed modeling & simulation, and automated manufacturing systems view lifecycle process and technical review as much more iterative than what is implied by current systems engineering guidance. Therefore, we found that the mission of

systems engineering modernization, contrary to much of the published digital engineering literature, should focus less on models and more on increasing responsiveness by promoting lifecycle processes that increase the number of iterations and shorten the cycle time between them. This led to our vision statement:

> The vision of systems engineering modernization is to use data and models to create system engineering practices that are more agile and responsive to changing stakeholder needs.

This article describes the rationale for systems engineeering modernization, and then what we found to be the value drivers for digital and model-based engineering initiatives. The article then describes the concept of the "supra-system model," its underlying theory, and its potential use in modernized systems engineering practice. Finally, the article concludes with how these might address some gaps between agile practice in the software world and agile systems engineering.

The full SEMOD research program identified a set of pain points and then a set of roadmaps that reflect modernization steps specific to systems engineering in US defense acquisition programs. These have not yet been generalized to all applications of systems engineering but can be accessed in the published SERC reports (McDermott and Benjamin 2022, McDermott, Mesmer, and Ergin 2023).

WHY SYSTEMS ENGINEERING MODERNIZATION?

Today systems are not only physical; they are software-intensive, highly connected, and have extensive automation and user configuration capabilities. Software engineering became a discipline in 1967, manufacturing automation (the third industrial revolution) began in the 1970s, and the World Wide Web was invented in 1989. In addition, the DoD's Defense Modeling and Simulation Office was opened in the early 1990s, and large-scale networked simulation of defense systems followed. These have continued to evolve the systems engineering discipline, not as a whole, but as sets of related subdisciplines (systems engineering, software systems engineering, information technology, and enterprise architecture, distributed modeling & simulation, and automated manufacturing systems).

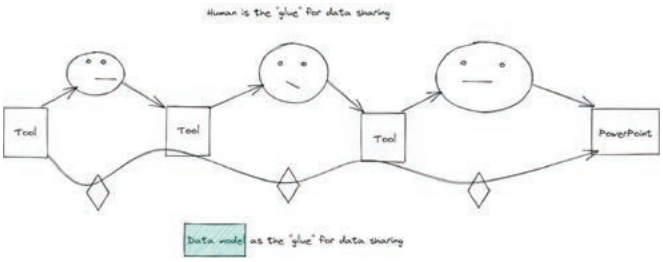
Following the successful evolution of the unified modeling language (UML®) in the software discipline, the systems modeling language (SysML®) was published in 2007. It started the growth in model-based systems engineering (MBSE) as an improved approach to managing technical and programmatic risk. "Industry 4.0" originated in 2011 and introduced the concept of a "digital twin" as a non-physical product realization. The DoD's digital engineering (DE) strategy was published in 2018, ushering in the vision of a digital era of systems engineering.

Throughout this change, the historical background of systems engineering has continued to be linked with the physical realization of large complex systems and other critical capabilities that are intended to persist for many years. The need for rigorous definition, analysis, and testing of these critical systems will always exist, but the lifecycle processes we choose to use must be tailored to the system's actual use and life. Modernized systems engineering discipline needs to be more model-based, agile, and responsive, which will be accomplished with more efficient lifecycle processes. Fundamentally, modernized systems engineering practices and lifecycle processes must define how data and models can be used to be more iterative and responsive to user needs. In this project, we found that it is not the mental model or vision of current policy and guidance related to these focus areas. Our systems engineering modernization vision is stated below:

> The **vision** of systems engineering modernization is to use data and models to create system engineering practices that are more agile and responsive to changing stakeholder needs.

DIGITAL TRANSFORMATION OF SYSTEMS ENGINEERING

At the core of systems engineering modernization is "shared and authoritatively managed data" that can be transformed through "shared and authoritatively managed models" and related tools to create digital artifacts that can be used by various decision-makers and others needing digital access to the design and descriptions of the system across its lifetime. These artifacts were almost always paper documents or drawings in the early years. Now they are based on digital technologies but far from "seamlessly integrated and interoperable." The cartoon in Figure 1 might best describe the current state of digital artifact development.



Digital Artifact – An artifact produced within, or generated from, the engineering ecosystem. These artifacts are generated **through transformation of data and models into views** in order to visualize, communicate, and deliver data, information, and knowledge to stakeholders.

Figure 1. Data transformation mental model (McDermott and Benjamin 2021) technologies but far from "seamlessly integrated and interoperable."

In the figure, the diamonds on the bottom represent data connections and transfers, as opposed to human connections and transfers at the top. Systems engineers have long used digital data and various modeling and analysis tools to produce digital artifacts for decision-making (such as PowerPoint slides). We do not see this flow as changing in modernized systems engineering. However, the underlying data models have not been "seamlessly shared" or likely not shared at all, and authority for that data has often been held by independent activities generally organized by discipline. Today, much of the "transformation" is still a manual interpretation of disparate data and analyses. This manual interpretation limits our ability to be iterative and responsive across disciplines and disciplinary tools. It is inefficient and also non-holistic. One might describe the current state of systems engineering as seeing the whole while looking through a set of soda straws. We desire a fully integrated, iterative workflow where the system is the focus, not the owner of the data or the particular element of a design. Today's primary challenge in digital engineering is not so much being "model based." It is understanding and creating the underlying data model that integrates across requirements, design, test, disciplines, and disciplinary processes, with it being shareable and shared.

This leads us to the **value statement** for systems engineering modernization, depicted in Figure 2 and the box below:

> The **value** of systems engineering modernization will be realized through a more seamless and efficient transfer of data and models, starting from underlying performance drivers through models to decisions and ease of drilling back down from decisions to data.

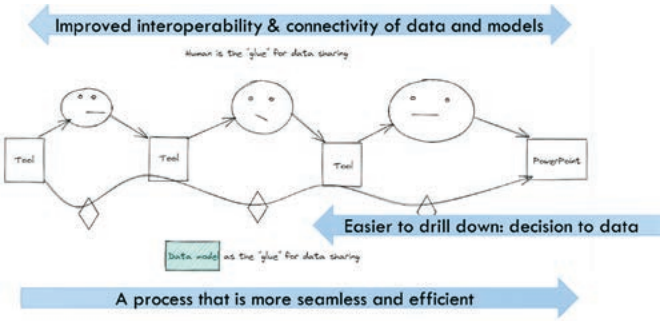


Figure 2. Systems engineering modernization value depiction

Systems engineering and related acquisition processes can be visualized as a set of iterative data transformations from sources of truth that produce artifacts for human consumption — across all stages of a system life cycle. Figure 3 redraws the widely depicted define→realize→deploy & use stages of the systems engineering lifecycle process stages in a circular process to represent it as a:

1. set of data transformations at the core;
2. layered across disciplines and tasks;
3. in continuous iterative processes that could be entered from any point.

In the figure, we generalize define, realize, and deploy as a "learn→build→measure" to be more consistent with current design literature.

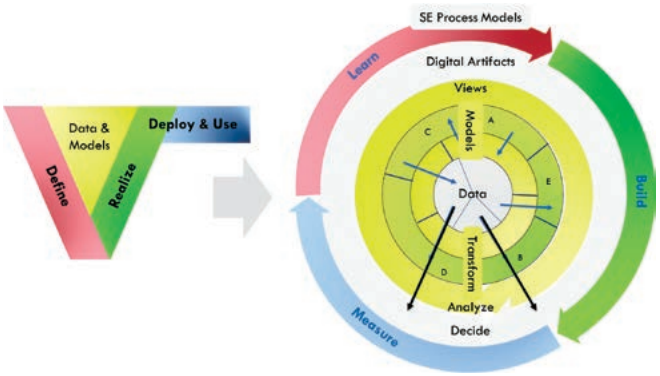


Figure 3. Circular processes with data at the core

In systems engineering technical and management processes, data is transformed through models into views, which support analyses leading to decisions. These transformations have traditionally produced decision artifacts severed from the underlying data and models and captured in independent static document or presentation forms. Digital artifacts may still be documents or presentable views but should remain digitally connected to the underlying data and models from which they draw context and explainability. This process flow reflects "data transformed into models then analyzed through views to make decisions documented in digital artifacts." This process flow has been the core of systems engineering technical and management processes within each lifecycle phase since the inception of systems engineering. It was a largely manual, inefficient process flow focused on presentability rather than context.

As defined by ISO/IEC/IEEE 15288, systems engineering lifecycle processes do not define a specific ordering of process areas, but much of the literature and existing mental models imply a process ordering that is started in the learn (define) stages (ISO/IEC/IEEE 2015). Systems engineering lifecycle processes have been used not just in critical systems where up-front system definition and learning are essential but also in process and system innovation, prototyping, and incremental definition activities where build-first is the pathway to learning; and in sustainment life cycles where deployed system measurement and learning should apply to both the system sustainment, but also to define the next build. This SEMOD circular mental model better recognizes that systems engineering technical and management processes can be applied to any life cycle in any type of system. Figure 4 visualizes the domains of systems engineering in association with the ordering of learn, build, and measure cycles.

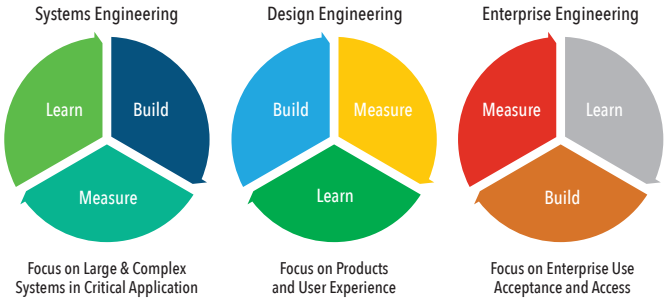


Figure 4. Different lifecycle ordering in different applications of systems engineering

Systems engineering modernization's challenge is maintaining appropriate systems engineering rigor and associated process definition in all discipline applications. **We contend that systems engineering rigor is maintained using the data→transform→analyze→decide flow of Figure 3, not through a specific ordering of SE processes**.

The workflow view in Figure 5 (on the next page), shows conceptually how shared and authoritatively managed data is transformed into digital artifacts in different life cycle stages in any pathway. This linear workflow model is familiar and comfortable to system engineers but does not visually represent that these data transformations into and out of the shared and authoritatively managed federations of data and models actually happen iteratively and continuously across the entire life of a system. This will be a distributed federation of data and models. These data and models might originate in any phase of a system's lifecycle and any function associated with engineering and management. In fact, this will always be the case. **Increasing responsiveness does not mean eliminating these critical systems engineering processes, just increasing the number of iterations and shortening the cycle time between them**. Also, "who owns the data and models" remains a pain point in this transformation.

Figure 5 is particularly relevant to systems engineering modernization, as "data management" is not currently defined as a disciplinary process in systems engineering standards. Data models and data storage systems are separate systems that must also be developed and deployed in support of the fielded system. These must be defined and built along with other system development aspects. These also have their own lifecycles.

This led us to the need for a new visually representative model for systems engineering modernization, which must address how shared and authoritatively managed data and models are defined, built, deployed, and used in systems:

> New systems engineering lifecycle processes must evolve that address shared and authoritatively managed sets of digital data and models associated with the system's entire lifecycle, not just a single engineering or program lifecycle.

In our interviews and workshops on this project, we found that the terms data, digital models, digital artifacts, digital threads, and

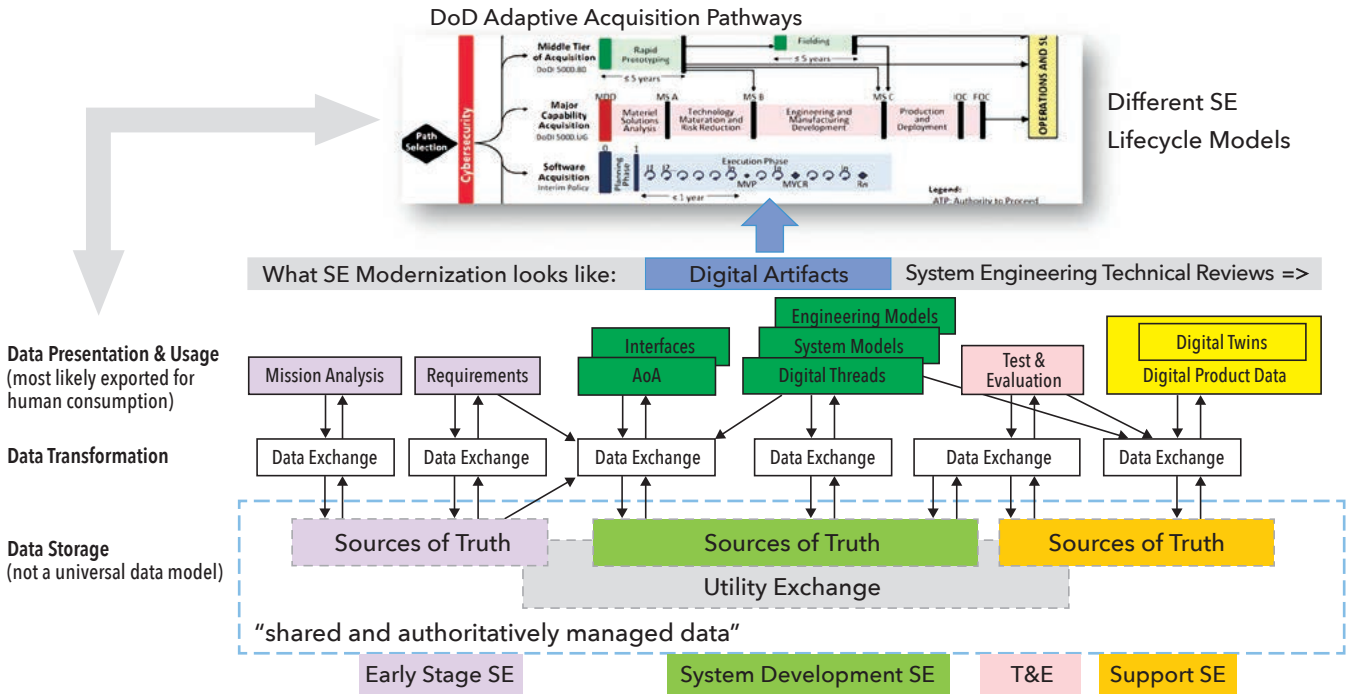


Figure 5. Data transformation into the life cycle

virtual systems, or "digital twins," have different definitions, uses, and driving forces behind their lifecycles. As a result, they are not being viewed in an integrated set of lifecycle and process models. In response, we developed a more integrative view of an systems engineering lifecycle model called "the supra-system model." This model was created as a discussion tool to distinguish historical systems engineering lifecycle and process models from a modernized approach needed to support today's activities and systems.

THE SUPRA-SYSTEM MODEL

Thullier and Wippler, in their chapter "Finding the Right Problem" from the book *Complex Systems and Systems of Systems Engineering*, caution us always to consider three lifecycles associated with any system, each with interdependencies and relative positions in the evolution of a system (Thullier and Wippler 2011):

- "*the system lifecycle*: the "experiences" of the system itself;
- *the program lifecycle* of the system: the rhythm of the project during study, development, production, etc. of the system;
- *the engineering cycle*: the processes and activities involved in engineering the system."

ISO/IEC/IEEE 15288:2015 refers extensively to "the life cycle of a system" and "stages of a system's life cycle" and then defines sets of system lifecycle processes used within organizations and projects (ISO/IEC/IEEE 2015). Historical systems engineering literature tends to portray system lifecycles and project lifecycles as simultaneous and combined. This may have been appropriate when most systems engineering activities were focused on large-scale physical systems, but with wider application of systems engineering system lifecycles, program lifecycles have become more distinct and separated in their purpose.

It is important to note that there are two established definitions of the term "lifecycle" (Merriam-Webster n.d.):

1. "the series of stages in form and functional activity through which an organism passes between successive recurrences of a specified primary stage" (multi-generational)
2. "a series of stages through which something (such as an individual, culture, or manufactured product) passes during its lifetime." (single generational)

Systems engineering and the "systems lifecycle" as defined by ISO/IEC/IEEE 15288:2015 and the Project Management Institute's (PMI) project lifecycle (Project Management Institute, n.d.) tend to follow the single generational lifecycle definition. Design engineering, software engineering, and enterprise engineering models tend to match the multi-generational lifecycle definition better.

15288 defines a set of process descriptions for affecting both the engineering cycles and project lifecycle, in the words of Thullier and Wippler. The "experiences of the system itself" will progress through a number of such engineering and program lifecycles. As systems engineering spreads broadly across all industries and applications, keeping these different lifecycles well distinguished is important.

Thullier and Wippler note that in the system's lifecycle, the "experience of the system" must be evaluated in periods and across "levels of temporal or time invariance." In their description, the actual lifecycle of a system progresses (experiences) from idea; to a virtual existence in models, documents, software, and today many digital artifacts; then to a physical existence. Systems engineering technical and management process divides these into stages. Systems engineering processes recognize "within each level [of abstraction], we may distinguish periods of time which we may observe the integrity of the structure and behavior of the system [as invariant]" (Thullier and Wippler 2011). We may use these periods to enable interdisciplinary and collaborative activities, referred to as phase gates or decision points. Virtual artifacts, by their nature, can cycle through more rapid periods of change than physical artifacts (Thullier and Wippler 2011). In other words, systems continually evolve but also have periods where their structure and behavior (virtual and physical) are invariant and support the establishment of baselines and review activities. We cannot assume that all types of systems and subsystems share common periods of invariance.

Thullier and Wippler also note that program lifecycle phases "are aligned (or mixed in) with key steps (or stages) of the system lifecycle. This allows us to fix program phases on integrated, coherent, and stable states of the system in question, and thus to make important decisions at precise moments in the life of the system" (Thullier and Wippler 2011). They further note that the engineering lifecycle is "the process that consists of moving from need…to an optimized solution – i.e., the best compromise integrating all constraints (cost/ time/performance) for the entirety of the phases and situations involved in the system lifecycle... This should not, however, be taken to mean that these processes must be carried out in a sequential manner" (Thullier and Wippler 2011). In other words, the idea that the system lifecycle, the program lifecycle, and engineering lifecycles can always be combined together is convenient but also a fallacy. There are "periods of temporal invariance" where we can view these lifecycles together to make crucial decisions; otherwise, they should be considered as independent. **Trying to force them to remain in lockstep limits our ability to be iterative and responsive**. This is perhaps the reason why agile/DevOps software practices have come to look and be regarded as so different from "traditional" systems engineering. For example, the necessary period of invariance to design and test the structural integrity of a physical aircraft wing will be much longer than the period to design and test a new software function or even a model of that wing.

Here, it is important to note that core systems engineering lifecycles and processes are not new; they have evolved in different ways since the first was envisioned in the 1960s. Stanley Shinners, in the 1967 book *Techniques of Systems Engineering* first introduced the concept of systems engineering as the methodological approach to define, realize, and deploy a system inherent in today's systems engineering lifecycle processes. Shinners defined these general techniques: understand the problem, consider alternative solutions, choose the most optimum design, synthesize the system, test the system, compare test results with requirements and objectives, and update the system characteristics and data (Shinners 1967). This process flow represents the basis for both classical systems engineering and software DevOps practice. The techniques are the same, only the process implementations are different. This is the engineering lifecycle that should be applied to all virtual and physical systems in any program management lifecycle. What is changing today with the advance of digital computing is how we maintain systems engineering rigor using the modernized data→ transform→ analyze→ decide flow of Figure 3 for any type of system, subsystem, or component.

There remains a fourth lifecycle that must also be considered in conjunction with the life of a system. Arthur David Hall, in *A Methodology for Systems Engineering* (1962), stated that systems engineering must consider the environment the system enters into: "The environment is the set of all objects outside the system: (1) a change in whose attributes affect the system and (2) whose attributes are changed by the behavior of the system." We cannot bind the system away from its external environment but must consider the experience of the system to be affected both by the technical and management processes that evolve the system and the external situations that seek to adapt the system (Hall 1962). In *General Systems Theory*, Ludwig von Bertalanffy noted that all systems could be divided according to levels of complexity into systems, supra-systems, and subsystems. The different levels interact and are not independent of each other (von Bertalanffy 1969). While the engineering lifecycle should be interested in decomposing the system into subsystems, the system itself should not be managed independently from its supra-system. The program lifecycle should ideally consider both subsystem and supra-system interdependencies. The supra-system is the next-level system that most closely interacts with the system of interest. Interestingly, "supra-system" is a commonly used term in social science but never used in engineering circles. Its time has come, as much of the authoritative data and models we are interested in actually describe system and supra-system boundaries.

Thus, there are four individual lifecycles that may affect the "experience of the system." These must be distinguished if we want a systems engineering process model that reflects any application of systems engineering with the rigor we have been accustomed to. One is the lifecycle of the system itself and potentially of the offspring it produces (both aspects of the lifecycle definition). Two others are the engineering and program or project lifecycles, which conduct processes internal to the system's life. Finally, the supra-system lifecycle reflects the direct experiences of the system itself in its operational context as related to the closest other systems it interacts with.

In addition to recognizing that each of the four lifecycle/process models may be individually relevant, each of these lifecycle processes must evolve to address shared and authoritatively managed sets of digital data and models associated with the entire lifecycle of the system itself, not just a single engineering or program lifecycle. Much of this data is contextual data in the supra-system. The established views that combine management processes/lifecycle and engineering processes/lifecycle do not fit well into the circular data-oriented mental model: technical (engineering) iterations and management (program) iterations in today's world have very different decision processes and respond to varying types of data, with some content overlap. Furthermore, systems engineering should be a holistic or systems-oriented problem-solving approach that reflects both the system and the supra-system, independent of the engineering cycle of a program. These are visualized together in Figure 6.

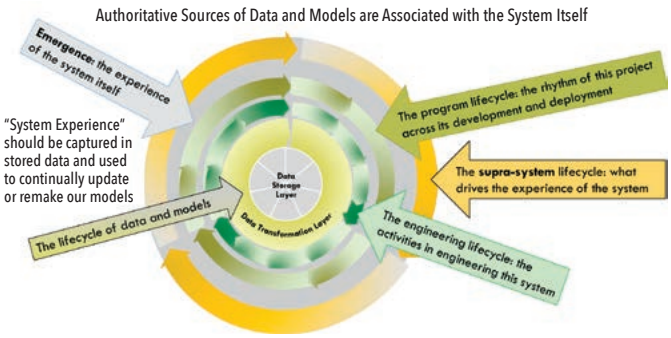


Figure 6. Multiple lifecycles of interest centered on data and models

THE SUPRA-SYSTEM MODEL APPLIED TO DEFENSE ENGINEERING AND ACQUISITION

In the SEMOD project, the team focused on the systems engineering lifecycles and processes defined and used in defense engineering and acquisition practices. To fully reflect the supra-system model of Figure 6, the team developed a new conceptual view of the full defense acquisition lifecycle shown in Figure 7. This is the supra-system model applied to the US Department of Defense (DoD). In this view, we attempted to capture everything associated with DoD engineering and acquisition activities in one mental model. It must be tailored and redrawn based on differing types of development, delivery, and support processes. This view is complex, but it becomes insightful in several ways with study. First, it illustrates systems engineering as a cyclic rather than a linear approach. Although almost all literature attempting to standardize a lifecycle model will say that activities are ongoing and should continue through the lifecycle, the circular illustra-

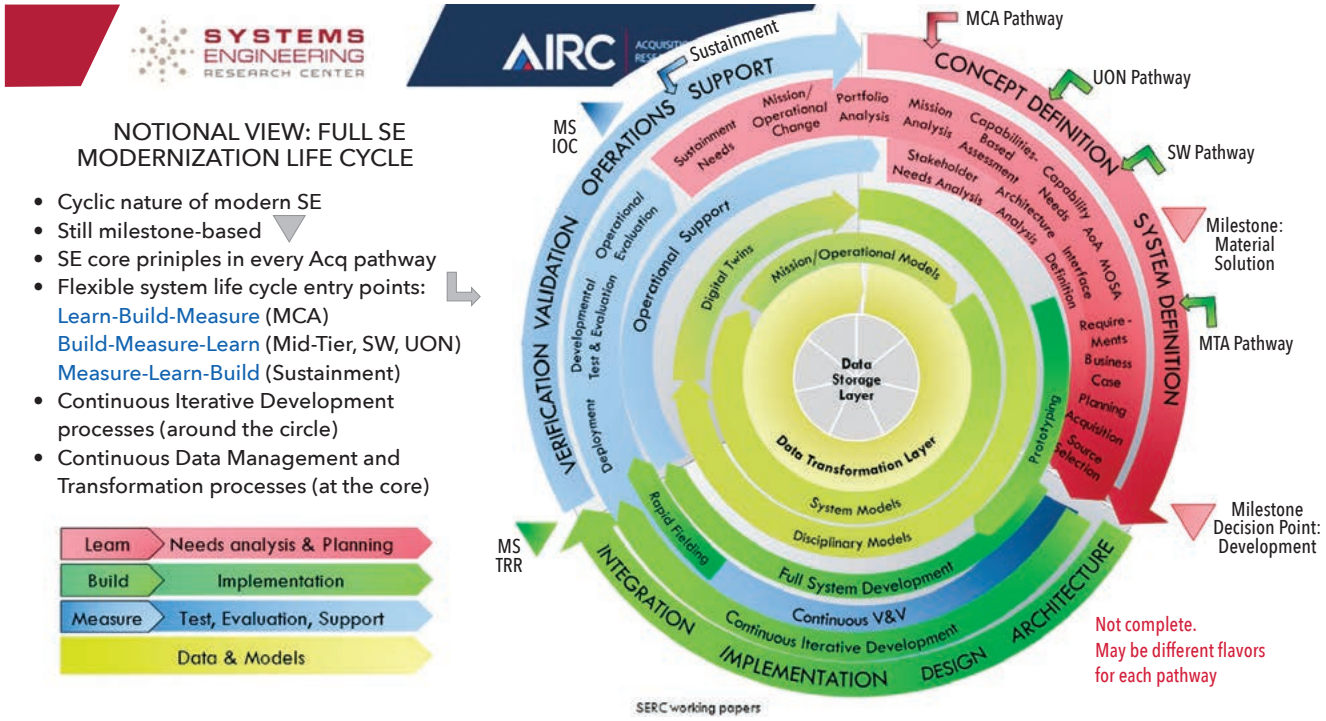


Figure 7. The supra-system model applied to defense acquisition (McDermott and Benjamin 2021)

tion drives this point home more visually and directly. Second, it captures the view that the "experience of the system itself" is a continuous journey that could be affected by multiple external supra-system evolutions, program cycles, and engineering cycles. Third, this view clarifies the digital transformation using a layered model with data storage and transformation at the core, models as the data transformation layer, and systems engineering process areas as the outer layers. Data and models can be associated with any activity in the system lifecycle and must live their lives with the entire experienced life of the system, not just a single program lifecycle. Lastly, it organizes the colors of the outer ring and related technical and management processes in the "build/measure/learn" context, capturing the underlying goal of continuous iterative development.

This view was enlightening to defense engineering and acquisition, particularly defense areas of focus like mission engineering, digital engineering, and agile development. It recognizes that data and models may come from any system experience, including stages that happen before and after what the DoD would define as an acquisition program. In particular, early concept development and later test and evaluation activities in the deployed environment explicitly learn and measure relationships between the system and supra-system and produce data that should be retained to inform other activities across the fully modernized systems engineering lifecycle. Mission engineering and operational test & evaluation are functionally separated from other defense acquisition functions today, and data and models are not rigorously shared.

The lifecycle model depicted here incorporates traditional DoD acquisition milestones (triangles). However, it highlights them in the context of the multi-faceted ongoing work and where they fall within the broader context. It highlights different DoD acquisition pathways, major capability acquisition (MCA), mid-tier acquisition (MTA) prototypes, urgent operational needs (UON), and software acquisition (SW) all have differing entry points and associate systems engineering process instantiations in a system's lifecycle. It highlights that the measurement activities (test and operations) of predecessor or similar systems are a major contributor of data to future systems. Finally, it highlights that data and associated models persist in the lifecycle as in physical system lifecycles.

RELEVANCE OF THE SUPRA-SYSTEM MODEL TO AGILE SYSTEMS ENGINEERING: IT'S ABOUT THE DATA, AND FLOW

Data is the core of the supra-system model. Data has been called the most valued asset in today's business world. Models are the transformation layer. Much of a system's "experience" today emerges from a set of behaviors that are coded in models built on data. The progression of a system "from idea to virtual existence to physical existence" remains true, but creating physical existence should not be the dominant mental model of the modern systems engineer. This does not mean that fundamentally new and valuable hardware solutions will not come into existence, just that we cannot let engineering and management processes that grew from the need to manage and control large complex physical systems drive the life of virtual systems anymore. We can be much more efficient today remaining virtual.

The supra-system model relates data and models to established systems engineering techniques and processes. Shinner's "systems engineering techniques" still apply, holistic problem solving remains the purpose. However, learning and building and measuring systems in their virtual existence is much more efficient than cycling through physical iterations. The power of data and models and associated tools comes from our ability to "shift-left" our learning of the emergent properties of the system into earlier developmental stages. This includes shifting left to the historical experience of previous generations of a system or similar systems if that provides data or models we can reuse. The most relevant measures of this shift are improvements in lead time (of the system), consistency in cycle times (of processes), and the number of anomalies found earlier in the lead time. The other important measure will be from automation: data will improve our ability to

automate the movement of data and models from one discipline to another, from one component to another, and from one phase to another (McDermott and Salado 2021).

The supra-system model envisions the system lifecycle as a continuous flow across internal engineering and program cycles and external supra-system driven change cycles. The core principle behind agile development is also flow: the flow of work should continue consistently across all engineering and program cycles. The cyclic nature of agile methodology encourages thin vertical slices of workflow across multi-disciplinary teams to deliver thin slices of value (capabilities) more rapidly and consistently to the supra-system. This is counter to large horizontal slices of workflow that are subject to planning bottlenecks and constraints. A process workflow that is built from thin-sliced periods that aggregate into larger slices is more responsive than a workflow based on longer slices. Dead time between steps in the flow should be isolated and removed. This requires that we view the whole supra-system, not just parts. Even a lifecycle process as large as the defense acquisition lifecycle can be envisioned as a set of activities in a more agile flow when we view it as the supra-system shown in Figure 7.

