Best Paper Award

33rd Annual INCOSE international symposium
hybrid event
Honolulu, HI, USA
July 15 - 20, 2023

Agile Systems Engineering – Eight Core Aspects

Rick Dove
Independent
Arizona, USA
dove@parshift.com

Kerry Lunney
Thales Australia
Sydney, NSW Australia
kerry.lunney@thalesgroup.com.au

Dr. Michael Orosz
Information Sciences Institute
University of Southern California
Marina del Rey, CA USA
mdorosz@isi.edu

Dr. Mike Yokell
Independent
Texas, USA
mike.r.yokell@gmail.com

Abstract. Agile engineering, of any kind, employs strategies for designing, building, sustaining, and evolving purpose-fulfilling creations when knowledge is uncertain and operational environments are dynamic. Strategies address what needs to be accomplished and why, without constraints or directions on how. How those strategies manifest as operational methods depends upon the engineering context. For instance, though single-domain software engineering is different than multi-domain systems engineering, both share the same goals and strategies. This article describes eight agility-supporting strategic aspects with application discussions and examples relevant to systems engineering and exposes common myths and misunderstandings. Each of the eight aspects can individually improve capability to deal with uncertain knowledge and dynamic environments.

Introduction

The purpose of this paper is to provide a basic agile systems engineering foundation for addressing a Vision 2035 challenge (INCOSE 2021, p. 58): "Systems engineering anticipates and effectively responds to an increasingly dynamic and uncertain environment."

INCOSE's Corporate Advisory Board, 120+ organizations, have designated Agile Systems Engineering as one of the top priorities for INCOSE to address. Desired guidance was communicated by CAB-chair Ron Giachetti directly to the Agile Systems & Systems Engineering working group as six questions:

1. What does it mean to be agile in the context of systems engineering?
2. What are the key practices that can make systems engineering agile?
3. How can organizations be more agile in their development of systems?
4. What benefits can be gained using of agile practices for systems engineering?
5. What is the relation between agility and model-based systems engineering (MBSE)?

6. Are there system characteristics and architectures that make some systems more amenable to agile development and others less so?

Questions one through four (Q1-Q4) roughly mirror the organization of this paper, with Q1 following this introduction. Q2 is addressed next in the section on Core Aspects, followed by Q3 in the section on Transition and Transformation, and Q4 in the Concluding Discussion. Q5 and Q6 are indirectly covered within the section on Core Aspects and clarified directly in the Concluding Discussion section. Located between the Transition and Concluding sections is a section outlining and correcting Myths and Misconceptions.

Agile System Engineering Meaning

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 *being* agile, not *doing* agile. Thus, Agile System Engineering is a what, not a how.

There are many *hows*, principally focused currently on the development phase, e.g., Evolutionary Development, Iterative Incremental Development (IID), Incremental Commitment Spiral Model (ICSM); and also many focused on a single (software) engineering domain, e.g. Scrum, Kanban, XP, and DevOps[1].

Agile systems engineering is best understood in contrast to sequential systems engineering in how the two relate to the system life cycle spectrum. Figure 1 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. It is questionable that a pure form of either depicted extreme would be effective in actual practice (INCOSE 2023).

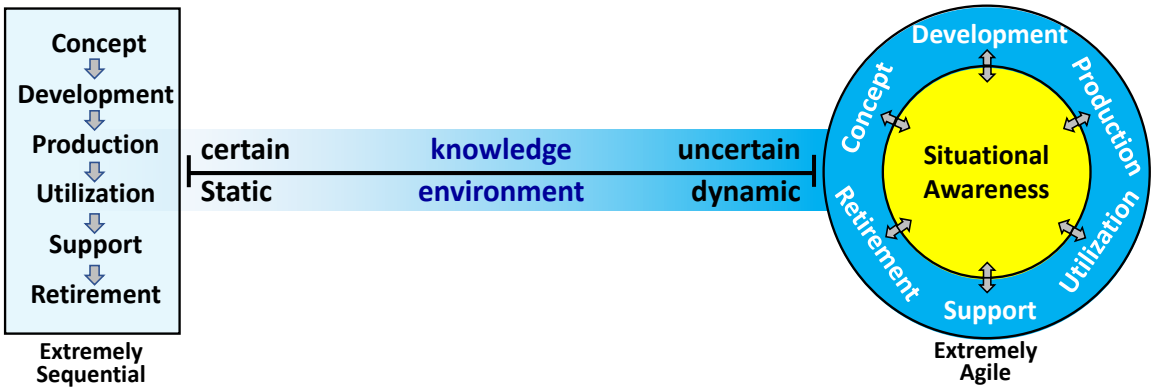

Figure 1: Systems engineering lifecycle spectrum – sequential to agile.

