26th Annual INCOSE International Symposium (IS 2016)
Edinburg, Scotland, UK, July 18-21, 2016

Introduction to the Agile Systems Engineering Life Cycle MBSE Pattern

Bill Schindel
ICTT System Sciences
schindel@ictt.com

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

Abstract. Engineered and other systems are under pressure to adapt, from opportunities or competition, predators, changing environment, and physical or cyberattack. Ability to adapt well enough as conditions change, especially in presence of uncertainty, is valued. Systems (including developmental and life cycle management) that adapt well enough, in time, cost, and effectiveness, are sometimes called “agile”. As environmental change or uncertainty increase, agility can mean survival. Agile systems and agile systems engineering are subjects of an INCOSE 2015-16 discovery project, described elsewhere. This paper introduces the underlying MBSE-based Agile Systems Engineering Life Cycle Pattern being used to capture, analyze, and communicate key aspects of systems being studied. More than an ontology, this model helps us understand necessary and sufficient conditions for agility, different approaches to it, and underlying relationships, performance couplings, and principles. This paper introduces the framework, while specific findings about methods and practicing enterprises studied will be reported separately.

Agile Systems, Agile Systems Engineering, and Why They Matter

Agile systems-engineering and agile-systems engineering are two different concepts (Haberfellner and de Weck 2005), but both are designed for change. They can be augmented with new functional capability. They can be restructured with different internal relationships among their subsystems. They can be scaled up or down for economic delivery of functional capability. They can be reshaped to regain compatibility or synergy with an environment that has changed shape. These types of changes are structural in nature, and require an architecture that accommodates structural change.

Agility is the ability of a system or process to thrive in a dynamic operating environment; facilitating the deployment of effective response to both opportunity and threat, within mission. A useful framework (Dove and LaBarge 2014) for characterizing the nature of a targeted operating environment considers five categories of potential dynamics. An agile system or process is appropriate when a UURVE environment is anticipated:

- Unpredictability: randomness among unknowable possibilities.
- Uncertainty: randomness among known possibilities with unknowable probabilities.
- Risk: randomness among known possibilities with knowable probabilities.
- Variation: randomness among knowable variables and knowable variance ranges.
- Evolution: gradual (relatively) successive environmental developments.

Central to agile systems-engineering is continuous learning and subsequent modification of work in process, as new knowledge is revealed. Consequently it is counterproductive to employ an agile systems-engineering process that doesn’t design and develop an agile system to facilitate time- and cost-affordable modification of work in process. Figure 1 depicts this interaction of agile systems-engineering and engineered agile-systems. The family of agile software development practices frequently obtain this necessary relationship by employing an object-oriented development platform. Although hardware development practices have not yet embraced an equivalent change-enabling development platform, related opportunities are demonstrated in rapid additive mechanical prototyping (3D printing), electronic FPGA methods, and variations of so-called live-virtual-constructive (LVC) platforms that can intermix human, simulated, and instantiated system components.

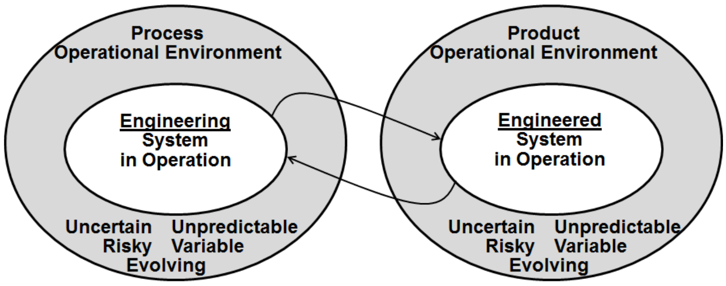

Figure 1: Two different operational environments defining necessary agile counterpoint for the systems they encompass.

The INCOSE Agile Systems Engineering Life Cycle Model Discovery Project

An INCOSE project-in-process is analyzing and building case studies of agile systems engineering in a variety of systems engineering applications, collectively covering agile software, firmware, and hardware systems engineering processes in experienced practice. The objective of the project is to discover and justify process principles as repetitively employed patterns, which are necessary and sufficient for any system engineering process that must contend effectively with an unpredictable, uncertain, and evolving engineering environment; and to document case studies that show how those principles are employed in the context of different engineering process environments.

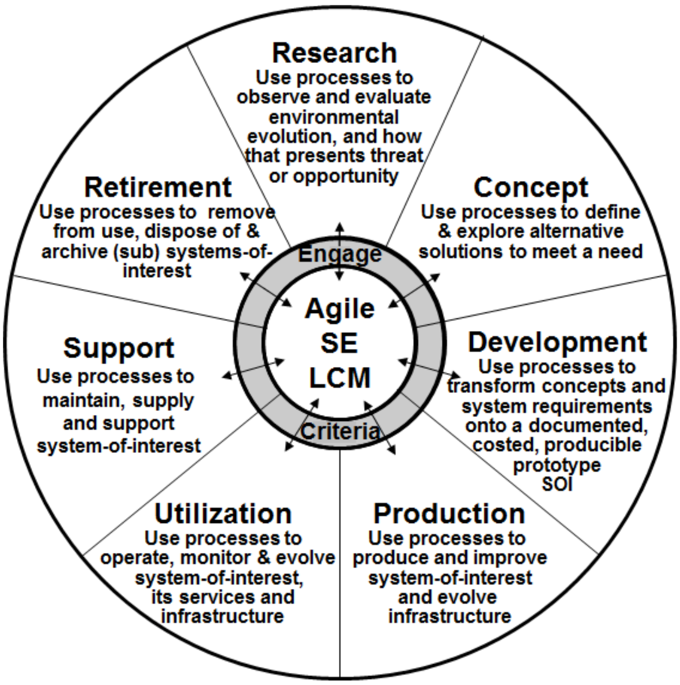

Figure 2: Seven asynchronously-invoked stages can be engaged repetitively and simultaneously

The project is analyzing systems engineering process activities as outlined in ISO/IEC 15288 System Life Cycle Processes. Figure 2 represents the life cycle model framework employed in agile systems engineering practices, and is consistent with ISO/IEC TR 24748-1 Guide for life cycle management: "…one applies, at any stage, the appropriate life cycle processes, in whatever sequence is appropriate to the project, and repeatedly or recursively if appropriate. While this may seem to be a total lack of structure, indeed it is not. Rather the structure has well defined parts that can be juxtaposed as needed to get the job done, flexibly but still in a disciplined manner, just as a real structure would be created."

The Agile Architecture Pattern

In (Dove and LaBarge 2014) the fundamental Agile Architecture Pattern (AAP) for agile systems of any kind employed the metaphor of an Erector/Meccano construction kit; showing modular components that can be used to compose and reconfigure a variety of target systems enabled by an interconnection and operation infrastructure. The focus in this paper, however, is on agile systems-engineering rather than agile-systems. The same AAP prevails in both cases, but here we employ the more apt example of the game of American football, as depicted in Figure 3.

The AAP graphic depiction typically shows a variety of assembled system configurations needed for different system-response situations. In the football example, each play (aka down) is an iteration toward a completed effort. Every play/iteration is a learning experience that is expected to inform strategy, resource configuration, and action for subsequent iterations. The customer, of course, is ever-present to feed back performance evaluation against expectations. Offensive plays are proactive and defensive plays are reactive. A series of offensive plays are intended to result in achieving the goal, but are often interspersed with needs for defensive plays. There are people with a variety of responsibilities, other than players, that are necessary to achieve the goal.