There is nothing about agile practice or this supra-system model that will change the fact that a critical safety assurance test of a large physical component may take months while a critical safety assurance test of a software component may take minutes. Engineering lifecycles will always have large temporal variability across disciplinary methods and component types. Program lifecycles should be designed and balanced to match the needs of engineering to learn, build, and measure, and not set program milestones that default to the longest engineering cycles. Likewise different systems will enter into a supra-system that may be or become more volatile and uncertain than others. Program lifecycles should reflect needs for adaptability of the system to supra-system, not as a simple first pass through the system development and transition. Current systems engineering literature does not distinguish between system, program, engineering, and supra-system lifecycle processes and current systems engineering mental models that "force them to remain in lockstep" should be retired. ■

REFERENCES

- Hall, A. D. 1962. *A Methodology for Systems Engineering*, Van Nostrand.
- INCOSE. 2035. *Systems Engineering Vision 2035*, International Council on Systems Engineering, https://www.incose.org/about-systems-engineering/se-vision-2035.
- ISO/IEC/IEEE 15288:2015. Systems and software engineering — System life cycle processes.
- McDermott, T., and A. Salado. 2021. SERC Final Technical Report SERC-2021-TR-024, Application of Digital Engineering Measures, November.
- McDermott, T., and W. Benjamin. 2022. SERC Final Technical Report SERC-2022-TR-009, Program Managers Guide to Digital and Agile Systems Engineering Process Transformation, September.
- McDermott, T., B. Mesmer, and N. Ergin. 2023. SERC Final Technical Report SERC-2023-TR-002, Systems Engineering Modernization Policy, Practice, and Workforce Roadmaps, April.
- Merriam-Webster online dictionary. n.d. https://www.merriam-webster.com/dictionary/life%20cycle.
- Office of the Deputy Assistant Secretary of Defense for Systems Engineering. 2018. Department of Defense Digital Engineering Strategy. June, Washington. US-DC.
- Project Management Institute. n.d. "What is Project Management?" https://www.pmi.org/about/learn-about-pmi/what-is-project-management.
- Shinners, S. 1967. *Techniques of System Engineering*, McGraw-Hill Education.
- Thullier, P., and J. Wippler. 2011. "Finding the Right Problem," in D. Luzeaux and J. Rualt, eds., *Large Scale Complex Systems and Systems of Systems*. ISTE.
- von Bertalanffy, L. 1969. *General System Theory; Foundations, Development, Applications*, New York, US-NY: G. Braziller.

ABOUT THE AUTHORS

Tom McDermott is the chief technology officer of the Systems Engineering Research Center (SERC) and a faculty member in the School of Systems and Enterprises at Stevens Institute of Technology in Hoboken, NJ. With the SERC he develops new research strategies and is leading research on digital transformation, education, security, and artificial intelligence applications. He previously held roles as faculty and director of research at Georgia Tech Research Institute and director and integrated product team manager at Lockheed Martin. Mr. McDermott teaches system architecture, systems and critical thinking, and engineering leadership. He provides executive level consulting as a systems engineering and organizational strategy expert. He is a fellow of the International Council on Systems Engineering (INCOSE) and recently completed 3 years as INCOSE director of strategic integration.

Dr. Kelly Alexander is the chief systems engineer at System Innovation. She has over 35 years of experience in systems engineering and project management. As a US Department of Defense civilian, she served as the director of systems engineering policy and program support, director for common operating environment at HQ US Army, and spearheaded the transition of modular software into the US Army's tactical software. She also led the Army's digital transformation and modular open systems using open standards and open architectures. Dr. Alexander earned a PhD in systems engineering from George Washington University, master's degrees from George Washington University (systems engineering) and National Defense University (national resource strategy), and a bachelor's degree in industrial engineering from North Carolina State University.

Dr. Richard Wallace is a senior research engineer in the Systems Engineering Research Division at the Georgia Tech Research Institute of Technology (GTRI) field office in Dayton, OH. At GTRI, he analyses and models complex systems and processes. He has over 40 years of experience in research, commercial, and DoD engagements for embedded and system-software development as an electrical computer engineer for federated-distributed systems and National SIGINT systems. Recent avionics programs include MQ-9 and RQ-170. In addition, he was an essential participant in the development of ANSI-MIL-STD-1815A (Ada), IEEE-1076 (VHDL), and IEEE-1850 (PSL). Dr. Wallace received his PhD in computer engineering from Complutense University, his MS in computer science from the University of Dayton, and his BS from the University of Idaho.

How Large Scale Agile Can Operate Systems Engineering in the Future

Laurent Alt, alt.laurent@bcg.com; and **Mikaël Le Mouëlli,** lemouellic.mikael@bcg.com

ABSTRACT
The significant shift happening today towards more connected, more automated, and more autonomous systems is bringing software inside all systems, and at the same time agile practices. Our experience of large-scale agile deployments in companies building or operating complex systems in automotive and aerospace shows that, whereas both approaches can easily coexist in isolated teams within the same company, major problems arise when coordinating them at the leadership level, where they are perceived as antagonist, and create misalignments, friction and quality issues. In this article, we propose to describe why it is important to make agile and systems engineering work together, how to do it, and how this impacts how we see value, systems, digital twins, and leadership. The following concepts of the FuSE agile roadmaps are addressed:
- Agility with long lead time components and dependencies
- Agility across organizations boundaries
- Orchestrating agile operations.

PERCEIVED DISCONNECT BETWEEN AGILE AND SYSTEMS ENGINEERING

In many industries, the increasing expectations on time-to-market, complexity, sustainability, regulations, and personalization are stressing the development processes in place, to make them more flexible and more adaptive. In addition, software is taking a large share of the added value of products, and forces organizations to expand their software development capabilities and adopt agile practices.

But there is a perceived opposition between systems engineering practices (which are often perceived as reliable but rigid) and agile (faster but permissive). Each approach has its benefits but also comes with constraints that seem to be at odds with the other.

So this "softwarization" trend raises the question of maintaining the existing processes in place, while putting in place agile ways of working. This also raises the question of which agile practices should be looked at, and if they should be adapted.

In the following, we consider agile in a very broad sense, including the end-to-end DevOps view based on feedback loops from real operations data, not constraining ourselves by any specific framework nor organizational implementation. We will simply consider agile as a collaborative and sustainable way of incrementally delivering value and learning, based on facts and data. This view encompasses both software startups and companies like SpaceX.

THE NEW CHALLENGES AHEAD

The best illustration of the numerous challenges that manufacturing industries are facing now is Tesla. It is true that the focus has mostly been put on the electrification side of the car market, but however significant this shift is for the economy, that technical challenge could be manageable for every original equipment manufacturer (OEM) by wisely using current engineering practices. It simply amounts to replacing one propulsion technology by another one.

But the problem is much bigger than that.

On top of electric propulsion, new important trends are actually impacting all automotive OEMs:
- more and more software in functions: for example, braking systems now heavily rely on sensors to trigger braking, and also make it possible to recover energy to the battery.
- new usage trends like ADAS (advanced driver-assistance systems) use numerous sensors, and many options can be turned on and off as a preference.
- independence of software from hardware, and increased platformization of the technology, in order to optimize reuse across car lines, especially due to the complexity of software.
- new business models based on user stickiness, increased connectivity, and more frequent updates.

- extension of the reach of software to a more global mobility scope, like charging stations.

These trends are not specific to automotive, of course, we can see them emerging in all markets, although not all at the same speed.

As a consequence of this, many organizations are adopting agile practices. But agile is often used in the information technology (IT) department and in teams doing the development of software applications. Besides, systems engineering practices are used for embedded software and hardware organizations. This situation is simply due to these organizations working in silos, each using what they are most familiar with. Both worlds are connected but do not operate consistently, and, consequently, they have a hard time defining shared priorities, speaking the same language, implementing requirements that are fulfilled by a mix of software and hardware, and usually discover quality issues too late.

Therefore, it is important to first explore how these agile and systems engineering teams can work together effectively, keeping the best of both worlds, in particular for products with long development lead times.

HOW AGILE AND SYSTEMS ENGINEERING CAN WORK TOGETHER

Without going against classical stage gate processes, which are designed to progressively derisk the delivery of complex products, here are a few guidelines to make the two systems work together.

- the agile iterations rhythm can be designed to match programs gates. One single agile cadence can be aligned on several programs, in order to deal properly with teams with fixed capacity.
- backlog items, which describe agile teams activities for example *user stories*, can be seen as studies with requirements as an input, and systems design documents as output (*definition of done*). According to the process stage we are in, the expected precision can be more or less precise (*acceptance criteria*). But they can also well be other types of activities, like testing, documenting, and so forth. A more detailed description of this incremental precision approach can be found in Krob (2019), for example.
- their *business value* can be used the usual way (related to the client), or reflect the expected gains from trade-offs (performance vs cost, for example), or even negatively as the risk of delaying them (Reinertsen 2009).

The benefit of this unifying approach is not local, it is global.

Locally, IT teams already working in agile will likely not be fond of formalizing requirements, until they must develop offboard features that are linked to onboard features. *Locally*, hardware teams will likely find little added value in slicing their work in small increments, until they become part of a mechatronic system that is evolving at the speed of software changes. But *globally*, the whole organization can communicate, develop, and integrate consistently.

Having now in mind an agile way of working that can cover all aspects of a complex, long lead time product, let's now look more closely at a few key aspects of product development.

DEFINING VALUE

The concept of "business value" is central to agile, since it governs how priorities are assigned to activities within a fixed capacity and fixed deadline constraint. Yet it is not so easy to define and manipulate, so much so it has sometimes been called "the elephant in the agile room" (Schwartz 2016). The question lies in how to define it, but also how to easily allocate it down to the level of teams and bring them actionable priorities very frequently to enable the so much needed teams empowerment.

Where is the problem? For agile teams in start-ups continuously delivering a service or an app to end users and being able to measure the outcomes of a new version on their business, it is easy to define some business value, and to connect it to everyone's daily work. But for large companies delivering complex products that take months or even years to deliver, it is different. A car or an aircraft are defined by several target attributes such as range, cost, NVH (noise, vibration, harshness), weight, and so forth, that are more or less strictly allocated across all the sub-systems upfront, and further design activities actually produce or refine trade-offs between those attributes. In addition, other properties as impact on manufacturability, sustainability, or delivery timeliness must be optimized too.

So, for complex systems, *value* is a multi-criteria concept that is defined in the context of a global product and organizational setting, and due to impacts, the overall convergence of the system design vs the target attributes can only be evaluated by integrating all the design elements. The more entangled the trade-offs, the more frequent updates we need.

Systems engineering practices make these choices possible. Frequency can be achieved by automation and intensive use of model-based systems engineering (MBSE) and simulation tools. But agile provides the incremental way of working that makes it possible to smoothly make the trade-offs converge as we move from upfront phases to more detailed design phases, and make them flow across the whole organization.

There is yet another aspect of value that must be considered.

Let's go back to the ever-increasing part of software in the design. The most important effect of this, is that that software is massively reused from a continuously evolving platform rather than specified against new requirements each time, increasing value even after launch through over-the-air (OTA) updates, hence creating the emergence of long-lived platforms that support whole product lines. Practically, this means that the concept of value must also include the contribution of an activity to a long-term architecture vision, and across several product lines, which must be balanced with short term priorities of single products.

LOOKING AT SYSTEMS AS PRODUCTS

The speed at which products are being delivered is progressively becoming a problem, especially as new constraints arrive. This is not new, since compliance to regulations have always been an important provider of requirements, and sustainability regulations are simply making the context more complex.

The problem is that, even with good systems engineering practices, many organizations already struggle with how to best balance innovation and carrying over existing design solutions in the context of new requirements. When requirements are cascaded too fast too early, this prevents proper reuse (or adaptation) of existing designs.

This is getting worse when one thinks of reusing across several product lines, where the variety of requirements is multiplied by the specifics of each line.

And even worse if we think that OEMs want to "own" the intelligence of their systems and therefore migrate the algorithmic part of their subsystems up to a vehicle software layer. And worse yet, considering the rapid evolution of sustainability regulations that force OEMs and suppliers to progressively have all their components fulfill requirements under more and more sustainability constraints, implying that components must follow a path towards more sustainability.

In a sustainable organization, this much change can only be managed with an *evolutionary* approach covering all aspects of the product, technology, knowledge, organization, testing facilities, and so forth.

A working mode where rigid expectations overcome reuse, and leave little room for trade-offs and adaptation, is called "project" mode in the agile world, as opposed to the "product" mode, where reuse and evolution are the default. This may sound counterintuitive, since agile is often believed to be very flexible and to encourage massive changes, but most of this agility comes from changing priorities, not from refactoring the technical stack nor disbanding teams and forming new ones. As Martin Fowler (one of the Agile Manifesto authors) puts it, design is not dead with agile, it simply becomes different (Fowler 2004). Enabling agility does not prevent good thinking upfront but is better achieved with the ability to make redesign possible when needed.

Therefore, agility is managed by integrating, at the core of the organization, the idea that each component must be considered as something that will evolve and that provides a service either internally or externally, that improves over time. This does not go against systems engineering, it simply means that organizations would better manage systems as products (meaning, systems associated with a vision, a development roadmap, a value delivered, competition, the means to deliver it, and so forth.

The solution consists in having the technical choices for a system be made in the context of a long-term vision of that system, a set of functions and structure progressively evolving along a roadmap, and assigning a product owner as the person who prioritizes the increments. This does not only hold for the system of interest, but also for sub systems on several levels.

Finally, organizing this way puts more emphasis on the importance of modeling and managing stable interfaces, how functions are fulfilled, how they evolve, in order to provide the necessary autonomy for product owners to manage their own work in an autonomous manner. Therefore, this product approach must reinforce good systems engineering practices.

DEVOPS AND DIGITAL TWINS

Agile is about working incrementally, and this ability to work by increments heavily relies on all stakeholders agreeing on shared facts. In software it is the famous principle of the Agile Manifesto "working software over comprehensive documentation." Everyone in a company doing a software product understands this software and what problems it solves for its users, so it is the best means to communicate and assess the work done. The word "working" has its importance too, since it means that whatever is considered done should have undergone a minimum of testing. This is the *raison d'être* for the DevOps chain.

In hardware, however, due to much longer cycle times, we cannot wait for a product to be delivered, even for a subsystem. So, one can rely on systems diagrams, 3D prints, digital mockups, prototypes, mules, etc.

However, if we look precisely at the verification and validation (V&V) part, and taking into account that the system of interest is a mix of software and hardware, if we wish to have the equivalent of a DevOps chain, the ideal artefact is a digital twin. This is so for two reasons. The first one is that the digital twin is agnostic regarding the kind of technology the system is using (hardware, embedded software, or software). The second one is that it can span the entire lifecycle of the product, even after launch, all the product configurations, and help simulate further software updates over the same hardware product.

An additional difficulty that will become more important is the integration of humans in the definition and operations of systems. Of course, this will definitely bring more complexity and more constraints for development and operations, but we also believe that this additional complexity will accelerate the use of data, the Internet of things (IoT), and artificial intelligence (AI), together with the definition of ethical principles.

Of course, it is a daunting task to model 100% of a system with MBSE. But there are ways to model things incrementally, by considering some of the components as black boxes, simulating their behavior with models, and progressively increasing the coverage of the whole system as needed.

LEADERSHIP AND CULTURE

It is well known that culture and leadership are the main issues in large scale agile transformations. Moving towards agile is probably the most difficult change in organizations since it implies a significant culture shift across all functions.

There are many aspects of this change: teams' empowerment, collaboration across teams and leaders, data driven decisions, value driven priorities, product mindset vs project mindset, being flexible on priorities, caring about organizational learning, focus on quality, etc.

Those can be found in the vast literature about agile leadership, but the most important thing about it is that the higher you go in hierarchy and responsibilities, the more difficult it is to make these changes happen due to the increasing pressure, time scarcity, and the longer history of control those leaders have acquired by reaching their position.

Here, we will simply mention one important point, that we find particularly relevant in the context of complex systems.

Since systems should be considered as products (or platforms) with a roadmap, leadership must have an increased ability to balance value and technical feasibility of these products more widely and more frequently. This does not mean that all leaders should master both skills, but rather that the organization should enable the seamless collaboration of value driven and technically savvy people in order to manage, on several levels of the system, balanced priorities between value delivery and technology feasibility.

In our experience, the pattern of disconnect between business and tech is pervasive. We have seen many situations where programme directives are so compelling from the start that they make reuse very difficult and impede the creation of technology roadmaps. But we have also seen many times systems architects spread across organizations, each having local influence but reporting to non tech-savvy leaders, or, on the contrary, being enough involved upfront to make important choices, but without having enough understanding of the business implications of their choices, due to the lack of ability to communicate across the leadership layers.

Of course, the agile ways of working make this connection more natural, since it is usually embedded in the sprint plannings and quarterly business reviews, or SAFe planning interval (PI) plannings. But when it is about executives, quarterly discussions are not enough to make a significant cultural change happen.

To us, this raises the question of how to structure communication, in a way that naturally connects both worlds. Systems engineering concepts need to be made more accessible, so that as many stakeholders as possible can be involved in decisions. We have successfully used high level "product maps" to align leaders on a shared understanding of a product view of their work, as opposed to simply delivering their work as a program. This has been successful in spreading a shared sense of the product, spot and share high level "invisible" dependencies at leadership level and early in the development plans.

Consequently, another topic that also needs to be addressed is the way leaders organize their agendas, in order to make these connection points as frequent as possible. One cannot simply expect to develop complex products and to change mindsets if the only alignment between stakeholders happens once a quarter. This point is addressed more in length in Alt-Le Mouëllic (2022).

CONCLUSION

With the increasing importance of software in complex products, agile ways of working are also becoming a standard in the development process. However, the perceived opposition between systems engineering and stage gate processes on one hand, and agile on the other hand, often creates dual organizations that struggle to work effectively.

Software organizations can be reluctant to adopt some systems engineering practices, as much as hardware organizations may not find a lot of added value in adopting agile. But the problem is not a local one, it is global. There are tremendous gains in unifying product development methods to encompass hardware and software, and this brings new insights in how agile should be considered at the scale of a company. ■

REFERENCES

- Alt, L., and M. Le Mouëllic. 2022. "How leaders can take ownership of their Agile Transformation." *LinkedIn*. https://www.linkedin.com/pulse/how-leaders-can-take-ownership-agile-transformation-laurent-alt/.
- Beck K. et al. 2001. Manifesto for Agile Software Development. https://agilemanifesto.org/.
- Fowler, M,. 2004. Is Design Dead? https://www.martinfowler.com/articles/designDead.html.
- Krob, D, 2019. "CESAF: an Iterative & Collaborative Approach for Complex Systems Development. Complex Systems Design & Management." Paris, FR. ffhal-02561389. https://hal.science/hal-02561389.
- Reinertsen, D. 2009. "The Principles of Product Development Flow: Second Generation Lean Product Development." Celeritas Publishing.
- Schwartz, M. 2016. *The Art of Business Value*. IT Revolution Press.

ABOUT THE AUTHORS

Laurent Alt is a seasoned leader (CTO, CEO) in technology development and innovation (notably at Dassault Systèmes and Lectra), and is now an expert in enterprise agility and systems engineering at BCG Paris, mainly supporting the move of automotive and aerospace companies towards software and agility, leveraging systems engineering practices.

Mikaël Le Mouëllic is a managing director and partner, at BCG Paris since 10 years, after 6 years in charge of production management at Vallourec. Mikaël is the head of BCG's R&D practice in Europe, helping automotive and aerospace companies in their transformation towards agile.

Model-Based Systems Engineering as an Enabler of Agility

Sophie Plazanet, sophie.plazanet@thalesgroup.com; and **Juan Navas,** juan.navas@thalesgroup.com

ABSTRACT

Model-based systems engineering (MBSE) with agility can help systems engineering programs which deal with both increasing complexity and frequent changes in environment and usages, shorter time-to-market, uncertainty of the needs, and more sophisticated industrial schemes. Agile approaches originated in software engineering can be extended and tailored to a certain extent to complex systems engineering and particularly to MBSE. Main benefits of agility are provision of a minimum viable product as early as possible in the schedule, early capture of changes of needs, enabling to deliver a system answering to the real needs, and securing of the value proposal. It includes also potential reduction in rework of the final system through regular customer feedback throughout development (left shift for the defect correction with early exposure), and efficiency of the use of resources. Concerning MBSE, the use of models as a single source of truth for completeness and consistency is useful to share and secure the design by improving communication within engineering teams and the building and support of the development strategy, and to help to automate some tasks such as model exchange and synchronization. In addition to the benefits of each approach, combining them may help to:

- **Organize and synchronize** the development and validation effort of one or multiple engineering teams.
- **Faster impact analysis** including trade-off studies/options and hence a **faster reaction to evolutions** in expectations and constraints, that is, the agility of systems.
- **Show regularly "end to end" value** to the customer and other stakeholders.

INTRODUCTION

In this article, we illustrate how model-based systems engineering (MBSE) may be an effective enabler of agility in systems engineering, focusing on dynamic learning and evolution (cf concepts of the INCOSE facilitated future of systems engineering (FuSE) roadmap), with a fictive testimony interview of a system engineer based on a fictive example (a drone-based product for inspection of electrical network), used to condensate experiences at Thales. Key concepts are presented, then the process of "warm-up/run/evaluation" is detailed, and we finish with the way to deal with an evolution request in a MBSE and agile context.

1. ***Could you tell more about your product and firm and yourself?***

I am a system engineer in the company Pythagoras. My firm develops and sells lightweight drone-based products for different markets: agriculture, aircraft exterior inspection, and public security enforcement. In addition to the drones themselves, these products embed mission control and data analysis software. These drone-based products feature manual and automated piloting, data acquisition using a wide range of technologies, live data processing, data recording, live, and post mission data analysis. After market analysis of the context, resulting in identification of the need of inspection of electrical networks, my firm has launched the development of a new product providing this service.

2. ***What means agility in systems engineering? And agility with MBSE?***

Agility in systems engineering refers to an engineering effort in which teams can adapt to new circumstances (for example changes in or new stakeholders needs) while meeting the customer expectations in terms of schedule, quality, and cost. Agile systems engineering is a principle-based method for designing, building, sustaining, and evolving systems when knowledge is uncertain and/or environments are dynamic.

Agility with MBSE refers to the use of key concepts favoring agility and co-engineering such as capabilities and functional chains, developed in an iterative and incremental way. And, to the use of system modelling tools to define these concepts, which allows additional engineering rigor and quality.

3. ***What are the key concepts you used for agile with MBSE?***

We focused on a subset of Arcadia concepts that are particularly useful in organizing the engineering effort and carrying value at the solution level: capabilities, functional chains, and scenarios. These engineering artefacts were the references for all engineering teams (systems, software, hardware, IVV, etc.).

We used agile concepts of increments, iteration, increment data packages, EPIC, and user stories.

- **Increment** is a working, tested subset of the system (or of the systems engineering artifacts) delivered regularly to the system stakeholders and built on top of an existing baseline. Value can be knowledge, risk reduction, new features, enhanced performance, etc.

  Each **iteration** is a standard, fixed-length timebox including several successive blocks of fixed duration, where agile teams deliver incremental value in the form of working, tested software and systems. The duration of these blocks may vary according to many factors, both system-related (for example, life-cycle phase, complexity of the capabilities included in the scope) and organization-related (for example, system or software engineering levels, available resources, and human resources policies)

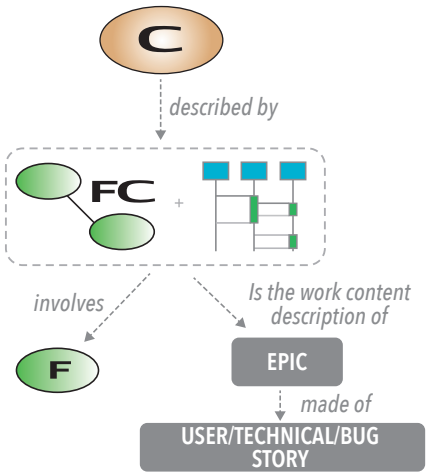


Figure 1. Relations between different concepts

- We refer to **EPIC** as an element of planning, which refers to a functional chain or scenario (or composition or pieces of them) and to other engineering data. It is used to define the expected content of a system increment and thus the value to be delivered to the user. It is defined in **user stories** that are developed in successive blocks. The content of the user stories was defined so that value (working software) was delivered after each software block (Figure 1).
- An **Increment Design Data Package** is a set of engineering data, related to an increment, that will be transferred as a baseline to either lower engineering levels teams (for example, a subsystem) or to other engineering teams (for example, verification and validation), becoming their inputs for subsequent iterations. In a MBSE context, it could be modelling artefacts (along the needs and contexts solution perspectives) described below, associated to textual and model requirements, constraints, justifications, and simulation-based analysis that are associated to these engineering artifacts (Figure 2).

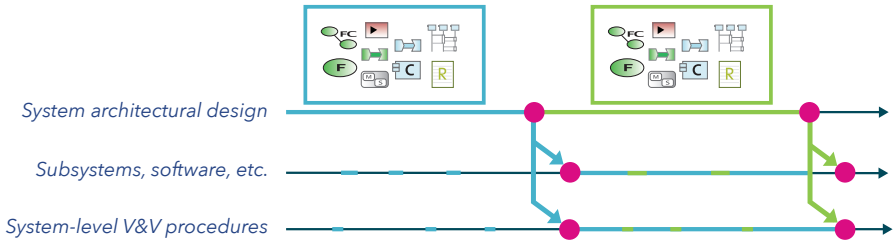


Figure 2. Increment packages dispatched to other agile teams at the end of iterations

4. ***How did you define the engineering workflow?***

To define the engineering workflow between the project's major milestones and associated reviews, we used a sports analogy. You need first to prepare your body (warm-up) before performing a continuous and strong effort (run), and then, if you want to improve, you need to measure and analyze your performance (evaluate).

Warm-up: The "warm-up" activities refer to tasks that will reduce the risk of the engineering efforts that will be done afterwards. It is about capture, selection, and prioritization of the work to be done to meet the objectives of the next milestone, providing the expected value to the stakeholders. It is also about the estimation of efforts required to do it and the definition of the schedule to do it.

Run: For an engineering team, the "run" activity is made of iterations or blocks, aiming at implementing product capabilities. This includes (non-exhaustive list) the detailed definition of product functions and exchanges involved in the capabilities, the development of the system and subsystems' architecture, the development of the software and hardware implementing expected behavior, and the verification and validation of what will be delivered at the end of an increment and to the customer.

Evaluation: The evaluation focuses on ensuring that the whole product increment produced during the iteration can be released; the major part of the integration, verification, and validation effort is performed incrementally during the run iterations, and the evaluation focuses on ensuring that the whole can be released. During early stages, evaluation may include multi-viewpoints analysis (safety, security, performance, reliability, testability, etc.), the preparation of a review of experts or the execution of simulations. Later it may include the approval by the customer or the packaging of software and hardware releases. To evaluate how was produced the engineering effort, the engineering team

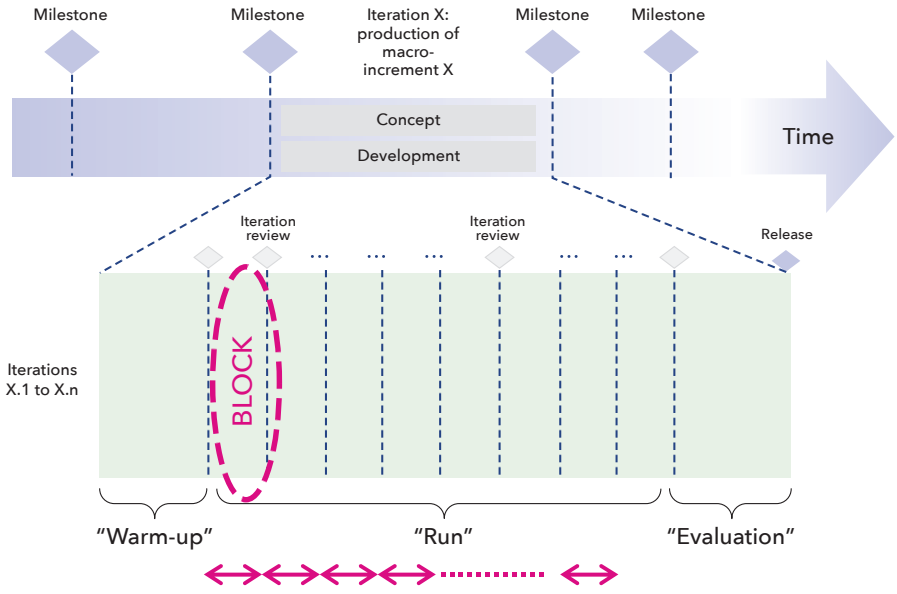


Figure 3. Engineering workflow of warmup, run, and evaluation, composed of several blocks/iterations

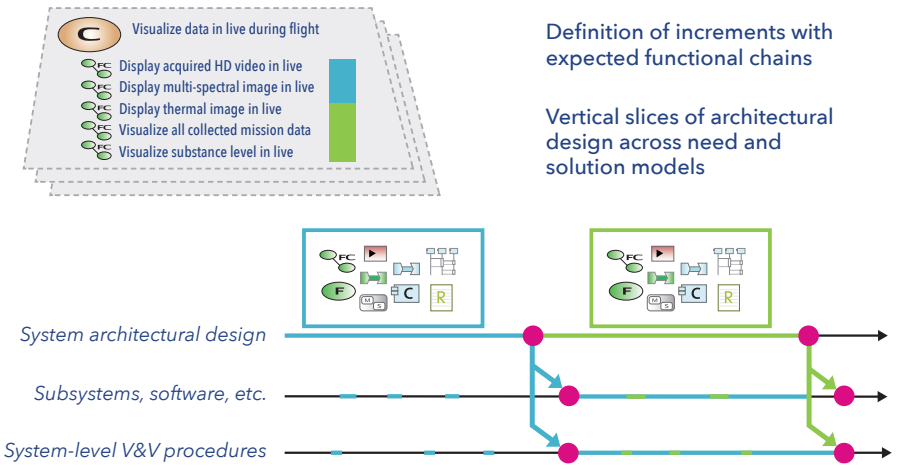


Figure 4. Example of the definition of increments for the capability "visualize data live during flight"

	ITERATION 1
	Acquire data (pictures, videos, scan...
	Acquire multi-spectral image
	Acquire thermal image
	Acquire 3D image
	Manually acquire data
	Manually trigger thermal image acquisition
	Manually trigger multi-spectral image
	Manually trigger 3D image acquisition
	ITERATION 2
	Acquire data (pictures, videos, scan...
	Acquire HD video of moving element
	Acquire HD video
	Acquire HD image
	Automatically follow a flight plan
	Automatically follow a moving target
	Visualize mission progress status

Figure 5. Extract of the repartition of functional chains in different iterations

members review the engineering practices, identify what went well and wrong, elucidate ways to improve the way they perform their engineering effort, including the dependencies with stakeholders, both outside and inside their organization (Figure 3).

Note that:

- The team defines the effort and time length allocated to the warm-up activity between two milestones.
- Warmup, run, and evaluate activities are not necessarily sequential, they can and are often executed in parallel: for example, some key members of the team can "warmup" by defining the scope, while others can "run" and pay technical debt that needs to be done at that moment.
- You can perform these activities several times.

