Agile Systems-Engineering AND Agile-Systems Engineering

Rick Dove, rick.dove@incose.org

Agile systems-engineering and agile-systems engineering are two different concepts that share the word agile. In the first case, the system of interest is an engineering process, and in the second case, the system of interest is what an engineering process produces. In both cases, the word agile refers to operational adaptability and the sustainment of that adaptability in an uncertain and evolving environment.

To many, the word Agile, with a capital *A*, is used as a noun, referring to a family of software development processes adhering to a set of principles published as the Agile Software Development Manifesto in 2001 (Fowler 2001). To the INCOSE Agile Systems and Systems Engineering (AS&SE) working group, the word agile has a small *a*, and is an adjective referring to a system's capability for operational adaptability in an uncertain and unpredictable evolving environment.

Agile-Systems Engineering

The fundamental needs and concepts of agile systems capability were developed throughout the nineties in two projects led by Lehigh University and funded by the US Department of Defense (DoD), which wanted early understanding of what would be the next competitive focus once the Japanese initiative in lean concepts were digested and employed internationally. See (Dove 2001, 2014a) for that history. The project involved over 1,000 people from all types of organizations, and analyzed real systems that exhibited agile characteristics. Out of this emerged a fundamental common architecture and set of design concepts that enable agile capability in any system domain.

Those projects, with subsequent wordsmithing, defined agility as the ability of a system to thrive in an uncertain and unpredictably evolving environment; deploying effective response to both opportunity and threat, within mission. Effective response has four metrics: timely (fast enough to deliver value), affordable (at a cost that can be repeated as often as necessary), predictable (can be counted on to meet the need), and comprehensive (anything and everything within the system mission boundary).

We are all very familiar with architectures that accommodate and facilitate structural change. Think of the construction sets we grew up with: Erector/Meccano sets (Figure 1), Tinker Toy, Lego, and Lincoln Logs. Just to name some of the classics. Each of these construction sets consists of different types of components, with constraints on how these components can be connected and interact. Though each construction set is better suited to some types of constructions than others are, they all share a common architectural pattern.

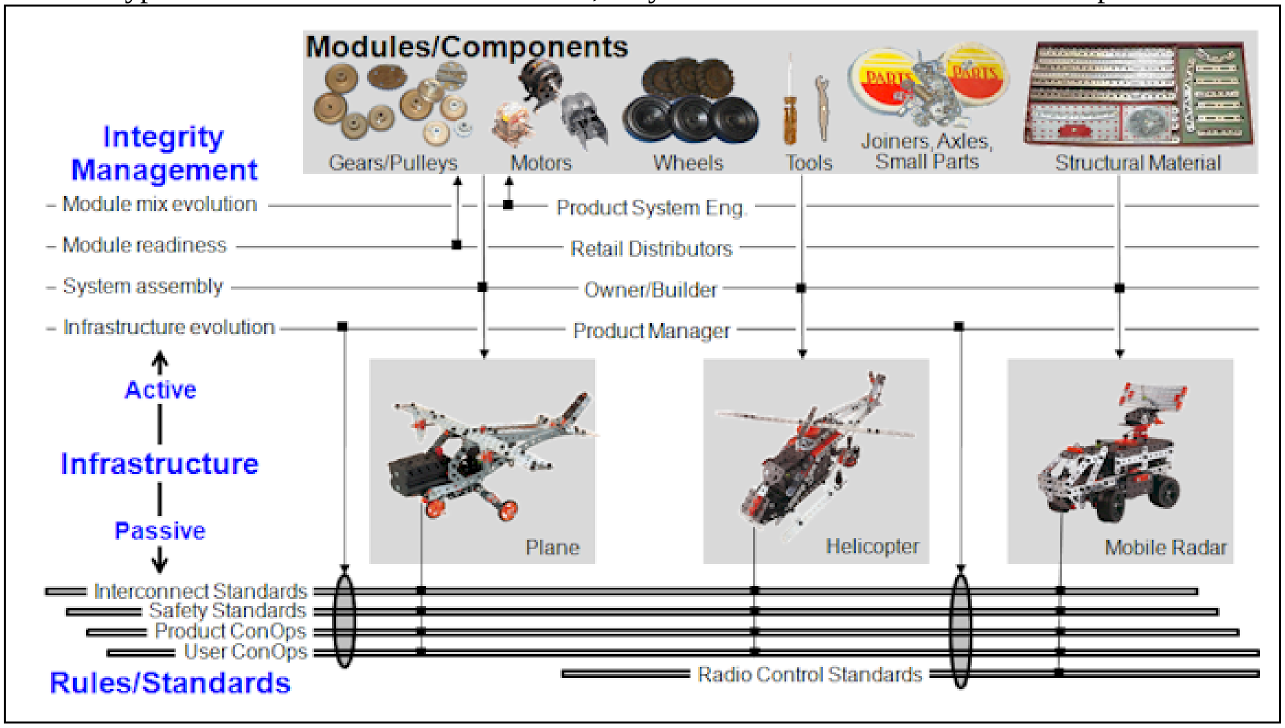

Figure 1. Notional agile system architecture employed by Erector/Meccano sets.

There are three critical elements in the agile architectural pattern: an inventory of drag-and-drop encapsulated modules, a passive infrastructure of minimal but sufficient rules and standards that enable and constrain plug-and-play interconnection, and an active infrastructure that designates four specific responsibilities for sustaining agile operational capability. The coverage here of these elements is necessarily brief, but see (Dove 2014a) for additional detail.

Here the word module is a generic term for system functional assets, which can be human or inanimate.

