Editorial of *INSIGHT* Special Feature

Enabling and Practicing Systems Engineering Agility – Prelude

Rick Dove, dove@parshift.com

ABSTRACT
Agile systems engineering must function effectively in an engineering environment that is capricious, uncertain, risky, variable, and evolving (CURVE). This article leads off with methods for determining and justifying response capability requirements in such environments. These methods answer the questions: why is agility needed, and what must agility address? Subsequently, this article shows a framework for an agile system engineering lifecycle model (ASELCM) that has emerged from an INCOSE project concerned principally with determining such a model by analyzing real-world cases of what works. Together, the first enables effective process design, the second provides guidance for effective process practice. Finally, the article presents a synopsis of accompanying articles supporting the theme of this *INSIGHT* issue.

INTRODUCTION

This article leads off with methods for determining and justifying response capability requirements for any agile systems engineering process. These methods answer the questions: why is agility needed, and what must agility address? Answering these questions is a prudent prelude to designing both enabling capability and practicing capability, and it provides a traceable rationale for justifying process evolutionary changes. Subsequently, this article shows agility as a full lifecycle concern, rather than a development-only concern, which necessarily broadens the nature of requirements thinking.

An understanding of what the phrase agile systems engineering means is necessary–at the encompassing conceptual level, not at the procedural or best practice level. This starts with succinct statements of need and intent.

- Need: Effective systems engineering in the face of uncontrolled change
- Intent: Effective response to a systems engineering environment that is capricious, uncertain, risky, variable, and evolving (CURVE)

The word effective means that systems engineers obtain a valued result from resource employment. At one extreme, a project canceled before completion should provide valuable and employable learning and artifacts. At the other extreme, a deployed system should provide sustainable relevance beyond a break-even return on investment. In the middle, responding to changes in the engineering environment should sustain innovative forward progress. Methods for achieving this are outside the scope of this article, but some may be found in (Schindel 2018) and in (Dove, Schindel, and Hartney 2017).

Uncontrolled change encompasses uncontrollable change but is not limited to that which one cannot control, just what is not controlled regardless of reason.

Many think the value proposition for an agile systems engineering process is faster, lower-cost system development. That is an appealing argument, but it is only a best-case side effect. The fundamental value proposition for agility is risk and opportunity management–sustainability of innovation/process/product at risk. This article will discuss methods for identifying the sources and natures of process risk and the necessary occurrence-mitigation capabilities next.

The Environment Drives the Need for Agility

Agile systems (processes included) are defined in counterpoint to their operating environments. Words used to describe the general nature of the target environment often include and combine dynamic, unpredictable, uncertain, risky, variable, and changing, with little attention to clear distinction among them. To design and develop a system that can deal effectively with changing environments, it is useful to articulate the nature of changes that we should consider. A practice employed in teaching design methods for agile systems considers five types of environmental dynamics: caprice (unpredictability), uncertainty, risk, variation, and evolution. This categorization originated from a desire to explain why it felt natural to talk about agile systems as ones that can deal with uncertain and unpredictable environments.

Is there a meaningful difference between uncertain and unpredictable, or was this just a lazy tendency to use two words when one can do?

Research yielded the wisdom of Frank Knight, who very carefully and logically separated the meaning of risk from the meaning of uncertainty in his 1921 doctoral thesis, which he subsequently published and is available as a classic economics book. Knight's work argues that random events come in two varieties: those with knowable probability and those with unknowable probability. Knight states that this distinction separates risk and uncertainty. His knowable/unknowable distinction can also be a key differentiator for unpredictability and variation, though these do not have the symmetrical relationship of Knight's risk versus uncertainty.

Our objective is a tool that directs the designer's mind to a multidimensional exploration of response needs, consistent with the expectations of an agile system. This is an ill-structured problem in that essential variables are not numeric, goals are vague and not quantitative, and computational algorithms are not available. Ill-structured problems lend themselves to heuristic techniques, approaches to problem solving, learning, or discovery that employ a practical method, not guaranteed to be optimal or perfect, but sufficient for the immediate goals.