5. ***How did you perform the warmup in the early stages of this development?***

We organize with the help of Pythagoras engineering coaches (agile, MBSE,…) orientation workshops, where all the teams have discussions to define the following points:

- **The articulation between engineering teams:** for example, what is a "contract" between engineering teams made of, what are the outputs from/inputs to each team, what is the development pace (length of iterations, for instance here 12 weeks was collectively decided)
- **The model-based engineering strategy:** what is the purpose of each model view? How will the views be structured? Are there existing building blocks to assemble? We defined a modelling plan.
- **The engineering tools and how they will be configured:** we chose the Team-4Capella tool that integrates natively the MBSE Arcadia method and allows engineers to work concurrently, which was an enabler to co-engineering.
- The identification and selection of the scope of work and its schedule, with a first vision of the product/system to develop: We did this by selecting the capabilities that will be developed, validated, maintained or retired, along with their related functional chains and scenarios (Figure 4).

We defined which functional chains or scenarios (or composition or pieces of them) should be delivered for which iteration, as part of which scope of work of increment (Figure 5).

We defined in a model an intentional architecture of the system, that is, the architectural principles in which further architectural definition work will be based. In further run and evaluation phases, updates or complements of these assets (operational analysis, capabilities, architecture, etc.) were done. Thus, MBSE has accelerated learning by building and revising models of the intentional architecture.

- **Organization and exploitation of models:** Each capability of the system was assigned to a capability leader (cf below), who was accountable for the associated functional chains. The capability leader coordinated the co-engineering with integration, verification, and validation (IVV) and software teams on their capability iterations after iterations (Figure 6).

The results of these workshops as well as the model-based engineering strategy were then formalized in the systems engineering management plan.

Deal with the uncertainty of the needs: We identified while modelling what we don't know and variability points. We

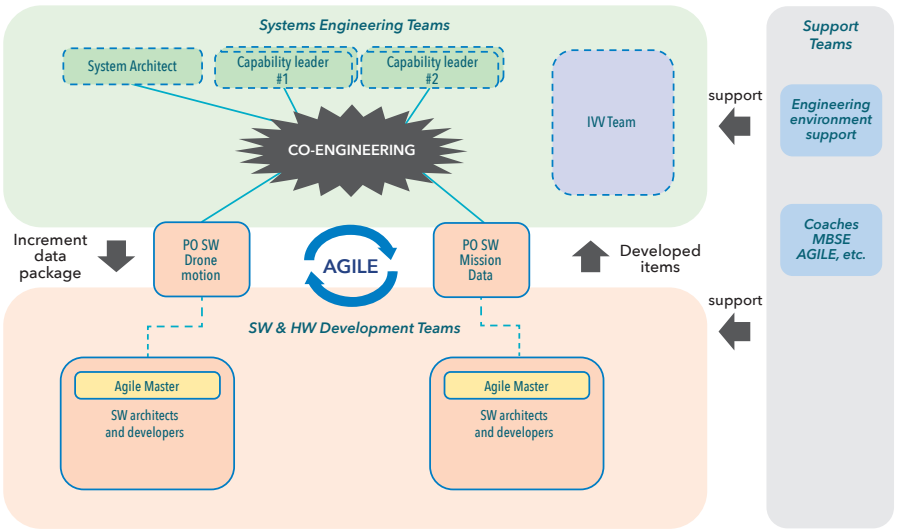


Figure 6. Example of Pythagoras organizational breakdown structure (OBS)

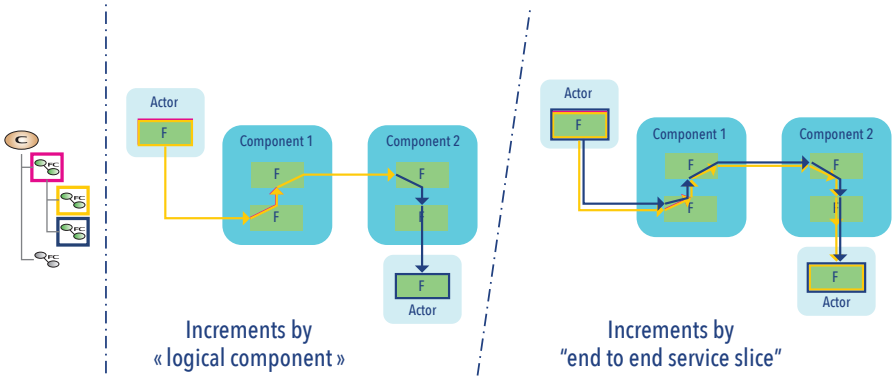


Figure 7. Increments by logical components or by end-to-end service slice

capitalized information in the model that could be for example inputs for decision to eliminate some architecture alternatives in a set-based approach.

Thanks to the testimonies of a previous project, we had learned the lessons that this warmup phase shall be performed with a sufficient scoping and not skipped.

6. ***How did you perform the run, designing, and implementing the increment design data packages?***

First, we defined the content of increments. It could be by for example an "end-to-end service" slice. As an alternative, it could be logical component, with the help of a simulator to simulate inputs/outputs and behaviors. Each slice, once implemented, is fully functional, creates value for the user, and makes user feedbacs possible. Taking care of the compatibility of interfaces between components is key especially for IVV work (with for example the delivery of interface data with identification of their version in increment data packages) (Figure 7).

Then we delivered the architectural design produced by systems teams to software and IVV teams, which was based on the capabilities and associated functional chains or scenarios describing them. In the example below, increment data package relates to the functional chain "manually control the drone motion."

Representative members of the software team participated to regular reviews of the increment data package current build by the systems team. Their role was to anticipate the feasibility and to make sure the system-level vision of the solution was compatible with the current software architecture. Participation of software architects in the agile co-engineering effort at system level was key for the developers to "accept" the models they will receive from the systems engineers. This effort helped to secure the design. In such reviews, there had been the presentation of physical architecture blank displaying the functional chain "manually control the drone motion," with the physical components involved (for example, micro controller, etc.), the expected behavior of these physical components, the operator external entity...

The development team received then for a run iteration n+1 the increment data package related to this functional chain, (cf Figure 8). Having inside the increment functional chain or scenarios (or composition or pieces of them) helped the team to better understand how the implementation of a piece of interface or a small feature fits in the product-wide picture. It helped them also to split in EPIC, applying the agile precepts, to refine the received EPICs in user stories that were developed in successive blocks. The content of the user stories was defined so that value (working software) was delivered after each block. For example, a block was about plugging the actual drone motion to the piloting graphical interface on the tablet. This increment data could contain system mode machine from solution perspective such as below, textual requirements associated to model elements, etc. (Figure 9).

The pace of the IVV team was aligned with the pace of the software development team. The releases were driven by IVV, allowing to test end-to-end services with system integration. The IVV team integrated in the system the components delivered by the software team every third week and run the test procedures written collaboratively in co-engineering during the previous iteration. Each test procedure "tells the story" of its corresponding functional chain, the test steps roughly matching the steps of the functional chain.

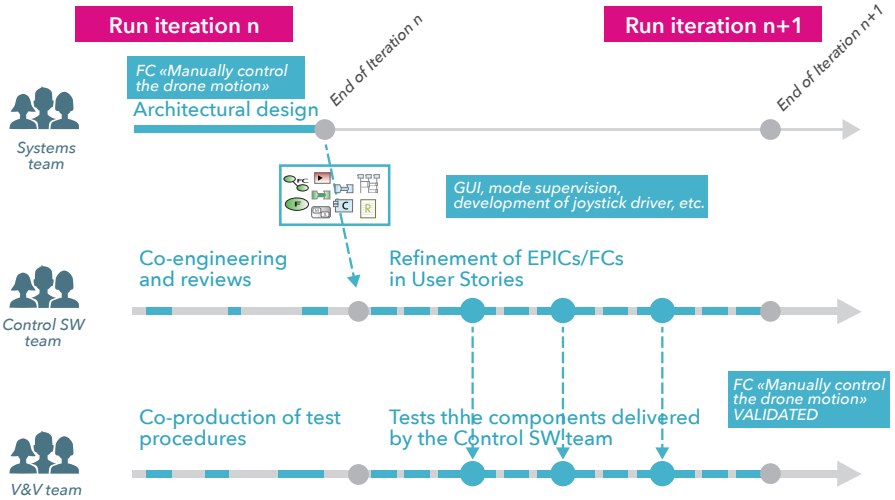


Figure 8. An increment package dispatched to other agile teams at the end of iterations

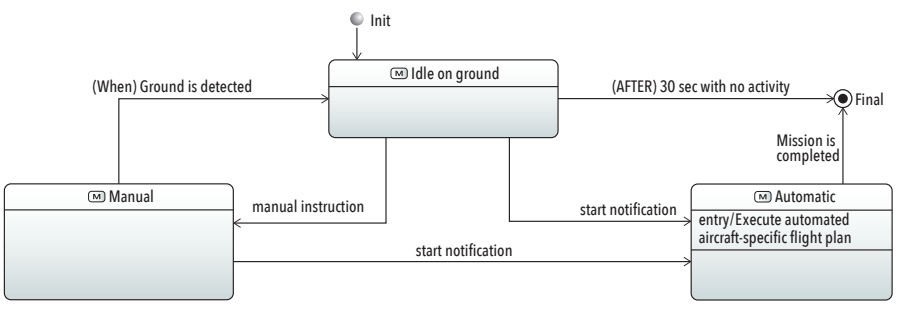


Figure 9. A piece of increment data package for the functional chain "manually control the drone motion"

Capability	% designed	% developed	% validated
Manually pilot the drone	40%	20%	10%
Automatically follow a flight plan	40%	20%	20%
Manually acquire data	30%	10%	10%
Automatically acquire data	60%	40%	40%
Visualize data during mission execution	70%	70%	30%
Visualize data after mission execution	100%	50%	40%
Analyze data during mission execution	0%	0%	0%
Analyze data after mission execution	50%	20%	0%

Figure 10. Progress status of design, development, and test per capability at the end of iteration 3

To obtain this result, IVV practitioners worked closely with the capability leaders in order to translate each need-perspective functional chain in a corresponding system-level test procedure. Models helped to share the design.

When a problem was encountered on a test step, finding the corresponding function or functional exchange in the model was straightforward. Using automated impact analysis, the investigation on the related data was also immediate. It is straightforward to locate the possible cause of a problem. This analysis of the model can have different outputs. If the model (need and corresponding solution) is correct, a defect is created on the faulty component. If the model is actually faulty, then a defect is created on the model itself, and an evolution request is created for the involved component.

7. ***How did you evaluate the design iteratively?***

For example, we organized a review of experts and the simulation about the product increment of the last iteration. We synthetize these results and run performed IVV results in a table such as in Figure 10 to evaluate and monitor the progress status. We also performed a retrospective to take a step back and improve our engineering practices.

8. ***When did the process of warm up/run/evaluation end?***

The process of warm up/run/evaluation ended when all the capabilities in Figure 2 were all released and accepted by the customer. Then the life cycle of the system transitions from development to full operation by the customer. The engineering organization is also an active actor of the evolutionary maintenance of the system: a "warm-up" iteration is currently performed to prepare the engineering teams for this new phase. We reuse both the previous work capitalized in the model and the previous process exposed.

9. ***How did you deal with an evolution request in a MBSE and agile context?***

Agile with MBSE helps to bring the value proposition and short loop for customer feedback. For example, during customer visibility milestones, the functional chains "manual drone control with joystick" and "manual drone control with tablet" were released and validated by the customer. Additional needs (obstacle avoidance and switch between automated piloting and manual piloting) expressed during this milestone were captured in the need-perspective model. We performed impact analysis with the help of queries or any other form of data extraction from the model to precisely compute the consequences of these evolution requests and consequently discussed with the customer to bring value corresponding to its request in further iterations.

CONCLUSION

In this article, we illustrated how MBSE may be an effective enabler of agility in systems engineering, focusing on dynamic learning and evolution (cf concepts of the FuSE roadmap). MBSE accelerates learning by building and revising models since the early stages and helps explore and get agreement on solutions when evolution is requested. Key concepts used in the contact of MBSE with agile were presented, then the process of "warm-up/run/evaluation" was detailed and we finished with the way to deal with an evolution request in a MBSE and agile context. Combining both approaches may help to:

- Organize and synchronize the development and validation effort of one or multiple engineering teams.
- Faster impact analysis including trade-off studies/options and hence a faster reaction to evolutions in expectations and constraints, that is, the agility of systems.
- Show regularly "end-to-end" value to the customer and other stakeholders. ■

ABOUT THE AUTHORS

Sophie Plazanet has been working in system engineering for several years, especially during the past 5 years in Thales. Passionate about MBSE, she joined Thales Corporate Engineering in 2021 where she is a MBSE coach, supporting the Thales engineering teams to adopt MBSE practices. She holds a Master of Engineering and Master of Research in advanced systems and robotics from Arts & Métiers ParisTech Engineering School.

Juan Navas is a systems architect with 15 years' experience on performing systems engineering activities and implementing innovative engineering practices in multiple organizations. He currently leads the modelling and simulation team at Thales Corporate Engineering and dedicates most of his time to expertise and consulting for Thales business units and other organizations worldwide, accompanying managers and architects when implementing MBSE practices. He holds a PhD in embedded software engineering, a MSc degree in control and computer science, and a degree in electronics and electrical engineering.

Agile MBSE: Doing the Same Thing We Have Always Done, but in an Agile Way with Models

Matthew Hause, matthew.hause@hotmail.com

■ **ABSTRACT**
Agile systems engineering is not new. Work has progressed on this for many years to the point that criteria has been established regarding best practice as well as a means of quantifying adherence. The future of systems engineering (FuSE) initiative is reexamining how agile systems engineering fits into the FuSE (Willette et al. 2021). As model-based systems engineering (MBSE) is also a FuSE theme, it is only proper to look at how agile systems engineering and MBSE complement and enable each other. This article examines some of the aspects of MBSE–specifically the Systems Modeling Language® (SysML) – and show at how an agile approach to MBSE can help with the concepts of stakeholder engagement, continual integration, and dynamic learning and evolution. For reasons of space, the article will only provide minimal definitions and explanations of the basics of MBSE, agile, and SysML and as these are well known concepts.

■ **KEYWORDS:** MBSE, SysML, agile, process

STAKEHOLDER ENGAGEMENT

An extract of the definition of systems engineering taken from the INCOSE website states that "systems engineering focuses on establishing, balancing and integrating stakeholders' goals, purpose and success criteria, and defining actual or anticipated customer needs, operational concept and required functionality, starting early in the development cycle…." (INCOSE 2023). A key aspect of translating stakeholder goals and customer needs to requirements and eventually systems is to effectively interact and communicate with stakeholders and customers. The definition further states that systems engineers "baseline and model requirements and selected solution architectures for each phase of the endeavor," hence models are a key aspect for all aspects of systems engineering, especially for stakeholder engagement. If the models can assist, inform, and at times provide the bulk of this communication, then the model will serve a double purpose for both the system engineers and the stakeholders. Additionally, the *Systems Engineering Handbook* version 4 in section 2.10 Systems Engineering Leadership describes "working with the stakeholders (including customers), representing their points of view to the team and the team's point of view to them." The team will already be familiar with models, so this further enhances the benefits of the model as well as the communications.

USE CASES AND CONTEXT

There are a variety of ways of collecting stakeholder needs including operational concept (OpsCon) definition, concepts of operations (ConOps), business and mission needs analysis. These can and often are expressed in model form. The Systems Modeling Language® (SysML) can be used to capture these stakeholder and customer needs with the use case diagram. A use case is a methodology used to identify, clarify, and organize system requirements. The use case defines possible sequences of interactions between systems and users in a particular environment and related to a particular goal. The use case description describes all the steps taken by a user to achieve a goal. There are three main elements to use cases:

- Actors: The stakeholder, customer, system user, cooperating system, etc. These can be abstract as well as concrete people, organizations, external systems, and abstract actors such as time.
- The use case: The use case itself represents the goal or final successful outcome that the stakeholder wishes to achieve. The use case text description outlines the process and steps taken to reach the end goal, including the necessary functional requirements and their anticipated behaviors.
- The system context: the circumstances in which the goal is meant to be achieved including constraints, environmental factors, interface, and timing limitations, etc.

USE CASE MECHANICS

The use case descriptions are written using stakeholder terminology and language to correctly capture the intended purpose and flow and to ensure that they accurately capture customer needs. This clearly defines functional requirements for the development team. Logical sequences and constructs can be difficult to follow in textual language, so activity diagrams can also be used to capture required behavior and then translated to text. These activity diagrams can also follow on from textual use case descriptions to further elaborate the system behavior. The defined activities can be allocated to structural elements as well as traced back to the original stakeholder needs and use cases.

An important aspect to capture in parallel is the system scope or context to determine those elements included in and excluded from the project. The system of interest (SoI) is shown in the center of the diagram with connections to the various stakeholders and other interfacing systems. This initial logical diagram can be translated to a more physical version defining specific interface types, data and other exchanges, timing aspects, capacities, etc. There may also be multiple contexts for the SoI depending on mobility, timing, and lifecycle as well as stakeholder variations. Having established this stakeholder baseline, it is validated with the customer, iterating as necessary. The model should include stakeholders throughout the project lifecycle not just the functioning system reflecting all stages of the 15288 standard (ISO 2015).

Use cases can also encapsulate a set of functional requirements to form a cohesive causal sequence providing customer value, features, and goals. The use case refines the functional requirements in that it expresses them in model format. Functional requirements can also be traced to use cases defined with the customer, so can be derived from requirements, or precede them.

As the system is developed, the implementation of the use case forms a useful increment in the system development: build the functionality and structure necessary to implement the use case. When appropriate, implemented use cases can be demonstrated to the customer to ensure that the goal is being met in an acceptable way. Although use cases are commonly used in software development, they also represent system functionality including physical user interface (UI), human factors interactions, physical system ergonomics, etc. Enabling this stakeholder involvement early and often ensures that the users are engaged and that their goals are being met. I have used these incremental delivery techniques throughout my career as a means of demonstrating "visible signs of life" in project development and to increase customer confidence. It has the added benefit of providing a platform for continuous integration and delivery as implementation often requires multiple cooperating system elements for successful implementation.

CONTINUAL INTEGRATION, AND CONTINUOUS DELIVERY

In the age of MBSE, system deliveries need to evolve past delivery of documents and physical systems. Models are being used throughout the development lifecycle starting with model-based acquisition and through to development and onto product line management (PLM), computer-aided design (CAD), and into manufacturing. Models are developed, delivered, and evaluated throughout the product lifecycle. In the same way that systems are developed incrementally with increments of components, subsystems, systems, etc., models and model elements can also be delivered incrementally. With agile software development, the product owner discusses the objective that the sprint should achieve and the items that, upon completion, would achieve the sprint goal. The team then creates a plan for how they will get them "done" before the end of the sprint (Schwaber 2021). In this same way, models, model architectures, model elements, and diagrams need to start with a clear set of goals, intended audience, and questions to be answered/addressed. These criteria guide what needs to be developed as well as guidance on how much and how little needs to be done. Without this clear guidance, the developers can start down an endless path of modeling without reaching a conclusion or fail to create a useful deliverable. In addition to the goals, the intended audience needs to be clearly determined. High level decision makers and stakeholders should not be given detailed technical drawings, but instead should be provided with consumable reports enabling them to make their decisions. In other words, appropriate deliverables need to be given to the appropriate people.

As described above in the previous section, use cases can provide a useful deliverable increment. As they often cut across systems and interfaces, these require integration of these different model and system elements including blocks, behavior, etc. Prior to starting this, the model structure or architecture needs to be defined. These include libraries of types, interfaces, components, package structure, and dependencies, etc. Major system elements and their interfaces should be defined early in the project and provide reusable components along the lines of modular open systems architectures (MOSA) (DoD 2018). These elements should also be deliverables and care should be taken to ensure that they are available and accessible to the entire team. This ensures interoperability and compatibility. The right architectures enable change rather than restrict change. Correct and fixed major interfaces ensure parallel development enabling integrated deliverables. Agreement and delivery on these model development milestones are also part of the continuous integration and delivery lifecycle.

AGILE AND THE DIGITAL THREAD

As well as project deliverables, attention also needs to be given to the integration and test of the tools comprising the digital thread. (This topic was covered extensively by Papke et al. (2023), so I will simply add a few comments.) Tools almost never integrate as advertised even if they come from the same provider. In addition, different versions from different providers may be incompatible or not provide the same levels of integration. This integration needs to be tested and prototyped using the production environment to ensure that deliverables can be generated at the right time and in the right format. Otherwise, projects may have a significant delay while reports, traceability, and procedures are generated or worse generated by hand. Impact analysis and traceability mechanisms and procedures need to be tested to ensure that the incremental changes can be integrated, analyzed, and tested. Rollback procedures should also be prototyped if the project has taken a wrong turn. Sadly, these issues are often only considered when things have gone badly wrong. The time to test your sprinkler systems is not when the factory is on fire. Finally, the digital thread forms an authoritative source of truth (ASOT), (DoD 2018) where data is entered once and referenced where, when, and how it is necessary. Duplication of information means that it can become untrustworthy or inaccurate.

DYNAMIC LEARNING AND EVOLUTION

To add to Benjamin Franklin's quote, the only things certain in life are death, taxes, and change. No project ever runs without errors, and all projects must have the capacity to adapt to the inevitable change. I often say that systems engineering is the process of correcting your mistakes, misunderstandings, miscomprehensions, omissions, miscommunications, assumptions, etc. Projects never have all the information they need at the beginning and part of modeling is to highlight the knowns and unknowns. The purpose of models is to document what you know when you know it to figure

out what you don't know. This often does not correspond to the traditional MBSE process since high and low-level requirements, needs, implementation constraints, assumptions, etc., can become known at various stages in the project. So, the order in which these are documented can be immaterial if these are documented and in the correct model location. The model can form the basis of a dynamic learning and evolution environment if the ASOT principles defined above are adhered to.

DO THE OODA

The OODA loop of observe, orient, decide, and act is a four-step approach to decision-making that focuses on filtering available information, putting it in context and quickly making the most appropriate decision while also understanding that changes can be made as more data becomes available. It was developed by military strategist and United States Air Force Colonel John Boyd (Boyd 1976). Boyd applied the concept to the combat operations process, often at the operational level during military campaigns. It is now also often applied to understand commercial operations and learning processes. The approach explains how agility can overcome raw power in dealing with human opponents. It is especially applicable to cyber security and cyberwarfare. Projects can also suffer from "analysis paralysis" or hit roadblocks. When a roadblock is met, flag it and move on. SysML has a problem concept that can be used to document the roadblock in whatever form it takes. Revisit the roadblock prior to the end of the scrum period to see if additional modeling and research has now revealed an answer.

SYSTEMS 1, 2, AND 3

Schindel and Dove (2016) outlined the agile systems engineering life cycle MBSE (ASELCM) pattern where:

- **System 1** is the target system under development.
- **System 2** is the system that produces, supports, and learns about the target system.
- **System 3** is the process improvement system, called the system of innovation that learns, configures, and matures system 2. System 3 is responsible for situational awareness, evolution, and knowledge management – the provider of operational agility.

For project success, there must be a clear delineation and understanding of systems 1, 2, and 3 in the process and during modeling. Responsible teams should be assigned for each to ensure continuous improvement. Post project, mandatory lessons learned sessions must be built in, scheduled, and budgeted for all projects. Without these sessions, lessons will not be learned, issues documented, and improvements implemented.

SUMMARY AND CONCLUSION

For many years, the thoughtful and laborious process of systems engineering looked at agile systems engineering as an oxymoron. Over the past two decades, agile has become a key enabler of system development and systems engineering, shortening timescales, and improving results. MBSE is also emerging as the way the systems engineering is done to the point that the "MB" in MBSE is becoming redundant. For FuSE concepts to advance, the various principals need to work in harmony with one another to enable the *Systems Engineering Vision 2035* published by INCOSE. This article looks at some of the ways in which agile and MBSE can work together to advance these concepts. I have used the principles of both agile and MBSE throughout my career of over 45 years as they just seemed the most obvious and natural ways to do things. My hope is that these become ubiquitous and "the way things are done" rather than new concepts. ■

REFERENCES

- Boyd, John R. 1976. Destruction and Creation. U.S. Army Command and General Staff College. (3 September). Available online at https://upload.wikimedia.org/wikipedia/commons/a/a6/Destruction_%26_Creation.pdf.
- US Department of Defense. 2018. "Department of Defense Digital Engineering Strategy", Office of the Deputy Assistant Secretary of Defense for Systems Engineering. 9 May. www.acq.osd.mil/se
- Willett, K, R. Dove, A. Chudnow, R. Eckman, L. Rosser, J. Stevens, R. Yeman, and M. Yokell. 2021. "Agility in the Future of Systems Engineering (FuSE) – A Roadmap of Foundational Concepts." INCOSE International Symposium, 31: 158-174.
- INCOSE. 2023. About Systems Engineering, Available online at https://www.incose.org/about-systems-engineering/system-and-se-definition/systems-engineering-definition, Accessed March, 2023.
- ISO/IEC/IEEE 15288:2015 Systems and software engineering — System life cycle processes, Available Online at https://www.iso.org/standard/63711.html, Accessed March 2023.
- Papke, B, M. Hause, D. Hetherington, and S. McGervey. 2022. "MBSE Model Management Pain Points – Wait, this looks familiar!" NDIA Systems and Mission Engineering Conference. November, NDIA Proceedings (dtic.mil).
- Schwaber, K., and J. Sutherland. 2013. "The Scrum Guide", Scrum.Org, https://www.scrum.org/resources/scrum-guide, 13 March.
- Schindel, W., and R. Dove . 2016. "Introduction to the Agile Systems Engineering Life Cycle MBSE Pattern", Proceedings International Symposium. International Council on Systems Engineering. Edinburgh, Scotland, July 18-21.

ABOUT THE AUTHOR

Matthew Hause is an SSI principal and MBSE technical specialist, a former PTC fellow, a co-chair of the UAF group and a member of the OMG SysML specification team. He has been developing multi-national complex systems for over 45 years as a systems and software engineer. He started out working in the power systems industry and has been involved in military command and control systems, process control, manufacturing, factory automation, communications, supervisory control and data acquisition (SCADA), distributed control, office automation, and many other areas of technical and real-time systems. His roles have varied from project manager to developer. His role at SSI includes mentoring, sales presentations, standards development, presentations at conferences, specification of the unified architecture framework (UAF®) profile, and developing and presenting training courses. He has written over 100 technical papers on architectural modeling, project management, systems engineering, model-based engineering, and many other subjects. He is a proud recipient of the INCOSE MBSE propellor hat award.

FuSE Agility as a Foundation for Sound MBSE Lifecycle Management

Barry Papke, barry.papke@3ds.com; **Matthew Hause,** matthew.hause@hotmail.com; and **David Hetherington**; david_hetherington@ieee.org

ABSTRACT
Over the past several years, numerous industries have increased their adoption of the systems modeling language (SysML®) and model-based systems engineering (MBSE) as a core practice within their engineering lifecycles. However, the introduction of SysML and MBSE methodologies has not yet yielded many of the originally envisioned benefits. System models are becoming larger and more complex and many large MBSE projects continue to experience problems with model integration, repository performance, and model lifecycle management. The root cause is the failure to recognize the MBSE digital environment as a complex engineering information processing system that requires the same rigor and development processes as the system-of-interest (SoI) it is designing. This article describes how three future of systems engineering (FuSE) agility foundation concepts (system of innovation, effective stakeholder engagement, and continuous integration) directly address some of the problems seen in adoption, deployment, and sustainment of the MBSE digital environment as an SoI.

MBSE ADOPTION, DEPLOYMENT, AND SUSTAINMENT IS NOT WITHOUT IT'S CHALLENGES

FuSE agility is one of the initiatives intended to shape the future of systems engineering and contribute toward the practical realization of the *Systems Engineering Vision 2035* published by INCOSE. FuSE agility includes nine foundational concepts to advance thinking and practice in "agile-systems engineering" (development of agile systems) and "agile systems-engineering" (an agile systems engineering process). Creating an agile system requires an agile organization using an agile process.

While the challenges of realizing the *Systems Engineering Vision 2035* are before us, there are a number of urgent challenges biting at our heels, specifically in the area of MBSE deployment and MBSE lifecycle management (Noguchi 2022 and Papke 2022). Many large MBSE projects have not fully realized the envisioned benefits. Common pain points include:

- Model integration issues between government-reference models, prime contractor models, and supplier models and across disconnected networks.
- Integration issues between descriptive models and computational/analytical models.
- Insufficient representation of their systems of interest, leading to stakeholder confusion, increased backlog of rework, and system architecture and design flaws
- Dramatic slowdown of modeling tool performance as the number of models users grew, model size grew, and during model delivery activities for major reviews.
- Insufficient verification and validation of model simulations and outputs.
- Lack of configuration control and lack of quality control in model content.

Industry is beginning to realize that the digital engineering environment is its own system of interest with requirements for what it must provide, functions it must perform, interfaces and data exchanges it must support, and performance parameters it must achieve. As a system it must also be based on a sound architecture (both the tool environment and the models it contains) that provides key architectural properties such as loose coupling, proper cohesion, modularity, maintainability, extensibility, etc.

This article focuses primarily on "agile systems-engineering" (an agile systems engineering process) and three specific FuSE agile concepts: system of innovation, effective stakeholder engagement, and continuous integration (Willette, et al. 2021) to provide actionable guidance in the context definition, deployment, operation, and sustainment of the MBSE digital engineering environment.

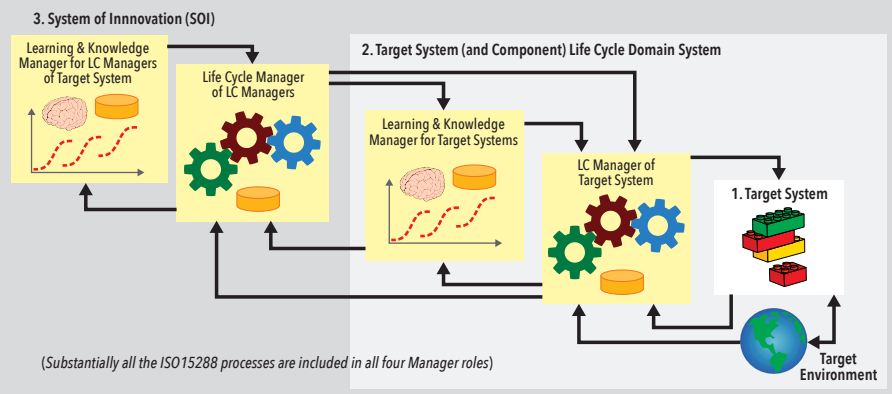


Figure 1. Agile systems engineering life cycle model pattern (Schindel and Dove 2016)

SYSTEM OF INNOVATION

The system of innovation was described in the paper "Introduction to the Agile Systems Engineering Life Cycle MBSE Pattern" (Schindel and Dove 2016). It consists of three (3) systems:

- **System 1:** the target system under development.
- **System 2:** the system that produces, supports, and learns about the target system. This is the logical system within which the target system will exist during its lifecycle.
- **System 3:** the process improvement system, called the system of innovation that learns, configures, and matures system 2. System 3 is responsible for situational awareness, evolution, and knowledge management – the provider of operational agility.