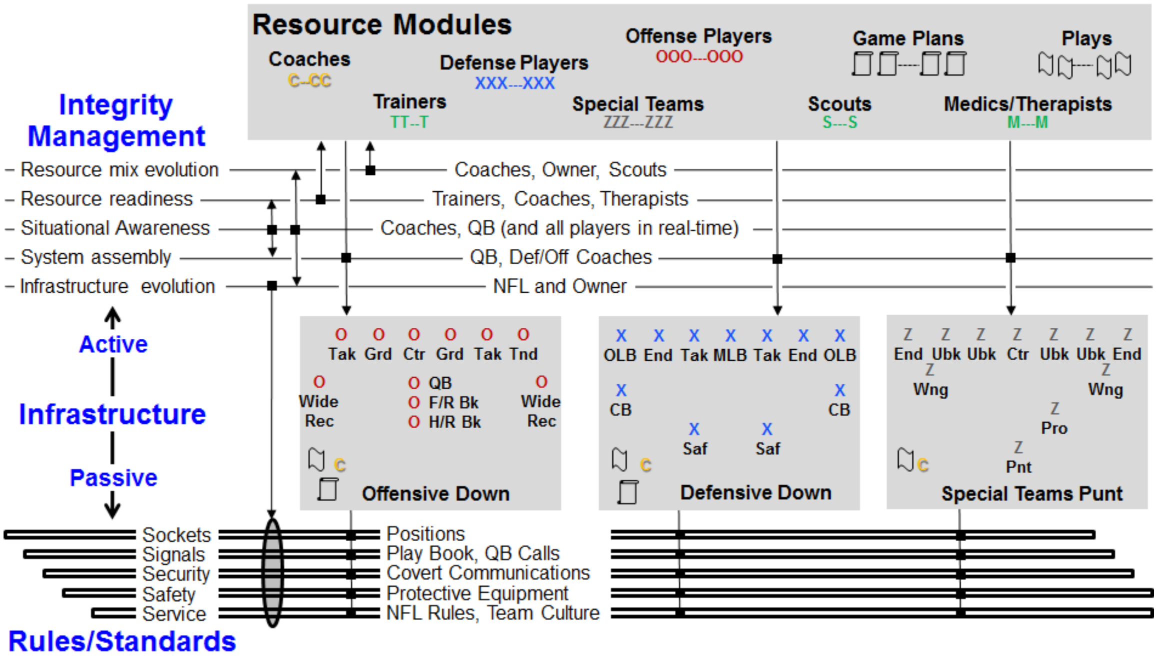

Figure 3: American football depicted as an agile-process architecture pattern, structured to pursue goals in an unpredictable, uncertain, evolving environment.

Agile Architecture Pattern Elements

There are three basic elements in the AAP: a roster of drag-and-drop encapsulated modules (resources), a passive infrastructure of minimal but sufficient rules and standards that enable and constrain plug-and-play interconnection, and an active infrastructure that designates five specific responsibilities for sustaining agile operational capability. The coverage here of these elements is necessarily brief, but explained further in (Dove and LaBarge 2014).

- Resources—Resources are self-contained encapsulated units complete with well-defined interfaces which conform to the plug-and-play passive infrastructure.

- Passive Infrastructure—The passive infrastructure provides drag-and-drop connectivity between resources.
- Active Infrastructure—An agile system is not something designed and deployed in a fixed event and then left alone. Agility is most active as new configurations are assembled in response to new requirements. In order for new configurations to be enabled when needed, five responsibilities are required:
    - Resource Mix Evolution—Who (or what process) is responsible for ensuring that existing modules are upgraded, new modules are added, and inadequate modules are removed, in time to satisfy response needs?
    - Resource Readiness—Who (or what process) is responsible for ensuring that sufficient modules are ready for deployment at unpredictable times?
    - Situational Awareness—Who (or what process) is responsible for monitoring, evaluating, and anticipating the operational environment in relationship to situational response capability.
    - System Assembly—Who (or what process) assembles new system configurations when new situations require something different in capability?
    - Infrastructure Evolution—Who (or what process) is responsible for evolving the passive and active infrastructures as new rules and standards become appropriate to enable next generation capability.

MBSE Formalization: Introduction to the ASELCM Pattern

This project includes creation of an MBSE representation of the configurable model (pattern) that formalizes the above metaphorically-described aspects as well as more extensive information, at multiple levels of detail, learned through the project's host site studies and analysis. That MBSE formalization is the ASELCM Pattern, introduced in sections below.

How Are Agile Systems Related (Or Not) to MBSE?

Agile Systems Engineering (ASE) and Model-Based Systems Engineering (MBSE) are not synonymous. How are they related or different? MBSE enters into this project in several ways:

ASE as a system. The current project is creating an MBSE description of ASE as a system, without assuming that ASE processes are themselves MBSE-based.

> **Definition**: A System is a set of interacting components. (Fig. 4) By "interact", we mean exchange of energy, force, mass, or information, resulting in component change of state. By "state", we mean the condition of a thing that influences its behavior in subsequent interactions.

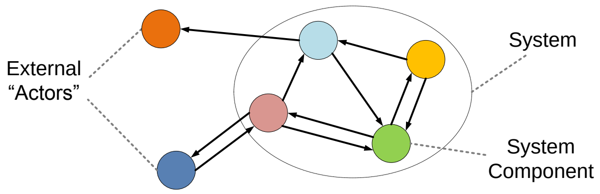

Figure 4: The System Perspective

We are using model-based representations to add clarity to the descriptions of ASE being uncovered by this project. Recognizing that there are multiple approaches to, and degrees of, agility, this reference is a configurable model (a formal pattern), that can be configured to represent different cases for examination. The resulting reference model is intended to be descriptive, not prescriptive, and should be usable to describe, from the point of view of its relative agility, any Systems Engineering / Life Cycle Management process, good, bad or otherwise. For targeted users of this model, it should help us answer questions about Agile Systems Engineering, such as: *What systems are involved? What are the nature, form, and behavior of those systems? Do we understand why they "work" or not? In what situations are various approaches to agile systems engineering of value (or not), and what are those values (capabilities)? What principles and practices support the ASELCM, and in what way? What quantitative relationships exist in this life cycle? How is this model informed by the current descriptions of ISO/IEC 15288 and current descriptions of various agile engineering practices? How does this model inform future versions of these standards and descriptions?*

ASE consumes and produces information. As shown in Figure 5, we know that information is a large part of what is produced and consumed by SE and other Life Cycle Management processes, whether agile or not. ASE asks us to re-think what information is really needed, and emphasizes discovery and validation of certain information by agile iterations. Systems engineers steeped in more traditional SE methods may understandably question the optionality of some traditionally valued SE information, discussed further in this project.

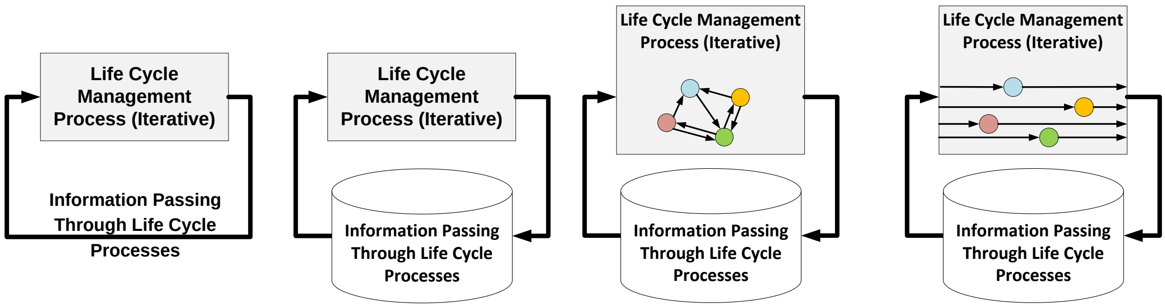

Figure 5: Engineering Produces and Consumes Information—Four Perspectives

Figure 5 provides four progressive views of information in the life cycle management system context. The progression from the first to the second view is to suggest that we will shift the historical emphasis, from mostly SE process description plus some SE information description, to a greater emphasis on the information itself, modeled in this study to understand its progression. The third and four views in this figure remind us that the Life Cycle Management Process (including its SE subset) is not just a simple set of interacting components—it is a complex network, with potentially highly varied interconnections and sequences, for which the underlying content of the information consumed and produced may be easier to understand than the structure of the processes. For these and other reasons, this project uses information models, not just process models, to examine the related questions about information in an ASE setting—but without necessarily assuming the use of MBSE in a given ASE.

Agile trajectories in target system configuration space. Have you ever experienced the following? Technical staff reports that all the procedural steps have been completed, but the resulting design is nevertheless found to be flawed. This common experience illustrates why an emphasis shift from process space to information space is important, and characteristic of ASE. Historically, in describing SE, more ink was always spent describing details of SE processes than the amount spent to describe details of the SE information those processes produce and consume. (The INCOSE SE Handbook [Walden et al 2015] and ISO15288 serve as examples.) It has been pointed out that this is at odds with the other engineering disciplines, in which information representing underlying phenomena is emphasized over procedure (Schindel 2015a). The "destination" that an ASE project seeks to reach is not the completion of a set of process steps, but reaching an acceptable location in the configuration space of its system of interest, to be discovered. As emphasized by Figure 6, ASE projects seek agile paths through subject system configuration space, but have not had a way to fully conceptualize this space until the arrival of strong enough system model foundations. Note the “information” in Figure 5 above describes points in the subject system configuration space emphasized in Figure 6.