Agile systems, by definition, are ones that have effective situational response options, within mission, when operating in a CURVE environment. The CURVE heuristic framework is useful for characterizing uncontrolled internal and external environmental forces that impact process and product as systems. This framework characterizes the problem space in which either system will exist and for which the systems should have appropriate solution-space capability:

- Caprice: randomness among unknowable possibilities; unanticipated system-environment change
- Uncertainty: randomness among known possibilities with unknowable probabilities; kinetic and potential forces present in the system
- Risk: randomness among known possibilities with knowable probabilities; relevance of current system-dynamics understandings
- Variation: randomness among knowable variables and knowable variance ranges; temporal excursions on existing behavior attractors (a reference to complex system behavior trajectories)
- Evolution: gradual (relatively) successive developments; experimentation and natural selection at work.

The difference between risk and variation in this framework is that risk is viewed as the possible occurrence of a discrete event (a strike keeps all employees away), while variation is viewed as the intensity of a possible event (absenteeism varies with the season).

Stated earlier, the value proposition of agility is risk management. Recent thinking about risk is recognizing the role of uncertainty in addition to more traditional probability-based risk. For instance, Klinke and Renn describe precaution-based risk-management consistent with agile capability to deal with uncertain environments (2002), while Weike, Sutcliffe, and Obstfeld (1999) and Aven and Bodil (2014) explore the management of risk with operational concepts that employ agile system concepts to sense and mitigate the sources of risk.

You might wonder how a solution space can be prepared to respond to a capricious (unpredictable) situation. Think about the ingredients available in a well-stocked kitchen run by a creative chef. The employment combinations of these ingredients and the amounts of each used in any combination are uncountable. The chef will never use most possible combinations. But the possibility to concoct an appropriate combination is available when the unanticipated situation arises. This may be an opportunity for the proactive creation of a new signature dish; or a need for creative reaction when a diner with an unanticipated but declared allergy attends; or the Queen comes to dinner with an out-of-repertoire request. Note that we do not need to characterize the environment comprehensively or precisely if we present enough characterization to cause preparation sufficient to accommodate that which we have not itemized.

Fleshing out the uncontrolled problem space in the CURVE framework is a useful first step toward developing effective agile response requirements. Developing response requirements uses another heuristic framework, referred to as response situation analysis (RSA), which I will not detail here as it has been around for some time and is adequately covered in Dove and LaBarge, 2014; but the example below ties CURVE and RSA together for justification and traceability.

Table 1 shows a CURVE example characterizing environment-imposed needs. Table 2 shows an RSA characterizing response capability intent to address those needs. The examples shown are two or three selected elements of larger sets of needs and intents drawn from a case study at Lockheed Martin's Integrated Fighter Group (IFG) in Fort Worth, US-TX (Dove, Schindel, and Garlington 2018).

Response requirement intents in Table 2 are traced to, and justified by, CURVE needs with parenthetical references. Some subsequent response feature specifications are traced and justified similarly to response requirements intents in Dove, Schindel, and Garlington, 2018, tying the features back to the requirements. I arranged Table 2 according to the RSA framework (Dove and LaBarge 2014).

It is important to note that I constructed the excerpted examples above after listening to people talk about what they were doing and why. I did not use the CURVE and RSA analysis frameworks explicitly as guides for process design. Nevertheless, the systems thinking that went into process design and evolution implicitly covered those same bases. Casting that thinking into the frameworks provides an explicit underpinning of rationale for the process requirements decisions that the engineers

Table 1. CURVE characterization of the process environment

Caprice: Unknowable event occurrences CC1: Urgent pre-emptive customer needs, sometimes called quick reaction notice events CC3: Project scope change	**Variation:** Knowable variables and ranges CV1: Multiple project-resource conflicts such as test facilities or key people CV4: Requirements of differing importance levels
Uncertainty: Randomness with unknowable probabilities CU1: Effectiveness of process CU4: Team-member engagement with agile approach	**Evolution:** Successive developments CE1: Open System Architecture (OSA) and Open Mission System (OMS) emphasis CE2: Customer mission needs
Risk: Randomness with knowable probabilities CR1: Cultural incompatibility CR2: Ability to keep and attract talent	

