Agility in Systems Engineering – Findings from Recent Studies

Rick Dove, Paradigm Shift International, dove@parshift.com

Bill Schindel, ICTT System Sciences, schindel@ictt.com

Working paper 14-April-2018 (with subsequent updates)

Abstract

This article is focused on five recent understandings that are crystalizing from INCOSE's Agile Systems Engineering Life Cycle Model (ASELCM) project. The ASELCM project analyzed effective agile systems engineering processes in 3-day structured analysis workshops hosted by organizations that want deeper understandings of what works, why it works, and how to apply those understandings across a broader base. Four case studies arising from these workshops have been published. This article cannot cover everything of interest being discovered, but will expose current thinking on five key findings: a CURVE problem-space characterization, an asynchronous/concurrent Life Cycle Model Framework, a set of SRE behavior principles, the encompassing ASELCM Pattern of three concurrent systems operating simultaneously in agile systems engineering, and general Agile SE Response Requirements.

Introduction

Waterfall based systems engineering processes are failing to meet schedule and cost constraints in their attempt to design, build and deploy *effective* systems. The requirements for an *effective* system continue to change during the engineering process; partly because requirements are insufficiently understood initially, but more so because the target environment and knowledge of it continues to evolve independently.

The increasing speed of change in a system's operational environment increases the need to accommodate change in the systems' engineering environment, and to accommodate change in a system's operational environment – if an effective system is to be deployed and sustained.

The expectation is that some sort of agile systems engineering is required. But what does that mean? Is there one such sort? Some would have us believe that the Agile (Software Development) Manifesto and the software development principles behind it are the answer. Some others go further down that software path, believing that branded software development practices, such as Scrum or SAFe, provide at least a framework appropriate for agile systems engineering. Akin perhaps to a single master recipe for all manner of dinner preparation, in any cuisine.

Codified systems engineering currently relies on the 15288 standard (ISO/IEC/IEEE 2015), the V diagram (Forsberg, Mooz 1991), defense acquisition policies (country specific), and companion guidance-for-practice in the INCOSE SE Handbook (INCOSE 2015). These have provided a somewhat unified *one-sort* of systems engineering.

Can agile systems engineering coexist compatibly with these current mainstays of understanding and guidance? Before this question can be answered 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 system engineering in the face of uncontrolled change.

Intent: Effective response to a systems engineering operational environment that is capricious, uncertain, risky, variable, and evolving.

The word *effective* means that a valued result from resource employment is obtained. 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 ROI. In the middle, responding to changes in the engineering operational environment should sustain innovative forward progress. The word *should* is used to indicate a necessary objective for an agile systems engineering process: value delivery.

Uncontrolled change encompasses uncontrollable change, but is not limited to that which can't be controlled, just what isn't controlled regardless of reason.

It is important to understand and respect the intimate dependence an agile systems engineering process has on the agility of the system it builds. Operating in an uncontrolled environment is necessarily about real-time awareness, learning, and employment of that learning. If employment of that learning is inhibited, the agility of the process is diminished. This requires a target system amenable to change and evolution as it is being developed, as well as after it is deployed.

Incremental and iterative development alone does not provide the necessary agility in the system of interest. Incremental implies successive feature development, iterative implies successive improvement cycles on a feature. Both of these techniques can be done with little or no effect on agility in the system of interest. They *can* contribute greatly to the agility of the system engineering process, if their purpose and outcome is learning applied to work in process.

This article is focused on five recent understandings that are crystalizing from INCOSE's Agile Systems Engineering Life Cycle Model (ASELCM) project (ASELCM various dates). The ASELCM project analyzed effective agile systems engineering processes in 3-day structured analysis workshops hosted by organizations that want deeper understandings of what works, why it works, and how to apply those understandings across a broader base. Four case studies arising from these workshops have been published. (Dove, Schindel, Scrapper 2016, Dove, Schindel 2017, Dove, Schindel, Hartney 2017, Dove, Schindel, Garlington 2018). This article cannot cover everything of interest being discovered, but will expose current thinking on five key findings: a CURVE problem-space characterization, an asynchronous/concurrent Life Cycle Model Framework, a set of SRE behavior principles, the encompassing ASELCM Pattern of three concurrent systems operating simultaneously in agile systems engineering, and general Agile SE Response Requirements.