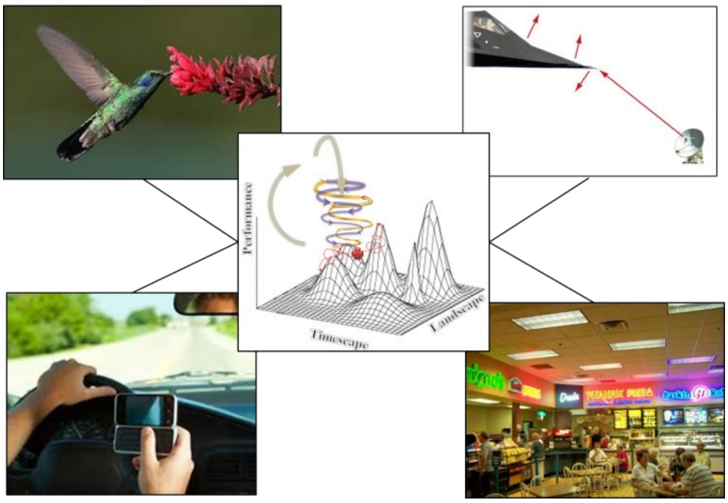

Figure 6: Systems Coevolve in System Configuration Space, Not Work Process Space

Although system configuration space sounds abstract, major progress occurred twice in the history of STEM from exactly such a formalization. The geometrization of algebra by Descartes and the geometrization of function space by Hilbert both made possible practical methods of modern engineering, discussed in (Schindel, 2015b). Recent anthropological studies similarly report an earlier shift in human conceptualization of geographic space, marked by movement from travel itineraries (analogous to SE process documents) to spatial maps (analogous to target system configuration space). (Schindel 2015a)

ASE emphasis on learning. ASE emphasizes learning in the presence of uncertainty and change. Some may be human learning by individuals and teams, but we know that the accumulation of experience in an organization can take a number of forms, symbolized in Figure 7. Accordingly, part of our project is examining how “information debt” can be managed to support sustained agility on a balanced basis, and this includes our investigation of MBSE-based Patterns in a role similar to their history in science—as repositories of accumulated (improving) knowledge about systems of interest. To the extent that we find that MBSE has a place within the practice of ASE, we know that MBSE processes can be made more agile by their retaining and efficiently re-using what we know as MBSE Patterns, thereby limiting MBSE overhead to new learning. Composable knowledge, like composable architecture, improves agility. Through learning, agile systems may accumulate formidable amounts of information, such as complex situation-based configuration rules, appearing intelligent, but in fact riding on a static accumulated base of learned knowledge:

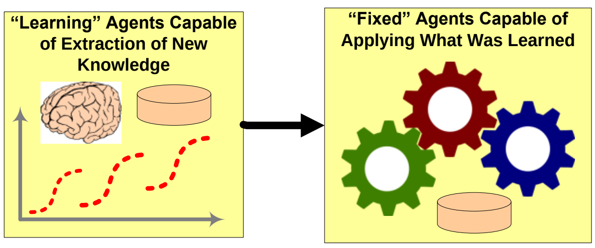

Figure 7: Experience Accumulates in Different Forms, for Later Uses

ASELCM Pattern Hierarchy and ASE Configurations. Because it is about system innovation and management of life cycles, the ASELCM Pattern inherits model content from several existing patterns (Fig. 8), thereafter configurable for different Agile SE approaches:

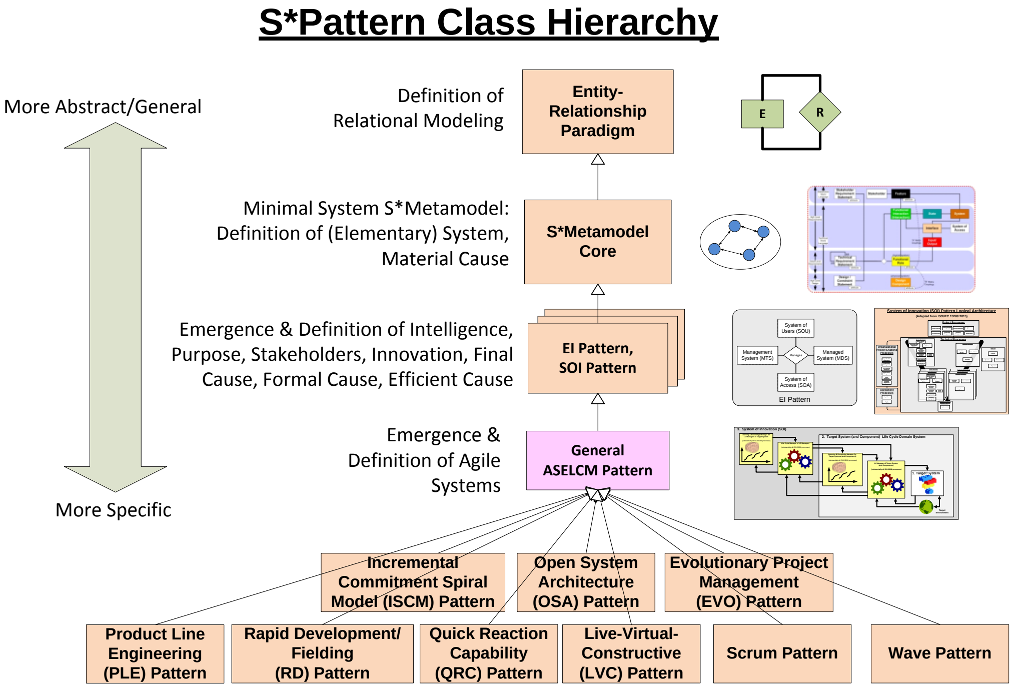

Figure 8: ASELCM Pattern inherits content from parent patterns, then is further specialized

At the top (most general) levels, Figure 8 shows that S*Metamodel, containing the minimum conceptual content necessary to model any system for purposes of engineering or science. This framework includes representation of elementary systems, emergence, and material cause, along with provisions for concepts (e.g., Stakeholder Value) that emerge later in Figure 8.

The ASELCM Domain Model: Key System Boundaries

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, all these start with the S*Model definition of System, depicted in Figure 4.

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 of Figure 9:

System 1: The **Target System (and Components):** (Refer to Figure 9.) The logical system of interest, which results from, or is subject to, innovation:

- Its behavior, characteristics, or performance are targets of the life cycle innovation (change, adaptation) process we'll introduce later.
- It is potentially agile. Assertion: for Systems Engineering to be fully agile, so must its target system also be agile—or else a competing SE system with an agile target system will out-perform it.
- Examples include aircraft, automobiles, telephones, satellites, the human immune system, software, restaurants, birds, and the health care delivery system.