Background Context

To put agile systems engineering in perspective a brief history is in order. In 1991 the US Department of Defense funded a project to investigate what would drive competition in manufacturing

[1]DevOps IEEE/ISO/IEC Standard <https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=653267>

enterprises after the then-active scramble to become more Lean had stabilized. The results of that project put the concept of agility and the word agile into play as a way to describe how organizations would deal with an increasing frequency of change in markets and technologies. For the next four years the Agility Forum, funded by DARPA (Defense Advanced Research Agency) at Lehigh University, led 1200 participants from 125 organizations through a collaborative discovery process across a broad base of business and engineering areas. The initial focus on agile manufacturing evolved quickly during the early '90s into agile system, agile enterprise, and agile command and control, which led to the 2001 adoption of the agile word to describe a variety of new agile ways to develop software (Dove and LaBarge 2014). INCOSE established agile systems engineering as one of its top priorities in 2014, the pursuit of which is informed by 25 years of prior related discovery and refinement, augmented by INCOSE-instigated case studies and refinement of agile systems engineering life cycle concepts (Dove and Schindel 2019).

Core Aspects

Eight core (Figure 2) aspects are each explained succinctly as a Why (need) and a What (behavior), with some examples of How (method). Examples may draw from domain engineering areas if they are clearly instructive as abstractions for application at the systems engineering level.

Though these aspects are core strategies for any kind of agile engineering, the purpose and descriptions here are for application at the systems engineering level rather than the domain engineering level.

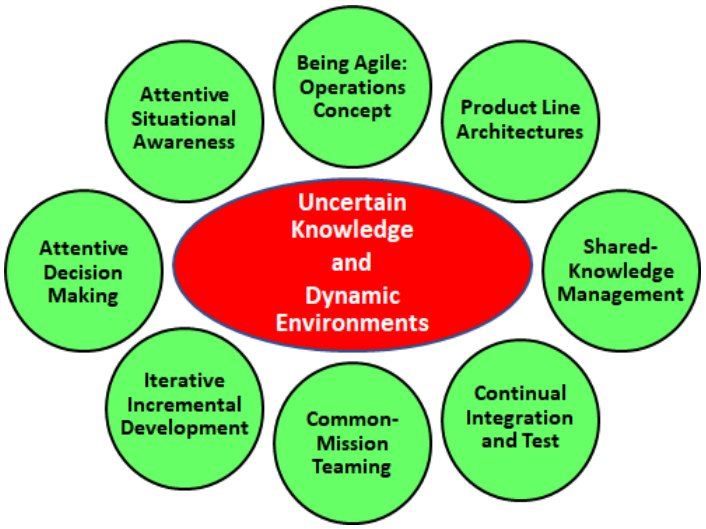

Figure 2: Eight core aspects of agile systems engineering.

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. Big bang

concurrent implementation of all aspects is not necessary to gain agility benefits. Incremental adoption can accommodate incremental appetites.

These eight aspects in their current form have emerged from the pooled knowledge of the authors of this article – knowledge gained from their considerable 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.

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 (Figure 3).

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, potentially informing multiple reference architectures.

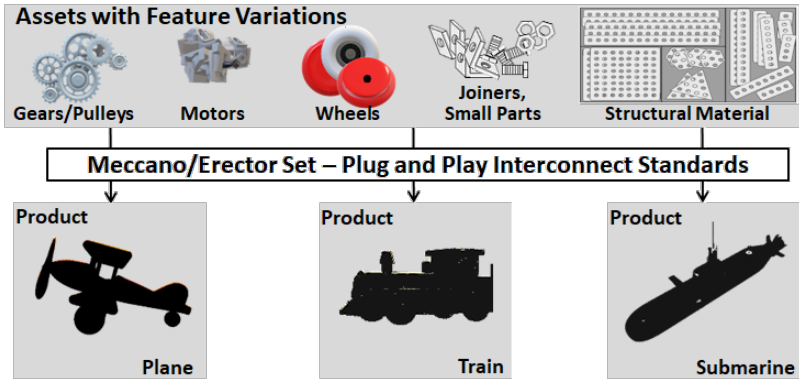

Figure 3: Agile Architecture Pattern depiction, adapted from (Dove and LaBarge 2014).