CURVE Framework for Characterizing an Uncontrolled Environment

What follows is a heuristic framework for characterizing uncontrolled internal and external environmental forces that impact process and product as systems. This framework, referred to as CURVE, is useful for characterizing the problem space in which either system will exist, and for which the systems should have appropriate solution-space capability.

- Caprice: Unknowable situations. Unanticipated system-environment change.
- Uncertainty: Randomness with unknowable probabilities. Kinetic and potential forces present in the system.
- Risk: Randomness with knowable probabilities. Relevance of current system-dynamics understandings.
- Variation: Knowable variables and associated variance ranges. Temporal excursions on existing behavior attractors (a reference to complex system behavior trajectories).
- Evolution: Gradual successive developments. Experimentation and natural selection at work.

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. Most possible combinations will never be used. 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 it is not necessary to characterize the environment comprehensively or precisely – if enough characterization is present to cause preparation sufficient to accommodate that which is not itemized.

Fleshing out the uncontrolled problem space in the CURVE Framework is a necessary first step toward developing effective agile response requirements. Developing response requirements uses another heuristic framework, referred to as Response Situation Analysis, which will not be detailed here as it has been around for some time and is adequately covered in (Dove, LaBarge 2014).

The four ASELCM case studies provide the CURVE environment characterized by each workshop Host. Todd Embry, of Sandia National Labs, provides a simplified example (from an SE Master's course term project) for conceptual display here:

> The Agile Optical Sensor Group creates innovative optical sensor systems on a very short time scale. These systems are short runs of one to 10 units, must meet all customer's needs, work out-of-the-box, and last for one to three years under often adverse conditions. The Group is a fairly stable team of engineers with diverse and somewhat overlapping capabilities. Problem-space CURVE characterization:
>
> Caprice: unknowable situations.
> - (reactive) next customer, team member availability, new technology availability.
> - (proactive) new designs the team devises without customer direction.
>
> Uncertainty: randomness with unknowable probabilities.
> - (reactive) how soon needed, test facility availability, quick-parts vendors.
> - (proactive) available previous designs/materials/parts.
>
> Risk: randomness with knowable probabilities.
> - (reactive) outside vendor response time, timely access to test facility.
> - (proactive) materials on hand for new designs.
>
> Variation: knowable variables and associated variance ranges.
> - (reactive) Common of the Shelf (COTS) delivery times.
> - (proactive) alternative COTS sources.
>
> Evolution: gradual successive developments.
> - (reactive) new team members, adoption of more integrated modeling tools.

- (reactive and proactive) changes to primary/backup-engineer relationships.
- (proactive) improvements to methods & rules.

Agile Systems Engineering Life Cycle Model Framework

Asynchronous and concurrent life-cycle stage and process activity, as shown in Figure 1, is a hallmark of effective agile systems engineering processes.

ISO/IEC Systems and Software Engineering — Life Cycle Management — Part 1: Guide for Life Cycle Management (ISO/IEC 2010) recognizes six "commonly encountered" system life cycle stages. Figure 1 adds a seventh life-cycle stage, *Awareness*, as a critically necessary element of an effective agile systems engineering life cycle model, as discovered in the ASELCM project.

Counter to the implication that a progression through stages is sequentially expected, technical report (ISO/IEC 2010, p 32) stated 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 life cycle processes, in whatever sequence is appropriate to the project, and repeatedly or recursively if appropriate."