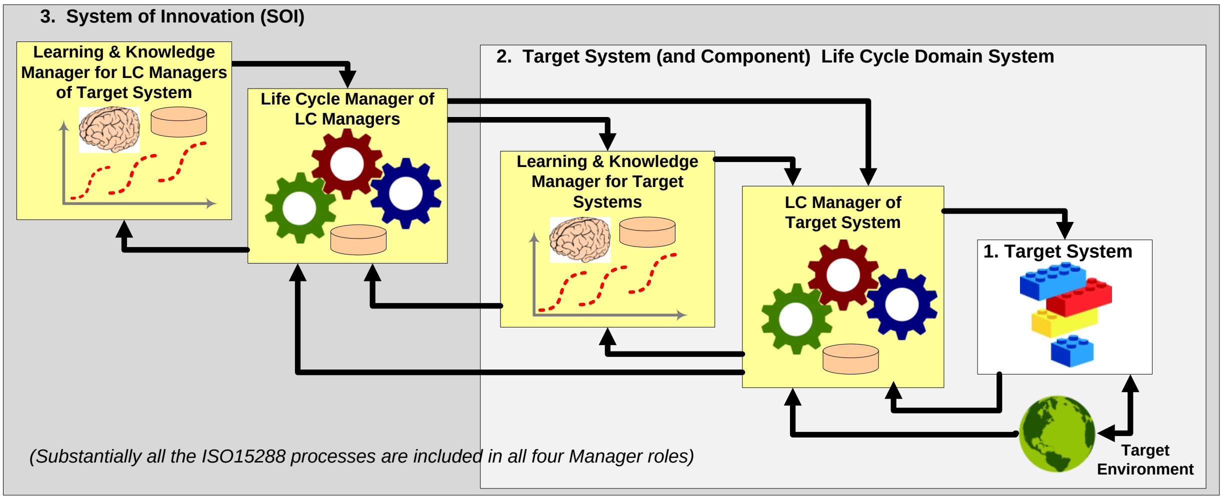

Figure 9: Iconic view of the ASELCM reference boundaries

System 2: The **<u>Target System (and Component) Life Cycle Domain System</u>:** (Refer to Figure 9.) The logical system within which the Target System will exist during its life cycle, when "in service" or otherwise. This domain includes *<u>all actors with which the Target System will directly interact any time during its life cycle</u>*: This includes (among others) any system that directly manages the life cycle of an instance of a Target System (or a Component)—development, production and integration systems, maintenance and operations systems, and others. The System 2 model (Figure 9) recognizes three sub-systems besides the Target System:

> <u>Target System Life Cycle Domain Actors</u>: All actors with which the Target System will directly interact during its life cycle—those in its operational domain (demanding agility) as well as all other direct actors. The next sub-system is a special case of those actors.
>
> <u>LC Manager of Target System</u>: Manages all life cycle aspects of the Target System, as recognized by ISO 15288. Note that this is more than just development or systems engineering—it includes manufacturing or acquisition, operations, maintenance, security, configuration management, and all the ISO 10040 System Management Functional Areas (SMFAs). However, it manages only "already known" aspects of System 1 and its Target Environment—it does not include responsibility of learning new things about them, which is allocated elsewhere.
>
> <u>Learning & Knowledge Manager for Target System (and Components)</u>: Responsible for learning new things about the Target System, its Components, and its Environment. This may include extraction of patterns or other knowledge from observations, planning experiments and extracting conclusions from their results, and other forms of learning. It also includes responsibility for accumulation and persistent memory of those learnings, and for providing the resulting knowledge for use by the LC Managers of the Target System.

Remember that these are logical (behavioral) roles. In realized physical systems, a single physical system may behave as both a Target System and a system that produces, modifies, reconfigures, or otherwise manages a Target System, by having roles from each allocated to it. For purposes of this logical roles description, they have been identified separately.

System 3: The **<u>System of Innovation</u>:** (Refer to Figure 9.) The logical system that includes System 1 and System 2, and that is additionally responsible for managing the life cycles of instances of any (System 2) Target System LC Manager. Recall that those System 2 Target

System LC Managers include Target System development, production, integration, maintenance, operations, and other management systems.

Why are the learning capabilities of System 2 and System 3 differentiated from other capabilities in System 2 & 3 models? Especially for understanding two aspects of Agility:

- We want to understand what capabilities (Figure 7, right side) can exist for “agile movement within what is already known”, for both System 2 and System 3, in nearly all of the ISO 15288 process areas, and . . .
- We also want to explicitly understand what is meant by “learning” (Figure 7, left side) in nearly all of the ISO 15288 process areas.

Learning includes observation, experimental discovery, questioning, noticing differences between what is observed and what is believed, and extracting patterns from instances.

The Formal Domain Model. Supporting the simple “iconic” diagram of Figure 9, there is a formal MBSE model that describes the ASELCM Pattern. A top level view of its formal domain model is shown in Figure 10. In this model, the disk icons represent the responsibility for persistent state information—whether by human memory, automated databases, or otherwise—concerning listed system descriptions of the following systems:

- The Target System, its Components, and its Environment
- The LC Managers of the Target System.

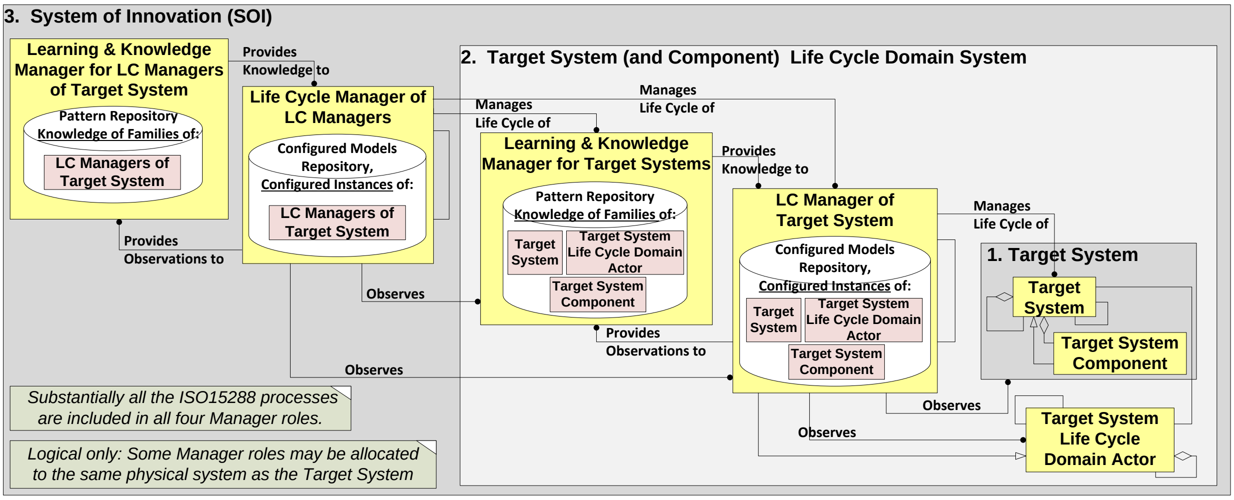

Figure 10: Top Level ASELCM Formal Domain Model

How ISO 15288 LC Management Processes Participate in the ASELCM Pattern

There are four subsystems of ASELCM Systems 2 and 3 that deal in the life cycle management of other systems. The (system life cycle management) processes of ISO 15288 are therefore within the scope of those systems. Likewise, the processes of ASE, even if believed distinct from the ISO 15288 processes, are also within the scope of those ASELCM Pattern systems. The (agile or not) equivalents of the ISO15288 Technical Processes (see Appendix III) appear four times in the ASELCM Pattern, as shown in Figure 11.