In the context of the MBSE digital environment, the target system (System 1) is the actual physical system. One of the key features of the agile systems engineering life cycle model (ASELCM) is that the target system provides feedback through the physical environment to the lifecycle manager. This feedback could be sensor data, user complaints, feature requests, or other feedback based on the deployed operation of the physical system. The lifecycle domain system (System 2) is the digital engineering environment that consists of tools, processes, and people that develop and maintain System 1. System 3 is the functional engineering and project management organizations that plan the definition, deployment, and sustainment of System 2 and also control its funding and resources.

In terms of the system that learns, configures, and matures System 2, System 3 has been in hibernation for decades. The people, processes, and tools for execution of the systems engineering lifecycle have been codified in standards, documented in engineering procedures, and specified in contract statements of work. There has been little innovation to the document-based systems engineering process since its first inception. Project planning is based on past metrics and standard processes.

Many of the project challenges and execution problems seen in large MBSE projects can be traced directly to the inability of System 3 to recognize and adapt to the impact of MBSE. The largely manual, document-based systems engineering process has been replaced by a highly complex, engineering information processing system that has interfaces and information exchanges with other systems in the digital engineering system-of-systems (Maier 1996).

Unlike documents, models are living entities that are also systems with an architecture, functions, interfaces, and performance requirements. While projects are still adapting to perform design reviews and associated quality assurance processes which focus on the verification and validation (V&V) of the system design, the need for such activity is clearly understood, in the context of System 1 (the system under development), V&V asks:

Verification – does the system satisfy the specified requirement within cost, schedule, and acceptable technical risk (did we build the thing right.)

Validation – does the system achieve its mission and business objectives and intended use in the operational environment (did we build the right thing.)

In MBSE, there is another system of interest (SoI) that must be considered, System 2 which is the model itself as an engineering artifact in the engineering lifecycle:

Verification – is the system model syntactically and semantically correct and does it contain all required engineering content (is it a good model-based representation of the design.)

Validation – does the model fulfill its objectives as a digital engineering artifact in the digital engineering environment (is the model useful to all applicable stakeholders throughout the engineering lifecycle.)

In the document-based world for Department of Defense projects, there was little, or no thought given to the document format and content specified as the "contract deliverable." Document based deliverables were defined by a system of data items descriptions (DIDs) and DD Form 1423 "Contract Data Requirements List." In the commercial domain, most organizations had similar pre-defined templates and formats.

SYSTEM OF INNOVATION ENABLER FOR SUCCESSFUL MBSE ADOPTION

Most large MBSE projects today are still "first attempts" to deliver systems using this methodology. Projects are attempting to do many more things with models as the "authoritative source of truth" such as implementation of the "digital thread" than were attempted with documents (DoD 2018)]. The growth in size and complexity of models during the engineering lifecycle continues to present new problems and challenges to model lifecycle management, model integration, and digital infrastructure performance. Projects can no longer depend on the predictability of their legacy document-based processes. They require an agile approach to address these challenges.

The system of innovation provides this agility through three basic principles: sensing, responding, and evolving.

- Sensing – MBSE projects must monitor and measure key indicators of both project progress (a measure of System 2 performance) as well as data from System 1. This requires more than assigning budget, scope, and schedule to the MBSE deployment activity. It means recognizing that many assumptions made about the MBSE workflows, MBSE model architecture, and network architecture may be incorrect and are yet to reveal themselves. As we used to say — Inside every small problem is a large problem struggling to get out!
- Respond – MBSE projects must make decisions about what they see and be prepared to react to address the actual performance of their MBSE deployment. Detailed planning and compliance are not sufficient. Initial plans will need to evolve as new information comes to light.
- Evolve – MBSE projects must embrace the fact that their process will evolve, and this evolution must be supported

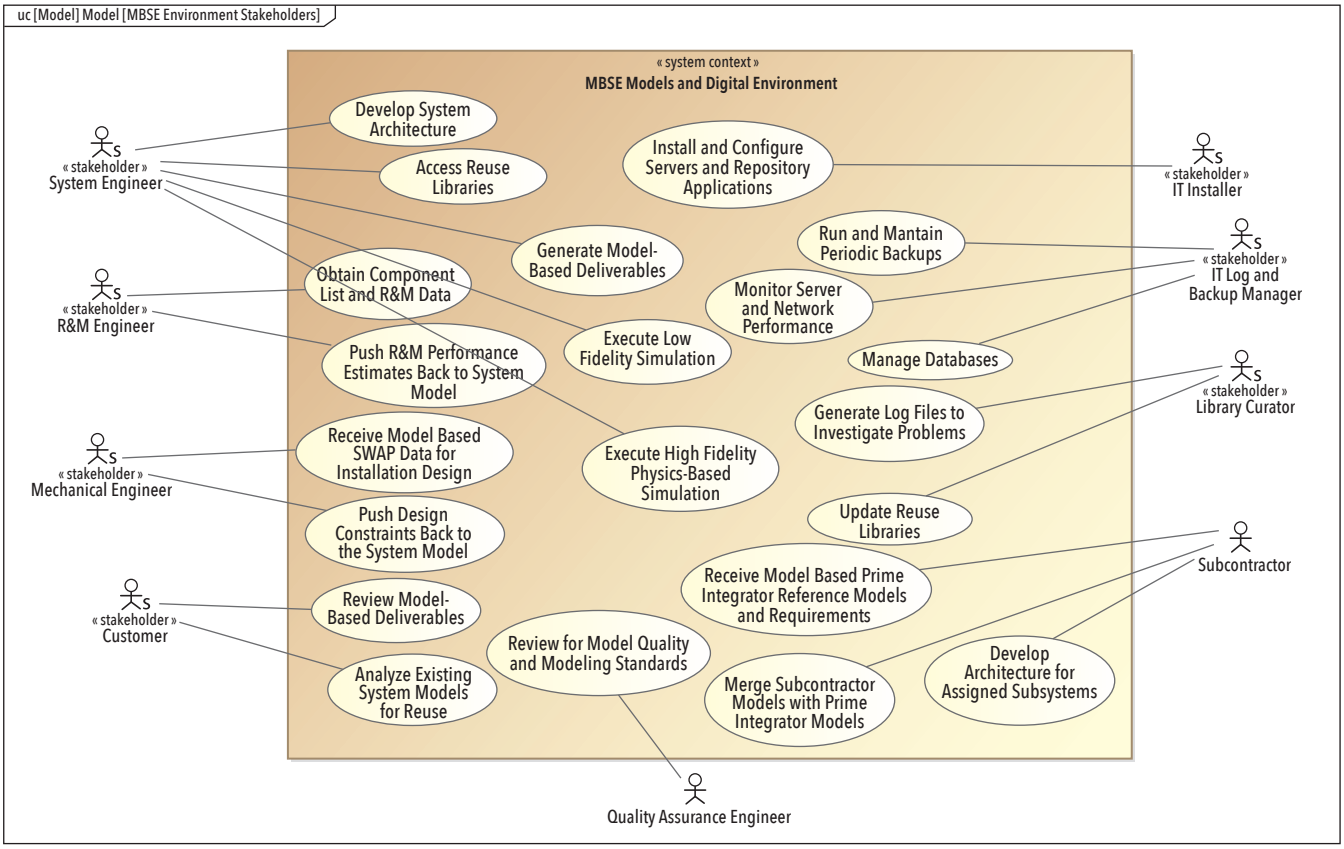


Figure 2. MBSE model and digital environment stakeholders have diverse needs and concerns

with a culture of experimentation, re-evaluation, and new institutional memories.

EFFECTIVE STAKEHOLDER ENGAGEMENT

Effective stakeholder engagement is another key FuSE agility concept that is well understood and generally applied correctly with respect to the SoI being developed, but is neglected in the context of deployment, operation, and sustainment of the MBSE digital engineering environment. As with the system of innovation, stakeholder engagement within document-based systems engineering processes is largely dormant. Stakeholder concerns are codified in the standards, DIDs, and statements of work. In MBSE projects, there are many new and significant activities and processes, that each have new stakeholders and new concerns. Those new concerns must be identified and addressed as part of the MBSE definition, deployment, and execution activities.

For example, in document-based projects, deliverables are managed through defined data management processes and organizations. The engineering information is controlled at the document level. The tools to create documents are "commodity office applications" that require minimum information technology (IT) resources and little "backend" management or support. Little has changed in document and data management technology.

In MBSE projects, engineering information is managed at the model element/property level. The model repositories are complex server applications that require significant back-end IT support. Models are living entities that access data and integrate with other models. Model re-use is a major benefit of MBSE and is typically implemented using reuse libraries, which require a library manager or curator.

The backend repository IT support and library curators are new stakeholders. Legacy stakeholders including customers, subcontractors, and other engineering disciplines have new and different stakeholder needs with respect to the MBSE models and digital environment.

EFFECTIVE STAKEHOLDER ENGAGEMENT AS AN ENABLER FOR SUCCESSFUL MBSE ADOPTION

As a symptom of new MBSE projects failure to apply the system of innovation concept and to perform their three key functions (sense, respond, and evolve), many MBSE projects fail to recognize how the technology, tools, and processes of MBSE have created the need to perform new stakeholder engagement analysis for definition, deployment, and operation of the MBSE modeling environment. The stakeholder analysis for the MBSE models and digital environment (System 2) is just as important as that for the SoI (System 1) itself. This is at the core of the second dimension of V&V as described earlier. If we don't understand stakeholder needs correctly, how can we know whether the models and their digital environment is producing models that fulfill their objectives as a digital engineering artifacts (validation)? Projects must understand the needs and concerns of both new and legacy stakeholders with respect to the models, tools, and processes of MBSE (Flyvbjerg 2023).

CONTINUAL INTEGRATION AS AN ENABLER FOR SUCCESSFUL MBSE ADOPTION

Continuous integration is yet another key FuSE agility concept that is well understood and generally applied correctly with respect to the SoI being developed (particularly in software intensive projects), but is neglected in the context of deployment, operation, and sustainment of the MBSE digital engineering environment. The concept of continuous integration (that is, continuous delivery) is at the core of agile. The key principle of agile development is to deliver products iteratively and incrementally, maximizing

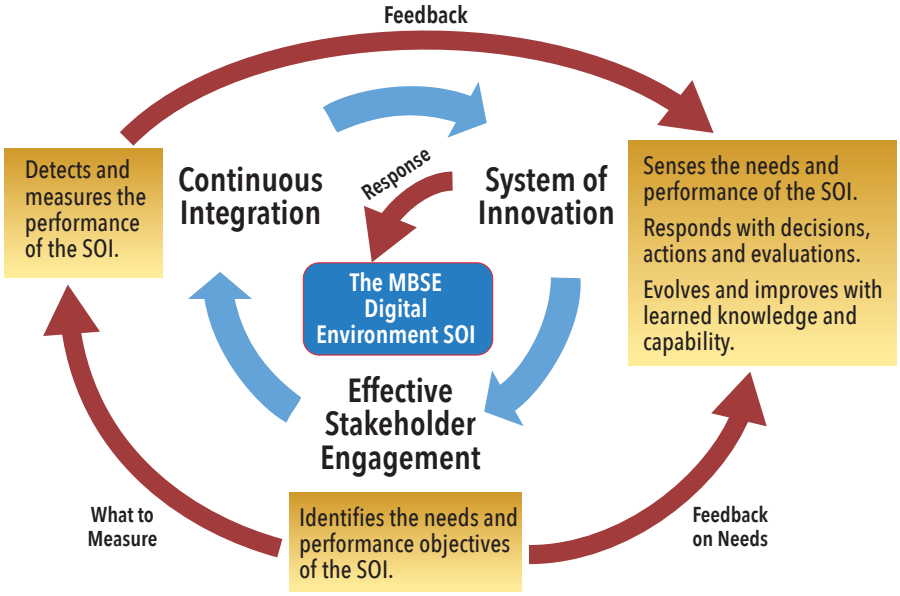


Figure 3. FuSE agility foundation concepts depend on each other

opportunities for feedback. Incremental deliveries of 'done' products to ensure a potentially useful version of working product is always available" (Schwaber and Sutherland 2013). This concept is about enabling integration as early and as often as possible revealing issues caused by incompatible or insufficient interfaces and information exchange specifications (Willette et al. 2021).

As mentioned previously, the MBSE digital engineering environment is a complex engineering information management system within a larger system-of-systems digital engineering environment. Within a given systems engineering department, the interfaces and engineering workflows between networks and tools may be well defined and mature. However, most of these interfaces and workflows are specific to the project and may be new or poorly defined and largely or entirely untested. In addition, as part of a system-of-systems, the tools and networks in customer and supplier environments and in other engineering disciplines evolve independently which may affect these interfaces. Continual integration provides the opportunity to begin exercising the interfaces and workflows between engineering departments, customers, and suppliers early, when the model content is relatively small and to establish performance baselines for critical MBSE processes such as model download and commit time, simulation execution, and other server/network resource demands. This does not mean propagating model changes instantly across the project. It refers to planned, incremental model integrations that coincide with incremental delivery of the System 1 design, as part of an agile project workflow.

In their application of agile systems engineering, projects must ensure that the output of each "sprint" is both a useful version of the design description, but also a useful version of the system-of-models that fulfill their intended role in the engineering lifecycle. The sprint objectives should also include model data exchanges between contractor/supplier/customer environment, branch merges and model integration and any end-to-end simulations, queries or other "digital thread" related workflows.

SUMMARY AND CONCLUSION

Once projects commit to adoption of MBSE in their engineering projects, they take on new challenges related to the deployment, operation, and sustainment of the complex digital engineering environment consisting of the models, tools, network, processes, and people involved. Because projects have operated so long with the mature and much simpler document-based systems engineering approach, they failed to realize the impact of such a transformational change in their engineering lifecycle. This article highlighted three of the FuSE agility foundation concepts (system of innovation, effective stakeholder engagement, and continuous integration) and described how each concept can help address the challenges that large projects are experience in their initial MBSE journey. Several other concepts are also key enablers such as technical oversight for agile projects, agility across the value stream, and orchestrating agile operations, but were beyond the scope of this article.

The article also highlights the dependencies between each of the FuSE agility foundation concepts. In this context, the system-of-interest is the set of models and the model based digital engineering environment that consists of the tools, IT infrastructure, processes, and people.

The system of innovation operationalizes agility by sensing, responding, and evolving. One of the key elements to sense is the stakeholder needs. If these are not understood, the response will be ineffective. Continuous integration provides feedback in how well the stakeholder needs are being met. If this feedback is not provided, there is no learned knowledge to drive the necessary evolution of the SoI. ■

REFERENCES

- Department of Defense. 2018. Department of Defense Digital Engineering Strategy. Office of the Deputy Assistant Secretary of Defense for Systems Engineering. 09 May. www.acq.osd.mil/se.
- Willett, K., R. Dove, A. Chudnow, R. Eckman, L. Rosser, J. Stevens, R. Yeman R and M. Yokell. 2021. "Agility in the Future of Systems Engineering (FuSE) – A Roadmap of Foundational Concepts." INCOSE International Symposium, 31: 158-174.
- Maier, M. W. 1996. "Architecting Principles for Systems-of-Systems." INCOSE International Symposium, 6(1), 565-573.
- Noguchi, R. 2022. "Converging MBSE Methodology" NDIA Systems and Mission Engineering Conference. November. NDIA Proceedings (dtic.mil).
- Papke, B., M. Haus, D. Hetherington, and S. McGervey. 2022. "MBSE Model Management Pain Points – Wait, this looks familiar!" NDIA Systems and Mission Engineering Conference. November. NDIA Proceedings (dtic.mil).
- Flyvbjerg, B. and D. Gardner. 2023. "How Frank Gehry Delivers On Time and On Budget." *Harvard Business Review.* January-February. Accessed 03 March 2023, https://hbr.org/2023/01/how-frank-gehry-delivers-on-time-and-on-budget.
- Schwaber, K., and J. Sutherland. 2013. "The Scrum Guide." Scrum.Org, 13 March. https://www.scrum.org/resources/scrum-guide.
- Schindel, W., and R. Dove. 2016. "Introduction to the Agile Systems Engineering Life Cycle MBSE Pattern." Proceedings International Symposium. International Council on Systems Engineering. Edinburgh, Scotland, 18-21 July.

ABOUT THE AUTHORS

Barry Papke is the manager for the cyber system industry process success team for CATIA/No Magic. He has thirty-five years of systems engineering and operations analysis experience in the aerospace and defense industry across the entire systems engineering lifecycle from concept development through integration, test, and post-delivery support. Throughout his career, he has been actively involved in application of model-based methods including requirements management, enterprise architecture, cost estimation, system design, and operations analysis. Barry has a BS in mechanical engineering from Texas A&M University and a MS in systems engineering from Steven's Institute of Technology. He is a member of the INCOSE Agile and Security Working Groups and the MBSE initiative.

David Hetherington is a principal systems engineer with System Strategy, Inc. He serves multiple defense and commercial industry sectors. He has extensive personal experience in designing and leading design teams for both software and hardware covering an unusually broad range of system types. These complex systems have varied from real-time control to software internationalization, to offshore oil drill ships, to enterprise software applications, to automotive radar chipsets, to electronic publishing, and more. In addition to MBSE, he has a strong concentration of domain knowledge in safety, reliability, maintainability, and diagnostics.

Matthew Hause is an SSI principal and MBSE technical specialist, a former PTC fellow, a co-chair of the unified architecture framework (UAF®) group, and a member of the OMG SysML specification team. He has been developing multi-national complex systems for over 45 years as a systems and software engineer. He started out working in the power systems industry and has been involved in military command and control systems, process control, manufacturing, factory automation, communications, supervisory control and data acquisition (SCADA), distributed control, office automation, and many other areas of technical and real-time systems. His roles have varied from project manager to developer. His role at SSI includes mentoring, sales presentations, standards development, presentations at conferences, specification of the UAF profile, and developing and presenting training courses. He has written over 100 technical papers on architectural modeling, project management, systems engineering, model-based engineering, and many other subjects.

An Agile Systems Engineering Process for Stakeholder Needs Identification and Solution Concept Design

Lymari Castro, lcastr2@radium.ncsc.mil

■ **ABSTRACT**
This paper presents a case study where an agile systems engineering process was used to identify stakeholder needs to design an improved cross-organizational proposal development process during the proposal formulation phase of a program. The agile systems engineering process leveraged the incremental application of design thinking techniques to engage the stakeholders and identify the care-abouts of an organization during proposal formulation in support of a change management effort. The goal of the change management effort was to design solutions that increased collaboration and engagement across the various internal and external stakeholders without changing the overarching corporate proposal development process. The identified solutions broke existing organizational silos and changed the dynamics of the organization impacting over 1,200 employees. The case study relates to the future of systems engineering (FuSE) concepts of stakeholder engagement and agility across organizational boundaries.

INTRODUCTION

In 2019, the Goddard Space Flight Center (GSFC) Engineering and Technology Directorate (ETD) initiated a change management effort to improve their participation in proposal formulation (in this paper referred as the proposal process). During the proposal formulation phase the organization provides feasible engineering concepts and cost estimates for new proposals. For many years, the organization faced the challenges of providing timely engineering support to the proposal process. In 2019, the organization began a change management effort to look at the organization as a system and codify its architecture (people, processes, tools, and behaviors) with the goal of optimizing the proposal process from the perspective of the organization to improve the ETD experience with the process.

BACKGROUND—THE NEED FOR CHANGE

The ETD organization designs feasible engineering concepts and develops cost estimates that are included in the proposals to compete for funding of new projects. Proposal development is a cross-organizational process, that includes the collaboration of science, engineering, and finance resources (among others) for 12 to 18 months to formulate proposals for new projects. In this process, multiple proposals are being worked simultaneously at different stages of the process. The organization did not have a full visibility into the announcement of opportunities for project proposals and often the organization was engaged by other organizations to provide engineering expertise late in the process, that is, six months before proposal due date.

The ever-changing nature of the scientific and technology environment, combined with competing internal and external priorities and an evolving engineering workforce, placed a significant burden in the organization to be able to meet the demand for engineering support. Over time, the demand for engineering support outpaced the organization's capacity to support the proposals, resulting in employee burnout. Organizational silos resulted in unclear priorities and limited ability to transfer knowledge and skills for reuse between proposal teams.

In 2019, the organization initiated a change effort to improve their engagement with the proposal process with the goal of taking a more proactive stance to support proposal formulation and increase employee engagement and satisfaction. The organizational goals were to identify the top challenges and care-abouts of the workforce supporting the proposals, and design solutions that will improve the experience of the engineering organization during proposal formulation while satisfying the needs of the entire organizational workforce and without changing the overcharging corporate proposal process.

Figure 1. Design thinking techniques used in the case study

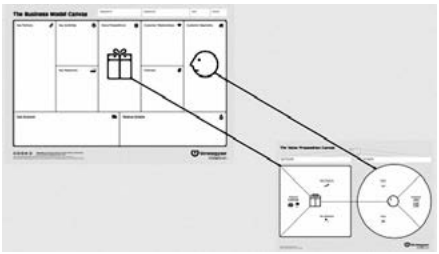


Figure 3. Value proposition canvas and business model canvas relationship (Ref. 6)

THE AGILE SYSTEMS ENGINEERING PROCESS

The change management effort used an agile systems engineering process that leveraged design thinking techniques for requirements identification and solution concept design in an incremental manner. The agile systems engineering process gradually engaged the various stakeholder roles in the organization to gather their needs, pain points and opportunities for improvement of the proposal development process. In addition, the agile systems engineering process used the techniques from lean startup such as the value proposition canvas and the business model canvas as tools to map potential solutions to pain points and opportunities from the perspective of each stakeholder segment in the proposal process.

Figure 1 shows the design thinking techniques that were incrementally used during the case study and the order in which they were used. The techniques used during the case study are from the LUMA Institute of Innovation (Ref. 1).

ACHIEVING STAKEHOLDER ENGAGEMENT THROUGHOUT THE CHANGE EFFORT

One of the biggest challenges was to bring all the stakeholders onboard of the change effort and make them active participants of the initiative. The first step to engage all the stakeholders into the change effort, was to gather their individual perspectives through a set of interviews.

a) **Structured Interviews:** During the initial three months of the change effort, individual structured interviews were conducted with 27 members of the workforce representing a total of seven work roles who contributed to the proposal process. Each interview was two hours long and questions were tailored to the organization (see Figure 2), and framed to address the following categories: (1) understand the stakeholder role, (2) discover the problems each role was experiencing in the process, (3) validate the problems, (4) discover opportunities for improvement, and (5) validate potential opportunities for improvement. Interviews resulted in stakeholder engagement throughout the change effort since the various types of stakeholders were able to have a better understanding about how their individual contributions impacted other stakeholders in the proposal formulation process.

b) **Value Proposition Canvas and Business Model Canvas:** The information gathered in the interviews was captured in a value proposition canvas (Ref. 2) for each stakeholder role. This provided a mapping of problems and potential solutions for improvement and helped visualize the notional solution-problem fit.

Data from the interviews highlighted common challenge themes. Common themes identified were: cost estimation, concept development, and planning/ governance of the proposal process. The next step was to use the business model canvas to frame each common theme and capture the key elements needed to make improvements in that area of the proposal process. Key elements captured in the business model canvas were: the stakeholder value proposition for each role in the process, partnering organizations, and stakeholders for making process improvements. Figure 3 shows the relationship between the value proposition canvas and the business model canvas.

The next step of the agile systems engineering process was to codify the business process from the perspective of the organization by using the experience diagram technique to capture the participation of the various stakeholders in the proposal process.

Example of Questions

Stakeholder Role
1. Tell me about your role in the organization.
2. How much time do you spend working the proposal process?

Problem Discovery (pains)
1. What is the hardest part of your role?
2. What tasks take most of your time?

Problem Validation
1. Tell me about your last experience working with the proposal process.
2. How important is it for you to fix this problem?

Opportunity Discovery (gains)
1. Is there anything that can be done differently to make your job easier?
2. What do you think of making changes to the process?

Opportunity Validation
1. Do you see any barriers to change the process?

Figure 2. Example of interview questions

c) **Experience Diagramming:** The experience diagramming (Figure 4) technique was used to visualize the existing proposal process workflow showing the various steps and the roles in which the organization engaged the proposal process. The technique provided a holistic view of the proposal process from the perspective of the organization. This activity baselined the organizations' participation in the proposal process thus capturing the true complexity of the process, entrance and exit points of the various ETD internal organizations and roles in the proposal process, key decision points, dependencies, and bottlenecks in the process caused by multiple iterations of work required to achieve a feasible concept design that met proposal cost constraints.

After all individual stakeholders' perspectives were gathered for the various roles in the organization and the proposal process was codified from the perspective of the organization, the next step of the agile systems engineering process was to bring each stakeholder segment of the organization to collectively focus on the problems they wanted to get solved and collaborate to create solutions for those problems. This was achieved through a series

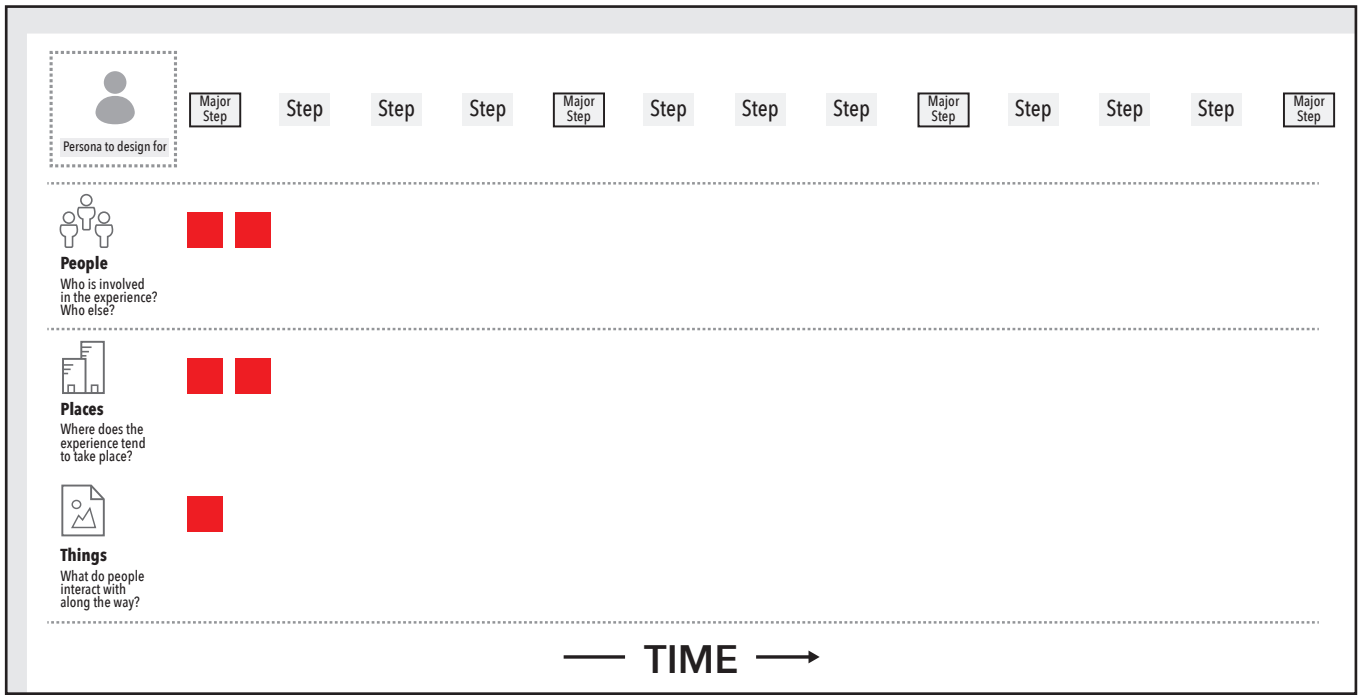


Figure 4. Experience diagram (Ref. 3)

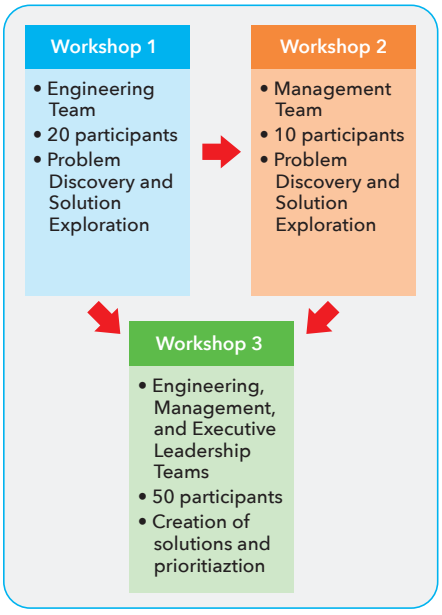


Figure 5. Incremental design thinking workshops

of design thinking workshops. Each workshop targeted a specific set of stakeholder segments in the organization.

Three design thinking workshops were conducted to incrementally engage the various stakeholders in the organization and guide them through the problem identification process and solution exploration process, as shown in Figure 1.

The first four-hour workshop targeted the systems engineering stakeholder segment of the organization. The second four-hour workshop targeted the management stakeholder group of the organization. The outputs of the first workshop were shared with the management stakeholder group before the second workshop, which helped them design solutions that worked for both stakeholder segments.

A third workshop engaged all the various stakeholders in the organization: systems engineers, management, senior executives, hardware engineers, software engineers, architects, mission leads, testers, and external stakeholders from other organizations. The goal of this eight-hour activity was to blend the perspectives of all the stakeholders to create a common vision for the future of the organization in the context of the proposal process. The activity ensured all stakeholder segments in the organization were represented and resulted in the design of solutions that addressed the needs of the entire organization instead of one portion of the workforce. The results from the previous two workshops were shared with all attendees in preparation for the third workshop, which helped the group craft solutions that built upon the previous designs completed by the management and the systems engineering groups.

The agile systems engineering process used the stakeholder mapping technique to understand the social network of the proposal process and the interactions of stakeholders internal and external to the organization.

d) **Stakeholder Mapping:** The stakeholder mapping technique was used to identify the social network of the proposal process in the context of the three challenge areas: concept development, cost estimation, and planning/governance. The technique created a view of the proposal process ecosystem and the interactions between internal and external stakeholders (Figure 6).

