Agile Systems Engineering – A Primer on Eight Core Aspects

May 10, 2023 – Enchantment Chapter

Rick Dove
Chair, Agile Systems & Systems Engineering WG

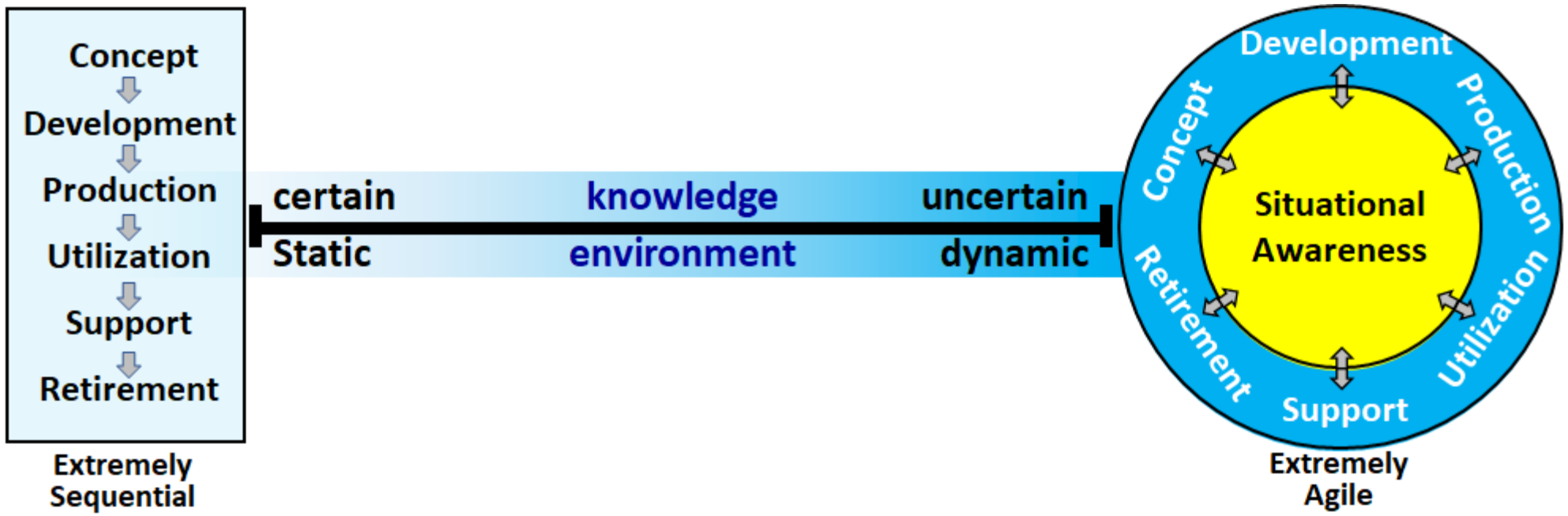
rick.dove@parshift.com, attributed copies permitted

For the Record …

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. Though single-domain software engineering is different than multi-domain systems engineering, both share the same goals and strategies. A primer on eight core strategic aspects, discussed as behaviors in fulfillment of needs, will be shown to enable and amplify systems engineering agility.

Bio: Rick Dove is an independent researcher, systems engineer, and project manager generally focused in the system security and system agility areas. He chairs the INCOSE working groups for System Security Engineering, and for Agile Systems and Systems Engineering. He is an INCOSE Fellow, and author of R*esponse Ability – the Language, Structure, and Culture of the Agile Enterprise*.