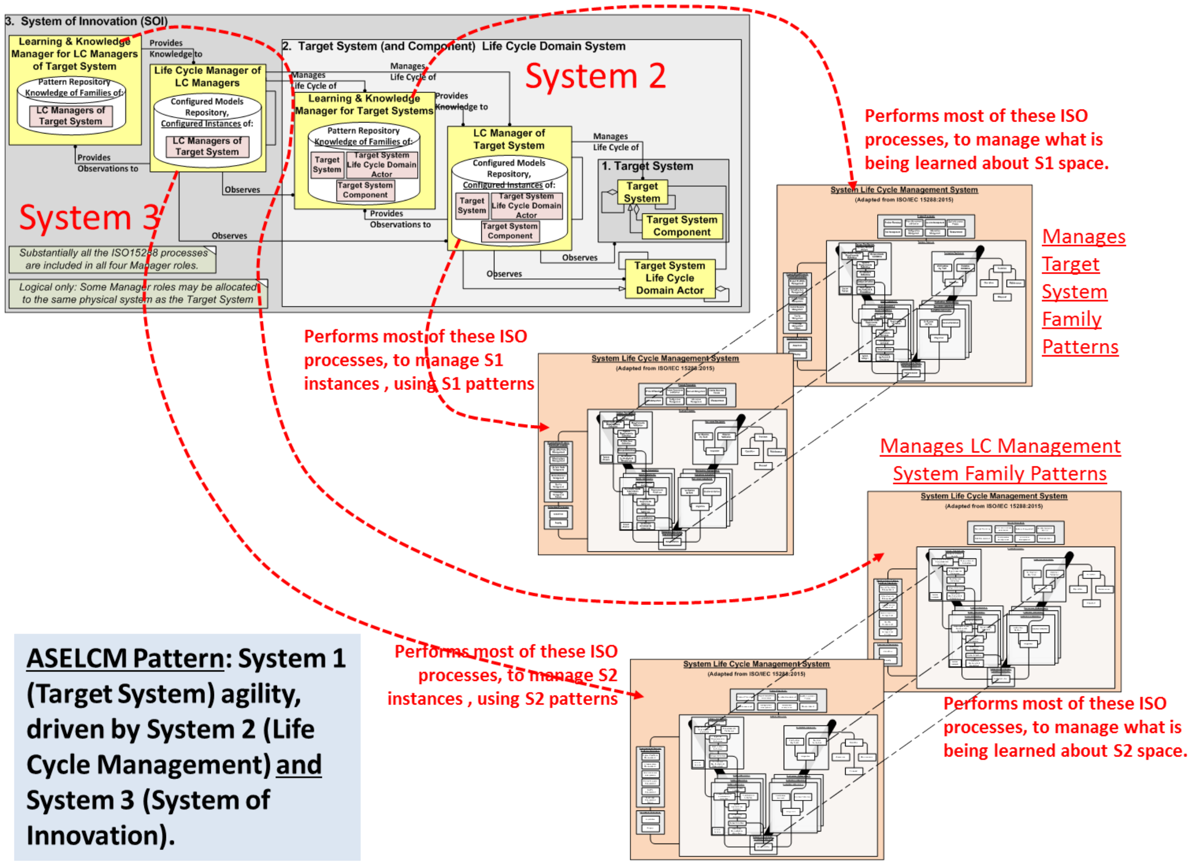

Figure 11: Four Instances of the LC Management Processes, Across the ASELCM Pattern

We believe that the ISO 15288 and ASE processes will be demonstrated by this project to be potentially alternate views of the same underlying reality, with ISO15288 traditionally emphasizing process management, and ASE processes typically emphasizing effective discovery, learning, and response in the presence of change and uncertainty, through rapid iteration. The configurability of the underlying pattern is driven by its Stakeholder Feature level optimization for different situations and different stakeholder objectives.

For example, Figure 18 of Appendix IV provides an alternate view of processes, configured in an Agile Scrum configuration, emphasizing agile sprint iteration cycles, and depicted in a form more likely to be recognized by the current Agile community.

Figure 16 is an informal representation of the ISO15288 processes, distributed across a "Vee" diagram easing its recognition by the traditional SE community, but depicted as potentially concurrent processes not meant to suggest any specific sequence or interdependence. Rather than suggest that the ISO 15288 processes are interconnected or sequenced in some single fixed way, Figure 16 illustrates them as a network of processes, interconnected only by the fact that all these processes produce and consume shared information about the target systems they manage. See also the fourth case of Figure 5 and (Braha and Bar-Yam 2007).

Agile Trajectories in Target System Space. Moving to a view of target system configuration space trajectories (Schindel 2015b) enables recasting the problem of managing the life cycle process as a problem of optimal control (of trajectory) in the presence of uncertainty, for which a large base of experience and literature exists. However, as noted in that reference, this requires a strong enough underlying metamodel to adequately represent the system configuration space for that purpose. To further illustrate that this is a realistic goal, it should be noted that ASE practice has already developed a less formal approach of that same nature, and at least one heuristic algorithm, that are informally in the spirit of such optimal control:

- The Backlog: A hallmark of ASE is establishment and management of "the backlog" of incremental tasks, from which next jobs are selected in such a fashion as to make the "best" progress in potentially shippable agile deliverables along an agile trajectory. (Leffingwell 2011)

- WSJF Algorithm: In the context of the backlog, ASE offers the Weighted Shortest Job First (WSJF) algorithm. This is an heuristic method by which a human strategist selects perceived highest value, shortest effort to deliver (note the potential conflict) as the next backlog task to perform. (Leffingwell 2011).

It is therefore reasonable to suggest that, in a configuration space that is strongly enough modeled, this may be further formalized as the optimal control problem noted. A key part of that space, described by the S*Metamodel (Appendix I), is the Stakeholder Feature subspace, which describes the fitness landscape of all stakeholder value, and which includes risk. That this can be cast as such a problem in dynamics is further supported by (Schindel, 2015c).

Agile Trajectories Include Learning

Agile trajectories in target system configuration space may occur without learning, when the trajectory is only applying what is already known to a current situation, as in managing the production, configuration, security, or operational performance of System 1. This in itself can still be very valuable to agile system stakeholders. However, agile trajectories can also extract new learning, as when System 2 extracts new information about either System 1 or its Environment. This can include, for example, discovery of requirements, characteristics of technologies, behavior of the environment, changes to any of those, or other new information.

Information Debt. This leads to a conundrum that might be perceived as a fault line between traditional SE and current ASE practice. In this project, we have called this the issue of "Information Debt", to contrast it to the "Technical Debt" more frequently referenced by ASE (Leffingwell 2011). Whereas Technical Debt refers to gaps in what is available to be delivered (System 1), we have used Information Debt to refer to gaps between what the (System 2) team has learned about System 1 versus information captured in a form suitable to sustain System 1 over its life cycle.

For example, ASE emphasizes team discovery and learning of what System 1 stakeholders want, and this it turned into potential deliverable System 1 solution content. But what else happens to what team members learned, other than it flowing into the delivered product? Some members of the traditional SE community may believe that they detect an ASE intention to not create a record of what was learned, as system technical documentation or otherwise, when they hear the Agile Software Manifesto statement of valuing "working software over comprehensive documentation". (Fowler and Highsmith 2001) Will a delivered System 1 that travels an extensive learning trajectory be sustainable over its subsequent life cycle?

Members of the traditional SE community might agree that there have been times when the traditional process failed because of emphasis on comprehensive documentation over testable product. Likewise, members of the ASE community might agree that complex, long-lived software that is undocumented except for its source code might at times present a support longevity problem. A concern in this project is to identify middle-ground principles in the

ASELCM Pattern that might be used to balance between these two extremes. This territory has been at least partially explored by others (Ambler 2015).

Initial study of this area has also suggested it may help to resolve the perceived gap between defense acquisition contractual review event policies and ASE approaches.

Emergence of Purpose. Another important learning case involves what has been termed the "emergence of purpose" (Schindel 2013). ASE should support a high level of innovation, and some learning can potentially include discovery of the existence stakeholders and purposes that lead to entirely new, unanticipated Target Systems. Learning is not just the extraction of patterns from signals. It also includes setting up conditions for those signals, such as being curious, asking challenging questions, designing experiments, constant observation, networking, and other skills. These "innovation competencies" are further catalogued in (Schindel, Peffers, et al 2011).

Additional ASELCM Pattern Content

The ASELCM Pattern as a whole contains more than the space of a single paper allows—Appendices II-IV and the References sample the additional scope of this S*Pattern:

- Stakeholder Features, Feature Attributes: System fitness (trade) space for S1, S2, S3
- Functional Interactions, Roles, Role Attributes: What actually happens
- States / Modes: Temporal aspects
- Attribute Couplings: Representing quantitative relationships, impacts, principles
- Configuration, Reconfiguration, and Adaptation

Initial Observations, Work to Follow, Participation

The purpose of this paper has been to introduce some of the high level structures and approach to the ASELCM MBSE Pattern. As the related project workshops continue and modeling work proceeds, observations are accumulating and will be the subject of separate papers and reports, such as (Dove, Scrapper, Schindel 2016). Some of the interesting observations already being pursued for such other reports include:

1. Risk allocation, sharing, and spreading techniques
2. Product Line Engineering (PLE) aspects
3. Impacts of transparency, awareness, and communications, by culture and technology
4. Information debt versus technical debt
5. Coupling ASE approaches with defense or other traditional acquisition contracting

This is an INCOSE community project, with heavy involvement from the Agile Working Group and the Patterns Working Group, and an open invitation to INCOSE members and enterprises to consider participating as discovery host sites or individual participants in 2016, by contacting the authors.

Appendix I: Applying the S*Metamodel to MBSE and PBSE

The ASELCM Pattern is being constructed as an S*Model. This means that it is an MBSE model which conforms to the underlying S*Metamodel, summarized in Figure 12 (Schindel 2005b, 2011), defining the minimal underlying concepts necessary to describe any system for engineering or scientific purposes. This does not prescribe a MBSE modeling language or modeling tools to use, and S*Models can be expressed using a number of contemporary modeling languages and toolsets. We expect to provide two such language-specific renderings of the ASELCM Pattern, for appeal to different groups.