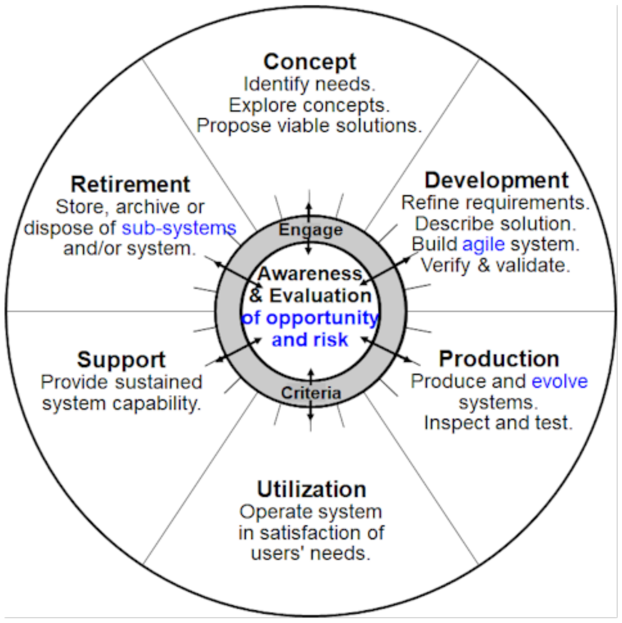
Concept
Identify needs.
Explore concepts.
Propose viable solutions.
Development
Refine requirements.
Describe solution.
Build agile system.
Verify & validate.
Production
Produce and evolve systems.
Inspect and test.
Utilization
Operate system in satisfaction of users' needs.
Support
Provide sustained system capability.
Retirement
Store, archive or dispose of sub-systems and/or system.
Engage
Awareness & Evaluation of opportunity and risk
Criteria

Figure 1. Purposes for each Life Cycle Model stage, adapted from (ISO/IEC 2016, p 17), with added Awareness stage.

That 2010 technical report has since been replaced by (ISO/IEC 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."

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 life cycle 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 life cycle.

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 life cycle 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 SE Life Cycle Model starts with default processes in each stage, per (ISO/IEC 2016), tailored and augmented for specific agile SE 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, 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 the (Dove, Schindel, Hartney 2017) case study viewed from the product line engineering perspective.

SRE Behavior Principles

Fundamentally the purpose of agility is survival in a CURVE environment. The architecture and structural principles that enable agility have been covered elsewhere (Dove, LaBarge 2014). Here the concern is with emerging understandings of behavioral principles that facilitate operational agility in action. Current thinking has three categories that encompass nine principles:

Sensing:
- External awareness (proactive alertness).
- Internal awareness (proactive alertness).
- Sense making (risk analysis, trade space analysis).

Responding:
- Decision making (timely, informed).
- Action making (invoke/configure process activity to address the situation).
- Action evaluation (Validation and Verification).

Evolving:
- Experimentation (variations on process ConOps).
- Evaluation (internal and external judgement).
- Memory (evolving culture, response capabilities, and process ConOps).

Sensing and Responding mirror John Boyd's OODA loop (Boyd nd), with Observe and Orient encompassed by Sensing, and with Decide and Act encompassed by Responding. The Evolving category wasn't overlooked by Boyd, but rather the ultimate purpose of his OODA loop. He valued the necessity to learn and improve the practice of OODA looping. It is instructive to understand that the accomplished OODA loop practitioner is not running through a sequence of four activities in incremental repetition, but rather engaged in all four activities simultaneously.

In Boyd's application of the OODA loop to fighter pilot dog-fight engagement, he recognized a human cognitive activity, and expected an increasing learned intuition. Putting this roughly in neuroscience terms, the brain is a pattern learner and recognizer that drives motor action. Increased learning experience drives these functions away from reasoning and closer to direct and immediate motor control. Action becomes systemically autonomic – the ultimate objective of Evolution in the SRE behavior framework.

Note the difference between Plan-Do-Check-Act (Deming nd) and OODA. PDCA has a sequential procedural feel, OODA has an in-the-moment dynamic-engagement feel. OODA is focused on awareness-driven re-evaluation of the changing problem space, rather than marching to convergence on a plan.

Explaining how things work has a reductionist quality to it. It can be off-putting to think that a new model needs to be learned and an old comfortable model abandoned. There is nothing new here. It is the natural way we navigate through life. It is, however, a new way of appreciating where the cul-de-sac of artificial, seemingly logical, approaches have taken us.