Table 2. Response situation analysis

Proactive response requirements	Reactive response requirements
▸ What must the process be creating or eliminating during its operational activity? RC3: Loading plans with spare capacity for unknowns/ inaccurate planning (CV1) RC5: Experience accumulation (CU1) **▸ What performance will the process be expected to improve during operational lifecycle?** RI3: Stakeholder, developer, and supplier alignment (CR1) RI5: Agility of existing integrated system (CU1, CE1) RI7: Effectiveness of distributed knowledge exchange (CU1, CR2) **▸ What major events coming down the road will require a change in the process infrastructure?** RM1: Evolution of customer missions (CE2) RM2: Cybersecurity and related standards (CC3) RM3: US Department of Defense (DoD) Open Missions approach (CE1) **▸ What modifications in employable resources might need to be made as the process is used?** RA1: Personnel that make up a team (CV1, CR2, CV4) RA5 Reallocation of work between prime contractor and other entities (CC1, CV1)	**▸ What can go wrong that will need a systemic detection and response?** RW2: Non-detection of variances (CU4. CV1) RW3: Insufficient identification and management of opportunities and risks (CR1) **▸ What process variables will need accommodation?** RV1: Process self-improvement and policing (CU1, CU4) RV3: Organizational acceptance and adoption of process (CU4, CR1) **▸ What elastic-capacity will be needed on resources/ output/activity/other?** RE1: System test capacity (CV1) RE2: Development capacity band to avoid disruption when work is more than expected in volume or difficulty (CC1, CC3, CV4) **▸ What types of resource relationship configurations will need changed during operation?** RR3: Priorities for requirements (CC3, CV1, CV4) RR4: Acquisition procedures/policies/ contract for situational and objectives reality (CC1, CE2)

made and implemented. As tools, thoughtfully populated frameworks can offer perpetual and traceable documentation for process evolution decisions and tradeoffs, even when the original thinkers are no longer involved. As a process environment evolves, and it will, the CURVE profile evolves. If engineers capture and update that profile, they can use it to drive reviews of response needs and response mechanisms for continued applicability and augmentation.

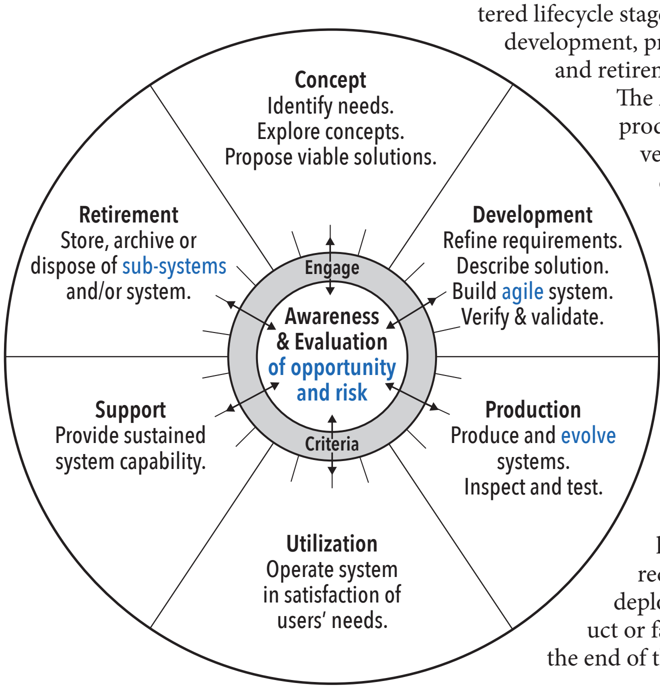

Figure 1. Purposes for each lifecycle model stage, adapted from (ISO/IEC 2016, p 17), with added awareness stage

AGILE SYSTEMS ENGINEERING LIFE CYCLE MODEL FRAMEWORK

The discussion of CURVE above used an example focused on the design of an agile systems engineering process specifically to accommodate the development of government contracted aircraft weapon systems. Systems engineering, however, encompasses additional lifecycle stages beyond system development and the conceptual stage that precedes development. ISO/IEC TS 24748-1:2016 recognizes six commonly encountered lifecycle stages for systems: concept, development, production, utilization, and retirement (ISO/IEC 2016).

The ASELCM project has produced case studies of four very different agile systems engineering processes (Dove, Schindel, and Scrapper 2016; Dove, Schindel, and Hartney 2017; Dove and Schindel 2017; Dove, Schindel, and Garlington 2018), and produced a working paper of common emergent findings (Dove, Schindel 2018). One thing all cases have in common is the recognition that the initial deployment of a system product or family of products is not the end of the development activity; evolution continues throughout the full life.

Asynchronous and concurrent life-cycle stage and process activity, as shown in Figure 1, is a hallmark of effective agile systems engineering processes. CURVE analysis specific to each stage can provide value, not just for the stage of interest, but for the implications the needs of each stage has on the nature of the total systems engineering process.