A hallmark of agile systems engineering is iterative incremental development (discussed next), which modifies work in process as suitability is repetitively evaluated. The agility of the process is dependent upon the agility of the product – so both process and product need to be easily changed.

Examples:

- Product: Automobile design for new model market entry and aftermarket modification.
- Process: SpaWar unmanned vehicle development (Dove, Schindel, Scrapper 2016) depicts assembly of IPT working groups, validation test activities, and frequent integration activities from available resources most appropriate for the activity of the moment.

Iterative Incremental Development

Needs: Minimize unexpected rework and maximize quality.

Behaviors: Incremental loops of building, evaluating, correcting, and improving capabilities (Figure 4).

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

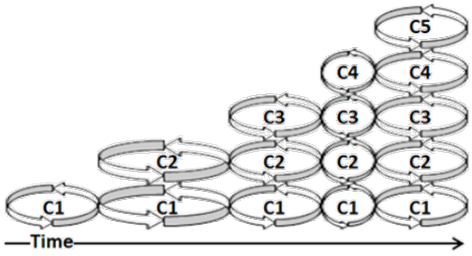

Figure 4: Iterative capability improvements (looping) and incremental capability additions (successive columns).

- 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.

Examples:

- Incremental Commitment Spiral Model (Boehm, Lane 2007).
- SpaceX: Rapid cycles of build-test-learn iterations[2] (Rasky n.d.a).
- SpaWar: Overlapping stages of subcontracted device development and government-led architecture, integration, and validation increments (Dove, Schindel, Scrapper 2016).
- Collins: asynchronous/unaligned integrated Domain Engineering (software, FPGA, ECB. mechanical) testable increments (Dove, Schindel, Hartney 2017).

Attentive Situational Awareness

Needs: Timely knowledge of emergent risks and opportunities.

Behaviors: Active monitoring and evaluation of relevant internal and external operational-environment factors (Figure 5).

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.


Figure 5: Alert in-the-moment constant attention.

Examples:

- Work in process demonstrations and reviews for stakeholder feedback.
- Periodic SE process-participant collaborative evaluations.
- Collins: Continual market and technology evolution monitoring and evaluation (Dove, Schindel, Hartney 2017).

[2] <www.youtube.com/watch?v=SMLDAgDNOhk&list=PL6vdik5frDGVL4USjKgYkJoOb76_7sdkS&index=12>

- Northrop: Systematic internet search for pending security threats and in-use COTS obsolescence (Dove, Schindel, Kenney 2017).
- SpaceX: constant internet search and rapid evaluation acquisition[3] (Rasky n.d.b).

Attentive Decision Making

Needs: Timely corrective and improvement actions.

Behaviors: Systemic linkage of situational awareness to decisive action (Figure 6).

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.

Figure 6: John Boyd's OODA loop (Philips 2021).

Examples:

- Satisficing – making a timely good-enough decision rather than an optimal time-consuming decision.
- Northrop: Systemic refactoring of development planning to shuffle resources needed to address real-time security-threat evolution (Dove, Schindel, Kenney 2017).
- SpaceX: "As soon as they would get to, we would joke, 51% probability, they would make a decision and move forward. … You keep making decision after decision after decision. If you find a problem you hadn't anticipated then you backtrack, make another decision and try it again. It allows you to progress very rapidly (Rasky n.d.b)."

Common-Mission Teaming

Needs: Coherent collective pursuit of a common mission.

Behaviors: Engaged collaboration, cooperation, and teaming among all relevant stakeholders (Figure 7).

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.