ASELCM Pattern of Three Concurrent Systems

Agile systems engineering encompasses three nested concurrent systems, depicted in Figure 2 as an iconic pattern (Schindel, Dove 2016).

The ASELCM Pattern establishes a set of system reference boundaries. Whether the systems of interest are small or large, human or inanimate, flying through the air or performing business processes.

This ASELCM Pattern particularly refers to three major system reference boundaries, and within those, six subsystem reference boundaries. These are all logical boundaries (defined by the behavior, not the identity, of systems), and are depicted by the iconic diagram in Figure 2.

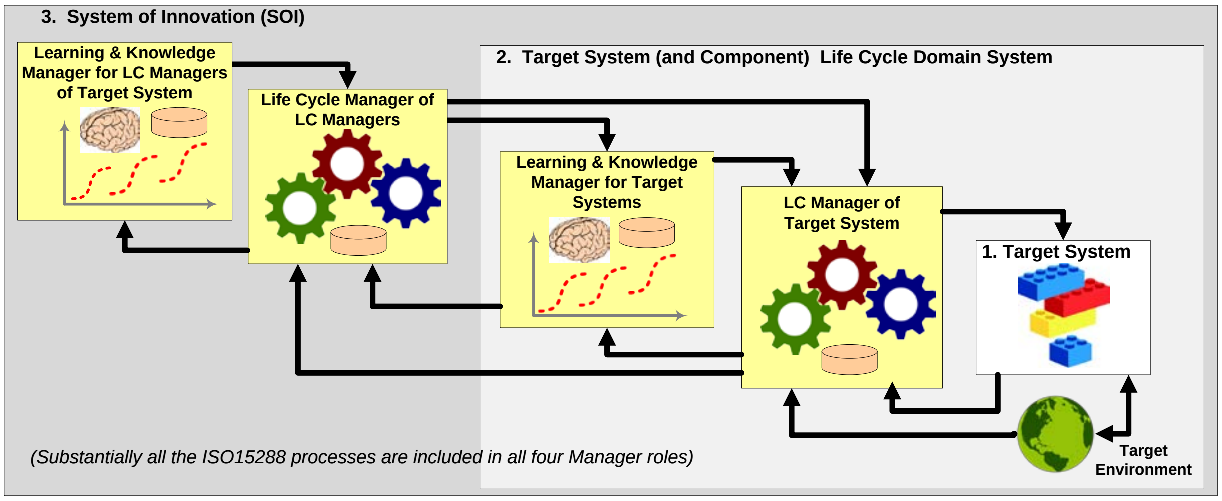
3. System of Innovation (SOI)
Learning & Knowledge Manager for LC Managers of Target System
Life Cycle Manager of LC Managers
2. Target System (and Component) Life Cycle Domain System
Learning & Knowledge Manager for Target Systems
LC Manager of Target System
1. Target System
Target Environment
(Substantially all the ISO15288 processes are included in all four Manager roles)

Fig. 2. Iconic view of the ASELCM Pattern reference boundaries (Schindel, Dove 2016).

- System 1: The Target System, the subject of innovation over managed life cycles of development, deployment, and support.
- System 2: The Target System Life Cycle Domain System, including the entire external environment of the Target System—everything with which it directly interacts, particularly its operational environment and all systems that manage the life cycle of the Target System. This includes the external environment of the operational target system(s), as well as all the (agile or other) development, production, deployment, support, security, accounting, performance, and configuration management systems that manage System 1.
- System 3: The System of Innovation, which includes System 1 and 2 along with the systems managing (improving, deploying, supporting) the life cycle of System 2. This includes the systems that define, observe, analyze (as in agile software process retrospective), improve and support processes of development, deployment, service, or other managers of System 1.