ISO/IEC TR 24748-1:2010(E) recognizes six commonly encountered system lifecycle stages (2010). Figure 1 adds a seventh lifecycle stage, awareness, as a critically necessary element of an effective agile systems engineering lifecycle model, as discovered in the ASELCM project.

Counter to the implication that a progression through stages is sequentially expected, ISO/IEC TR 24748-1:2010(E) states clearly that asynchronous and simultaneous activity in any and all stages is within expectations: "…one can jump from a stage to one that does not immediately follow it or revert to a prior stage or stages that do not immediately precede it. … one applies, at any stage, the appropriate lifecycle processes, in whatever sequence is appropriate to the project, and repeatedly or recursively if appropriate" (ISO/IEC 2010).

That 2010 technical report has since been replaced by ISO/IEC TS 24748-1:2016, which doesn't use the same words but has many more statements supporting the same concept, perhaps best illustrated on page 22 by "An individual system life cycle is thus a complex system of processes that will normally possess concurrent, iterative, recursive and time dependent characteristics. These statements are true for both a system-of-interest and any enabling systems (ISO/IEC 2016)."

Unlike sequential-stage waterfall, agile systems engineering processes may be conducting activities in any and all of the stages concurrently without progressive sequencing. The agile lifecycle model framework features the addition of a central awareness seventh stage. The awareness stage attends to active external and internal situational awareness–and engages all of the other stages. A distinguishing feature of effective agile systems engineering, previously unrecognized explicitly for its fundamental driving role, is proactive awareness of situational opportunities and risks to process and product alike, throughout the systems engineering process and the target system's lifecycle.

This proactive awareness and subsequent mitigation activity breathes life into the agile systems engineering process, taking it beyond a repetitive execution of traditional development increments fulfilling a backlog of planned features.

The lifecycle model framework does not have fixed starting and ending points. It implies and accommodates perpetual evolution beyond initial delivery and requires that the product development stage produces an agile system-of-interest in order to support evolution.

The retirement stage recognizes that subsystems and older system versions are retired frequently, as the current system evolves. This has implications for maintenance, disposal, and reversion processes.

Fleshing out a generic agile systems engineering lifecycle model starts with default processes in each stage, per ISO/IEC TS 24748-1:2016, tailored and augmented for specific agile systems engineering differences generally noted earlier. This awaits the ASELCM project's final report, anticipated as a 2019 INCOSE publication. Adapting the generic model to a specific organization's process does well to tailor and augment the generic model as the organization's evolving document of record.

For a software system application example, see the (Dove, Schindel, and Kenney 2017) case study that generally runs around the circle sequentially every six months with the next evolution of the system-of-interest, while simultaneously and asynchronously invoking specific stage processes in production, support, utilization, and retirement for system versions in certification and operation. For a mixed-discipline example, see "Case Study: Agile Hardware/Firmware/Software Product Line Engineering at Rockwell Collins," which takes the product line engineering perspective rather than the satisfaction of a single project contract (Dove, Schindel, and Hartney 2017).

Lesson Learned from Practice

This theme issue contains articles on experience and knowledge gained from the actual practice of agile systems engineering methods. Generally, each article focuses on one or a few key areas within the broader context of agile systems engineering.

Managing Awareness in a CURVE-y World: Agile Systems Engineering for Autonomous Vehicles

Bill Schindel of ICTT Systems Sciences provides a case study of the central role awareness plays in agile systems engineering. Agility as a core capability of systems engineering offers a means to respond effectively in a CURVE-y environment; but as a capability it awaits employment and is of marginal value without vigilant and broad awareness of events that demand attention. This article describes the active awareness seen at SpaWar System Center Pacific in their program that develops innovative off-road unmanned vehicle technology.

A New Muscle Memory: Training Systems Engineers in the Agile Culture of Trust

Sharon Fairbairn of Raytheon relates the effects of working with two teams of systems engineers on the methods and benefits of trust when working in agile development teams. One team was transitioning a legacy multi-discipline ground-based radar defense program from a documentation-centric waterfall process, the other team worked on a supervisory control and data acquisition (SCADA) model driven software development project. Trust improved when systems engineers practiced behaviors related to four competencies: willingness to experiment, community building, effective knowledge sharing, and listening to others.