Systems Engineering Life Cycle Spectrum
Sequential to Agile

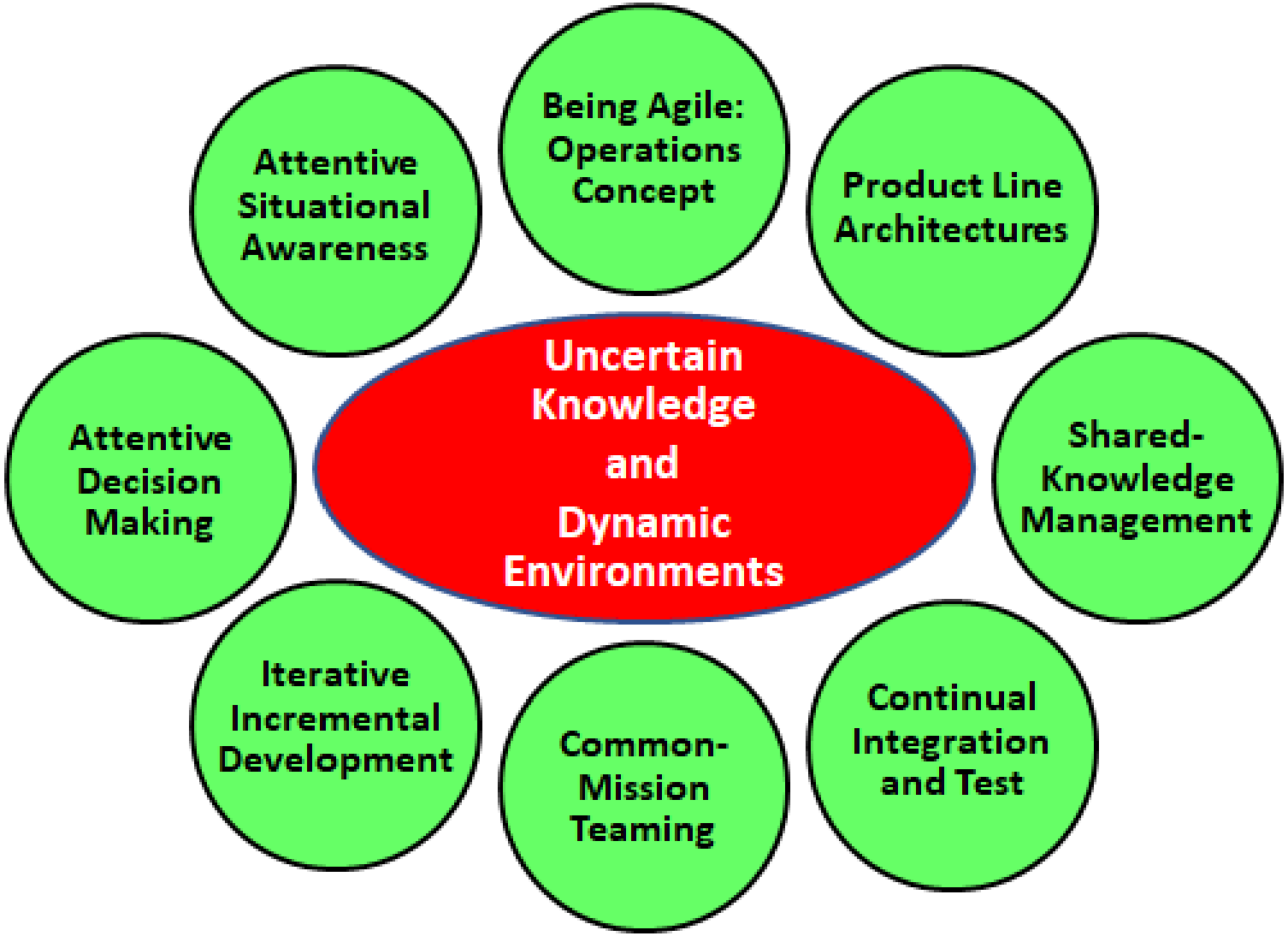


Agile systems engineering is a strategy-based method for designing, building, sustaining, and evolving systems when knowledge is uncertain and/or environments are dynamic

Agile System Engineering is a what, not a how.
There are many hows, principally focused on the development phase,
e.g.
Evolutionary Development
Iterative Incremental Development (IID)
Incremental Commitment Spiral Model (ICSM)
et al.

and also many focused on a single engineering domain,
e.g.
Scrum, Kanban, XP, DevOps, et al.