Figure 7: Tightly integrated coherent operation

[3] <https://www.youtube.com/watch?v=yit0FvjDtkw&list=PL6vdik5frDGVL4USjKgYkJoOb76_7sdkS&index=10>

Examples:

- Integrated product teams – multidisciplinary groups of people who are collectively responsible for delivering a defined product or process.[4]
- High-performance teams – groups of people with complementary skills, committed to a shared vision, working towards a common objective.
- SpaWar: Of particular note in the SE process was its successful objective and ability to integrate outside contractors as full team members, forming a family-like relationship of all-for-one and one-for-all (Dove, Schindel, Scrapper 2016).
- Mine-Resistant Ambush Protected (MRAP) program: Plagued with discordant relationships among a variety of service agencies, contractors, and manufacturers, Paul Mann credits the eventual acclaimed success of the MRAP program to the many people who pulled together in a process that enveloped them all in the mission of program success, rather than local optimization of individual needs or contract performance independent of the effect on all others in the program (Dove, Schindel, Scrapper 2016).

Shared-Knowledge Management

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

Behaviors: Facilitated communication, collaboration, and knowledge curation (Figure 8).

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.

Examples:

- Periodic status meetings and information radiators, e.g., war room status displays.
- Collaboration tools.
- Model based systems engineering (MBSE) tools.
- Product Lifecycle Management (PLM) tools.
- SpaWar: The "Continuous Integration Environment" (CIE) is a home-grown data-driven repository of knowledge, with customized viewing templates for different needs and viewers. The intent is to facilitate a real-time collective consciousness, where all team members are plugged in to all information associated with full project success, as well as to the information of relevance to their specific responsibilities and tasks (Dove, Schindel, Scrapper 2016).

Figure 8: Depicted books represent information containers of any kind; but typically digital.

[4] https://acc.dau.mil/CommunityBrowser.aspx?id=24675&lang=en-US

Continual Integration & Test

Needs: Early revelation of system integration issues.

Behaviors: Integrated demonstration and test of work-in-process (Figure 9).

Figure 9: SpaWar iteratively evolving unmanned technology integration platform (Dove, Schindel, Scrapper 2016).

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. The examples below show some effective alternatives.

Examples:

- Digital engineering.
- Iron bird – a physical system mockup for prototyping and integrating aircraft systems during development.
- Lockheed: ANTE (Agile Non-Target Environment) is used to compose integrated systems consisting of simulated devices, real devices, software simulations and work-in-process, and temporary low fidelity proxy devices. Subcontractors are required to provide device simulations to ANTE specs. (Dove, Schindel, Garlington 2018)
- SpaceX: "While design and simulation are extremely important at SpaceX, they do not try to perfect a design before they try it. They design, and they simulate, but they also build and test often. They feel that they learn more by building something and pushing it to failure than they would learn in a hundred simulations" (Berg 2019).

Being Agile: Operations Concept

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

Behaviors: Sensing, responding, evolving (Figure 10).

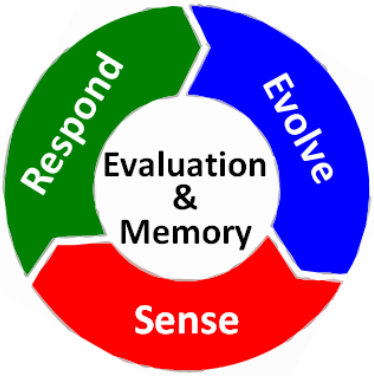

Figure 10: Three principles that operationalize agility

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.

Examples:

- Scrum provides introspective sense-respond-evolve examples in agile software engineering with its frequent product review and process retrospective ceremonies. These concepts are independent of the engineering domain.

- SpaceX provides extrospective sense-respond-evolve examples, cycling through instrumented in-the-environment operational testing for rapid design evolution (Rasky n.d.a), and actively searching the internet for innovative technology solutions with rapid acquisition capability for decisive evaluation (Rasky n.d.b).
- Colins combines both extrospective and introspective examples in their external market and technology evolution monitoring and evaluation compared against internal skills and competency needs (Dove, Schindel, Hartney 2017).

Transition and Transformation

Systems engineering is agile to various degrees on all projects – nobody is stuck at either extreme of the life cycle spectrum (Figure 1).

But how agile is agile enough? And how much is too much? The environments that systems engineering processes and systems engineered products operate in is the measure of that. "Systems engineering anticipates and effectively responds to an increasingly dynamic and complex environment (INCOSE 2021, p. 58)."