Restructuring Requirements Analysis Using Model-Based Systems Engineering and Agile Systems Engineering

Warren Smith and Lymari Castro explain a process they used when faced with the need for a seventy percent reduction in the time needed for requirements development on a US Department of Defense (DoD) program. Their process employed a highly effective approach to removing time in the requirements review cycle, synchronizing multiple teams, and increasing communications among exceedingly differing stakeholders. The logic, benefit, and acquisition compatibility of this approach makes one wonder why systems engineers have not widely adopted this process–perhaps because an appropriate audience has not had an opportunity to consider it before this writing.

Balancing Systems Engineering Rigor with Agile Software Development Flexibility

Glenn Tolentino and John Wood relate their experience in reconciling the cultural and procedural conflicts between software developers steeped in US DoD acquisition tradition and a younger influx of developers expecting the benefits of an agile development approach. An open collaboration identified the advantages and disadvantages of the two approaches and identified the areas of conflict. This guided a subsequent reconciliation that reduced or eliminated the conflicts with balanced approaches for documentation needs, design decision making, and change management.

Agile Dynamics at Scale

A team of five authors from The Mitre Corporation and one from MIT present the nature and early results of what they call a "management flight simulator," for exploring the adoption of a scaled agile systems engineering process and adjusting process parameters to see the effect on outcomes. The simulator is not everything they want as yet but has already had impact on actual programs. In one case, program managers revised their contractors' requirements to mandate the use of a dedicated system team. In another case, as a result of interacting with the simulator, decision makers on another government project chose to adopt continuous integration.

Overcoming the Challenges of Agile for Globally Distributed Industrial Research

A team of six authors from ABB and one from CII Group relate the lessons learned and successes of rolling out agile practices tailored for globally distributed, industrial research. Notably, software plays a small role in the mixed discipline research realities of control, electro-magnetics, materials, mechanics, power electronics, sensors, and switching. These agile research pilots have already demonstrated values in three targeted objectives: stronger engagement with business unit customers, faster handover of research results and prototypes, and improved transparency.

Using Agile Systems Engineering Workshops and Model-Based Systems Engineering to Drive Agile Development

Harry Koehnemann from Scaled Agile and Mark Coats from General Dynamics share experienced means for combining frequent collaborative planning workshops with model-based systems engineering to achieve systems engineering benefits. They center the article's context on Lean-Agile concepts in a SAFe-like development framework.

Synergy: Agile Systems Engineering and Product line Engineering at Rockwell Collins

As theme editor for this issue of *INSIGHT*, and chair of the Agile Systems and Systems Engineering Working Group, I close the article series with selected highlights from a case study of the Rockwell Collins process for evolving a product family of military radios. This case study shows that product line engineering and agile systems engineering are synergistic.

CONCLUSION

The definition of agile systems engineering is rooted in what it does, not how it does it. The how can be satisfied many ways, as chronicled in the four ASELCM case studies (Dove, Schindel, and Scrapper 2016; Dove, Schindel, and Hartney 2017; Dove and Schindel 2017; Dove, Schindel, and Garlington 2018). Discussed in this article, agile systems engineering responds effectively in CURVE environments, and operates asynchronously and potentially simultaneously in at least seven lifecycle stages. As to compatibility with defense acquisition policies, all four ASELCM case studies analyzed processes applied to defense projects, all employed agile systems engineering, three of the four deliver hardware as well as software, and all included preliminary and critical design reviews and other typical contract gates. The case studies show how, over time, program management and contracting offices participated in some adjustments to contract terms as they came to appreciate the benefits that could come from employing agile systems engineering practices.

The ASELCM framework, as depicted in Figure 1, is compatible with the ISO/IEC/IEEE 15288 standard and companion specifications, and the venerable V diagram. At heart, an effective agile systems engineering process is an organic complex system with many specialized processes working and reacting in mutually dependent concert.