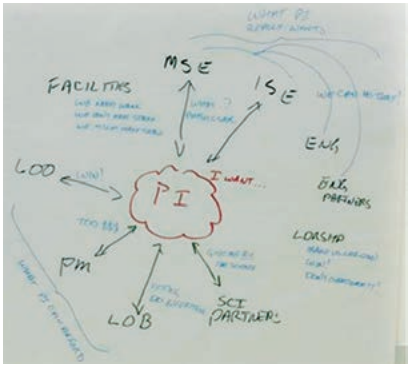


Figure 6 Stakeholder mapping example

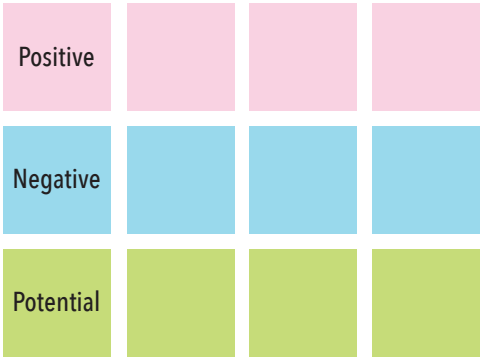


Figure 7 Rose, bud, thorn template

The next step was to gain a better insight into the challenges and areas for improvement. This was accomplished by using the *rose*, *bud*, *thorn*, and the affinity clustering techniques.

e) **Rose, Bud, Thorn:** The *rose, bud, thorn* technique was used to quickly identify positives (roses), negatives (thorns), and opportunities (buds) of the proposal process in each of the challenge areas (Figure 7). The activity provided deeper insights into the challenges and opportunities for improvement.

Figure 8. Affinity clustering example

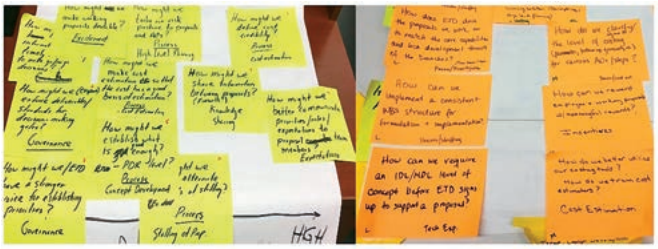


Figure 9. Problem statements framing example

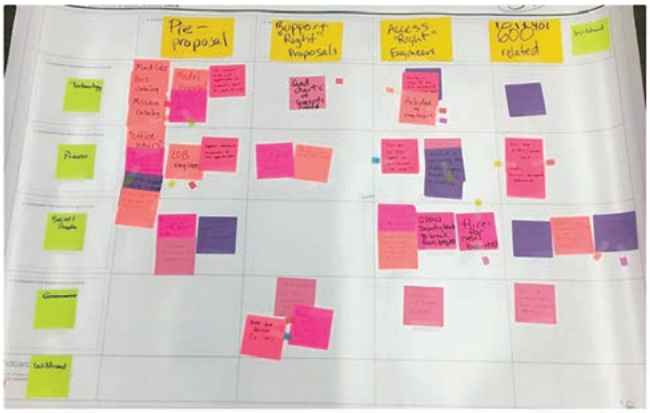


Figure 10. Creative matrix example

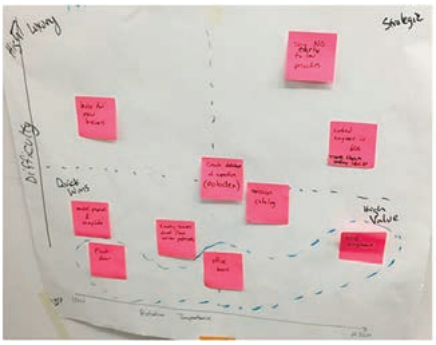


Figure 11. Impact-difficulty matrix example

f) **Affinity Clustering:** With the affinity clustering technique, the organization grouped the positives, negatives, and opportunities by similar categories, therefore creating a graphical representation of the issues, sharing insights between stakeholders, and gaining collective understanding of the common themes and their relationships (Figure 8).

After achieving common understanding amongst the stakeholders about the problem areas, the next step was to narrow down the scope of each problem by using the "How might we…" statement and *visualize the vote* techniques.

g) **"How Might We" Statements:** For each cluster, problems were framed by re-stating each cluster in a prescriptive approach that converted the problem into a question. Questions were framed as "How might we…" to narrow down the scope of the problem area and focus on a very specific aspect of the problem. The technique gave the opportunity for the team members to share their individual perspectives and brainstorm ideas for solutions.

h) **Visualize the Vote:** Each group voted for the top three "How Might We…" statements they wanted to explore in more detail during the solution exploration phase.

After consensus was achieved between all the stakeholder groups about the top challenges to be solved, the agile systems engineering process used the solution exploration design thinking techniques (see Figure 1). During solution exploration, stakeholders brainstormed potential solutions, then identified the most valuable solutions that were feasible, viable and desirable; and created concept designs for those solutions using storyboards and concept posters for their subsequent implementation.

i) **Creative Matrix:** By using the creative matrix technique, the organization explored the intersection of the various dimensions of the proposal process such as Leadership, Governance, Technology, and Process against the steps in the proposal workflow thus identifying ideas for solutions in each of the intersections in the creative matrix (Figure 10).

j) **Impact-Difficulty Assessment Matrix:** The impact-difficulty matrix technique plots the relative importance and level of difficulty of implementing each solution. The technique promoted deliberation between stakeholders until consensus was achieved about the relative importance of each proposed solution. Solutions were categorized as luxuries, quick wins, high value, or strategic. Relationships and dependencies between solutions were identified, creating a roadmap for their implementation (Figure 11).

k) **Storyboards:** Storyboards created the context for the implementation of high value solutions and built common understanding among the stakeholders about how the solution will work, thus providing the requirements needed for the implementation of the solution (Figure 12).

l) **Concept Posters:** Concept posters (Figure 13) captured how high value solutions will work, the resources needed for their implementation, and a notional timeframe for implementation. This technique created a common vision and roadmap for the future state of the proposal process. Combined, the storyboards and concept posters provided the requirements and the concept design for solutions.

Figure 12. Storyboard example

Figure 13. Concept poster example

ACHIEVING AGILITY ACROSS ORGANIZATIONAL BOUNDARIES DURING SOLUTION IMPLEMENTATION

After the storyboards and concept posters were created for the high value solutions that were viable, desirable, and feasible for the organization, cross-organizational teams were created to implement each of the high value solutions. As a result of the third design thinking workshop, the stakeholders in the organization were fully engaged and eager to implement the solutions. This high level of engagement resulted in people self-identifying as leads for the implementation of solutions. Once the leads were in place, the organizational leaders invited others to participate in the implementation of solutions. This resulted in self-organized implementation teams with representation from all segments of the organization.

By using the information from the impact-difficulty matrix, the concepts posters and storyboards as guidance, each team developed long term strategic plans and one-year action plans. The one-year action plans and the strategic plans were used to create a backlog of tasks for each team to realize the full implementation of each solution. By applying the concepts of agile development, teams broke each task into subtasks to manage the implementation work in sprints. They also engaged other external stakeholders as champions of the changes in their respective organizations.

As result of the change effort, the organization implemented solutions that provided them the ability to be more flexible and proactive in their interactions across the organizational silos to be more effective during proposal development, and improve cross-organizational collaboration with more visibility, early awareness of proposal request announcements for better matching the demand to the capacity for proposal development support, and better transfer knowledge across teams and organizations as a way of optimizing the end-to-end workflow to support proposal development. This resulted in emotional and process innovation (see Figure 14). The one-year change management effort provided the foundational requirements for the successful implementation of improvements to the proposal process and created the foundation for future optimization that will result in technology innovation such as digital engineering.

Agility across organizational boundaries was achieved by implementing the following changes:

- Process Innovation: Organizational management identified earlier points of engagement in the proposal process during the proposal formulation phase. This resulted in changing some of the roles and responsibilities of the stakeholders, and the creation of new roles. In addition, new points of engagement with organizational external stakeholders were identified in the earlier phases during proposal pre-concept formulation to ensure the organization was ready to provide support to proposal development when needed by the external stakeholders.
- Emotional Innovation: The organization identified new ways of sharing knowledge and lessons learned across proposals and teams. Reusing knowledge across proposals accelerated concept development and cost estimation processes.
- Emotional Innovation: The organization created new ways of recognizing the engineering workforce in a timely manner when working proposals. This resulted in new types of incentives and awards for working proposals, even when the proposal did not get awarded. This resulted in increased workforce engagement and motivation throughout proposal development.

CONCLUSION

The agile systems engineering process relied on the use of design thinking practices to incrementally guide the organization through problem solving and to identify solutions that were

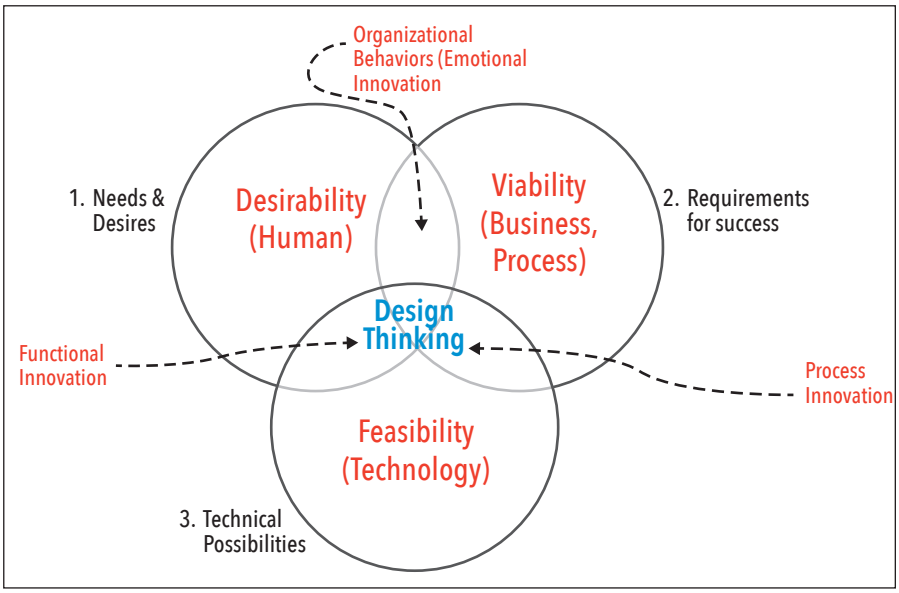


Figure 14. Design thinking is at the intersection of desirability, viability, and feasibility (adapted from Ref. 4)

feasible, viable, and desirable (Figure 14), which resulted in emotional and process innovation. The agile systems engineering process used in the case study, anchored on identifying the needs and care-abouts of the organization (that is, desirability for change) leading to changes in organizational behaviors, and identifying the requirements for the change effort to be a successful activity that brought value to the organization resulting in process innovation (that is, viability).

The incremental design thinking process allowed to identify the true requirements for the change effort, and actively engaged stakeholders in the change process to design solutions that worked in their environment, thus creating motivation and ownership. By using incremental design thinking workshops, the perspectives of all stakeholders in the organization were captured, therefore creating a single vision for the organization. The identified solutions broke existing organizational silos and changed the dynamics of the organization impacting over 1,200 employees. ■

REFERENCES

- 1. LUMA Institute. n.d. https://www.luma-institute.com/about-luma/luma-system-explore-methods/.
- 2. Lean Startup. n.d. https://steveblank.com/2014/10/24/17577/.
- 3. Experience Diagram Template. n.d. https://www.mural.co/templates/experience-diagramming.
- 4. Design Thinking Venn Diagram. n.d. IDEO Global Design and Innovation Company, https://www.ideou.com/pages/design-thinking.
- 5. Design Thinking Venn Diagram. n.d. https://www.uxdesigninstitute.com/blog/desirability-viability-and-feasibility/.
- 6. Strategyzer. n.d. https://www.strategyzer.com/canvas.

ABOUT THE AUTHOR

Lymari Castro is a systems engineer and effort leader at the Department of Defense (DoD). Mrs. Castro holds a BS in physics from the University of Puerto Rico, a masters in engineering physics from Cornell University, and a masters in systems engineering from Stevens Institute of Technology. Mrs. Castro has a total of 23 years of working experience. Prior to joining DoD, she worked as an instructor of physics at the Polytechnic University of Puerto Rico and at Raytheon Technologies as a radar systems engineer. She has assumed a variety of technical roles and positions at DoD. She has been an analyst, project systems engineer, lead systems engineer of a DoD major acquisition program, lead systems engineering of a portfolio comprised of 50 software data management products, and effort lead of large organizational transformational efforts. In 2016, she was recognized with the National Intelligence Meritorious Unit Citation and she was selected for the competitive senior technical development program to enhance her expertise in agile systems engineering. She completed external assignments at the Systems Engineering Research Center, the Carnegie Mellon University Software Engineering Institute, and NASA Goddard Space Flight Center. Mrs. Castro is an avid agile practitioner, and she is a certified scaled agile framework consultant, large scale scrum practitioner, scrum at scale practitioner, enterprise business agility strategist, and agility team health facilitator. In addition, she is practitioner of lean startup and design thinking methods. She has multiple publications in international journals and briefed at several international conferences. She is a member of the International Council of Systems Engineering, Agile Alliance, Society of Women Engineers, and the National Defense Industrial Association. (LinkedIn Profile: www.linkedin.com/in/lymaricastro).

Applying Agility for Sustainable Security

Larri Ann Rosser, larri.rosser@rtx.com

ABSTRACT
Systems engineering faces ongoing challenges due to the pace of change in technology and needs as well as the complexity, resilience, and adaptability demanded of solutions. System security needs and challenges are a prominent factor in the increasing demands placed on solutions and the systems engineers who design and develop them. The adoption of program level agile execution is one strategy for addressing these escalating challenges. In this article we describe how the broadly adopted technical processes from the ISO/IEEE/IEC 15288:2015 standard can be executed using agile methods to realize a large complex solution. In addition, we provide specific recommendations for executing these processes in a manner that enables systems to be sustainably secure – that is, to retain the desired level of security throughout the life cycle.

OVERVIEW

Agile practitioners recognize three general phases of agile execution. *Inception* is the period in which the need is identified, the project is initiated, and the design envelope is defined. *Realization* is the period in which the solution is defined, developed, integrated, and tested in a series of short cycles. *Transition* is when the solution is moved into its operational context and begins to be used. This article describes agile operation and sustainable security techniques across the complete life cycle with special emphasis on the inception phase and the definition of a solution architecture that enables sustainable security. The goal of sustainable security is achieved by embedding security concerns and solutions in every aspect of the architecture – needs, behaviors, structure and characteristics and then leveraging that architecture description to drive design, development, integration, verification, and validation.

The approach defined in this article is recommended as a best practice for implementing large, complex programs in environments of rapid change, interoperability, and innovation. This type of adaptive execution is required for the future of systems engineering (FuSE).

Systems engineering today is a challenging field and trends suggests that challenges will only grow over time. The *Systems Engineering Vision 2035* published by INCOSE encourages systems engineers to prepare to "deal with the continuously changing environment, be more responsive to stakeholders, and become more competitive" (INCOSE 2022). The document also identifies three specific goals for future systems engineering practice:

- Architect balanced solutions that satisfy diverse stakeholder needs for capability, dependability, sustainability, social acceptability, and ease of use
- Adapt to evolving technology and requirements
- Manage complexity and risk.

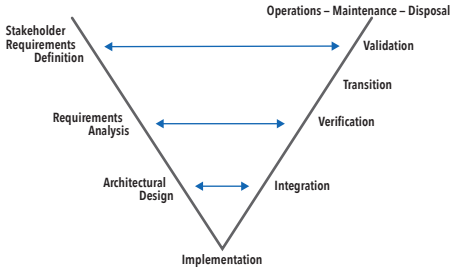


Figure 1. ISO/IEC/IEEE 15288 systems engineering technical processes depicted in the vee model

This article seeks to demonstrate application of systems engineering processes in the life cycle of a large-scale agile project, both to achieve the above goals in general, and also the more specific goals of creating systems that are sustainably secure — that is, they are not only secure when designed but maintain the desired level of security throughout their operational life.

SYSTEMS ENGINEERING TECHNICAL PROCESSES IN THE CONTEXT OF AN AGILE LIFE CYCLE

The focus of this article is the application of the technical processes, known as the 15288 standard (ISO/IEC/IEEE 2015), to a large complex project in an agile life cycle with the intention of defining, realizing, and operating a sustainably secure system. We acknowledge the role of the other process classes (technical management, agreement and organizational project enabling processes) defined in 15288 as applicable in this use case, but in order to control scope we limit our focus to the technical processes which have the most direct impact on our goals.

The INCOSE *Systems Engineering Handbook* (INCOSE 2015) states that the

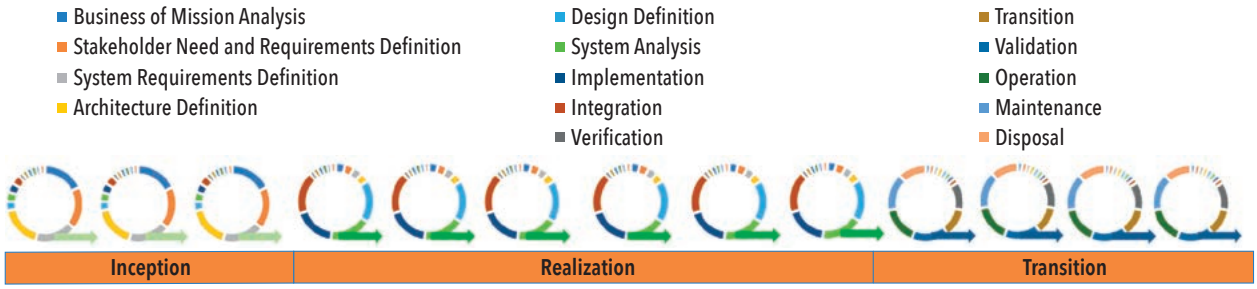


Figure 2. Technical processes in agile iterations

technical processes can be applied within any life cycle model. In a waterfall lifecycle, these processes are often represented as being applied over the systems engineering vee, as shown in Figure 1.

This representation illustrates a generally sequential execution of the processes with linkages between definition processes on the lefthand side of the vee and confirmation processes on the righthand side. Although it is acknowledged in the discipline that feedback cycles and iteration exist within the waterfall model, the general flow is from one process to the next, with delivery into operation occurring after the processes involved with design, delivery, and verification are essentially complete.

An agile lifecycle applies the same processes within a different operational cadence, as shown in Figure 2. Agile practitioners acknowledge three broad phases of activity within the lifecycle – inception, in which a need is articulated and the design envelope of the solution is defined; realization, in which the solution is designed, implemented, and, tested; and transition, in which the solution is moved into its operational use. Because agile execution proceeds in short cycles and incorporates feedback rapidly, all technical processes are applied throughout the lifecycle, but at different levels depending on which phase is underway. During inception, business/mission analysis, stakeholder needs and requirements, system requirements definition and architecture definition are primary. More detailed design and implementation processes are employed as needed (in prototyping and trades, for example) and operational processes are considered. In the realization phase, design, analysis, implementation, integration, and verification are the primary processes, with earlier processes revisited as updates are suggested by discovery, and the operational processes are considered in planning and invoked for early operational testing and similar activities. In the transition phase, transition, operation, validation, maintenance, and disposal are the primary processes, with the other processes being invoked when updates are determined to be needed. The execution lifecycle shown in Figure 2 is recommended for projects desiring to create sustainable security and is assumed by other recommendations in this article.

General Recommendations for Sustainable Security

Two semesters of academic directed research yield the following recommendations for enhancing the sustainable security posture of solutions through specific applications of the process system described by ISO/IEC/IEEE 15288.

Execute using an agile lifecycle: Agile execution enhances the ability to respond to changes in context and technology, which is critical to maintaining the desired security posture.

Include security personnel and considerations early in the inception phase: Both literature review and workshops with system security engineers highlight opportunities to better embed security into the solution architecture by explicit inclusion early in the concept development cycle.

Elicit security needs and challenges in business/mission analysis and stakeholder needs and requirements: These standard early processes provide an opportunity to evaluate security needs from the perspective of owners and users in the context in which the system will operate.

Include loss-driven and malicious user perspectives in analyses: Loss-driven engineering provides an approach to refine understanding of customers' risk tolerance, which is not likely to be uniform across all possible areas of risk. Inclusion of known malicious use cases allows the identification of specific high-risk scenarios for this customer and solution.

Express security elements in requirements and architecture: Working sessions highlight that security is frequently treated as a design detail that does not impact functional requirements or architecture. This leads to erosion of security posture through trades and refactoring as well as limiting the inclusion of security considerations in regression testing.

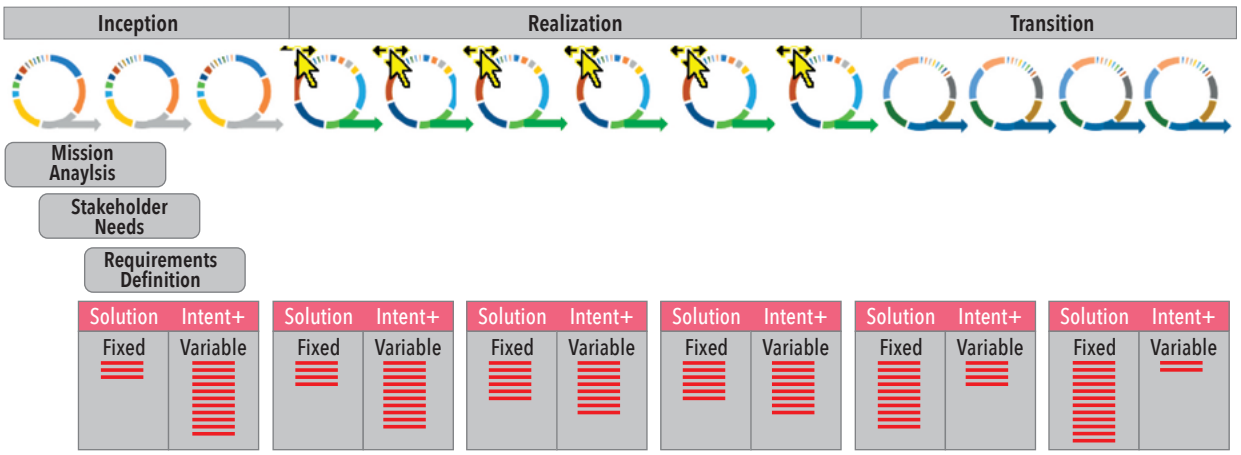


Figure 3. Agile approach to requirements definition

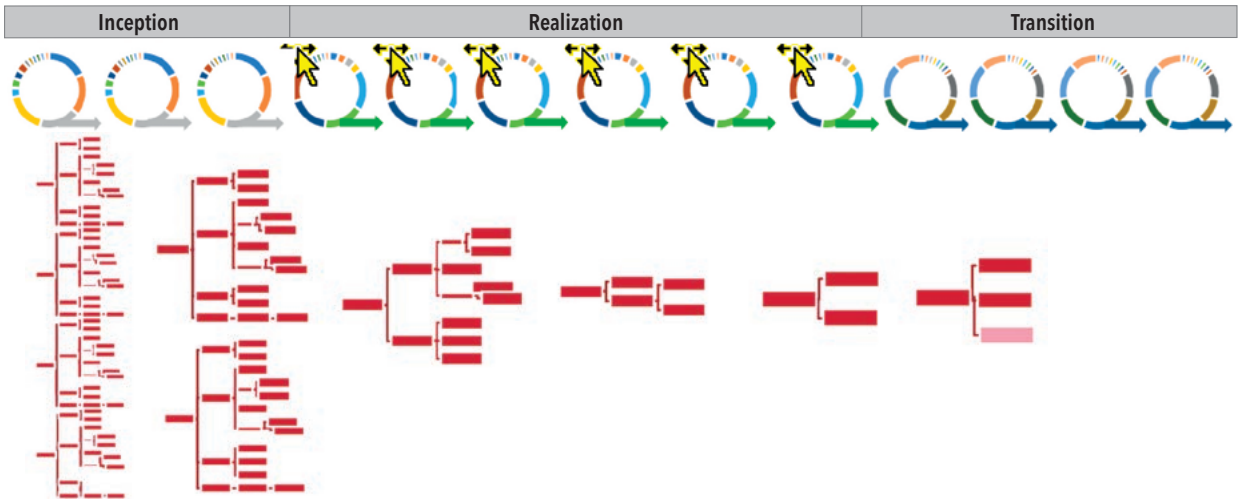


Figure 4. Evolving the solution set via trades and AoA

Agile Approach to Requirements

Requirements definition in an agile life cycle is an intentionally iterative process. Initially only those requirements that drive the architecture or fundamental top level solution decisions are defined and baselined. The rest are left as high-level needs which will be elaborated and baselined later based on the agile execution concept of making decisions at the "last responsible moment" (Balbes 2022). This paradigm of decision making recognizes that making decisions earlier than required increases the risk of rework and sub-optimal solutions due to changes in needs or context over time.

The concept of fixed and variable solution intent (Scaled Agile 2021) can be used to illustrate this approach. Requirements articulate the solution intent or respond to a stated need. Those parts of the intent that must be solidified to move forward with solution realization are defined as requirements and baselined. These represent the fixed solution intent.

At inception, only those elements of the need that define the design envelope are fixed. As realization progresses, analyses are performed, lessons are learned. and more parts of the solution intent are expressed as requirements and baselined. Figure 3 illustrates this approach.

Requirements Recommendations for Sustainable Security

Identify key security concerns through loss driven analysis: Once the capabilities needed for the solution are identified through mission/business analysis and stakeholder needs definition, identify potential losses associated with the required capabilities and determine which are of critical concern to the stakeholders.

Use unacceptable losses to select key misuse cases: Identify both careless and malicious acts that could trigger critical losses and express them as misuse cases.

Establish functional and performance requirements to address critical malicious behaviors and prevent unacceptable losses: Define functional requirements that prevent or mitigate the impact of misuse cases and protect against unacceptable loss.

Agile Approach to Analysis of Alternatives

Trade studies and analysis of alternatives (AoA) are an ongoing part of agile execution, one of the key practices that drive evolution of solution intent. Agile solution definition can be considered as iterations of set based design, with trades and AoA based on learning from previous iterations and changes in context driving the pruning or refactoring of the solution set tree. Figure 4 illustrates this concept.

Analysis of Alternatives Recommendations for Sustainable Security

Include key security characteristics (quality qttributes or QAs, defined with the architecture) as criteria for trades and AoA. In order for this approach to be effective, the defined quality attributes need to be specific. It is difficult to evaluate whether an approach is "secure" because security has many meanings and many facets. When defining QAs related to security, focus on attributes that specifically combat the misuse cases that can lead to unacceptable losses identified for the solution in question. For example, a system for which

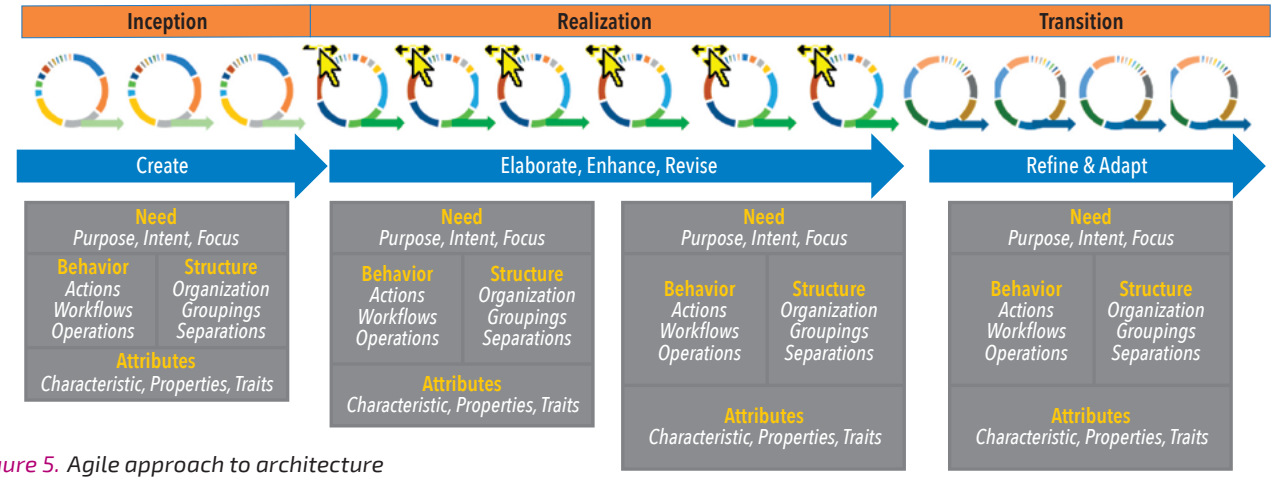


Figure 5. Agile approach to architecture

uptime is critical and distributed denial of services (DDoS) attacks are a key misuse case, quality attributes such as resilience and scalability are more helpful in identifying suitable solutions than "security."

Agile Approach to System Architecture

In an agile lifecycle, architecture is an iterative activity focused on defining a design envelope within which the solution is realized and then making and communicating critical system wide decisions that shape the solution.

In the inception phase, the "bones" of the architecture are defined, providing fundamental definitions and decisions that embody the essential nature of the solution. Currently, no attempt is made to define an exhaustive architecture. In fact, some elements are intentionally left undefined because the architect judges that better decisions can be made later in the cycle.

In the realization phase, the architecture is fleshed out lessons are learned, choices are made, and user feedback is received. At any point in the realization process, there are a set of possible architectures for the solution, and as realization progresses, the set becomes more refined based on learning from previous cycles, evolution in technology and mission and the architectural decisions that are made.

In the transition phase (while the system is in operation) the architecture continues to be refined and adapted as needed. If wholesale architecture changes are needed, the solution may spawn another cycle of realization with the intent to either replace or evolve the existing solution. Figure 5 illustrates this approach to architecture.