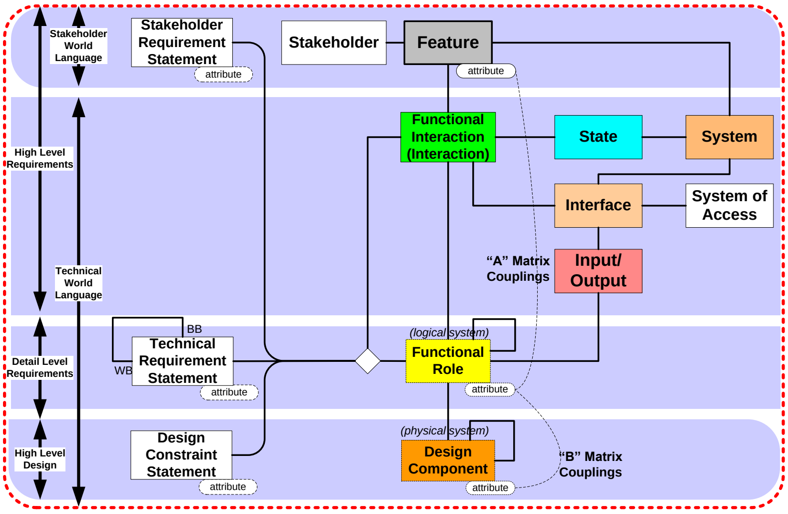

Figure 12: Summary Extract of S*Metamodel

An S*Pattern is an S*Model that describes a family (or platform or product line) of systems that can be configured differently for different situations, applications, or uses. Because we are describing a general reference model for Agile Systems Engineering Life Cycles, there are multiple possible configurations (Figure 13). This is further illustrated by Figure 8.

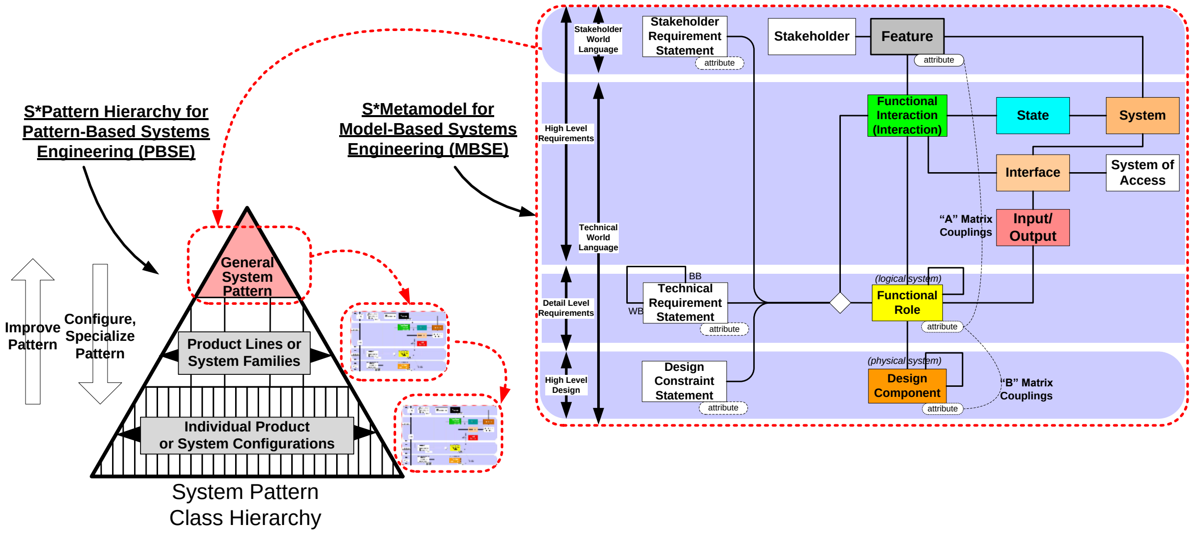

Figure 13: An S*Pattern is an S*Model that can be configured for different needs

Patterns Methodology has been summarized by the INCOSE Patterns Working Group, supporting this project (Schindel et al 2015; Schindel 2005a; Schindel and Peterson 2013).

Appendix II: Applying the Embedded Intelligence (EI) Pattern

Embedded Intelligence (EI) Pattern. The Embedded Intelligence (EI) Pattern (Peterson and Schindel 2014; Schindel and Smith 2002) contributes model-based representation of System Life Cycle Management across the System Management Functional Areas (SMFAs) of ISO 10404. This includes management of system Performance, Configuration, Security, Faults, and Accounting. Stakeholder Features associated with those capabilities describe what stakeholders value, summarized in Figure 14:

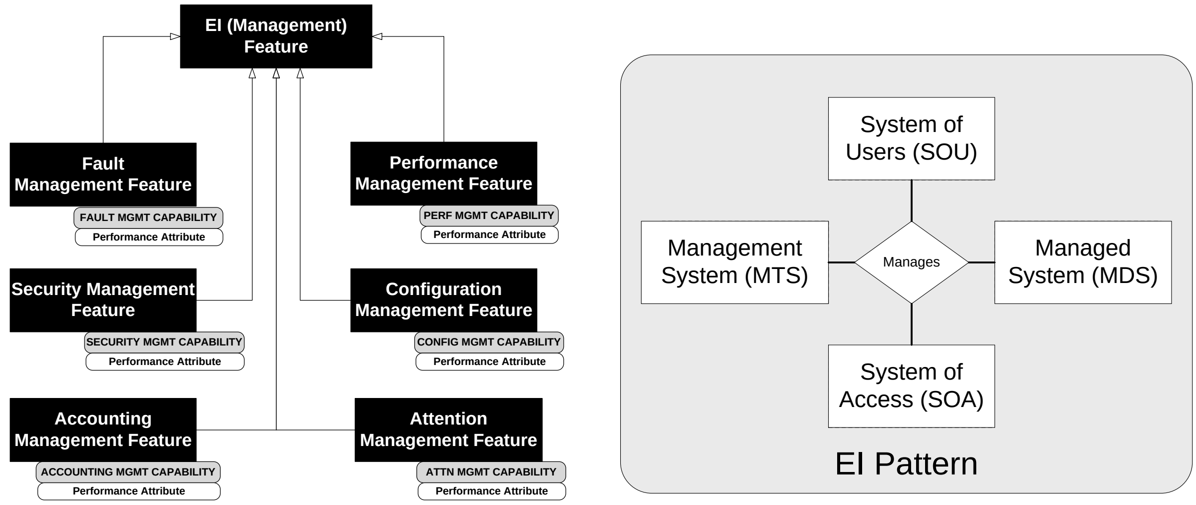

Figure 14: Major Stakeholder Features and Roles of the Embedded Intelligence (EI) Pattern

The EI Pattern describes a situation-based system regulatory framework for “handling” or “resolving” management situations in the five SMFAs, maintaining a form of dynamic system equilibrium as situations arise that demand resolution. The types of situations are reflected in the types of ISO SMFAs. Classical controls, for example, typically fall in the Performance Management SMFA. As these dynamic situations arise and are resolved, Situation Resolution States Models of the EI Pattern describe the rise and resolution of various conditions, such as:

- Major mission cycles, from mission start to completion
- Fault resolution cycles, including prevention
- Configuration change cycles, including adaptations
- Fulfillment of requests for services
- Security condition resolution cycles
- Other situation resolution cycles

Figure 15 illustrates the cyclic form of Situation Resolution State Model that the EI Pattern uses to represent these capabilities in time.