Agile systems engineering is a journey, not a destination. The dynamism and complexity of the systems engineering environments will continue to evolve over time. The nature of effective response will necessarily evolve accordingly. The how-part can evolve rapidly, the what-part (behavior strategies) will evolve at a slower rate, and the why-part (need) likely remains fairly stable with perhaps augmented interpretations.

Basically, there are two ways to become more agile: transition (incremental) and transformation (big bang).

A big bang transformation requires a top-down management mandate to be successful and is best accompanied by standardized process training for all executives, managers, and engineers in the development chain. A case study of Lockheed Martin's Integrated Fighter Group provides an example (Dove, Schindel, Garlington 2018), in which they trained 1,200 people in two-day classes on the SAFe process with an additional one day of home-grown custom tailoring.

An incremental transition can be a bottom-up initiative; and like any pursuit of expertise, it requires what is referred to as "deliberate practice" (Ericsson 1996):

1. A well-defined task (performance improvement objective to achieve next).
2. Appropriate difficulty level (a stretch within reach).
3. Informative feedback (continual progress analysis).
4. Opportunities for repetition and correction of errors (converge on the objective).

It is not necessary to have all eight core aspects in place before targeting one or some for incremental improvement. What is necessary is to have a measurable improvement target and a desire to succeed.

Challenges: inertia and hard times. The incremental approach effectively wears away the inertia with a carefully chosen (least resistant, low hanging fruit) path of incremental improvements. The hard times issue is difficult if management reverts to old rigid ways during a crisis incorrectly

believing a return to progressive ways can occur when the crisis is over (Li, Mukherjee, and Vasconcelos 2022).

Myths and Misunderstandings

Myths and misunderstandings associated with agile software engineering often influence concerns about agile systems engineering (Carlson 2017, Deloitte 2014, Tzemach, 2022). What follows is a brief summary of common myths, with an associated correction to the causal misunderstanding.

Paperwork is not required. Myth buster: The emphasis in agile approaches is to develop a working product with only the necessary documentation required for understanding to aid in sustainability and future development.

There are no checks and balances. Myth buster: Checks and balances are achieved through the demonstration of working products that meet customer needs and through project governance, including cadence of reviews and frequent customer feedback.

The speed of development is faster. Myth buster: Agile approaches deliver working product sooner to the customer, but that doesn't necessarily mean that the overall project will be completed faster. Unlike traditional sequential efforts where all capabilities are delivered once and at the end of a project, agile approaches deliver capabilities to the customer throughout the project, giving the appearance that the project is proceeding more quickly than traditional approaches.

Agile approaches are only for software. Myth buster: No, although there are many codified processes in practice for software development and fewer, more proprietary for other engineering domains. For example, Relativity Space's "Stargate Factory" (Relativity Space 2022) reportedly plans to produce 95% of their rocket using additive manufacturing, allowing them to reduce the build time of a rocket from 24 months to a predicted two months, and the design iteration time from 48 months to six.

Agile approaches aren't for highly regulated industries. Myth buster: Often more challenging to codified agile approaches with insufficient contextual flexibility; but not inherently antithetical to a principle-based agile approach that addresses regulations as functional requirements. For example, agile approaches have successfully been implemented in US Department of Defense applications (Orosz, *et al.,* 2022).

There is less planning involved in agile approaches. Myth buster: Arguably there is more planning and replanning in an agile approach, with an emphasis on just-in-time dynamic planning based on evolving knowledge rather than conjectured expectations (i.e., plan, act, observe, feedback, replan, act, observe, etc.).

Agile approaches are new concepts. Myth buster: Although introduced in 2001, the Agile Manifesto (Fowler and Highsmith 2001) was not the beginning of agile approaches. Agile-like approaches have been implemented for many years (see earlier section on Background Context). For example, the concept of successive repetitions of the process of interlaced testing and design of the system ultimately becoming the system was introduced at the 1968 NATO conference on Software Engineering (Naur and Randell 1969, p. 32).

Agile approaches favor developers. Myth buster: In agile approaches the teams (often composed of systems engineers, developers, testers, integrators, customer representatives and others) are given the authority to implement their scope. That is, teams, not the developers, self-organize and make decisions to implement their allocated features/stories. The focus here is on teams, not developers.