Architecture Recommendations for Sustainable Security

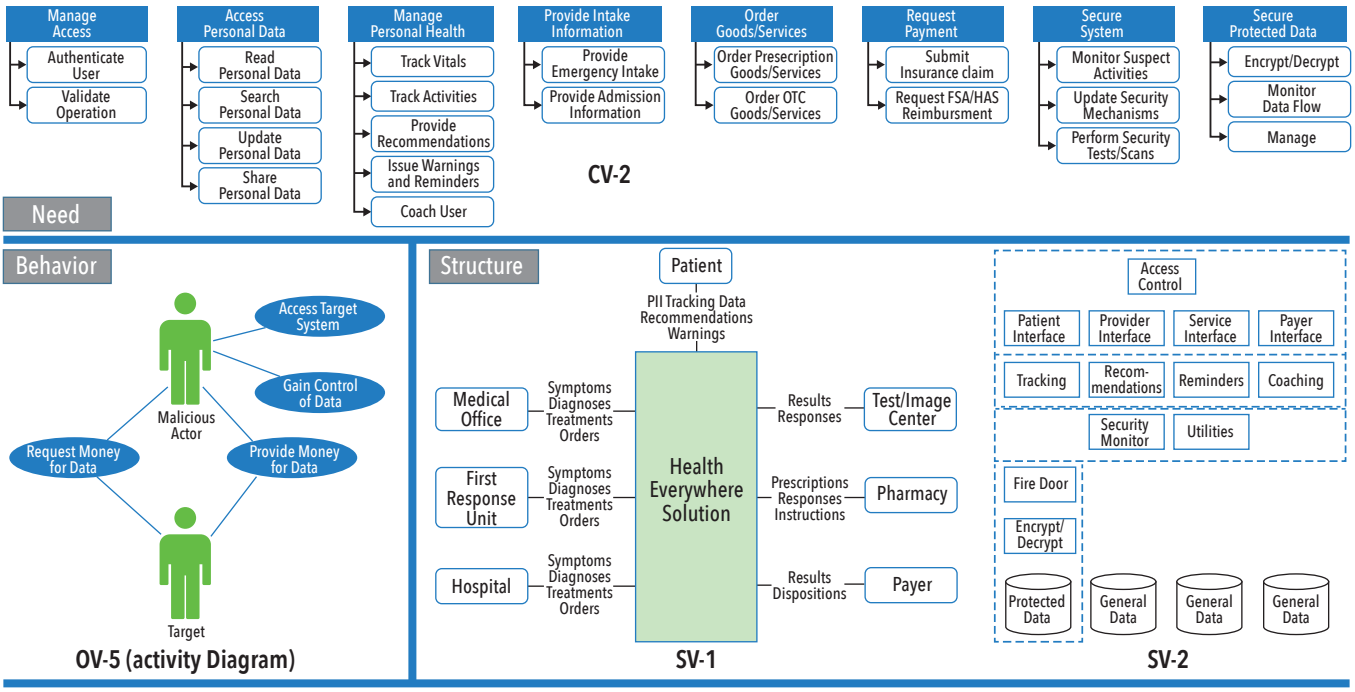
Integration of system security concerns into solution architectures is a critical step to ensuring sustainable security. Solution architectures guide design, development, testing and deployment so leveraging the architecture to communicate security concerns, characteristics, and decisions ensures that they are embedded in the solution. The following paragraphs describe some techniques for integrating security into architecture, using a medical records access system as an example.

An architecture describes four essential aspects of a solution: the need it is to fulfill, the behavior it must perform, the structure it must exhibit, and the characteristics it must embody. Adherence to these elements drives the design, development, and evolution of the solution, and proper articulation of the architecture ensures that the solution meets the stated needs (Figure 6).

Need: We express the solution needs, identified through mission/business analysis, stakeholder needs definition and system requirements definition, as capabilities in a DoDAF CV-2 capability taxonomy diagram. One of the security-related capabilities uncovered during loss driven analysis is to secure protected data (PII).

Behavior: Since exposure of PII is most often associated with criminal hacking, we include a misuse scenario "money for data" that describes the malicious acts that must be guarded against to protect the data. This misuse case will inform validation activities from peer reviews to validation security challenge scenarios.

Structure: In a DoDAF internal block diagram (SV-2) we identify a separate segment in the architecture to house only protected data. This segment can only be accessed with supplemental authorization and can be isolated when an attack is detected using "fire doors". This structural choice will inform the design of virtual machines, containers, and storage units and drive the configuration of firewalls and guards.



Characteristics

Qualify Attributes

Attribute	Priority	Use Case	Principle
Data Protection	Very High	PII Exfiltration Challenge	Limited access, isolation, access validation
Useability	Very High	Emergency Intake	User centered design
Affordability	High	First Year Cost	Just enough capability
Accessibility	High	Rural International Access	RESTful transactions, limited NRT data transfer
Resiliency	High	DDos Attack	Diverse communications, resource auto-spawn
Modularity	High	PII Isolation	Encapsulation, limited global data, loose coupling
Scalability	Medium	DDos Attack	Auto-scale with guardrails
Maintainability	Medium	Security Update	Light off operation

Design Principles

Figure 6. Example: embedding security considerations in architecture

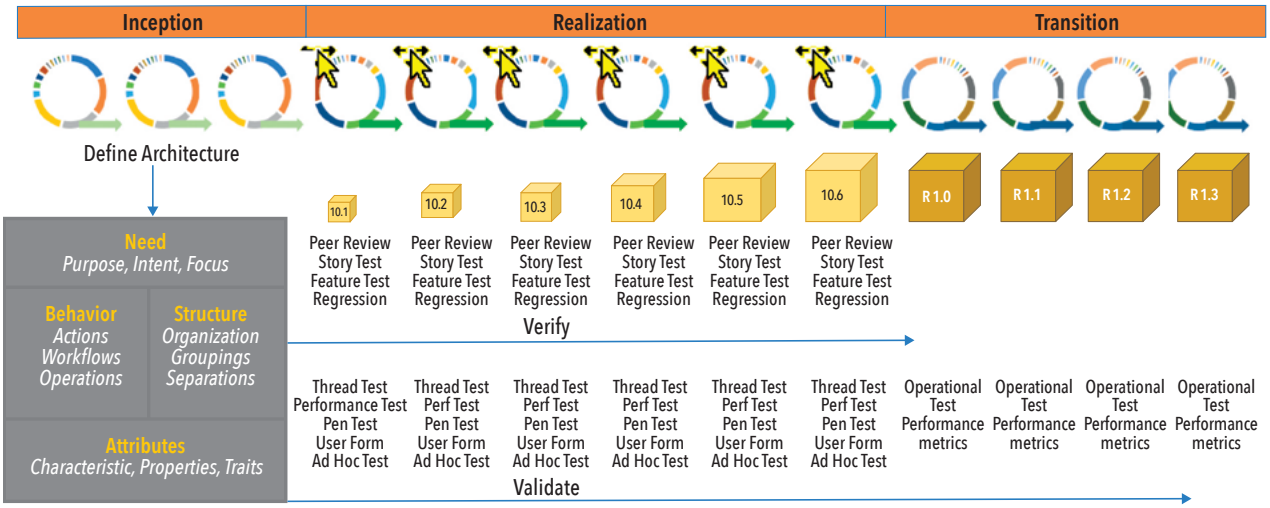


Figure 7. Agile approach to system test

Characteristics: We define a high priority quality attribute of modularity to ensure the isolation of segments from one another in order to maintain a high level of protection. This characteristic will supply criteria for trades and analysis of alternatives.

TEST AND EVALUATION

Agile Approach to Test and Evaluation

In agile systems, both verification and validation begin early in realization. Verification comes in the form of unit tests for features and stories. A best practice in agile development is the use of test driven development (TDD) in which the tests are defined and implemented using test automation tools such as Cucumber or Jmeter before the code is developed. The developers run the tests against their code until it passes, then the code is submitted for build and higher-level test. If issues are found, they are corrected in the code, and when possible, the unit test is updated to catch the error. Once both the code functionality and test validity are verified, both the code and its associated test are checked into the baseline. The unit test will be used against future updates to this code and in regression testing, which occurs at some level at least daily.

To begin validation early, a variant of the TDD process, called acceptance test driven development (ATDD) is used. This technique defines a series of system level scenarios or workflows which exercise the capabilities of the system in the manner that emulates actual use. These scenarios are vetted with customers and users to ensure that the are representative of expected usage. Tests to exercise these scenarios are developed and then all functions are stubbed out, resulting in an initial test that steps through the workflow without exercising any functionality. As stories and features are completed, they are fitted into the appropriate scenario test and run. This allows validation of the new functionality within a mission-relevant scenario or scenarios — some core functionality appears in many scenarios. Once the updated test runs properly, it is submitted to the baseline to be used in regression and to support the testing of additional functionality for that scenario.

These automated tests form a critical part of the demonstrations that occur at the end of each sprint and increment. Test procedures are discussed, test reports shared, and tests run in the presence of the customer. In some cases, users will replicate the automated tests manually or perform free-form exploratory testing of the functionality in order to provide feedback on the functionality as it will be used in operations.

The architectural description of a solution plays a significant role in agile test planning. The behavioral description in the architecture (capabilities, services, workflows, use cases, etc.) inform the mission scenarios used for validation. The behavioral description also guides the breakdown of functionality into features and stories to be implemented and the development of tests to verify their completion.

The structural portion of the architecture description identifies interfaces and defines their functionality, which is used to develop automated interface tests as well as to integrate information exchanges into scenario testing.

The architectural characteristics (quality attributes and architectural principles) enable analysis for peer reviews. Along with the needs, structural and behavioral descriptions of the architecture, they support operational validation activities such as red/blue exercises and day-in-the-life (DITL) tests.

As shown in Figure 7, all these verification and validation activities begin early, with formal test run-throughs happening at the end of the very first sprint or increment. It is understood that early tests only exercise a small subset of the system, and that each iteration of testing will incorporate both test of new functionality and regression and validation of the cumulative baseline.

Test and Evaluation Recommendations for Sustainable Security

Include response to malicious acts mission thread(s): Create scenarios driven by a malicious actor attempting to inflict harm on the system. The scenario starts with an attack by a malicious actor and exercises the solution's ability to detect, repel, work through, and recover from the attack.

Include test of defense against key losses as part of regression: Include then mission scenarios described above in each regression run to keep track of the solution's overall security posture. Since these scenarios may be long and resource intensive, consider alternating among attack scenarios with each run.

Include tabletop and live red/blue exercises in response to unacceptable losses as part of increment demonstrations: While automated tests provide an efficient method of identifying likely security issues, human-in-the-loop tests do a better job of identifying new threats and corner cases. Plan tabletop and on-system red/blue exercises to ferret out security weaknesses throughout the system life cycle.

RISK MANAGEMENT

Agile Approach to Risk Management

The basic artifacts and techniques of risk management — risk identification, risk evaluation, risk register, risk board, risk handling, etc. — are in general applicable in an agile lifecycle. Differences occur in the details of implementation as shown in Figure 8.

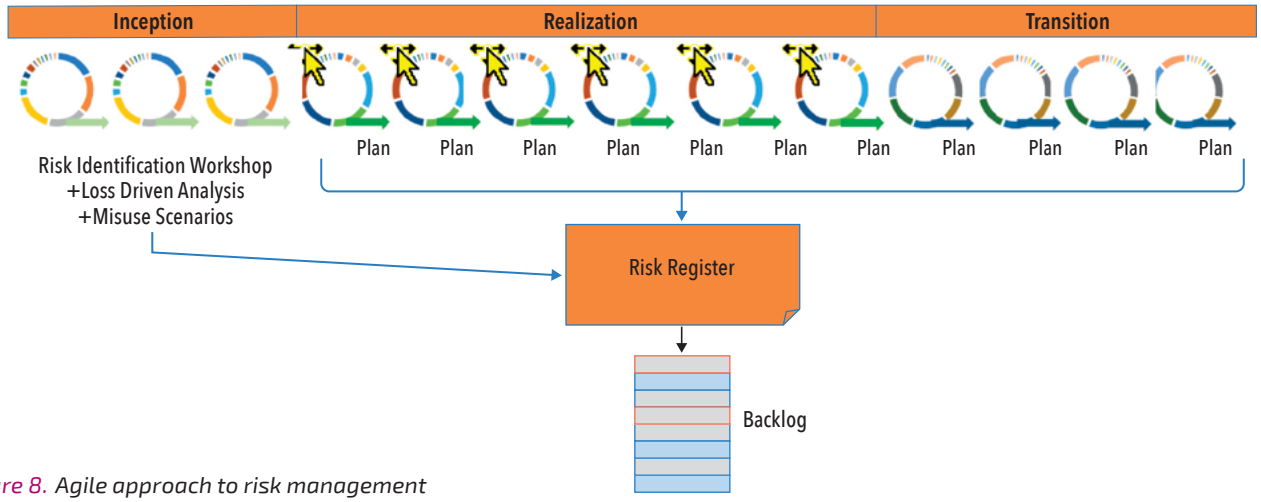


Figure 8. Agile approach to risk management

Agile planning events such as backlog grooming, sprint planning, and increment planning provide additional opportunities to identify risk. Many risks identified by individual teams can be managed within the team, but some are significant or widespread enough that they are presented to the program risk board for management.

The agile backlog provides a mechanism for scheduling and performing risk handling activities. Once a risk mitigation plan is prepared, it is broken into features and stories like any other work, added to the backlog and prioritized. This provides outstanding visibility into whether the risk activities are being performed and how they are being prioritized relative to other activities.

Risk Management Recommendations for Sustainable Security

Use loss-driven analysis during the inception phase to identify critical customer risks: To enhance the solution's sustainable security posture, apply loss-driven analysis techniques, which can enhance risk identification through articulation of unacceptable losses and the identification and analysis of key misuse cases that apply to the solution context.

Make review of current security threats a standard part of risk review process: Since specifics of security risk are constantly changing, a system security subject matter expert (SME) should review the current state of the security risk environment with the risk management board and the board should discuss whether that state merits any changes in recorded risks or their handling plan.

An example risk from a medical records access system provides an illustration of novel aspects of risk management in an agile context. "Mass PII exfiltration" was identified by the customer as an unacceptable loss in the loss-driven analysis of stakeholder needs. Table 1 shows the risk register entry developed for this risk.

Table 1. Risk register entry

Ref #	MRP-001
Date Identified	01/15/2023
Risk Title	Mass PII exposure
Risk Description	IF PII data is made public THEN the company will be liable or remediation costs, be exposed to civil lawsuits and suffer reputational damage that can lead to loss of business
Probability	0.2
Consequence	$10M
Expected Monetary Loss	$2M
Risk Adjusted Loss	$2.25M
Risk Response	Mitigate
Contingency Action Trigger	Data exfiltration detected
Annual Sales	$50M
Risk Tolerance	$32

This risk will be handled through mitigation, but a contingency plan will also be put in place in case mass exfiltration does occur. Contingency plans are common for ongoing security risks.

The risk mitigation plan for this risk is shown in Table 2.

Step 1 of the risk mitigation plan focuses on embedding exfiltration protection in the solution architecture in the inception phase. Notice that this step has been broken out into individual smaller activities that are inserted into the backlog of work for the current increment. They are scheduled for specific sprints and assigned to specific individuals.

During realization, specific design, analysis and test activities will ensure that the architectural constructs are implemented and doing their job. During transition (operations, sustainment, and disposal) the operational security team, guided by the operational security plan, will perform activities such as periodic stress testing and disciplined disposal of assets to protect against data exfiltration. These steps are not yet broken down into story-sized activities but are placed on the backlog in conceptual form to act as planning placeholders for future risk mitigation work.

Figure 9 shows the change in risk probability over time. This particular risk is mitigated primarily by reducing the probability of occurrence as there is little that can be done to reduce the impact if it occurs. The color bands show the general risk level at any point in time — above 15% is considered

Table 2. Risk mitigation plan

Ref #	Step	Time Phase	Owner
1	Architect for Exfiltration Protection	Inception	Architect
1.1	Include exfiltration-related misuse cases in operational architecture	Sprint I-1	Architect
1.2	Define exfiltration-related attributes in Quality Attributes Workshop	Sprint I-2	Architect
1.3	Define exfiltration resisting capabilities in Operational Architecture	Sprints I-3, I-4	Architect
1.4	Partition architecture to protect PII	Sprints I-3, I-4	Architect
1.5	Include exfiltration resistance scenario in tabletop evaluation of architecture	Sprint I-5	Security SME
2	Include Exfiltration Evaluation in Definition of Done	Realization	Product owner
3	Perform Exfiltration Resistance Testing as Part of Regression	Realization	Test Lead
4	Include Exfiltration Resistance in Operational Security Test Plan	Transition	Operational security Lead

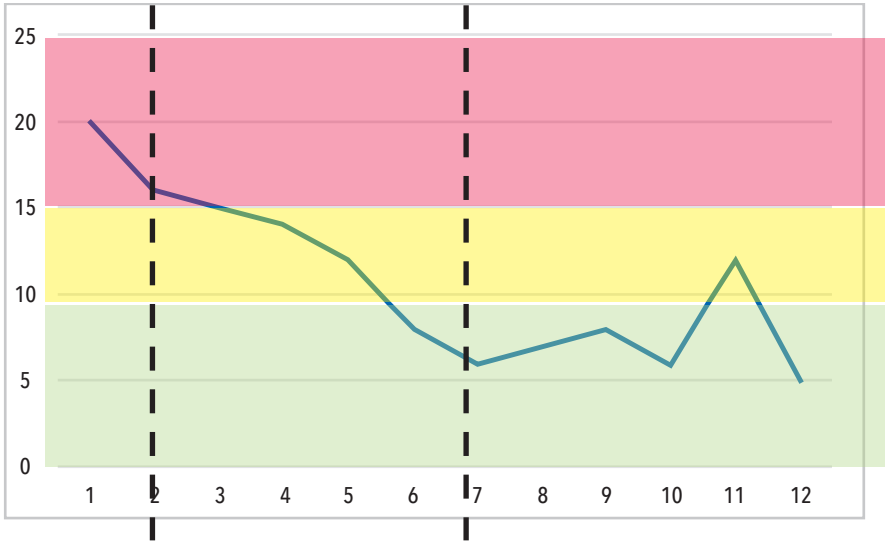


Figure 9. Change in risk probability over time

high risk, 10 to 15 percent is considered medium risk and below 10% is considered low risk. The lifelines indicate life cycle phase transitions. The left period is inception, the center period is realization, and the right period is transition. Risk drops sharply at the end of inception when security features have been designed. During realization, the risk moves down incrementally as the features are implemented and tested. In operations, the risk will rise based on exposure, operator behavior, and new threats, and then lower again based on patches training and procedure updates and refactor.

Agile Approach to Life Cycle Management

Since agile projects emphasize early and repeated value delivery, the processes associated with the transition phase — validation, operation, maintenance, and disposal — begin relatively early in the project's life span and continue until the system is retired, as shown in Figure 10. Large complex agile systems use a roadmap to identify key milestones, events, and transitions that occur throughout the life of the solution. These can include customer events, strategic operational changes, and solution lifecycle events that impact the capabilities and activities associated with the system. The roadmap provides a high-level framework for planning and prioritizing work associated with delivering system value.

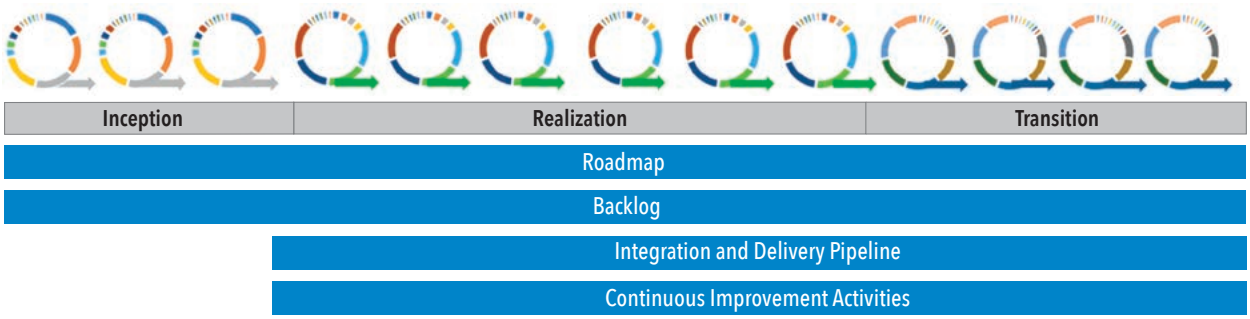


Figure 10. Agile approach to life cycle management

The backlog specifies the capabilities, features, and stories that create solution value. Capabilities are the large details of the work to be done in the new future of the solution. The backlog items prioritized include new or enhanced functionality along with enabling activities such as studies and plan development. During each backlog refinement event, future roadmap elements are considered and when necessary, activities in support of them are prioritized to be worked. In addition, the agile execution cadence includes periodic assessments (retrospectives) of the execution process and identification of improvement activities that are added to the backlog for implementation. As the solution moves into operation, the content of the backlog will include operations, sustainment, and disposal activities.

The integration and delivery pipeline is stood up as soon as realization begins. It is used to schedule and automate integration, verification, and validation work, including peer reviews, system builds, testing of new features, regression testing, performance testing, and penetration testing. In modern systems, an instance of the integration and delivery pipeline is delivered as part of the solution and continues to be used to integrate, test, and deploy bug fixes, security updates, and recapitalization content. Agile programs integrate continuous improvement activities into their cycles of activity and provide a mechanism for the solution to evolve.

Process System: Life Cycle Management Recommendations for Sustainable Security

Perform periodic reviews of current security threats to identify potential changes: Some general security threats persist throughout the life of a solution, but both solution vulnerabilities and malicious actor tactics may change over time. To sustain the solution security posture, the operational security team should evaluate security risks for such contextual changes and, when necessary, update training, procedures or technology in response.

Include security requirements and characteristics in analysis of updates, additions, and removals: When system changes are planned during the transition phase, impact to the system security posture should be considered and the change should be executed in a way that preserves system security.

SUMMARY

Systems engineering is in a period of change and evolution based on demand for increasingly complex, adaptable, and interoperable solutions to a wide range of needs. Escalating system security demands are an instance of this pattern. In this paper, we describe how the effective and well-established processes of ISO/IEC/IEEE 15288 can be implemented in the context of an agile life cycle in order meet the challenges of future systems engineering in general and sustainable security in specific.

This approach embraces the inevitability of change and treats solution realization as ongoing refinement of a solution set focused on meeting the established need. Rather than executing in a primarily sequential fashion, technical processes are revisited frequently, refining the solution intent based on learning gained from activities completed as well as changes in mission need, operational context and both social and technological landscapes. In this paradigm, architecture provides guidance for execution the technical processes of realization and transition while the architecture itself evolves and matures in response to frequent feedback.

This modern approach to systems engineering employs well-honed processes and methods in a dynamic environment. The goal is not to resist change or enforce a static solution definition, but rather to harness change to produce solutions that meet needs in a forward-leaning fashion and remain relevant and useful throughout the solution life. ■

REFERENCES

- Balbes, M. 2022. *Want to Make the Right Decision? Procrastinate Until 'The Last Responsible Moment* [Online] Viewed 19 February 2023. https://adtmag.com/articles/2022/06/09/the-last-responsible-moment.aspx.
- Dove, R., and W. Schindel. 2016. "Introduction to the Agile Systems Engineering Life Cycle MBSE Pattern." INCOSE International Symposium 26(1):725-742.
- Health Sector Cybersecurity Coordination Center (HC3). 2019. A Cost Analysis of Healthcare Sector Data Breaches. [Online] Viewed February 26, 2023. A Cost Analysis of Healthcare Sector Data Breaches | Technical Resources | ASPR TRACIE (hhs.gov).
- INCOSE. 2023. *Systems Engineering Vision 2035: Engineering Solutions for a Better World Executive Summary*, International Council on Systems Engineering, San Diego, US-CA.
- INCOSE. 2018. *INCOSE System of Systems Primer* International Council on Systems Engineering, San Diego, US-CA.
- INCOSE (2015). *Systems Engineering Handbook: A Guide for System Life Cycle Process and Activities* (4th ed.). D. D. Walden, G. J. Roedler, K. J. Forsberg, R. D. Hamelin, and T. M. Shortell (Eds.). San Diego, CA: International Council on Systems Engineering. Published by John Wiley & Sons, Inc.
- ISO/IEC/IEEE. 2015. Systems and Software Engineering — System Life Cycle Processes. ISO/IEC/IEEE 15288:2015(E) first edition 2015-05-15. CH.
- Ross R., M. McEvilley, and M. Winstead. 2022. Engineering Trustworthy Secure Systems. NIST Special Publication (SP) NIST SP 800-160v1r1. National Institute of Standards and Technology, Gaithersburg, US-MD.
- Rosser, L. 2022. "SYS598 Directed Research problem analysis results, July 2022" (unpublished work)
- Rosser, L. 2022. "SYS598 Directed Research solution recommendations, November 2022" (unpublished work)
- Scaled Agile. 2022. "SAFe for Lean Enterprise" Scaled Agile [Online] Viewed 1 February 2023. https://scaledagileframework.com.
- Scaled Agile. 2021. "Solution Intent." Scaled Agile [Online] Viewed 19 February 2023. https://www.scaledagileframework.com/solution-intent/.
- Scholl, M., et al. 2010. Security Architecture Design Process for Health Information Exchanges (HIEs). National Institute of Standards and Technology Interagency Report 7497. National Institute of Standards and Technology, Gaithersburg, US-MD.

ABOUT THE AUTHOR

Larri Ann Rosser has worked in engineering and technology for four decades as an electrical engineer, software engineer, systems engineer, and architect. She holds a BS in information systems and computer science from Charter Oak State College, and a Master of Science in systems engineering from Worcester Polytechnic Institute. She holds multiple patents in the man portable systems domain and is a CAP certified architect, a SAFe program consultant, and a Raytheon six sigma expert. She is the co-chair of the INCOSE Agile Systems and Systems Engineering Working Group, a member of the INCOSE Complex Systems Working Group, and a member of the NDIA SED Architecture Committee and the NDIA ADAPT working group. At Raytheon, she works with programs and product lines to apply modern methods to system realization.

JUNE 2023
VOLUME 26/ ISSUE 2

Agile Programs Need Agile Reviews

Larri Rosser, larri.rosser@rtx.com

■ **ABSTRACT**
Current technical oversight approaches used for government programs (for example, stage-gate reviews) are not agile — their expectations are not aligned with agile development cadences, and they are not adequately responsive to continuous unpredictable change. This article explores ways to provide insight and responsive forward looking actionable guidance for agile projects in the context of government and defense programs. It proposes a general oversight approach that produces minimal drag and disruption and keeps pace with agile product development.

INTRODUCTION

For at least a decade, the US government has been exploring the use of agile methods on defense and other federal acquisitions in order to reduce costs while more effectively supporting missions. Compared to commercial application software development efforts from which agile practices initially emerged, these acquisitions tend to be complex, completion-based efforts to realize systems, not just software. These challenges encourage evolution in agile execution, leading to the definition of numerous approaches to scaling and expanding agile practices. Descriptions of these advanced agile approaches can be found in commercial scaling frameworks such as scaled agile framework (SAFe) (Scaled Agile Framework 2021), disciplined agile (DA) (Program Management Institute 2020), and enterprise unified process (EUP) (Enterprise Unified Process 2020) as well as execution case studies from Lockheed Martin (Dove, Schindel, and Garlington 2018), Northrup Grumman (Dove and Schindel 2017), and Rockwell Collins (Dove, Schindel, and Hartley 2017), among other sources.

During this time, various agencies of the US government have studied agile execution and released several documents discussing the application of agile practices to government contracts. These include *Agile Assessment Guide: Best Practices for Agile Adoption and Implementation* (US Government Accountability Office 2020), *Design and Acquisition of Software for Defense Systems* (Defense Science Board 2018), and *The TechFAR Handbook* (TechFAR Hub 2022). In addition, they have worked with industry to provide instructions for applying specific agile practices to government programs, including earned value (National Defense Industrial Association 2019) and agile contracting (Office of Management and Budget 2012).

In the area of technical oversight, however, the standard recommended by the Department of Defense (Office of the Under Secretary of Defense for Acquisition, Technology, and Logistics 2017) is ISO/IEC/IEEE 15288.2, *IEEE Standard for Technical Reviews and Audits on Defense Programs* (IEEE Computer Society 2014). Although tailoring of criteria is permitted and even encouraged, tailoring this standard appropriately for agile programs presents challenges, as the standard is organized around the phases in the traditional waterfall lifecycle.

AGILE DEVELOPMENT VS. STAGE-GATE REVIEWS

Agile Systems Engineering Life Cycle Model for Mixed Discipline Engineering describes agile systems engineering as "effective system engineering in the face of uncontrolled change" and identifies "an asynchronous/concurrent life cycle model framework" as one of the hallmarks of agile systems engineering. (Schindel and Dove 2019) This approach to systems engineering does not fit comfortably with current prevalent gate review practices, which tend to assume that product realization activities occur sequentially and that reviews evaluate final outputs of one phase of engineering activities that has recently completed to determine if the project can proceed to the next (sequential) phase (Lapham et al. 2016 p. 19). Mismatches between the expectations of agile engineering and traditional gate reviews can lead to several undesirable effects, including incorrect evaluation of solution maturity, unhelpful or unactionable findings, and inaccurate assessment of risk.

This observation does not imply that either agile product development or traditional stage-gate reviews are wrong, but rather to point out that, as currently implemented, they are ineffective together, and to propose alternatives that preserve the intended value of both concepts.

At a conceptual level, the intent of gate reviews is to assess progress and determine if the program is able to proceed to their

Table 1. Agile values application to gate reviews

Agile Value	Review Application
Individuals and interactions over processes and tools	Group reviews of core topics supplemented by subject matter expert (SME) reviews
Working (capabilities) over comprehensive documentation	Prioritize review of working program artifacts including models over static documents
Customer collaboration over contract negotiation	Focus on how the contract allows the program to collaborate with the customer to create value
Responding to change over following a plan	Frequent light-weight touchpoints rather than rigid go-no go decisions

next set of planned activities. Gate reviews facilitate interaction between program personnel, stakeholders, and independent experts to identify risks and propose approaches for handling them.

The International Council on Systems Engineering *Systems Engineering Handbook* (INCOSE 2015) identifies the following outcomes of gate reviews:

- Ensure that the elaboration of the business and technical baselines are acceptable and will lead to satisfactory verification and validation (V&V)
- Ensure that the next step is achievable, and the risk of proceeding is acceptable
- Continue to foster buyer and seller teamwork
- Synchronize project activities.

From this generic perspective and detached from experience with a typical system requirements review, critical design review, or test readiness review, these outcomes are just as valuable to agile projects as they are to those using the waterfall lifecycle. Unfortunately, conflicts in assumptions and expectations between agile execution and traditional review approaches can limit the ability of the review to identify gaps and risks and decrease the value output of the execution team by focusing them on activities that do not support value delivery. The remainder of this article explores ways to effectively provide value added independent reviews to agile projects.

APPLYING AGILE VALUES TO GATE REVIEW INTENT

The manifesto for agile software development does not provide a comprehensive approach for expressing agility in systems engineering, but it does offer some key ideas for enabling agility (Beck et al. 2001). *Principles for Agile Development* proposes wording changes to the manifesto and its supporting principles to make their applicability to systems engineering clear. The value statements of the manifesto are shown to be applicable to systems overall as well as software, with a single adjustment of focusing on working capabilities rather than working software (Marbach, Rosser, and Osvalds 2015). Table 1 summarizes these values and their use in structuring gate reviews for agile projects.

Individuals and Interactions – the critical value of individuals and interactions to agile product development manifests in the use of small, cross-functional teams who plan and coordinate frequently in face-to-face group discussions. This approach can be applied to reviews by identifying a small group of reviewers who, as a group, have the skills and experience to provide valuable insight on the program's progress and recommendations for improving performance and reducing risk. This group of reviewers should review the program progress holistically as a team rather than taking a "divide and conquer" approach. Agile progress is measured against system capabilities, and the reviews should not sub-optimize around individual disciplines, domains, or activities.