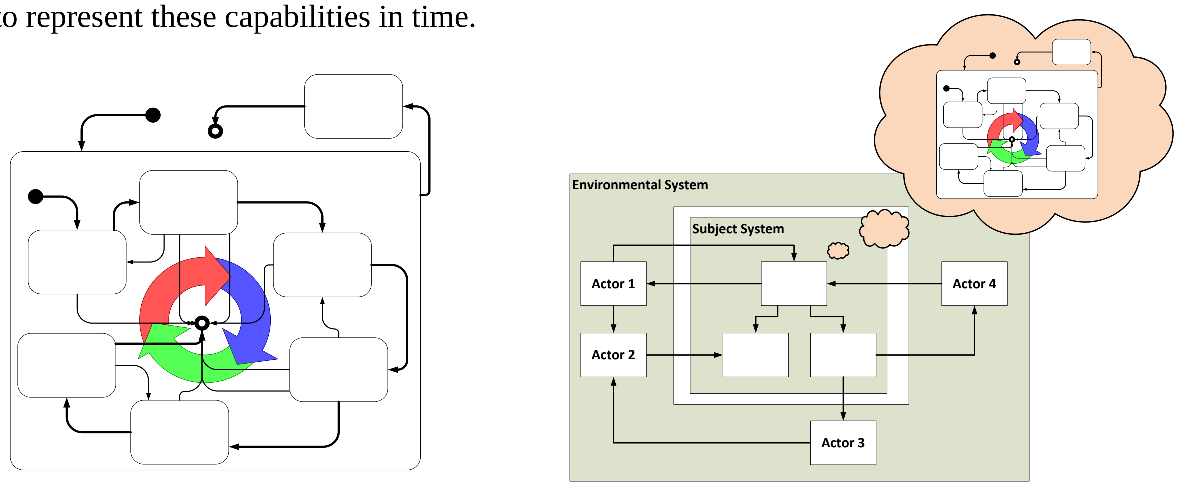

Figure 15: EI Management State Model

A backlog of situations of various priorities requiring management may arise, and this may at times exceed the time and resources capacity of the management system. For that reason, the EI Pattern includes an Attention Management Feature (Figure 14) that represents the required ability to queue and priority manage situations exceeding the resource (attention) capacity of the system.

Some EI systems may have to be informed of the occurrence of a situation requiring resolution. A system that is capable of not only traversing a situation resolution cycle, but also recognizing that a triggering situation has arisen in the first place is said to be "Situationally Aware":

> If a human operator control panel has a "mode switch", the system relies on the human to be aware of situations, launching the appropriate cycles
>
> More advanced systems recognize these situations autonomously, as in right side of Figure 15.

Appendix III: Applying the System of Innovation (SOI) Pattern

System of Innovation (SOI) Pattern: This pattern contributes model-based representation of the Life Cycle Management Processes of ISO 15288. At its top level, the Logical Architecture of this system pattern is summarized by Figure 16, where for recognition purposes the logical processes are spread across a "Vee" background that is not part of the formal model.

There are no inter-process architectural relationships shown in Figure 16, reflecting communication between processes, or any kind of sequence shown there for these potentially concurrent activities. This represents a complex network of nodes that interact with each other through produced/consumed information shown in the right side of Figure 5. Arranging these to provide "agility" is the goal of the ASELCM Pattern, which inherits this composable structure from the SOI Pattern, as shown in Figure 8.

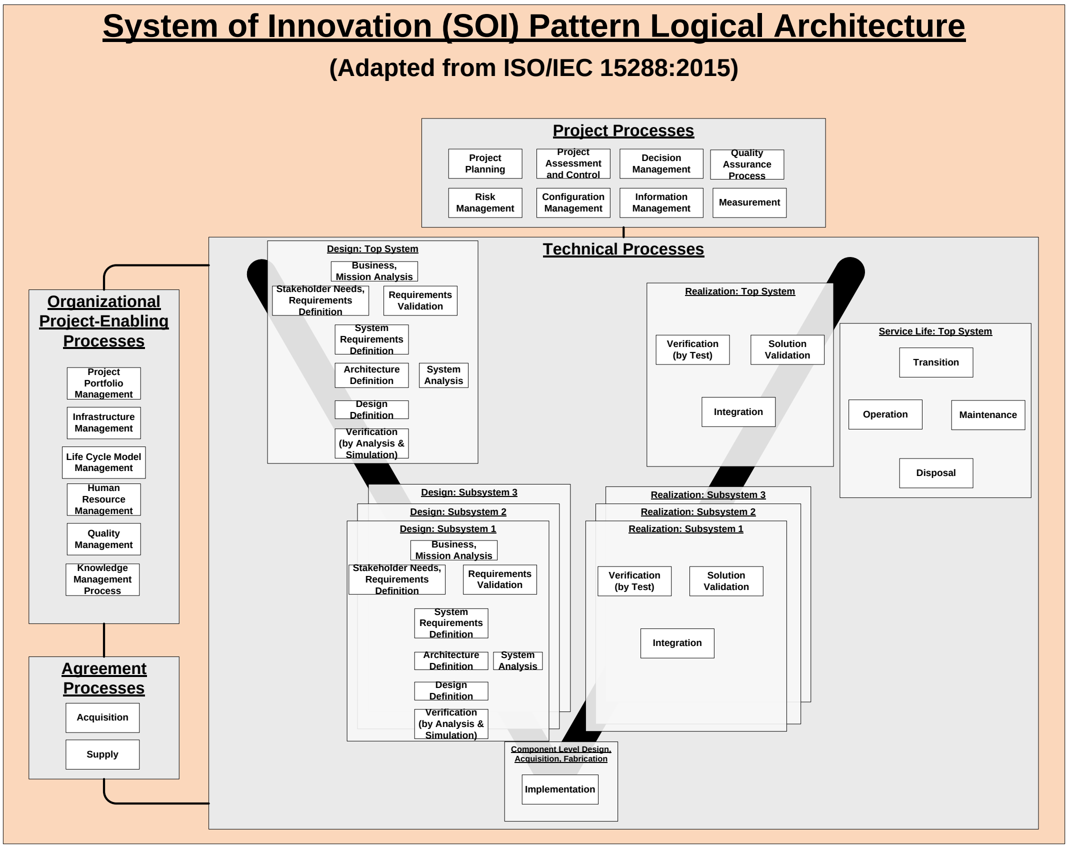

Figure 16: SOI Life Cycle Management Processes, Following ISO 15288

The related SOI Stakeholder Features, shown in Figure 17, represent capabilities of the ISO15288 process areas, further specialized in the ASELCM Pattern.

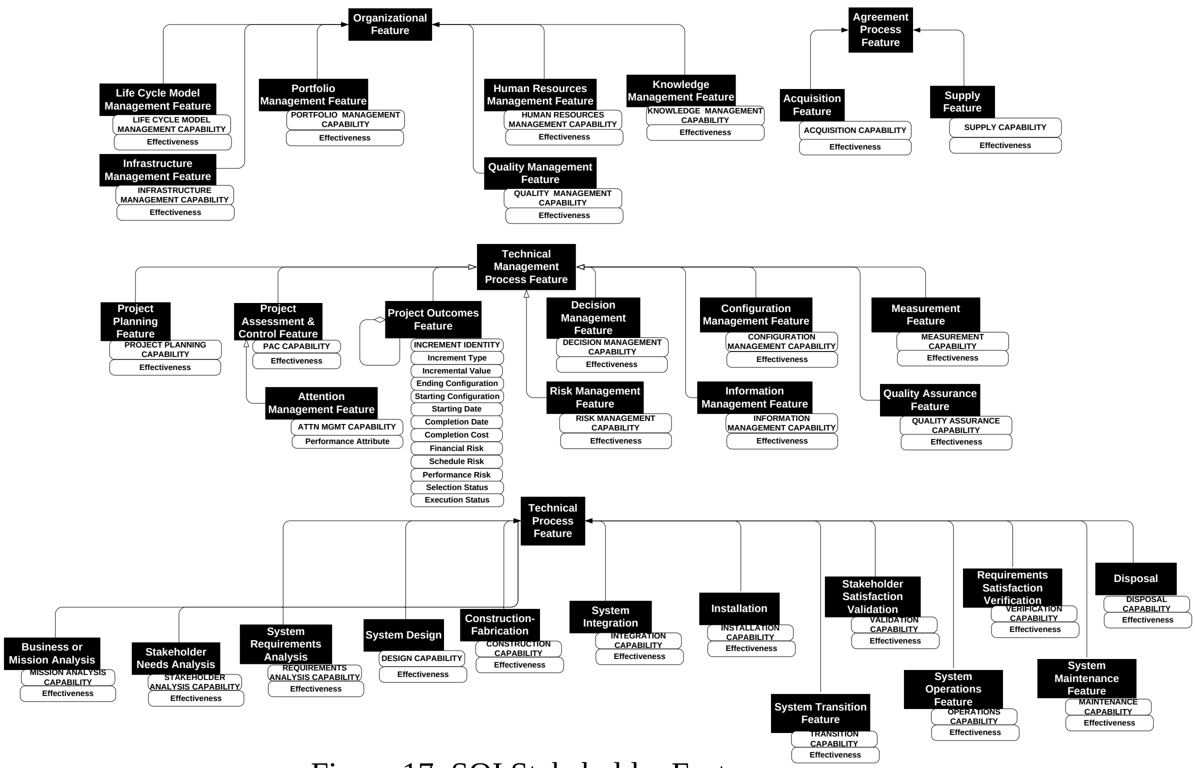

Figure 17: SOI Stakeholder Features

Appendix IV: MBSE Model of an Agile Scrum Iteration Process

Traditional Scrum Sprint Perspective
(Activity Diagram, with Swim Lane Roles)