Eight Fundamental Aspects Enable Agility



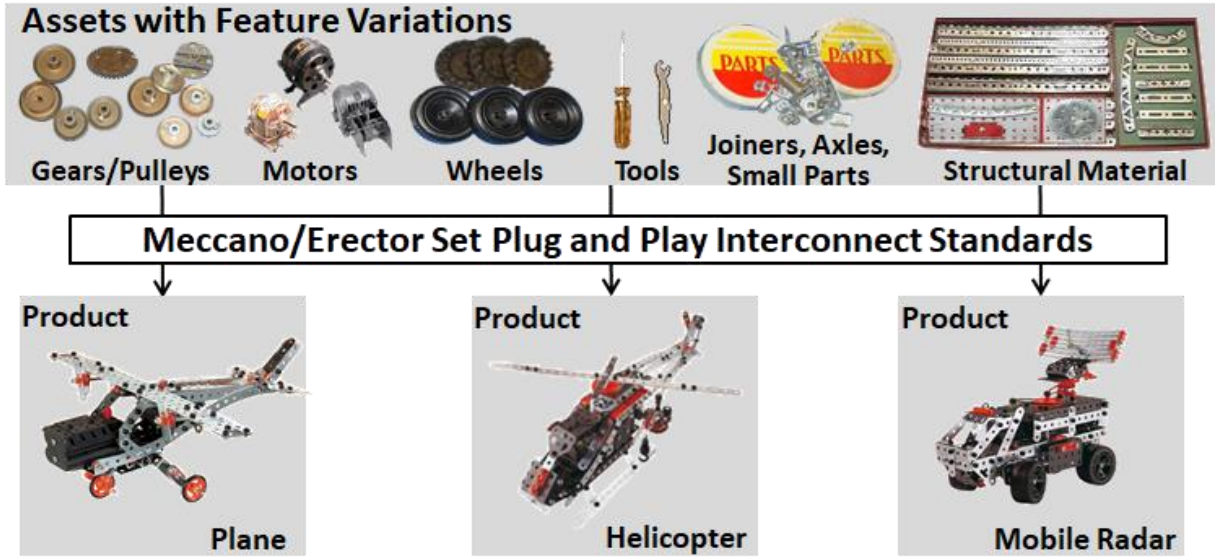
Product Line Architectures

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 on the agility of the product – so both process and product can be easily changed.



Iconic Agile Architecture Pattern

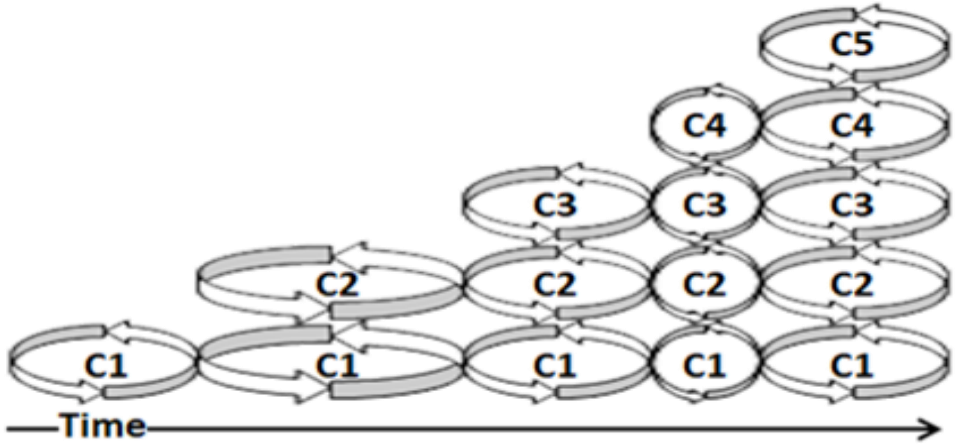
Iterative Incremental Development

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 columns)

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 you doing things right (internal awareness) and doing the right things (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 constant attention

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

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 information containers of any kind; but typically digital

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.