An agile approach is hampered by architecture. Myth buster: There is always the need to understand the overall goals, parameters, and architecture to determine smaller development runs. Co-engineering is a must!

An agile approach doesn't scale. Myth buster: It isn't straightforward to scale; however, hierarchical approaches that emphasize cross-team collaborations and coordination have been successfully used to scale agile development.

Regular customer feedback through development minimizes/removes the need for formal validation. Myth buster: Validation of the final system solution in its intended operating environment is necessary; however, it may be streamlined based on the previous levels of testing and the risk profile of the project.

Concluding Discussion

We have shown that agile systems engineering practicing eight core aspects is able to satisfy the challenge: "Systems engineering anticipates and effectively responds to an increasingly dynamic and uncertain environment. (INCOSE 2021)." Attentive Situational Awareness is the explicit aspect of "anticipates," but it relies on every other aspect to contribute implicitly. Attentive Decision Making is the explicit aspect of "effectively responds," but it relies on every other aspect to enable that response effectiveness.

We have also addressed the six questions posed by a collective of potential users (INCOSE's Corporate Advisory Board) in search of guidance. Aspect contributions shown in item 4 below are indicative, not comprehensive, as virtually every aspect is involved in delivering benefits:

1. What does it mean to be agile in the context of systems engineering?
   - Your processes and people are prepared to navigate a systems engineering project through dynamic and uncertain operating environments.

2. What are the key practices that can make systems engineering agile?
   - Eight core aspects were discussed as key practices.

3. How can organizations be more agile in their development of systems?
   - Employ and continuously improve performance of the eight core aspects.
   - Incremental transition or big-bang transformation.

4. What benefits can be gained using agile practices for systems engineering?
   - Minimal rework time and cost through incremental iterative development and continual integration (faster, less expensive).

- Delivery of value within time and budget constraints through employment of all eight aspects, with incremental and iterative development prioritizing most viable product and evolutions in the sequence of increments.
- Higher quality results through iterative incremental development, common mission teaming, shared knowledge management, and continual integration and test (better fit with customer needs)
- Motivated, engaged, and productive employees through common mission teaming shared knowledge management and iterative incremental development.
- Leading edge competitive capability through mastery of all eight aspects.

5. What is the relation between agility and model-based systems engineering (MBSE)?
   - They are synergistic with rapid and frequent design prototyping (an experiment and learning aid) and with improved stakeholder and developer communication.
   - MBSE can provide cross domain and domain independent design interaction communication.
   - MBSE can demonstrate integrated functionality of a total system design long before functional capability has physical presence.

6. Are there system characteristics and architectures that make some systems more amenable to agile development and others less so?
   - More amenable systems can be architected as modular open systems.
   - Less amenable systems have external and uncontrollable dependencies.

The eight aspects of agile systems engineering deal with the agility enabling and operational parts of systems engineering. In this respect they are a tailoring of a systems engineering process that encompasses many more process considerations.

Agile systems engineering is a journey, not a destination (Figure 11). The operational environments of systems engineering and of engineered systems will continue to evolve, requiring that agile systems engineering continues to evolve. This paper identified eight core aspects that represent the starting points for that journey. Those that have already started are moving on to maturing and evolving their capabilities (Willett et al. 2021).