Working (capabilities) – the agile focus on measuring progress by evaluating working capabilities can easily be extended to gate reviews by focusing on demonstration of what has been achieved, that is, evaluating those vertical threads of functionality that have been implemented from end to end, requirements through integration and test, rather than looking at a single part of the development work for all functionality whether it is working yet or not. This focus on working functionality can be extended to include the review of working program artifacts such as the backlog, system models, and automated test reports rather than static plans.

Customer collaboration – in typical government programs, there is a contract which binds the government and the contractor to perform tasks and provide outcomes. When the parties intend to employ agile methods, it's necessary to focus on how the contract enables continuous change, and reviews need to focus on how well these change processes are working to enable response to change within the framework of the contract.

Responding to change – because agile projects are designed to respond rapidly to frequently occurring change, there is no economy of scale in infrequent large batch reviews. Instead, reviews should be frequent, enabling small course corrections implemented at the speed of change. For this to be viable, reviews must be light-weight, minimally intrusive and focused on the work that is on deck to be done. Reviewers who remain with the program throughout its lifecycle and a focus on working artifacts, with little or no material prepared specifically for the review are recommended best practices for keeping the reviews lean and effective.

FRAMING OVERSIGHT FOR AGILE PROJECTS

To deliver valuable technical oversight to agile projects, reviews must be framed to coordinate and synchronize with agile product development in the same way that current stage-gate reviews align with waterfall development. In this section, we examine some key areas where re-framing can increase the effectiveness of reviews for agile projects.

Program Stages

Traditional reviews tend to perceive certain realization activities as being performed in phases that terminate with a review of the work for that stage, thus the concept of requirements reviews, design reviews, test reviews and so on. In agile engineering, these activities are not performed in this sort of discrete grouping, but rather in iterative or concurrent fashions, which makes the traditional gate review approach ineffective at assessing progress and adjusting course. However, there are detectable phases of activity on agile projects, which can inform the timing and focus of reviews.

Figure 1 summarizes the general phases of agile engineering development. These phases, or analogues thereof, appear in several agile frameworks and process flow descriptions including disciplined agile (Program Management Institute 2022), enterprise unified process (Enterprise Unified Process 2020), and the agile software development lifecycle (Eby 2016). Some works describe additional phases or sub-phases, but from the perspective of technical oversight, these three phases provide adequate granularity.

Inception is the phase that begins with contract award and ends when the program is ready to begin iterative capability

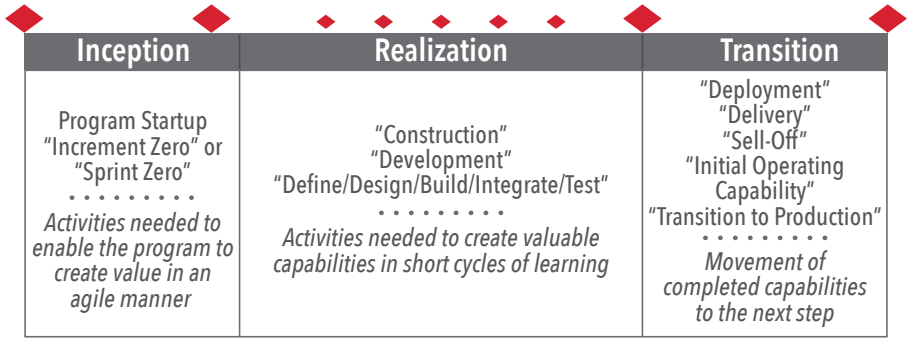


Figure 1. Agile phases and boundaries

development. This phase includes both traditional program startup activities and those engineering and technical activities required to make the program ready to commence agile development. *Systems Engineering for Software Intensive Projects Using Agile Methods* refers to these activities as "pre-planning" activities and describes them in some detail (Rosser et al. 2014 p. 6). From the perspective of review planning, the end of this period provides a good point to evaluate the program's readiness to set the agile realization process in motion. Plans, requirements, staffing, tooling, and artifacts can all be evaluated, not to determine if they are finished, but rather if they are adequate to start.

Realization is the phase in which capabilities are designed, implemented, and tested using agile methods. During this time, work is done in small batches, often repeated many times. Concurrent activities occur, and reprioritization of work and pivoting in direction is common. During this phase, large formal reviews are costly and ineffective and should be eliminated in favor of frequent, tightly scoped light weight course correction touch points. The focus of these touchpoints should primarily be capabilities just completed and those that are selected for completion in the upcoming iteration. Requirements, design, and test plans/results for the capabilities being reviewed are all considered.

Transition – at some point, agile capability development ceases, or at least pauses. In some cases, the contract may end, and the program shuts down. In other cases, an initial operating capability may have been achieved, or a first article completed and transition to operations or production required. This is another useful time for an extensive review, this time to determine if the deliverable items are ready for the pending transition. For government and defense contracts, transition reviews are a good point to evaluate against required standards such as technology readiness level (TRL) or manufacturing readiness level (MRL) or to assess the solutions readiness to request operational permissions such as authorization to operate (ATO). There may be multiple transition events in the context of a single program if multiple phases or effectivities have been contracted.

Review Focus

All reviews should focus on concerns that have high impact on the project or product being reviewed, and agile project reviews are no different in this regard. However, the specific elements that have high impact may not be the same as in reviews for other project types. The following paragraphs highlight some areas worth evaluating on agile projects.

Working capabilities are the primary yardstick of progress on agile projects. A critical part of an agile review should be focused on what's working. Newly working capabilities should be assessed for fulfillment of customer needs, previously implemented elements should be checked to ensure that they are still working, still meet needs and are still aligned with overall program direction. Progress to date can also be evaluated compared to the overall mission requirement and remaining time and budget.

Integration and synchronization are critical to success on agile projects. Discipline in continually identifying and managing dependencies at all levels is critical, as there is typically not a lengthy "get well" integration period planned near the end of the program.

Health of baselines must be evaluated regularly. The possibility of multiple significant changes occurring close together means that baselines must be up to date, working, and configuration controlled at all times. Deficiencies such as technical debt and performance shortfalls must be tracked and kept visible due to their impact on decisions and prioritization of work.

Working execution assets such as backlog, models, regression test results, and demonstration outcomes provide excellent insight into program progress, process, and risk without requiring program personnel to construct presentations.

Best practices for whatever agile program execution model that is employed should be a part of any agile review. This is particularly important since most of these practices are fairly new and potentially unfamiliar to team members or at odds with established expectations or adjacent processes.

Review Content

The content of agile reviews should include artifacts that contribute to or indicate the health of program execution. Some items to evaluate are described below.

Working plans – This may include the master phasing schedule and integrated master schedule but should also include any agile planning artifacts such as backlogs and roadmaps.

Integrated test plan and results – most agile practices encourage early and frequent integration and test, and both the approach and results are valuable objects of review.

Traceability is exceptionally important on agile projects where changes occur frequently and anywhere in the chain from customer requirements to operational systems. Bi-directional traceability from end to the end of realization should be evaluated.

Tools and automation is a force multiplier for agile projects. Effective integrated toolsets can increase both quality and productivity when properly configured and employed.

Baseline maturation and quality – as the central value output of agile product realization, baselines should be monitored for current state of quality, maturity, and performance as well as trends that may need attention.

Continuous improvement based on short cycles of learning is an intrinsic advantage of agile development, but only when improvement opportunities are identified, and improvements made.

Review Approach

The general approach to agile reviews should reflect agile values and principles. Reviews should be cross functional and collaborative, not stove-piped. The focus should be on delivery of value not completion of intermediate artifacts. They should be collaborative, not confrontational, and seek to course correct without limiting velocity, with goals of continuous improvement rather than striving for immediate perfection.

Review Outcomes

The outcomes of an agile review should align with the principles and characteristics of agile project realization. Some desirable outcomes are described below.

Enhanced value delivery – agile projects measure success in valuable capabilities

delivered to the customer. Reviews should strive to make actionable recommendations that enhance value delivery in some way. This can include increased quality, faster delivery, lower cost, decreased risk, or better fulfillment of mission needs.

Predictable performance – given that agile projects encourage ongoing modifications to the solution intent based on changes in the mission, technical and business environment, it's critical that the team performance can be reliably predicted. This enables effective replanning and reprioritization. Reviews strive to make actionable recommendations that enhance performance transparency through metrics, retrospectives, management of dependencies, and risks.

Ongoing improvements – successful agile projects leverage the short cycles of learning inherent in the agile practice to continually improve in a wide range of areas throughout their period of performance. Reviews strive to make actionable recommendations for improvement. Prioritize those areas where the program may be struggling, but don't ignore those areas that are "good enough.

Continuous learning for all – for programs, review teams, and organizations "the best we can do" last time may not be the best we can do this time. Reviews strive to make actionable recommendations for improvements not only to the program under review, but to the review team, the review process, and the organization overall.

Summary

The key to agile reviews is to review as you develop — agilely. Encourage inline quality improvements rather than waiting for outsiders to initiate corrections. Focus on valuable outcomes, fast feedback, and continuous learning.

CHALLENGES AND OPPORTUNITIES

Implementing an agile approach to reviews is not without its challenges. Existing norms and processes may need to change to admit new ways of working. Standards and processes that assume sequential execution of engineering activities need to be refactored to address end to end completion of small units of work. The perception that reviews are "grades" and the program's goal is to "pass" must evolve to see reviews as checkpoints that identify opportunities to improve. Criticism needs to give way to collaboration, where all participants are focused on enhancing value delivery. Cultural habits of covering our backsides or hiding our challenges and shortfalls needs to transform into eagerness for inquiry and improvement. And we must make these changes in an environment of constantly accelerating change. However daunting that sounds, it's the best option before us. If the world is changing rapidly, we need agile practices, including agile oversight, to respond effectively to that change.

Fortunately, there are nuggets in new practices and changing technologies that promise to help us provide effective oversight that keeps pace with change. Integrated digital engineering and the use of models at all levels of the system of interest provide direct insight into the current state of baseline capability and maturity. Increasing automation offers several opportunities. Automated checks of concerns like traceability and test coverage can reduce human effort in reviews as they also enhance quality. The use of end-to-end DevSecOps pipelines offers the possibly of in-line zero preparation reviews, in which data collection and presentation become an ongoing automated function of the pipeline and meetings can focus on discussing concerns and ways to overcome them. The data and information gathered from these reviews can be fed back to every level of our organization to ensure that we continue to improve on pace with the challenges we face. ■

REFERENCES

- Beck, K., et al. 2001. "Manifesto for Agile Software Development." viewed 21 November 2021. https://agilemanifesto.org/.
- Defense Science Board. 2018. "Design and Acquisition of Software for Defense Systems." 14 February.
- Dove, R., W. Schindel, and K. Garlington. 2018. "Case Study: Agile Systems Engineering at Lockheed Martin Aeronautics Integrated Fighter Group." Proceedings International Symposium, International Council on Systems Engineering, Washington, US-DC, 7-12 July.
- Dove, R., and W. Schindel. 2017. "Case Study: Agile SE Process for Centralized SoS Sustainment at Northrop Grumman." Proceedings International Symposium. International Council on Systems Engineering, Adelaide, AU, 15-20 July.
- Dove, R., W. Schindel, and R. Hartley. 2017. "Case Study: Agile Hardware/Firmware/Software Product Line Engineering at Rockwell Collins." Proceedings 11th Annual IEEE International Systems Conference, Montréal, Québec, CA, 24-27 April.
- Eby, K. 2016. "Understanding the Agile Software Development Lifecycle and Process Workflow." *SmartSheet*. Viewed 4 March 2022. https://www.smartsheet.com/understanding-agile-software-development-lifecycle-and-process-workflow.
- Enterprise Unified Process c. 2020. "Life Cycle Phases." Viewed 4 March 2022. http://www.enterpriseunifiedprocess.com/essays/phases.html.
- IEEE Computer Society. 2014. ISO/IEC/IEEE 15288.2, IEEE Standard for Technical Reviews and Audits on Defense Programs.
- INCOSE. 2015. *Systems Engineering Handbook: A Guide for System Life Cycle Process and Activities* (4th ed.). D. D. Walden, G. J. Roedler, K. J. Forsberg, R. D. Hamelin, and T. M. Shortell (Eds.). San Diego, US-CA: International Council on Systems Engineering. Published by John Wiley & Sons, Inc.
- Lapham, M. et al. 2016. RFP Patterns and Techniques for Successful Agile Contracting. Carnegie Mellon University, Software Engineering Institute Special Report, November.
- Marbach, P., L. Rosser, and G. Osvalds. 2015. "Principles for Agile Development." Proceedings International Symposium. International Council on Systems Engineering, Virtual Event, 17-22 July.
- National Defense Industrial Association. 2019. *An Industry Practice Guide for Agile on Earned Value Management Programs*.
- Office of Management and Budget. 2012. "Contracting Guidance to Support Modular IT Development." Viewed 20 March 2022.https://obamawhitehouse.archives.gov/sites/default/files/omb/procurement/guidance/modular-approaches-for-information-technology.pdf.
- Office of the Under Secretary of Defense for Acquisition, Technology, and Logistics. 2017. *Best Practices for Using Systems Engineering Standards (ISO/IEC/IEEE 15288, IEEE 15288.1, and IEEE 15288.2) on Contracts for Department of Defense Acquisition Programs*.
- Program Management Institute c. 2022. "Full Delivery Life Cycles." viewed 3 March 2022. https://www.pmi.org/disciplined-agile/process/introduction-to-dad/full-delivery-lifecycles-introduction.
- Rosser, L., G. Osvalds, P. Marbach, and D. Lempia. 2014. "Systems Engineering for Software Intensive Projects Using Agile Methods." Proceedings International Symposium. International Council on Systems Engineering, Las Vegas US-NV,30 June – 3 July.

> continued on page 65

Project Lifecycle Development for a Next Generation Space Suit Project

Michael A. Cabrera, michael.a.cabrera@nasa.gov; and **Steve Simske,** steve.simske@colostate.edu

ABSTRACT
A hypothesis for an optimized, project lifecycle development method was formulated by understanding (i) the project environment of implementation, (ii) applicable, current state-of-the-art frameworks, and (iii) eliciting feedback before, during and after testing from those individuals participating in the lifecycle development framework. While traditional waterfall methods have their place, high uncertainty projects instigate exploratory work and as such, agile implementations were created to allow projects to quickly adapt (PMI 2017). In the context of dynamic environments and niche products, NASA is no stranger. Understanding the current state of the project and current state-of-the-art facilitates an approach that allows for well-established techniques in the way of lean and agile to benefit project development. Additionally, these inclusions may help expose knowledge gaps in the current state-of-the-art and also lend credibility to approaches derived to help close those gaps. This article describes the modified agile concept (MAC) and its multi-disciplinary approach to a sampling of various lean and agile methods integrated alongside traditional, waterfall methods (such as a hybrid model) to support the hypothesized project lifecycle development. This approach was developed as part of a case study with a design and test team responsible for building test stations to qualify components of the life support system on the next generation space suit. This article will outline exclusively the scrum and lean methods in the MAC with a cursory overview on kanban development supporting the MAC.

INTRODUCTION

In 2019, the presidential administration proposed a return to the moon by 2024, allocating additional funding and a condensed schedule specifically for the next generation spaces suit (Simon 2020). Two years later, NASA's Office of the Inspector General (OIG) released an audit indicating that current forecasting projected that a flight-ready suit was years away from completion and that the government would spend over $1 billion dollars on design, testing, qualification, and development efforts. As a result, the next generation space suit, the Exploration Extravehicular Activity Mobility Unit (xEMU) project, would end and the portfolio of work would be transferred to contractor-developed suits instead of building the xEMU qualification and flight suits "in-house" (Martin 2021). As part of a case study on the systems engineering challenges associated with xEMU development, one of the specific areas examined was the history and development of the ground support equipment (GSE) used to qualify flight components on the exploration portable life support system (xPLSS). These GSE test stations require a similar pedigree as their space suit flight components and as such, demand a strict standard against quality and safety. In a similar respect to neighborin g projects across the xEMU suite of projects, xPLSS GSE development too had struggled with respect to cost and schedule expectations. One of the goals of the case study on GSE development was to understand the current state of the project by meeting with team members to understand the current sentiment on project lifecycle development, review project documents to understand root causes of cost and schedule challenges, review the current state-of-the-art, and identify applicable project lifecycle development approaches. The goal is to ultimately determine a development method by which the GSE test station team may be able to optimize lifecycle and team development so as to more effectively approximate cost and schedule expectations.

The current hypothesis regarding the formulation of the proposed project

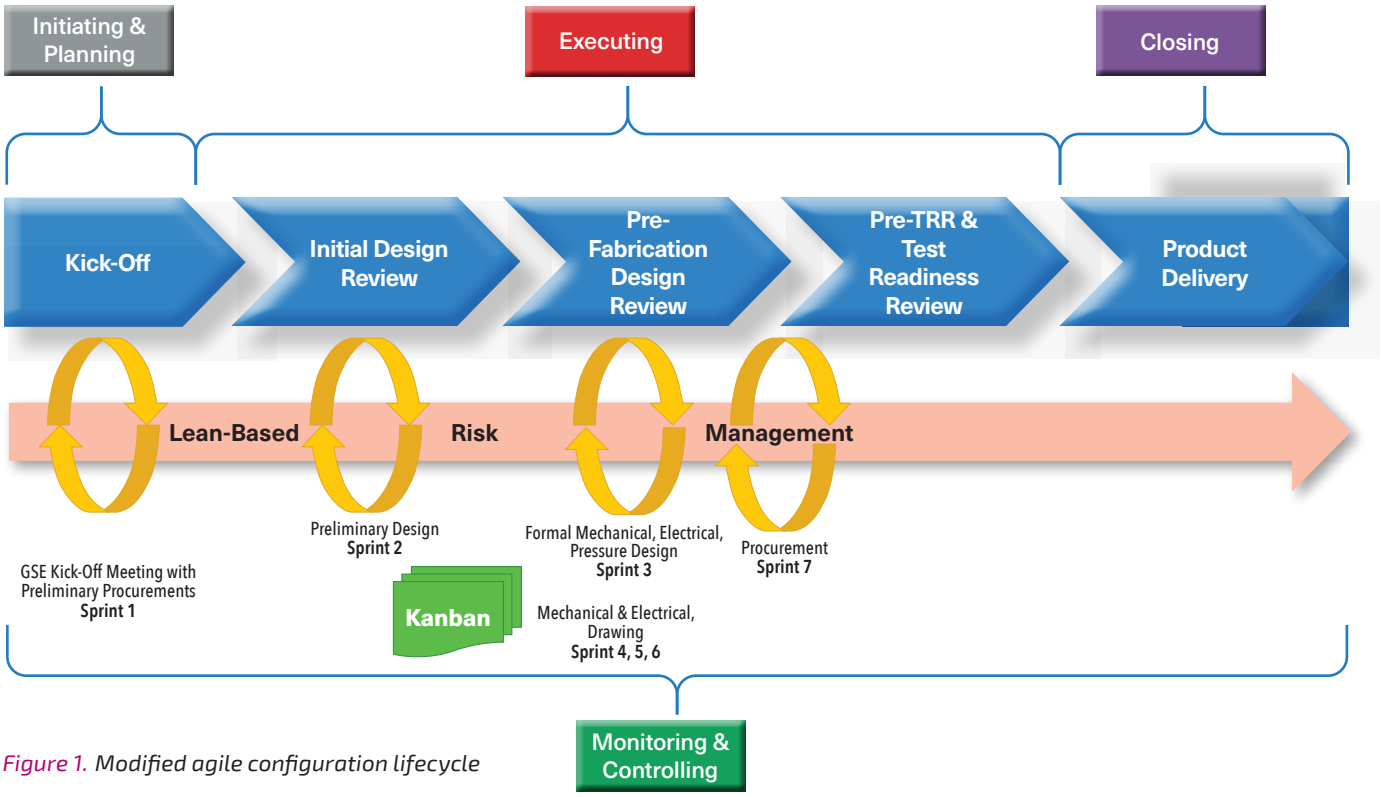


Figure 1. Modified agile configuration lifecycle

lifecycle development asks, "will a modified, agile-hybrid project lifecycle development model applied to waterfall teams develop a superior product within time and schedule constraints in a hardware-intensive environment?" Analysis to test the hypothesis includes iterative and incremental project lifecycle model tempering where focus groups of subject matter experts evaluate the additions of agile and lean-based lifecycles in a project lifecycle model until the tempered model across three iterations is satisfactorily accepted by panel approval. Likert scale, panel scoring, and failure modes and effects analyses (FMEA) were the quantitative measures used as success criteria measurements as resources were not available to effectively run a project in its entirety as a function of the MAC for vetting against the project's current lifecycle model. Results from the research currently indicate that schedule time is reduced if specific areas of scrum, lean and kanban are implemented, effectively reducing budget while scope is preserved. It must be noted that while multiple tempered model tests across multiple subject matter experts has approximated a solution that is specific for the GSE team's needs, by no means is this a definitive or fully optimized model. With regards to the GSE team's specific needs and within the context of the work performed, tempered model testing helped disqualify certain approaches or facets of approaches while preserving those that testing deemed satisfactory for implementation. The initial hypothesis was verbatim with the exception that the question was phrased for scrum alone in comparison to agile-hybrid as the current hypothesis prompts. This change was a result of vetting various alternatives to approximate an answer and taking the approach of discovering data to disqualify the prevailing hypothesis to further approximate the most correct project lifecycle approach.

It is important to note that is strictly a hypothesized approach to project lifecycle development. The ideas and approaches discussed were not fully exercised in practice in part due to the nature of the type of work, funding, and resources required. This should be considered as a postmortem of a project that by Likert scaling polling had requested the consideration of a hybrid lifecycle approach to GSE test station development. The metrics, project documents, and existing project personnel aided in supporting the testing via focus group paneling through three iteration testing of an approach and not a usage of the hypothesized approach of a project development of a test station. This hybrid took the form of different developments, including waterfall, agile, lean, extreme programming (XP), kanban, and feature driven development (FDD).

A HYBRID PROJECT LIFECYCLE

The MAC project lifecycle is a hybrid of waterfall with lean and agile philosophies added and was organized specifically to address the needs of xEMU's xPLSS GSE projects. While the organization of the MAC is specific to GSE test station development, the MAC approach provides any engineering team with a user's manual to allow for transformation of said team from a pure waterfall development into a platform combining various methods into a hybrid framework. The following hybrid methods are augmented and included in the waterfall development skeleton of the project framework, which preserves all facets of the project that are not as receptive to agile or lean methodologies, which are best reserved for operations or externalities that are needed by but not controlled by project (that is, calibration, cleaning, and fabrication of hardware).

The following methods are a comprehensive list which include facets of prototyping, incremental modeling, and spiral modeling within the context of scrum, lean, and kanban frameworks:

- Scrum (iterative and incremental delivery project model)
  - supports for schedule velocity modifications, iterative, and incremental deliveries of hardware, rapid prototyping, and an empowered team development.
  - supplemental and interrelated to scrum are XP and FDD which assist in schedule velocity, and pair programming to cross-train cross-functional engineering groups, help assist in story pointing, and break

Table 1. Projected task effort

				Week 1					Week 2					Week 3					
Work Remaining	**Task Projected Effort** (Burn Down Rate)	Points	Day 0	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Day 9	Day 10	Day 11	Day 12	Day 13	Day 14	Day 15	Day 16
	Meet with Component Owner and Review Rig Requirements	128	128	120	112	104	96	88	80	72	64	56	48	40	32	24	16	8	0
	Meet with Resource Managers and Receive ROMs	128	128	120	112	104	96	88	80	72	64	56	48	40	32	24	16	8	0
	Create Rig Cost Estimate	128	128	120	112	104	96	88	80	72	64	56	48	40	32	24	16	8	0
	Create Completion Form PD	128	128	120	112	104	96	88	80	72	64	56	48	40	32	24	16	8	0
	Create Rig Schedule	128	128	120	112	104	96	88	80	72	64	56	48	40	32	24	16	8	0
	Define Rig Scope	128	128	120	112	104	96	88	80	72	64	56	48	40	32	24	16	8	0
	Create Mechanical P&ID	128	128	120	112	104	96	88	80	72	64	56	48	40	32	24	16	8	0
	Create Electrical Block Diagram	128	128	120	112	104	96	88	80	72	64	56	48	40	32	24	16	8	0
	Create Rough CAD Model	128	128	120	112	104	96	88	80	72	64	56	48	40	32	24	16	8	0
	Create Powerpoint presentation	128	128	120	112	104	96	88	80	72	64	56	48	40	32	24	16	8	0

Table 2. Actual task effort

				Week 1					Week 2					Week 3					
Work Remaining	**Task Actual Effort**	Points	Day 0	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Day 9	Day 10	Day 11	Day 12	Day 13	Day 14	Day 15	Day 16
	Meet with Component Owner and Review Rig Requirements	128	128	120	112	104	96	88	80	72	64	56	48	40	32	24	16	8	0
	Meet with Resource Managers and Receive ROMs	128	128	121	111	101	99	92	87	81	75	65	55	45	42	34	24	9	0
	Create Rig Cost Estimate	128	128	120	112	104	79	74	69	59	49	39	29	19	9	0	0	0	0
	Create Completion Form PD	128	128	108	106	104	99	89	79	69	64	54	44	34	29	19	9	0	0
	Create Rig Schedule	128	128	120	112	104	96	88	80	72	64	56	48	40	32	24	16	8	0
	Define Rig Scope	128	128	127	117	107	99	91	83	75	74	64	54	44	34	24	19	9	0
	Create Mechanical P&ID	128	128	126	116	106	96	86	76	71	66	61	56	46	36	26	16	6	0
	Create Electrical Block Diagram	128	128	123	118	113	108	98	88	85	82	67	52	37	22	7	0	0	0
	Create Rough CAD Model	128	128	126	116	106	96	91	81	76	71	68	63	48	38	28	18	8	0
	Create Powerpoint presentation	128	128	127	126	125	124	123	122	121	120	119	84	34	0	0	0	0	0

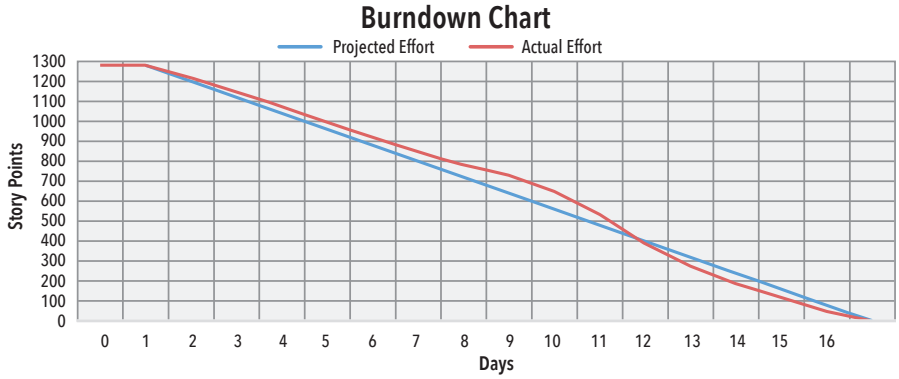


Figure 2. Burndown chart of projected vs. actual effort on story points

down work into smaller packages.

- Lean (waste identification project model)
  - supports the identification and quantification of the seven forms of lean waste with risk management used to illustrate how trading minimal increases to risk with regards to quality and safety allow for significant decreases to schedule and cost while preserving scope.
- Kanban (visualization project model)
  - supports for value map streaming and visualization of all project artifacts to ensure process control and proper work in progress limits.

SCRUM APPROACH FOR HARDWARE

Scrum finds success in software-intensive teams but has limitations for full acceptance in the context of hardware-intensive development due to constraints of physicality (Schmidt et al. 2017). However, there are areas where scrum has its place in hardware development projects with regards to prototyping and schedule estimations (Peterson et al. 2021). Due to the nature of the work performed in the case study, it is advisable that schedule re-estimations be performed as a part of wave rolling planning when the project work does not have sufficient historical data or is expected to change (Briatore et al. 2021). Using an inspired variation of earned value metrics (EVMs) in tandem with story pointing techniques from FDD allow for re-estimations against work packages when respective packages are comparable.

For context, the GSE team will create story points for projected task effort and once the work is completed will record the actual effort. A reconciliation between the projected vs. actual effort is used to inform future sprints. This effort should be done during the sprint retrospectives as a function of the "what could we have done differently?" question prompted at the conclusion of a sprint. The table provided shows a sample sprint from one of the earlier activities on the project, which uses an eight-hour effort during each day to help complete the work package(s).

The first step is to organize the task effort by assigning story points in a manner in which a burn rate (that is, a periodic measurement of task velocity to complete story points) can be organized. Typically, this burn rate should be constant and follow, when possible, a linear progression.

Table 3. Earned value metric augmentation on story points

				Running Average New Point Value
SMI	**Point Value**	**Itemized Point Value**	**Group**	***Point Value Divided By SMI***
1.07	128	Meet with Component Owner and Review Rig Requirements	Project Management	120
1.07	128	Meet with Resource Managers & Receive ROMs	Project Management	120
1.33	128	Create Rig Cost Estimate	Project Management	96
1.14	128	Create Contract	Project Management	112
1.07	128	Create Rig Schedule	Project Management	120
1.07	128	Define Rig Scope	Project Management	120
1.07	128	Create Mechanical P&ID	Mechanical	120
1.23	128	Create Electrical Block Diagram	Electrical	104
1.07	128	Create Rough CAD Model	Mechanical	120
1.45	128	Create PowerPoint presentation	Mechanical, Electrical	88

The second step is to record the actual effort to demonstrate the reality of the effort that is performed. The third step is to reconcile the differences in an effort to re-estimate task efforts for future scheduling. This is done by creating a burndown chart (that is, a graphical depiction of projected vs. actual schedule velocity against story point completion over time) to characterize projected effort vs. actual effort and a recalculation of equivalent story points moving forward by dividing the original story point (that is, an analogous planned work EVM for task effort estimation) by the story point modifier (SMI) (that is, an analogous schedule performance index (SPI) from the EVM).

In terms of scrum implementation, a schema was developed and a sample sprint shown which pairs with the EVM-inspired metric and projected vs. task effort story pointing. In addition to the story point and schedule re-estimation efforts, prototyping and the idea of a "moving prototype" that follows the build of the GSE hardware, allows for proof of concept and early troubleshooting when traveling with the actual product parallel in development. Prototypes take the following forms and for the MAC, prototypes include both physical and conceptual models:

- Physical: Scrum is limited in its ability to fully integrate with hardware projects as opposed to the tested and proven software-intensive projects. With certain hardware, in particular GSE hardware, physical breadboards can be made and modified over time. The majority of test stations in the case study are electrical harnessing, tubing, structure, fasteners, valves, sensors, instrumentation, a graphical user interface with LabVIEW®, vacuum chamber, vacuum system, and interfaces with facility and test articles. In most cases, the majority of existing hardware onsite (that is, NASA Johnson Space Center) may be used for these breadboarding efforts. During fabrication, mechanical and electrical issues may be troubleshot early by keeping a physical prototype to include the following:
  - A mechanical loop with all instruments to simulate the environment in which the test article will reside. This model may be updated periodically as continued purchased parts are added to the build and verifying functionality as the design progresses.
  - An electrical loop with all instruments to simulate the environment in which the test article will reside. This model may be updated periodically as purchased parts are added to the build and verifying functionality as the design progresses.
- Conceptual: Hardware projects may greatly benefit from a conceptual prototype as much of the time, these efforts are inherently built into the project structure. These conceptual prototypes may include:
  - Test station mechanical piping and instrumentation diagrams (P&ID).
  - Creo® mechanical computer-aided design (CAD) models.
  - Visio® electrical schematics.
  - LabVIEW® pseudo-code software schemas.
- Software: GSE test stations are run with LabVIEW® software and as such may follow a prototypical scrum lifecycle. The architecture is developed alongside the purchased hardware and much like typical software scrums, each iteration will improve upon the existing build.