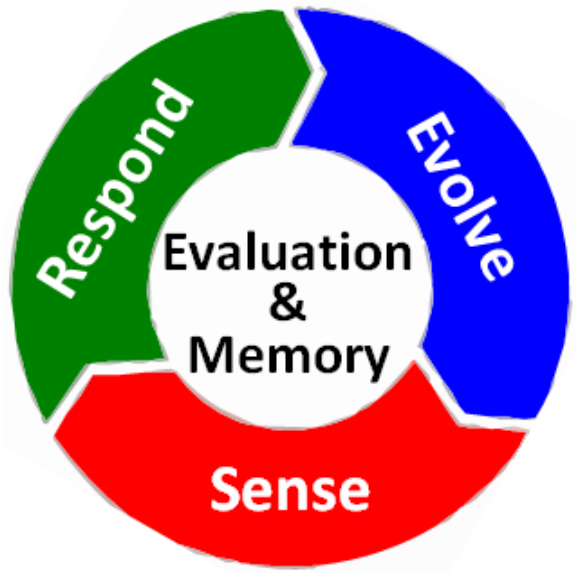
SpaWar iteratively evolving unmanned technology integration platform.

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

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.

Notional 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 you doing things right (internal awareness) and doing the right things (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 constant attention

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.

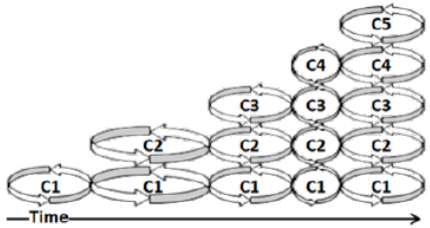
Iterative Incremental Development

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

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 information 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

Aspect Application is Context Dependent

Agile engineering, of any kind, is a principle-based method for designing, building, sustaining, and evolving purpose-fulfilling creations when knowledge is uncertain and operational environments are dynamic.

Principles are abstractions for what needs to be accomplished and why, without constraints or directions on how.

Single-domain software engineering is different than multi-domain systems engineering.

Aspect (principle)	**Software Engineering**	**Systems Engineering**
Feature-Based PLE Architectures	Standard interface	Proprietary interface
Iterative Incremental Development	Tight	Loose
Attentive Situational Awareness	Introspective	Extrospective
Attentive Decision Making	Simple	Complicated
Common-Mission Teaming	Homogeneous	Heterogeneous
Shared Knowledge Management	Code libraries	PLM
Continual Integration & Test	Common platforms	Proprietary platforms
Operations Concept (OpsCon)	Do Agile	Be agile

Many application contrasts can be listed in each row above depending upon project context.
Examples are chosen to provoke thinking and discussion of contrasts in context.