System 1 is contained in System 2, which is contained in System 3. All are (or at least should be) happening simultaneously, effectively an organic complex system motivated by self-preservation to evolve suitably in an uncontrolled operational environment. Think of the arrow-pointed pipes of Figure 2 as a circulatory system – not as channels for intermittent communication; but rather as pipes with constantly circulating information fluid. This circulatory system brings nourishing information and also provides regulation – policing the effectiveness, removing the dysfunctional, correcting the crippled.

General Agile SE Response Requirements

A framework for Response Situation Analyses that guides the identification of agile system and process response needs in eight different response domains has been employed effectively since the mid-nineties (Dove, LaBarge 2014). From ASELCM case study reflection, this framework is used to identify general agile SE process response needs as follows:

Domain		General Response Requirements
Proactive	Creation & Elimination	What will the process be creating or eliminating in the course of its operational activity? • Opportunity and risk awareness/knowledge • Acculturated memory • Response actions/options • Decisions to act
	Improvement	What performance will the process be expected to improve during its operational life cycle? • Awareness/Sensing • Effectiveness of response actions/options • Memory in acculturation, inventoried response options, and ConOps
	Migration	What major events coming down the road will require a change in the process infrastructure? • New fundamentally-different types of opportunities and risks
	Modification (add/sub capability)	What modifications in resources-employed might need made as the system is used? • Response action appropriate for specific response need • Personnel appropriate and available for a response action

R e a c t i v e	Correction	What can go wrong that will need a systemic detection and response? • Insufficient/inadequate awareness • Wrong decisions • Ineffective response actions/options
	Variation	What process variables will need accommodation? • Effectiveness of response actions/options • Effectiveness of response evaluation
	Expansion & Contraction	What elastic-capacity ranges will be needed on resources/output/activity/other? • Capacity to handle 1-? critical response actions simultaneously
	Reconfiguration	What types of resource relationship configurations will need changed during operation? • Elements of a response action • Response managers/engineers

Summary and Conclusion

This article has not been shy in offering assertions short of unequivocal justification. Supporting justification exists, but awaits the ASELCM project's final report with very many more pages, and a few Journal articles. This article is written as an initial exercise to articulate and touch on some of the key concepts emerging from the ASELCM project.

The definition of agile systems engineering is rooted in what it does, not how it does it. The how can be satisfied many ways, some already chronicled in the four ASELCM case studies to date. As discussed in this article, agile systems engineering responds effectively in CURVE environments, operates asynchronously and potentially simultaneously in at least seven life cycle stages, appears to behave according to nine operating principles, and melds target system, development system, and learning system into one interdependent system.

The Agile SE Life Cycle Model Framework, as currently depicted, 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 – of particular interest when systems engineering agility encompasses multidiscipline projects.

The ASELCM Pattern accommodates both V-diagram practices and 15288 processes throughout, as well as the guidance-for-practice in the INCOSE SE Handbook. This article gives a different way of viewing what is going on and a different way of appreciating how value is generated. "An individual system life cycle [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).

As to compatibility with defense acquisition policies: all four case studies are defense projects, all employ agile systems engineering, three of the four deliver hardware as well as software, and all include 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 be realized.

So yes to the question posed earlier, the current ASELCM project findings can exist compatibly with the current mainstays of systems engineering understanding and guidance.

The ASELCM project is now entering Phase 2. It would like to continue the method of on-site workshop analysis and employment recommendations of the findings. Host sites are being sought. The value proposition for the Host is rooted in new fundamental understandings from structured analysis of an existing process, and from workshop application of these understandings to vexing open issues. Contact the authors to inquire about Hosting.

Acronyms

ASELCM	Agile Systems Engineering Life Cycle Model
ConOps	Concept of Operations
COTS	Common Off The Shelf
CURVE	Capriciousness, Uncertainty, Risk, Variation, Evolution
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronic Engineers
ISO	International Standards Organization
INCOSE	International Council on Systems Engineering
SRE	Sensing, Responding, Evolving
OODA	Observe, Orient, Decide, Act
PDCA	Plan, Do, Check, Act
ROI	Return on Investment

SE	Systems Engineering

Acknowledgements

The ASELCM project leadership team consists of Rick Dove, Kevin Forsberg, Jack Ring, Garry Roedler, and William (Bill) Schindel. INCOSE has provided travel expenses for some members of the leadership team to participate in ASELCM workshops.