This article gives a different way of viewing what is going on and a different way of appreciating how we generate value. "An individual system lifecycle [specifically for agile systems engineering] is thus a complex system of processes that will normally possess concurrent, iterative, recursive and time dependent characteristics" (ISO/IEC 2016: 22). ■

REFERENCES

- ASELCM Project. 2015. https://connect.incose.org/ProgramsProjects/aselcm/Pages/Home.aspx, alternatively at www.parshift.com/ASELCM/Home.html.
- Aven, T., and B. S. Krohn. 2014. "A New Perspective on How to Understand, Assess, and Manage Risk and the Unforeseen." *Reliability Engineering and System Safety* 121:1-10. www.sciencedirect.com/science/article/pii/S0951832013002159.
- Dove, R., and R. LaBarge. 2014. "Fundamentals of Agile Systems Engineering – Part 1." Paper presented at the International Council on Systems Engineering, International Symposium, Las Vegas, US-NV, 30 June-3 July. www.parshift.com/s/140630IS14-AgileSystemsEngineering-Part1.pdf.
- Dove, R., W. Schindel, and C. Scrapper. 2016. "Agile Systems Engineering Process Features Collective Culture, Consciousness, and Conscience at SSC Pacific Unmanned Systems Group." Paper presented at the International Council on Systems Engineering. Edinburgh, GB, 18-21 July. www.parshift.com/s/ASELCM-01SSCPac.pdf.
- Dove, R., W. Schindel, and R. Hartney. 2017. "Case Study: Agile Hardware/Firmware/Software Product Line Engineering at Rockwell Collins." Paper presented at the 11th Annual IEEE International Systems Conference. Montreal, CA, 24-27 April. www.parshift.com/s/ASELCM-02RC.pdf.
- Dove, R, and W. Schindel. 2017. "Case Study: Agile SE Process for Centralized SoS Sustainment at Northrop Grumman." Paper presented at the International Council on Systems Engineering. Adelaide, AU, 17-20 July. www.parshift.com/s/ASELCM-03NGC.pdf.
- Dove, R., and W. Schindel. 2018. "Agility in Systems Engineering – Findings from Recent Studies." Working paper, 14 April. www.parshift.com/s/ASELCM-AgilityInSE-RecentFindings.pdf.
- Dove, R., W. Schindel, and K. Garlington. 2018. "Case Study: Agile Systems Engineering at Lockheed Martin Aeronautics Integrated Fighter Group." Paper presented at the International Council on Systems Engineering, International Symposium, Washington, US-DC, 7-12 July. www.parshift.com/s/ASELCM-04LMC.pdf.
- Dove, R, W. Schindel, M. Kenney. 2017. "Case Study: Agile SE Process for Centralized SoS Sustainment at Northrop Grumman." Paper presented at the INCOSE International Symposium. Adelaide, AU, 17-20 July. www.parshift.com/s/ASELCM-03NGC.pdf.
- ISO/IEC (International Organization for Standardization/International Electrotechnical Commission). 2010. ISO/IEC TR 24748-1:2010(E). Systems and Software Engineering — Life Cycle Management — Part 1: Guide for Life Cycle Management. Geneva, CH: ISO/IEC.
- ISO/IEC (International Organization for Standardization/International Electrotechnical Commission). 2016. ISO/IEC TS 24748-1:2016. Systems and Software Engineering—Life Cycle Management—Part 1: Guidelines for Life Cycle Management. Geneva, CH: ISO/IEC.
- Klinke, A. and O. Renn. 2002. "A New Approach to Risk Evaluation and Management: Risk-Based, Precaution-Based, and Discourse-Based Strategies." *Risk Analysis*, 22(6). https://pdfs.semanticscholar.org/b053/8f279cc602d866a4ec71620a1b2141023188.pdf.
- Schindel, W. 2018. "Managing Awareness in a CURVE-y World: Agile Systems Engineering for Autonomous Vehicles." *INSIGHT* 21 (2).
- Knight, F. H. 1921. *Risk, Uncertainty and Profit*. Boston, US-MA: Houghton Mifflin.
- Weick, K. E., K. M. Sutcliffe, and D. Obstfeld. 1999. "Organizing for High Reliability: Processes of Collective Mindfulness." R. S. Sutton and B. M. Staw (eds), *Research in Organizational Behavior*, 1: 81-123. https://pdfs.semanticscholar.org/e6c8/abc864d527258ceae63e6b2d775cb9d311b1.pdf.

ABOUT THE AUTHOR

Rick Dove is an INCOSE fellow, and chairs the working groups for Agile Systems and Systems Engineering and for Systems Security Engineering. His is CEO of Paradigm Shift International and an adjunct professor at Stevens Institute of Technology, teaching graduate courses in basic and advanced agile systems and systems engineering architecture and ConOps. He leads the current INCOSE project on discovering agile systems engineering lifecycle model fundamentals.