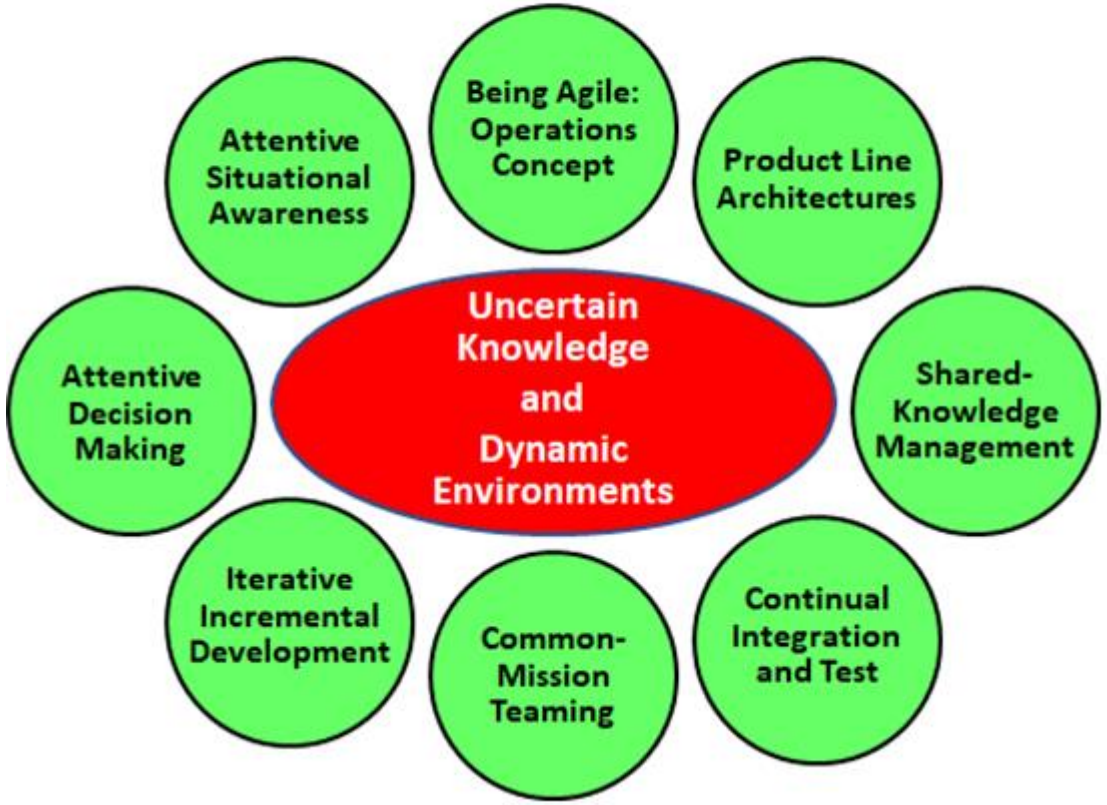
Supporting Material

1. Fundamentals of Agile Systems Engineering – Part 1 & Part 2. Dove, R., R. LaBarge. International Council on Systems Engineering. International Symposium, Las Vegas, NV, June 30-July 3, 2014. www.parshift.com/s/140630IS14-AgileSystemsEngineering-Part1&2.pdf
2. Introduction to the Agile Systems Engineering Life Cycle MBSE Pattern. Schindel, W., R. Dove. Proceedings International Symposium. International Council on Systems Engineering. Edinburgh, Scotland, July 18-21, 2016. www.parshift.com/s/160718IS16-IntroToTheAgileSystemsEngineeringLifeCycleMBSEPattern.pdf
3. [Case Study:] Agile Systems Engineering Process Features Collective Culture, Consciousness, and Conscience at SSC Pacific Unmanned Systems Group. Dove, R., W. Schindel, C. Scrapper. Proceedings International Symposium. International Council on Systems Engineering. Edinburgh, Scotland, July 18-21, 2016. www.parshift.com/s/ASELCM-01SSCPac.pdf.
4. Case Study: Agile Hardware/Firmware/Software Product Line Engineering at Rockwell Collins. Dove, R., W. Schindel, R. Hartney. Proceedings 11th Annual IEEE International Systems Conference. Montreal, Quebec, Canada, April 24-27, 2017. www.parshift.com/s/ASELCM-02RC.pdf
5. Case study: Agile SE Process for Centralized SoS Sustainment at Northrop Grumman. Dove, R, W. Schindel, M. Kenney. Proceedings International Symposium. International Council on Systems Engineering. Adelaide, Australia, July 17-20, 2017. www.parshift.com/s/ASELCM-03NGC.pdf.
6. Case Study: Agile Systems Engineering at Lockheed Martin Aeronautics Integrated Fighter Group. Dove, R., W. Schindel, K. Garlington. International Council on Systems Engineering, International Symposium, Washington, DC, July 7-12, 2018. www.parshift.com/s/ASELCM-04LMC.pdf
7. Agile Systems Engineering Life Cycle Model for Mixed Discipline Engineering. Dove, R., W. Schindel. Proceedings International Symposium. International Council on Systems Engineering. Orlando, FL, July 20-25, 2019. www.parshift.com/s/ASELCM-05Findings.pdf
8. Systems Engineering the Conditions of the Possibility (Towards Systems Engineering v2.0). Willett, K.D. Proceedings International Symposium. International Council on Systems Engineering. July 20-22, 2020. www.researchgate.net/publication/343306942_Systems_Engineering_the_Conditions_of_the_Possibility_Towards_Systems_Engineering_v20
9. Agility in the Future of Systems Engineering (FuSE) – A Roadmap of Foundational Concepts. Willett, K.D., R. Dove, A. Chudnow, R. Eckman, L. Rosser, J.S. Stevens, R. Yeman, M. Yokell. Proceedings International Symposium. International Council on Systems Engineering. July 17-22, 2021. www.parshift.com/s/210717IS21-FuseAgilityRoadmap.pdf
10. Complete Dan Rasky Interview (SpaceX Secrets), Knowledge @ NASA. https://www.youtube.com/watch?v=MxIiiwD9C0E