GETTING LEAN

The lean philosophy emphasizes cutting waste and inefficiencies (Kupiainen et al. 2015) and identifies seven forms of waste. These have been established primarily for manufacturing processes but have also found their way in software development. In the context of hardware development, outlined are the seven forms of waste for software intensive projects (Griffiths 2012), which are analogous to GSE development of hardware.

1. Defects
   a) Defects are items delivered to the customer that do not fulfill scope with regards to hardware and/or documentation. These could also be bugs associated with software builds for the test stations.
2. Hand-offs
   a) This is considered the effort to facilitate motion to communicate information from one group to another. Examples include if teams are not co-located and the loss of productivity associated or the loss

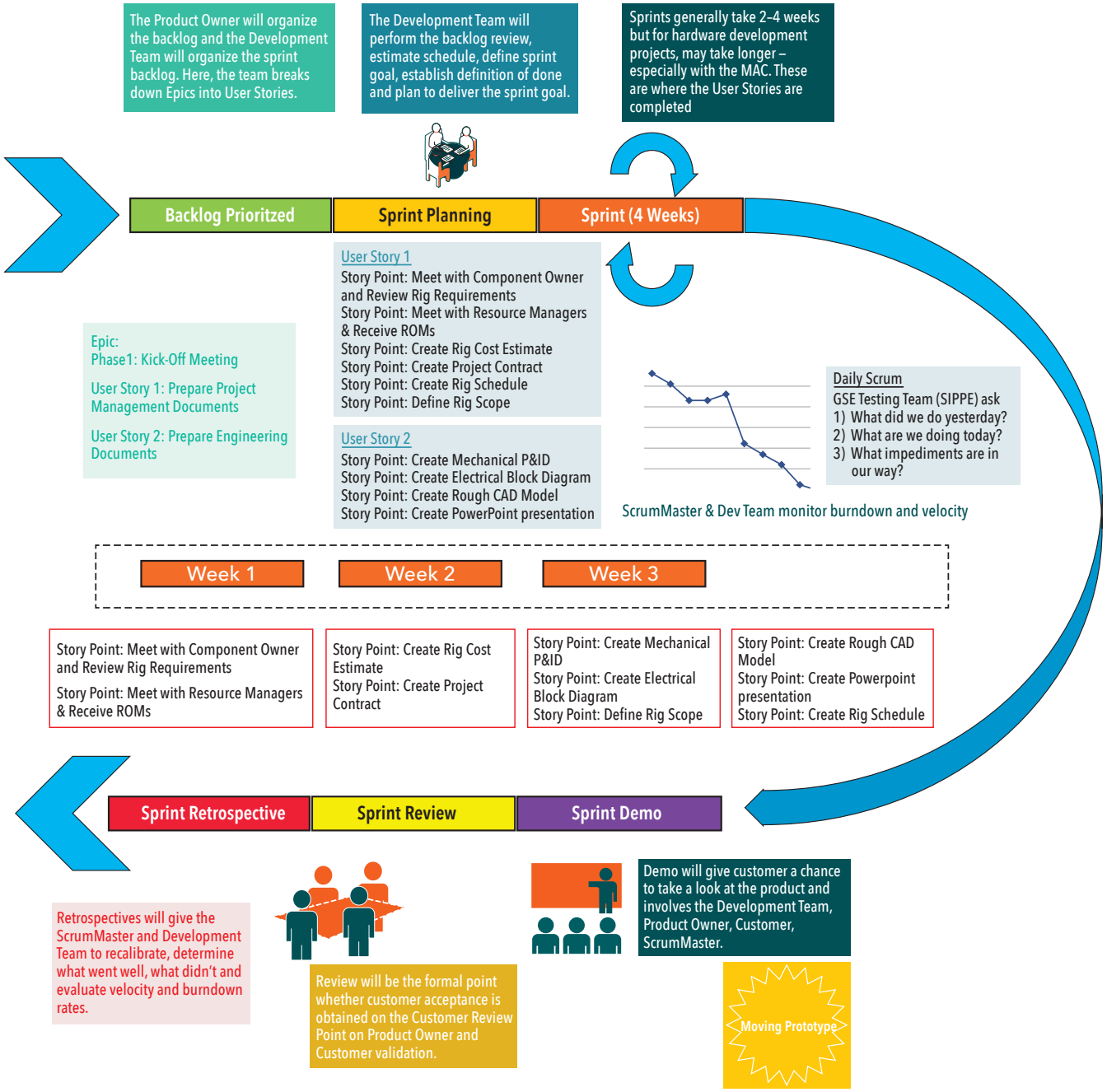


Figure 3. Typical scrum-style sprint

of productivity during the transit of work.

3. Waiting/Delays
   a) These are delays associated with approvals and reviews. These could be approval signatures on documents or drawings.
4. Task Switching
   a) This is the multi-tasking between several different projects. Lean experts converge on as much as 40% degradation in productivity during task switching (Cherry 2012). This could include resources working multiple tasks for multiple projects.
5. Extra Processing of Extra Documentation
   a) This includes additional work that does not provide value to the customer. These include unused documents or unnecessary approvals for several deliverable types such as drawings, fabrication documents, procedures, etc.
6. Unneccessary Features
   a) These are extra pieces of functionality which, while nice to have, are not entirely necessary. These can include gold-plated items that either are not necessary or were requested by the customer that were not formally approved.
7. Incomplete/Partial Work
   a) Partial work completed introduces entropy into the systems engineering process and does not deliver value. This includes drawings started that were never finished due to descoping, documents that were created but no longer needed, etc.

The first step is to identify the areas across the project lifecycle development where lean can be implemented. The reasons may be as follows and were particularly effective for the GSE team and may be one or any combination of the following:

- internal processes that were developed to preserve quality or add value that

Table 4. Prompt list example

ID#	Area of Concern	Project Management Area(s) Affected	Lean Waste Catagory	Additional Information	Average Wait Time	Potential Wait Time	Potential Corrective Action?
4	Purchasing & Quality Assurance	Schedule, Cost	Extra Processing, Unnecessary Features	The GSE team has had challenges with delivery on certain items sent to the purchasing department with contractually imposed quality codes to GSE hardware.	X	0.5X	By reducing the quality codes on certain GSE procurements that increase quality and safety to a marginal yet acceptable level, lead times can be diminished. These include the lead times to find a vendor that can provide certain documents to satisfy quality codes and also the lead time dedicated to the vendor providing said codes.

Table 5. Consequence ranking table

CONSEQUENCE RANKING					
Category	**1**	**2**	**3**	**4**	**5**
Quality	Remote loss of quality	Minimal loss of quality	1 standard deviation away from quality standard	2 standard deviations away from quality standard	3 standard deviations away from quality standard
Safety	Remote risk of injury	Minimal risk of injury	Minor injury	Severe injury	Loss of life
Cost	< $50K impact	$50k to $100K impact	$100K to $250K impact	$250K to $500k impact	> $500K impact
Scope	Remote impact to scope objectives	Minimal impact to scope objectives	Considerable impact to scope objectives	Major impact to scope objectives	Severe impact to scope objectives
Schedule	Major disruption of service not involving client interaction and resulting in either associate re-work or inconvenience to clients	1 to 2 month impact	3 to 4 month impact	5 to 6 month impact	> 7 month impact to schedule

Table 6. Likelihood ranking table

LIKELIHOOD RANKING		
Score	**Description**	**Probability Range**
1	Very Unlikely	< 10 %
2	Unlikely	10% to 30%
3	Possible	> 30% to 60%
4	Likely	> 60% to 90%
5	Very Likely	> 90 %

were deemed to either increase cost or schedule and do not deliver intended quality assurance or control, which may include performing organization processes on drawing and drafting review, purchasing of hardware, and signature approval review timeframes for documentation.

- external processes that are needed preserve quality or add value that while are deemed valuable may have processes or standard operations that may introduce unnecessary cost or schedule delays, which may include contractual agreements on purchasing and quality assurance or customer suggested processes that may not delivered the anticipated value.
- waiting periods for external review such

as signature review cycles, drafting engineering discipline group review, technical process specification (TPS), and discrepancy report (DR) review and signature.

The team records these areas in a prompt list, risk breakdown structure or include directly in a FMEA spreadsheet. The provided template is a sample. The next step includes determining a method by which to qualify and/or quantify metrics for areas classified as lean waste. For the majority of lean waste for the GSE team, the primary recording method was days. Days could correlate to either schedule, cost, or both. Many of these metrics could be derived from project documents or repositories.

Mean wait times across the project were found from historical records while potential wait times were determined either by the Delphi technique, polling, or by estimation methods including triangular or beta distributions. The GSE team was able to gather metrics for cleaning, calibration, drafting, procurements, document signatures (TPSs/DRs/procedures) strictly from repositories. Recommendations on time improvements (elimination of paperwork, reduction in calibration cycles, reduction in signature cycle) was performed by inspection, Delphi method, and beta or triangular distribution estimation.

A sample, prompt list line item is given. While several parameters are given notionally, average and potential wait times are unlisted as the data regarding those is project sensitive. For this sample, only one form of waste is given for one category while the GSE team found several areas of potential lean wastes.

Once the lean waste and metrics are populated in the provided templates, those selected forms of waste will be populated to the FMEA. The tool works twofold both as a risk management tool and a FMEA. The tool contains the following categories for the first step of the process:

- Risk number
  - What is the risk number associated with the process?
- Name
  - What is the name of the threat/opportunity associated with the process?
- Identification numer
  - What is the associated identification number of the threat/opportunity?
- Description
  - What the risk associated with the process?
- Itemized from description
  - How would these threats/opportunities decompose from the parent risk listed in the description?
    - Instead of a traditional failure mode, next level effects and end effects to the cascading effects are consolidated into one category for simplicity while still preserving and effectively illustrating the process.
- Impact areas
  - Safety? Schedule? Quality? Cost? Scope?
- Threat or opportunity distinction
  - One of the hallmarks of this augmented FMEA which differs from traditional approaches is that it is modified to work inversely when compared to a typical FMEA. For example, one of the central purposes of a traditional FMEA is to reduce risk, which this FMEA functions as by identifying a risk with an associated likelihood and a consequence at the onset. An updated likelihood and consequence evaluation are calculated after actions to correct the current project posture are proposed. In addition, the tool also functions as a means to understand if an opportunity that can improve schedule, budget, or scope is sensitive to fluctuations in reduced quality or higher risks to safety. If the post likelihood and consequence are within an acceptable limit (that is, in the green zone of the likelihood and consequence matrix), it could be deemed and a viable option to exploit the opportunity while safety and quality are still at acceptable levels.

After population of the preliminary information is completed, an assessment the following categories allows for a risk posture to be established.

- Consequence: How severe is the impact should the risk manifest?
- Likelihood: What is the probability of this risk manifesting?

Templates for each of the consequence and likelihood categories are given and are tailorable for the user. In many cases, the categories are presented with a general, non-numerical value so that the user may modify them to suit their needs.

At the conclusion of the consequence and likelihood assignment will be the population of the risk priority number (RPN). This RPN is given twice: once before analysis of alternatives and recommendations and once after analysis of alternatives or recommendations. The range is a number between 1 and 25. The RPN is a product of the two risk categories.

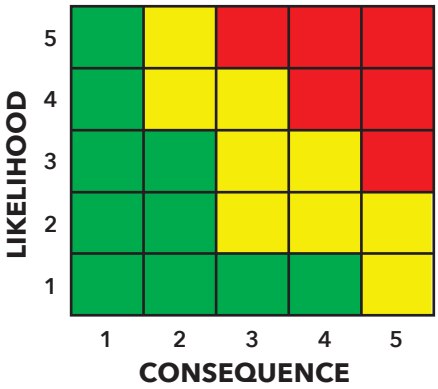


Figure 4. Likelihood vs. consequence matrix

After identification of the primary categories of potential issues and effects and assignment of a RPN, the next steps will be to identify what risk mitigation efforts, if any, should be implemented:

- Action recommended: What are the possible actions to remedy the requirement?
- Responsible party: Who is responsible for making sure the actions are completed?
- Actions taken: Will the Action Recommended be taken with respect to RPN?

If the user implements corrective actions in the form of alternatives or recommendations from the previous step, the user will update of the RPN with the intention of reducing the risk posture. As indicated previously, the RPN is given twice: once before analysis of alternatives and recommendations and once after analysis of alternatives or recommendations. Current limitations to this approach across the three iterations of model tempering include finite number of subject matter expert participants, finite risks specific to GSE development, and iterations limited to project resource allowance dedicated to the experiment. While sensitivity across each expert per each risk was a limitation of the study, the approach of an FMEA to gather data from individuals of varying disciplines allow for normalization and convergence of an estimate that is merited. The sample template of the FMEA is shown below for Item #4 from the prompt list alongside with a visual matrix representation of reduced or increased threat and opportunity, respectfully.

Once a posture on a threat or opportunity is quantified, the data can be used to inform the following but not limited to:

- inform upper management on processes thay may result in extended schedule and

Table 7. FMEA for lean waste management

Risk #	Name	Description	ID #	Risk/Opportunity	Itemized from Description	Impact Areas?	CONSEQUENCE (1–5)	LIKELIHOOD (1–5)	RPN (1–25) (BEFORE MANAGES)	Action Recommended	Responsible Party	Actions Taken	CONSEQUENCE (1–5)	LIKELIHOOD (1–5)	RPN (1–25) (AFTER MANAGES)
What is the risk/opportunity number associated with the process?	What is the name of the risk/opportunity associated with the process?	What is the risk associated with the process?	What is the identification?	Is this a risk or opportunity for the project?	How would these risks/opportunities decompose from the parent risk listed in the description?	Safety? Schedule? Quality? Cost? Scope?				What are the possible actions to remedy the potential risk or exploit opportunity?	Who is responsible for making sure the actions are completed?	Will the Action Recommended be taken with respect to RPN?			
4	GSE Non-Critical Hardware Procurement Purchased as Strict GSE	Procurements for GSE hardware are prolonged in terms of schedule and more expensive when compared to their Class III or Class I-E counterparts if they are bought strictly as GSE. Column D assumes if purchased as strict GSE, Column I assumes bought as Class III, Non-Critical GSE or Upgraded.	4A	RISK	4A) RISK: Extended time in finding vendors: With GSE, purchasing will spend more time securing a vendor that can provide the necessary paperwork (i.e., certifications, traceability, etc.)	Schedule	5	3	15	Procure as GSE Non-Critical or Class III and Upgrade	Project Manager, Design Lead.	Yes	2	3	6
			4B	RISK	4B) RISK: Difficulty obtaining quality codes (Q-code): Even if the vendor is secured, the Q-codes (quality codes) historically are not guaranteed to be met. This delays schedule.	Schedule	5	4	20	Procure as GSE Non-Critical or Class III and Upgrade	Project Manager, Design Lead.	Yes	1	2	2
			4C	RISK	4C) RISK: Higher costs in procuring as GSE: This is twofold (1/2). Even if the vendor is secured, the Q-codes historically are not guaranteed to be met. This increases costs to find new vendors.	Cost	3	5	15	Procure as GSE Non-Critical or Class III and Upgrade	Project Manager, Design Lead.	Yes	2	3	6
			4D	RISK	4D) RISK: Higher costs in procuring as GSE: This is twofold (2/2). Producing Q-codes increases costs from the vendor.	Cost	3	5	15	Procure as GSE Non-Critical or Class III and Upgrade	Project Manager, Design Lead.	Yes	2	3	6
			4F	OPPORTUNITY	4E) OPPORTUNITY: While not all items need to be procured with the same pedigree, (i.e., Q-codes, certificates, traceability), many do not. As such, quality can still be maintained and still deliver a sound product.	Quality	1	1	1	Procure as GSE Non-Critical or Class III and Upgrade	Project Manager, Design Lead.	Yes	2	2	4
			4F	OPPORTUNITY	4F) OPPORTUNITY: While not all items need to be procured with the same pedigree, (i.e., Q-codes, certificates, traceability), many do not. As such, safety can still be maintained and still deliver a sound product.	Safety	1	1	1	Procure as GSE Non-Critical or Class III and Upgrade	Project Manager, Design Lead.	Yes	2	2	4

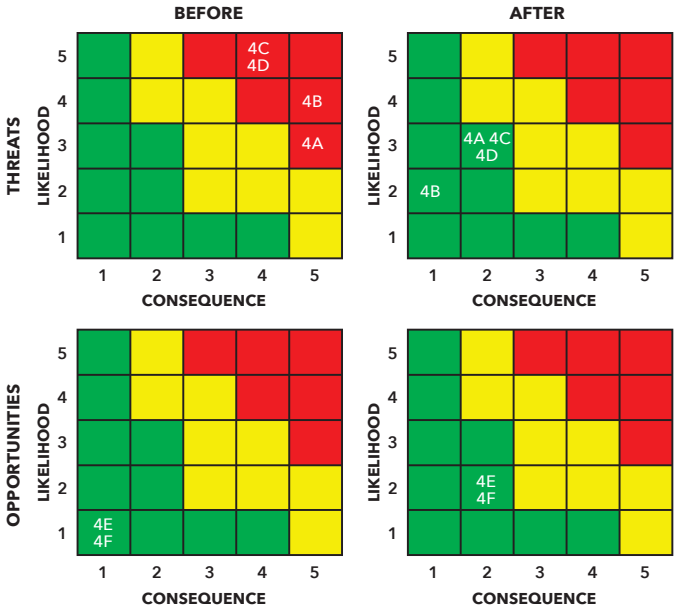


Figure 5. Risk matrix with before and after lean mitigation

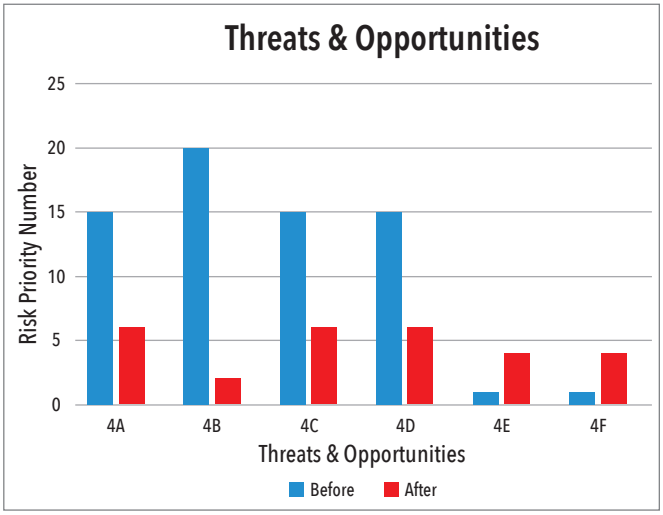


Figure 6. Threats and opportunity before and after chart

budget that may be reduced or eliminated while keeping safety and quality secured.

- inform the customer of certain threats or opportunities they may wish to consider when accepting a certain scope of work.
- populate project schedules to contrast before and after postures of threat and opportunity forecasting.

CONCLUSION

The MAC, a hypothesized approach to project lifecycle development, examines various, potentially viable offers for project lifecycle development. While the intended use is to exploit areas of alternative lifecycle approaches holistically, it is possible to extract possible derivatives of independent ideas if they may be applicable to peripheral projects with comparable needs. Limitations in this study include the scope of only utilizing GSE developments in the examination and the inability to fully vet the hypothesized lifecycle development in actual project work. ■

REFERENCES

- Briatore, S., and A. Golkar. 2021. "Estimating Task Efforts in Hardware Development Projects in a Scrum Context." *IEEE Systems Journal*, 15(4), pp.5119-5125.
- Cherry, K. 2012. Multitasking: The cognitive costs of multitasking. About, Cognitive Psychology. http://psychology.about.com/od/cognitivepsychology/a/costs-of-multitasking.htm.
- Griffiths, M. 2012. *PMI-ACP Exam Prep*, 2nd Edition. Minnesota: RMC Publications, pp.56-58.
- Ings, S. 2020. "*Boots on The Moon*." New Scientist 246, no. 3284 (2020): 24.
- Kupiainen, E., M. V. Mäntylä, and J. Itkonen. 2015. "Using Metrics in Agile and Lean Software Development–A Systematic Literature Review of Industrial Studies." *Information and Software Technology*, 62, pp.143-163.
- Martin, P. K. 2021. NASA's Management of Artemis Missions. Report No. IG-22-003.Pdf. https://oig.nasa.gov/docs/IG-22-003.pdf (November 15).
- Peterson, M., and J. Summers. 2021. "When worlds collide–a comparative analysis of issues impeding adoption of agile for hardware." *Proceedings of the Design Society, 1*, pp.3451-3460.
- Project Management Institute (PMI). 2017. *Agile Practice Guide*. First Edition. Newton Square, PA.
- Schmidt, T. S., A. Chahin, J. Kößler, and K. Paetzold. 2017. "Agile development and the constraints of physicality: a network theory-based cause-and-effect analysis." In DS 87-4 *Proceedings of the 21st International Conference on Engineering Design (ICED 17) Vol 4: Design Methods and Tools, Vancouver, Canada*, 21-25.08. 2017 (pp. 199-208).

https://www.scaledagileframework.com

ABOUT THE AUTHORS

Michael A. Cabrera attended Texas Tech and received undergraduate degrees in accounting and mechanical engineering. He then attended the University of Houston – Clear Lake and received graduate degrees in physics and engineering management. Currently, he is a PhD candidate in systems engineering at Colorado State University. Professionally, he has experience as a robotic operations analyst on Shuttle and ISS operations, an offshore design engineer on an oil rig, a project manager for the treadmill aboard the ISS, and currently works as a project manager supporting the next generation space suit. He has a passion for management, team building, teaching, and contining education.

Steve Simske is a professor of systems engineering at Colorado State University. Steve was at Hewlett Packard (HP) from 1994 to 2018 and was an HP fellow, vice president, and director in HP Labs. He has authored more than 450 publications and more than 200 US patents. Steve is an IEEE fellow and an NAI fellow. He is an IS&T fellow and its immediate past president (2017-2019). Steve is the steering committee chair for the ACM DocEng Symposium, which meets annually and benefits from University of Nottingham CS Professors Brailsford and Bagley being active leaders. Dr. Simske was a member of the World Economic Forum Global Agenda Councils from 2010-2016, including illicit trade, illicit economy, and the future of electronics. In his 20+ years in the industry, Steve directed teams in research on 3D printing, education, life sciences, sensing, authentication, packaging, analytics, imaging, and manufacturing. His books Meta-Algorithmics, Meta-Analytics, and Functional Applications of Text Analytics Systems bring computer science patterns and principles to address intelligent (AI/ML) systems. At CSU, he has a cadre of on-campus students in systems, mechanical, and biomedical engineering, along with a larger contingent of online/remote graduate students researching in a wide variety of disciplines.

Rosser continued from page 56

- Scaled Agile Framework c. 2021. "SAFe 5 for Lean Enterprises." viewed 8 March 2022. https://www.scaledagileframework.com
- Schindel, W., and R. Dove. 2019. "Agile Systems Engineering Life Cycle Model for Mixed Discipline Engineering." Proceedings International Symposium. International Council on Systems Engineering, Orlando, US-FL, 20-25 July.
- TecFAR Hub c. 2022. "TechFAR Handbook." viewed 20 March 2022 https://techfarhub.cio.gov/handbook.
- US Government Accountability Office. 2020. Agile Assessment Guide: Best Practices for Agile Adoption and Implementation.

ABOUT THE AUTHOR

Larri Ann Rosser has worked in engineering and technology for four decades as an electrical engineer, software engineer, systems engineer, and architect. She holds a BS in information systems and computer science from Charter Oak State college, and a Master of Science in systems engineering from Worcester Polytechnic Institute. She holds multiple patents in the man portable systems domain and is a CAP certified architect, a SAFe program consultant, and a Raytheon six sigma expert. She is the co-chair of the INCOSE Agile Systems and Systems Engineering Working Group, a member of the INCOSE Complex Systems Working Group, and a member of the NDIA SED Architecture Committee, and the NDIA ADAPT working group. At Raytheon, she works with programs and product lines to apply modern methods to system realization.

Systems Engineering: The Journal of The International Council on Systems Engineering

Call for Papers

The *Systems Engineering* journal is intended to be a primary source of multidisciplinary information for the systems engineering and management of products and services, and processes of all types. Systems engineering activities involve the technologies and system management approaches needed for

- definition of systems, including identification of user requirements and technological specifications;
- development of systems, including conceptual architectures, tradeoff of design concepts, configuration management during system development, integration of new systems with legacy systems, integrated product and process development; and
- deployment of systems, including operational test and evaluation, maintenance over an extended life-cycle, and re-engineering.

Systems Engineering is the archival journal of, and exists to serve the following objectives of, the International Council on Systems Engineering (INCOSE):

- To provide a focal point for dissemination of systems engineering knowledge
- To promote collaboration in systems engineering education and research
- To encourage and assure establishment of professional standards for integrity in the practice of systems engineering
- To improve the professional status of all those engaged in the practice of systems engineering
- To encourage governmental and industrial support for research and educational programs that will improve the systems engineering process and its practice

The journal supports these goals by providing a continuing, respected publication of peer-reviewed results from research and development in the area of systems engineering. Systems engineering is defined broadly in this context as an interdisciplinary approach and means to enable the realization of successful systems that are of high quality, cost-effective, and trustworthy in meeting customer requirements.

The *Systems Engineering* journal is dedicated to all aspects of the engineering of systems: technical, management, economic, and social. It focuses on the life-cycle processes needed to create trustworthy and high-quality systems. It will also emphasize the systems management efforts needed to define, develop, and deploy trustworthy and high quality processes for the production of systems. Within this, *Systems Engineering* is especially concerned with evaluation of the efficiency and effectiveness of systems management, technical direction, and integration of systems. *Systems Engineering* is also very concerned with the engineering of systems that support sustainable development. Modern systems, including both products and services, are often very knowledge-intensive, and are found in both the public and private sectors. The journal emphasizes strategic and program management of these, and the information and knowledge base for knowledge principles, knowledge practices, and knowledge perspectives for the engineering of systems. Definitive case studies involving systems engineering practice are especially welcome.

The journal is a primary source of information for the systems engineering of products and services that are generally large in scale, scope, and complexity. *Systems Engineering* will be especially concerned with process- or product-line–related efforts needed to produce products that are trustworthy and of high quality, and that are cost effective in meeting user needs. A major component of this is system cost and operational effectiveness determination, and the development of processes that ensure that products are cost effective. This requires the integration of a number of engineering disciplines necessary for the definition, development, and deployment of complex systems. It also requires attention to the lifecycle process used to produce systems, and the integration of systems, including legacy systems, at various architectural levels. In addition, appropriate systems management of information and knowledge across technologies, organizations, and environments is also needed to insure a sustainable world.

The journal will accept and review submissions in English from any author, in any global locality, whether or not the author is an INCOSE member. A body of international peers will review all submissions, and the reviewers will suggest potential revisions to the author, with the intent to achieve published papers that

- relate to the field of systems engineering;
- represent new, previously unpublished work;
- advance the state of knowledge of the field; and
- conform to a high standard of scholarly presentation.

Editorial selection of works for publication will be made based on content, without regard to the stature of the authors. Selections will include a wide variety of international works, recognizing and supporting the essential breadth and universality of the field. Final selection of papers for publication, and the form of publication, shall rest with the editor.

Submission of quality papers for review is strongly encouraged. The review process is estimated to take three months, occasionally longer for hard-copy manuscript.

Systems Engineering operates an online submission and peer review system that allows authors to submit articles online and track their progress, throughout the peer-review process, via a web interface. All papers submitted to *Systems Engineering*, including revisions or resubmissions of prior manuscripts, must be made through the online system. Contributions sent through regular mail on paper or emails with attachments will not be reviewed or acknowledged.

All manuscripts must be submitted online to *Systems Engineering* at ScholarOne Manuscripts, located at:

https://mc.manuscriptcentral.com/SYS

Full instructions and support are available on the site, and a user ID and password can be obtained on the first visit.

CALLING ALL SYSTEMS

Series Sponsor DASSAULT SYSTEMES

Product Line Engineering:

Are You Missing a Piece in Your Digital Engineering Puzzle?

Session sponsor BigLever Software, Inc.

Dr. Charles Krueger
BigLever Software

Marco Forlingieri
IBM Engineering

Dr. Bobbi Young
Worcester Polytechnic Institute

Rowland Darbin
INCOSE PLE Working Group

Brian Pepper
Dassault Systemes

The Future of MBSE

Session sponsor SIEMENS

Todd Tuthill
Siemens Digital Industry Software

Brett Hillhouse
IBM Sustainabiltiy Software

Stephanie Sharo Chiesi
Stevens Institute of Technology

Troy Peterson
SSI

Watch the replays at *incose.org/callingallsystems*

INCOSE
International Council on Systems Engineering
A better world through a systems approach / www.incose.org

INCOSE

Future events

Future events

AUG 08
INCOSE Los Angeles August Meeting: Digital Transformation & MBSE in a Heterogenous Environment
El Segundo, CA USA

Aug 17
INCOSE Chicagoland: System Engineering Competency Assessment Guide Overview
Schaumburg, IL USA

Aug 21–25
INCOSE SESA: Australian Simulation Congress 2023
Adelaide, AUSTRALIA

SEP 12–16
SAE and NASA: Energy and Mobility Conference and Expo
Cleveland, OH USA

SEP 14–17
2023 Annual INCOSE Western States Regional Conference (WSRC)
Richland, CA USA

SEP 20–22
12th Nordic Systems Engineering Autumn Tour 2023
Linköping • Copenhagen • Hamburg

OCT 11–14
AOSEC 2023: *Digitalization for Engineering Complex Systems*
Bangalore, INDIA

OCT 12–13
2023 INCOSE New England 5th Annual Fall Workshop
Storrs, CT USA

OCT 30–31
CSD&M (Complex Systems Design & Management) International Conference
Beijing, CHINA

NOV 21–22
INCOSE UK Annual Systems Engineering Conference
Liverpool L7 3FA, UNITED KINGDOM

INCOSE