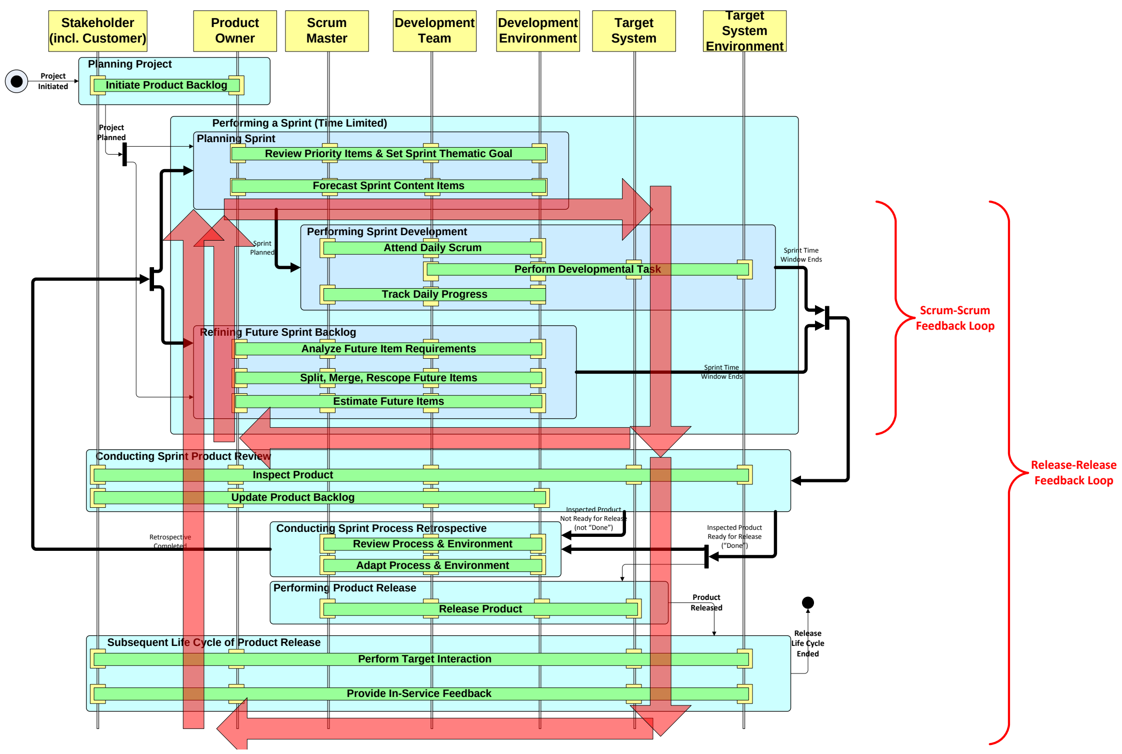

Figure 18: Agile Scrum Loops for Sprints and Releases

References

Ambler, S., “Agile/Lean Documentation: Strategies for Agile Software Development”, retrieved 2015 from http://agilemodeling.com/essays/agileDocumentation.htm

Beihoff, B., and Schindel, W., “Systems of Innovation I: Models of Their Health and Pathologies”, Proc. of INCOSE International Symposium, 2012.

Braha, D., Bar-Yam, Y., “The Statistical Mechanics of Complex Product Development: Empirical and Analytical Results”, *Management Science*, V53, No7, July2007, pp. 1127–1145.

Dove, R, and LaBarge, R., “Fundamentals of Agile Systems Engineering—Part 1” and “Part 2”, INCOSE IS2014, July, 2014.

Dove, R., INCOSE Agile Systems Engineering Life Cycle Model Discovery Project, project web site, loaded 2015 from: http://www.parshift.com/ASELCM/Home.html

Dove, R., Scrapper, C., and Schindel, W., “Agile Systems Engineering Process Features Collective Culture, Consciousness, and Conscience at SSC Pacific Unmanned Systems Group”, to appear in *Proceedings of INCOSE 2016 International Symposium*, 2016.

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

Haberfellner, Reinhard and Olivier de Weck. 2005. Agile SYSTEMS ENGINEERING versus AGILE SYSTEMS engineering. INCOSE International Symposium, Rochester, NY, 10-15 July. http://strategic.mit.edu/docs/3_59_INCOSE-2005-AGSEvsEAGS.pdf.

Highsmith, J., *Agile Software Development Ecosystems*. Addison-Wesley Professional, 2002.

INCOSE/OMG Patterns Working Group 2015, http://www.omgwiki.org/MBSE/doku.php?id=mbse:patterns:patterns

ISO/IEC 15288: Systems Engineering—System Life Cycle Processes. ISO 2015.

ISO/IEC TR24748-1: Systems and software engineering — Life cycle management — Part 1: Guide for life cycle management, ISO 2010.

Leffingwell, Dean, *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise*. Addison-Wesley 2011

Peterson, T., and Schindel, W., “Pattern Based Systems Engineering – Leveraging Model Based Systems Engineering for Cyber-Physical Systems”, *Proc. of NDIA GVSETS*, 2014.

Schindel, W., “Pattern-Based Systems Engineering: An Extension of Model-Based SE”, INCOSE IS2005 Tutorial TIES 4, 2005a.

-----------, “Requirements statements are transfer functions: An insight from model-based systems engineering”, *Proceedings of INCOSE 2005 International Symposium*, 2005b.

-----------, W., “What Is the Smallest Model of a System?”, *Proc. of the INCOSE 2011 International Symposium*, International Council on Systems Engineering 2011.

-----------, W., “Systems of Innovation II: The Emergence of Purpose”, *Proceedings of INCOSE 2013 International Symposium* 2013

-----------, W., “Maps or Itineraries?: A Systems Engineering Insight from Ancient Navigators”, *Proc. of INCOSE International Symposium*, July, 2015a.

-----------, W., “System Life Cycle Trajectories: Tracking Innovation Paths Using System DNA”, *Proc. of INCOSE International Symposium* 2015, July, 2015b.

-----------, W., "Got Phenomena? Science-Based Disciplines for Emerging Systems Challenges", *Proc of INCOSE Great Lakes Regional Conference*, Cleveland, OH, 2015c.

Schindel, W., and Peterson, T., "Introduction to Pattern-Based Systems Engineering (PBSE): Leveraging MBSE Techniques", in *Proc. of INCOSE IS2013*, Tutorial, June, 2013.

Schindel, W., and Smith, V., "Results of applying a families-of-systems approach to systems engineering of product line families", SAE International, TR 2002-01-3086 (2002).

Schindel, W., et al, "MBSE Methodology Summary: Pattern-Based Systems Engineering (PBSE), Based On S*MBSE Models", V1.5.5A, INCOSE Patterns Working Group, retrieved 2015 from: http://www.omgwiki.org/MBSE/doku.php?id=mbse:pbse

Schindel, W., Peffers, S., Hanson, J., Ahmed, J., Kline, W., "All Innovation is Innovation of Systems : An Integrated 3-D Model of Innovation Competencies ", *Proc. of ASEE 2011 Conference*, American Association for Engineering Education, (2011).

Walden, D., et al, *INCOSE Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*, Version 4, International Council on Systems Engineering (2015).

Biography

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.

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 Discovery Project. He is author of *Response Ability, the Language, Structure, and Culture of the Agile Enterprise*.