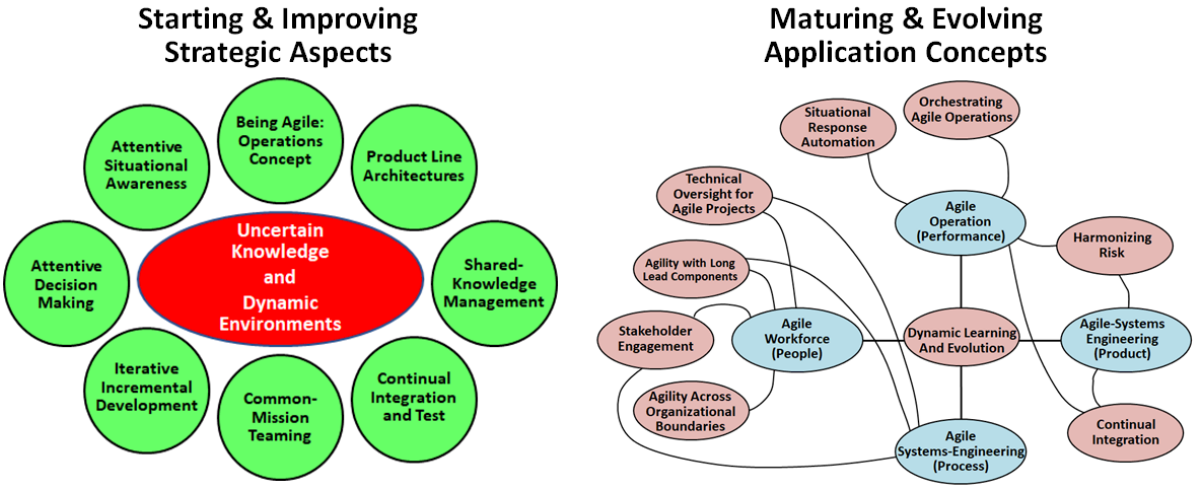

Figure 11. Large organizations likely have units working in both early and advanced stages.

References

Berg, C. 2019. SpaceX's Use of Agile Methods. Accessed 24 Sep 2022: https://cliffberg.medium.com/spacexs-use-of-agile-methods-c63042178a33

Boehm, B. and J. A. Lane. 2007. Using the Incremental Commitment Model to Integrate System Acquisition, Systems Engineering, and Software Engineering. CrossTalk, October. https://www.researchgate.net/publication/228699789

Carlson, D. 2017. Debunking agile myths. CrossTalk, 30(3), pp.32-36. June. https://zlmonroe.com/CSE566/Readings/7.Debunking_Agile_Myths.pdf

Phillips, M. S. 2021. Revisiting John Boyd and the OODA Loop in Our Time of Transformation. Defense Acquisition University. Defense Acquisition Magazine, October 21. www.dau.edu/library/defense-atl/blog/revisiting-john-boyd

Deloitte. 2014. 9 Myths About Agile. Wall Street Journal. 25-March. Accessed 23 November 2022: https://deloitte.wsj.com/articles/9-myths-about-agile-1395720095

Dove, R. and R. LaBarge. 2014. Fundamentals of Agile Systems Engineering – Part 1 & Part 2. International Council on Systems Engineering. International Symposium, Las Vegas, NV, June 30-July 3. www.researchgate.net/publication/264219672_Fundamentals_of_Agile_Systems_Engineering_-_Part_1_and_Part_2

Dove, R., W. Schindel, and C. Scrapper. 2016. Agile Systems Engineering Process Features Collective Culture, Consciousness, and Conscience at SSC Pacific Unmanned Systems Group. Proceedings International Symposium. International Council on Systems Engineering. Edinburgh, Scotland, July 18-21. www.researchgate.net/publication/308083752_Agile_Systems_Engineering_Process_Features_Collective_Culture_Consciousness_and_Conscience_at_SSC_Pacific_Unmanned_Systems_Group

Dove, R., W. Schindel, and R. Hartney. 2017. Case Study: Agile Hardware/Firmware/Software Product Line Engineering at Rockwell Collins. Proceedings 11th Annual IEEE International Systems Conference. Montreal, Quebec, Canada, April 24-27. www.researchgate.net/publication/316280809_Case_Study_Agile_HardwareFirmwareSoftware_Product_Line_Engineering_at_Rockwell_Collins

Dove, R., W. Schindel, and M. Kenney. 2017. Case study: Agile SE Process for Centralized SoS Sustainment at Northrop Grumman. Proceedings International Symposium. International Council on Systems Engineering. Adelaide, Australia, July 17-20. www.researchgate.net/publication/319406885_Case_Study_Agile_SE_Process_for_Centralized_SoS_Sustainment_at_Northrop_Grumman

Dove, R., W. Schindel, and K. Garlington. 2018. Case Study: Agile Systems Engineering at Lockheed Martin Aeronautics Integrated Fighter Group. International Council on Systems Engineering, International Symposium, Washington, DC, July 7-12. www.researchgate.net/publication/327072122_Case_Study_Agile_Systems_Engineering_at_Lockheed_Martin_Aeronautics_Integrated_Fighter_Group