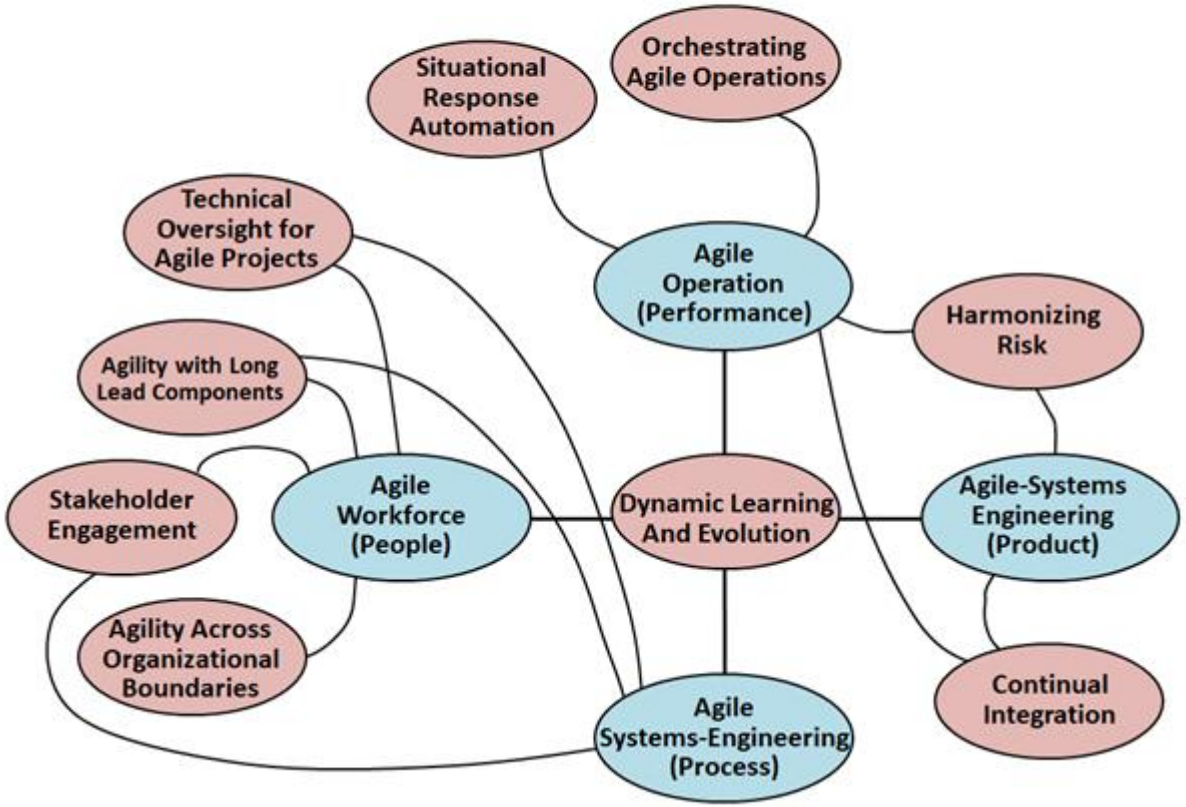
Background

Agile SE is a Journey

Starting & Improving Strategic Aspects



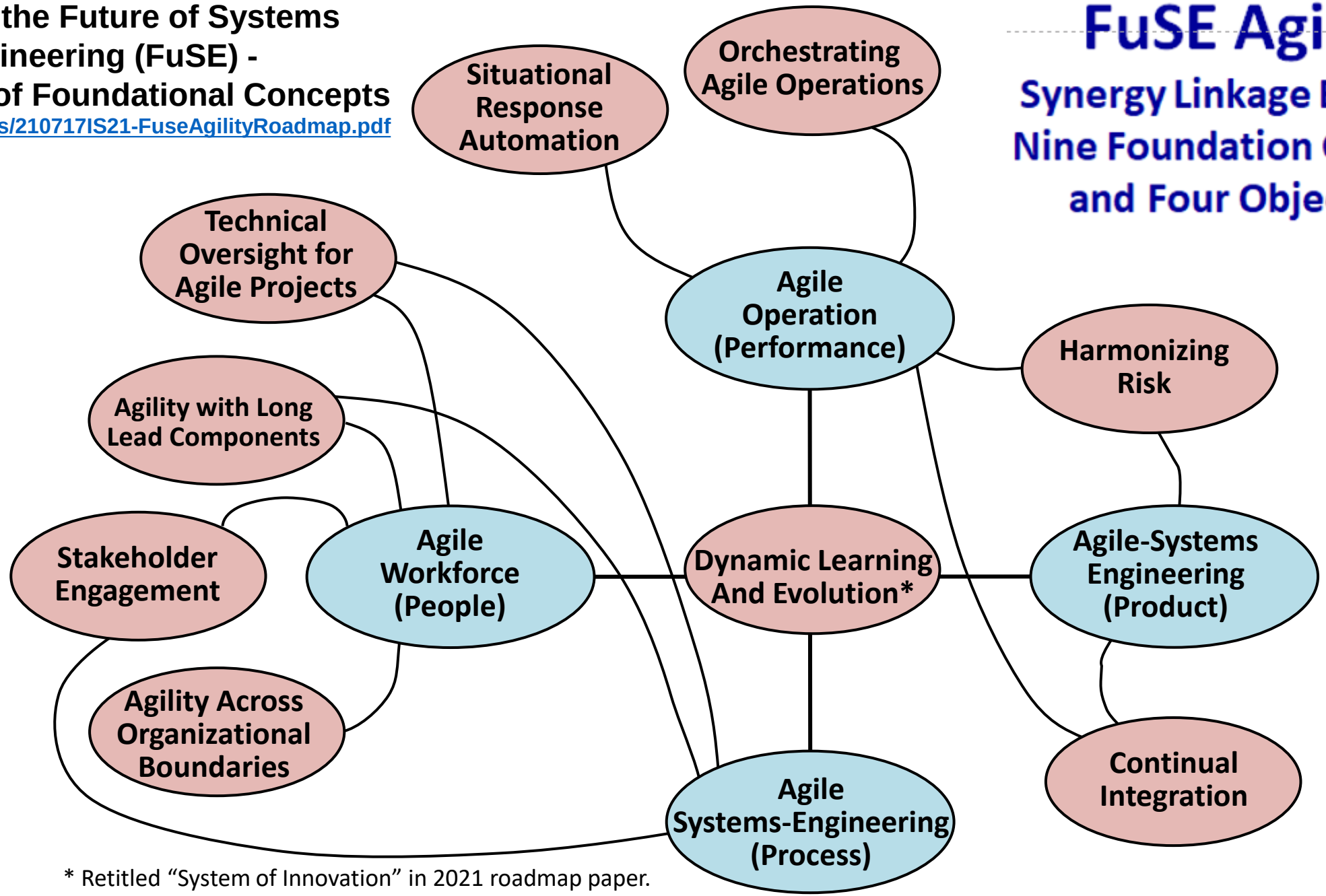
Maturing & Evolving Application Concepts



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

Agility in the Future of Systems Engineering (FuSE) - A Roadmap of Foundational Concepts

www.parshift.com/s/210717IS21-FuseAgilityRoadmap.pdf



* Retitled “System of Innovation” in 2021 roadmap paper.

FuSE Agility Roadmap Concepts

Concept Title	General Problem to Address	General Needs to Fill	General Barriers to Overcome
1. 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.
2. **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.
3. **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.
4. **Agility Across Organizational Boundaries**	Incompatible siloed cultures and languages.	Common language; less handoffs; product-based teams; common metrics.	Functional organizational silos.
5. **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.
6. **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 prior to HW being ready.
7. **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.
8. **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.
9. **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.