References

ASELCM Project. various dates. Documents at https://connect.incose.org/ProgramsProjects/aselcm/Pages/Home.aspx, alternatively at www.parshift.com/ASELCM/Home.html

Boyd, J. nd. OODA Loop, https://en.wikipedia.org/wiki/OODA_loop

Deming, W. E. nd. Plan-Do-Check-Act, https://en.wikipedia.org/wiki/PDCA

Dove, R., W. Schindel, 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, 18-21 July. www.parshift.com/s/ASELCM-01SSCPac.pdf

Dove, R., W. Schindel, 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, 24-27 April. www.parshift.com/s/ASELCM-02RC.pdf

Dove, R, W. Schindel. 2017. Case study: agile SE process for centralized SoS sustainment at Northrop Grumman. Proceedings International Symposium. International Council on Systems Engineering. Adelaide, Australia, 17-20 July. www.parshift.com/s/ASELCM-03NGC.pdf

Dove, R., W. Schindel, K. Garlington. 2018. Case Study: Agile Systems Engineering at Lockheed Martin Aeronautics Integrated Fighter Group. Proceedings International Symposium. International Council on Systems Engineering. Washington, DC, 7-12 July. www.parshift.com/s/ASELCM-04LMC.pdf

Dove, R., R. LaBarge. 2014. Fundamentals of Agile Systems Engineering – Part 1 and Part 2. International Council on Systems Engineering, International Symposium, Las Vegas, NV, 30Jun-3Jul. www.parshift.com/s/140630IS14-AgileSystemsEngineering-Part1&2.pdf

Forsberg, K., H. Mooz. 1991. The Relationship of Systems Engineering to the Project Cycle. Presented at the joint conference sponsored by: National Council On Systems Engineering (NCOSE) and American Society for Engineering Management (ASEM). Chattanooga, TN, 21–23 October. www.damiantgordon.com/Videos/ProgrammingAndAlgorithms/Papers/The Relationship of System Engineering to the Project Cycle.pdf

INCOSE. 2015. INCOSE Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities, 4th Edition. Editors: D. Walden, G. Roedler, K. Forsberg, R. Hamelin, T. Shortel. John Wiley and Sons. July.

ISO/IEC. 2010. Systems and Software Engineering — Life Cycle Management — Part 1: Guide for Life Cycle Management. ISO/IEC TR 24748-1:2010(E) first edition. Switzerland.

ISO/IEC. 2016. Systems and Software Engineering — Life Cycle Management — Part 1: Guidelines for Life Cycle Management. ISO/IEC TS 24748-1:2016.

ISO/IEC/IEEE. 2015. Systems and Software Engineering — System Life Cycle Processes. ISO/IEC/IEEE 15288:2015(E) first edition 2015-05-15. Switzerland.

Schindel, W., 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. www.parshift.com/s/160718IS16-IntroToTheAgileSystemsEngineeringLifeCycleMBSEPattern.pdf

Biographies

Rick Dove is CEO of Paradigm Shift International, specializing in agile systems research, engineering, and project management; and an adjunct professor at Stevens Institute of Technology teaching graduate courses in agile and self-organizing systems. He chairs the INCOSE working groups for Agile Systems and Systems Engineering, and for Systems Security Engineering, and is the leader of the INCOSE Agile Systems Engineering Life Cycle Model project. He is an INCOSE Fellow, and author of *Response Ability, the Language, Structure, and Culture of the Agile Enterprise*.

Bill Schindel is president of ICTT System Sciences. His engineering career began in mil/aero systems with IBM Federal Systems, included faculty service at Rose-Hulman Institute of Technology, and founding of three systems enterprises. Bill co-led a project on Systems of Innovation in the INCOSE System Science Working Group, co-leads the Patterns Working Group, and is a member of the lead team of the INCOSE Agile Systems Engineering Life Cycle project.