Dove, R. and W. Schindel. 2019. Agile Systems Engineering Life Cycle Model for Mixed Discipline Engineering. Proceedings International Symposium. International Council on Systems Engineering. Orlando, FL, July 20-25. www.researchgate.net/publication/334772706_Agile_Systems_Engineering_Life_Cycle_Model_for_Mixed_Discipline_Engineering

Ericsson, K. A. 1996. *The Acquisition of Expert Performance: An Introduction to Some of the Issues*. Chapter 1: The Road to Excellence – The Acquisition of Expert Performance in the Arts and Sciences, Sports and Games. Edited by K. A. Ericsson. Lawrence Erlbaum Associates, Inc. Free eBook: https://oiipdf.com/the-road-to-excellence-the-acquisition-of-expert-performance-in-the-arts-and-sciences-sports-and-games

Fowler, M. and J. Highsmith. 2001. The Agile Manifesto. Dr. Dobb's Journal, August. www.drdobbs.com/open-source/theagile-manifesto/184414755

INCOSE. 2021. *Systems Engineering Vision 2035*. International Council on Systems Engineering.

INCOSE. 2023. *Systems Engineering Handbook: A Guide for System Life Cycle Process and Activities* (5th ed.). Section 4.2.3, Agile Systems Engineering. D. D. Walden, T. M. Shortell, G. J. Roedler, B. A. Delicado, O. Mornas, Y. S. Yip, and D. Endler (Eds.). San Diego, CA: International Council on Systems Engineering. Published by John Wiley & Sons, Inc.

Li, J., A. Mukherjee, and L. Vasconcelos. 2022. What Makes Agility Fragile? A Dynamic Theory of Organizational Rigidity. Management Science. http://www.amukherjee.net/Rulebooks.pdf

Naur, P. and B. Randell, Editors. 1969. Software Engineering. Report on conference sponsored by the NATO SCIENCE COMMITTEE, Garmisch, Germany, 7th to 11th October 1968. Published January, 1969. <https://www.scrummanager.com/files/nato1968e.pdf>

Orosz, M., Spear, G., Duffy, B., and Charlton, C. 2022. Introducing Agile/DevSecOps into the Space Acquisition Environment. *Proceedings, Naval Postgraduate School 19th Annual Acquisition Research Symposium Proceedings*. Vol 1: 405-416. Naval Postgraduate School. https://dair.nps.edu/bitstream/123456789/4541/1/SYM-AM-22-028.pdf

Philips, M. S. 2021. Revisiting John Boyd and the OODA Loop in Our Time of Transformation. Defense Acquisition University, Defense Acquisition Magazine, Sept/Oct. https://www.dau.edu/library/defense-atl/DATLFiles/Sept-Oct_2021/defacq-datl_Phillips_SeptOct2021.pdf

Rasky, D. n.d.a Space X's Rapid Prototyping Design process. National Aeronautics and Space Agency. https://www.youtube.com/watch?v=SMLDAgDNOhk&list=PL6vdik5frDGVL4USjKgYkJoOb76_7sdkS&index=12

Rasky, D. n.d.b Applying Software Design Process to Aerospace. National Aeronautics and Space Agency. www.youtube.com/watch?v=yit0FvjDtkw&list=PL6vdik5frDGVL4USjKgYkJoOb76_7sdkS&index=10

Relativity Space. 2022. *Factory of the Future.* <https://www.relativityspace.com/stargate>

Scrapper, C., R. Halterman, and J. Dahmann. 2016. An implementer's view of the evolutionary systems engineering for autonomous unmanned systems. IEEE Systems Conference (SysCon 2016). Orlando, Florida, 18-21 April. https://zenodo.org/record/1280529/files/article.pdf

Tzemach, D. 2022. Top Agile Myths & Misconceptions. Lambdatest, 27-September. Accessed 23 November 2022: https://www.lambdatest.com/blog/agile-myths-misconceptions

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. Proceedings International Symposium. International Council on Systems Engineering. July 17-22. www.researchgate.net/publication/353348381_Agility_in_the_Future_of_Systems_Engineering_FuSE_-_A_Roadmap_of_Foundational_Concepts

Biography

Rick Dove is an independent researcher, systems engineer, and project manager generally focused in 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 USA. She also participates in a number of 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. Michael 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.