- Modules—Modules 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 modules. Its value is in isolating the encapsulated modules from each other so that we minimize unexpected side effects when changing modules and new operational functionality is rapid. We should consider at least five categories of standards and rules: sockets (physical interconnect), signals (data interconnect), security (trust interconnect), safety (of user, system, and environment), and service (system assembly concept of operations and evolutionary agility sustainment).
- 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 system configurations are assembled in response to new requirements – something that may happen very frequently. In order for new configurations to be enabled when needed, four actively dispatched responsibilities are required:
  - Module Mix Evolution—Who (or what process) is responsible for ensuring existing modules receive upgrades, addition of new modules, and removal of inadequate modules, in time to satisfy response needs?
  - Module Readiness—Who (or what process) is responsible for ensuring that sufficient modules are ready for deployment at unpredictable times?
  - 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.

Agile Systems-Engineering

Systems engineering is a disciplined activity that delivers engineered solutions to problems and opportunities – an activity often involving multiple stakeholders, coordination across multiple engineering disciplines, and complexity in both problem and solution (Sheard 2000). Unlike other engineering disciplines that employ natural laws to guide and govern engineering design with certainty within a single discipline, systems engineering deals with the social, political, and technical aspects of managing projects that span multiple disciplines. These projects can be quite large and complex, need cross-discipline unifying architectures and operational concepts, require multi-party accommodations to resolve tradeoffs, and often exhibit unexpected emergent behaviors as the project progresses.

The growing acceptance and adaptation of agile software development methods has passed the tipping point in the software world, and is now motivating expectations in broader domain-independent systems engineering. The popularity of Scrum as a project management process, and the Siren song of the Agile Software Development Manifesto, has created for some an expectation of a clear path to broader application.

On the opposite extreme are those who make a clear case for inapplicability, e.g., (Carson 2012). Neither camp actually focuses on agility, but rather on specific software development management practices and principles that share *agility* as a family name.

Fundamental agile capability is part of the classic agile software development processes with the same architecture as an agile system, though these fundamentals get hidden behind popularized project management and technical-process practice dogma. See (Dove 2014b) for the popular Scrum process depicted in the agile architecture pattern. However, according to a recent study by the US Defense Contract Management Agency (Blank 2014), there remains considerable diversity in agile software development employment, at least among US defense contractors.

There is no *a priori* reason to expect domain specific software development practices to be applicable in domain independent systems engineering. For a simple disconnect example, (Carson 2012) observes that in software development the code designer is also the code fabricator; whereas in hardware, the designer's product is an intermediate document that is intended to drive the separate activities of a fabricator with a different world view. Integrated product teams attempt to address some of the communication issues, but inherently hardware design effort and fabrication effort are sequential activities of different time durations and different costs – at least currently.

Nevertheless, the ball is in motion toward the goal of an agile systems-engineering discipline. Perhaps many different balls are in motion, as the pressure to do systems engineering under accelerating environmental dynamics is not waiting for a common disciplined understanding.

Agile Systems-Engineering and the System Life Cycle

Life cycle, as a term applied to systems, traditionally demarcates the progressive maturity flow of a system through a linear sequence of stages, from concept to disposal. Inherent in this model are the notions that a system is in one and only one stage at any point in time, and progresses from one single-state stage to another in a proscribed sequence.

One cannot argue against the necessity to develop a system before utilization can occur. Here, however, the argument is against the continued notions of non-repeating stages and of single-state existence by depicting an agile life cycle in Figure 2 as having progressively concurrent repeated stages.

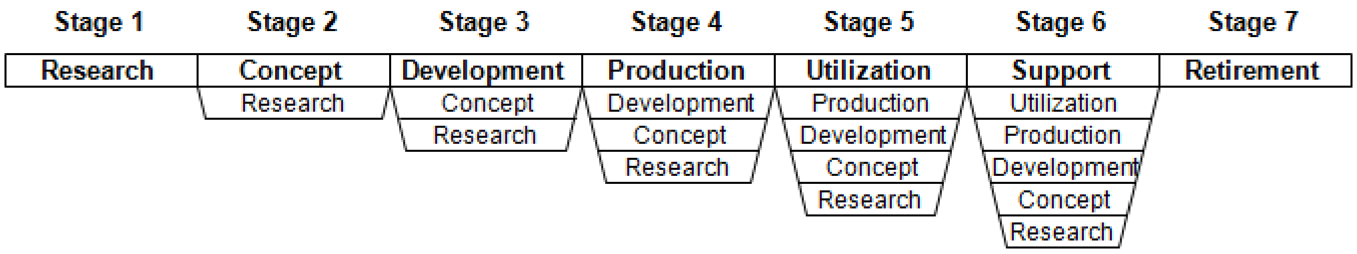

Figure 2: Framework for an agile system engineering life cycle model.

An effective agile systems engineering process must converge on sufficient completion of each of the primary stages to warrant the transition to the next primary stage, presumably on schedule and on budget regardless of how flexible or rigid these might be. This is in the Figure 2 life cycle depiction showing diminishing emphasis on the lower concurrent stages as maturity through primary stages progresses.

It is time to develop an agile systems-engineering life cycle model. This model, if a single one is sufficient, must take into account at least the three different types of systems engineering, articulated well in (Sheard 2000): Discovery (very high complexity in problem space), Programmatic (complexity in solution space and possibly organizational), and Approach (complexity in the variation of applications, and possibly product lines).

Developing an agile systems-engineering life cycle model might start with the framework displayed in Figure 2, take guidance from the ISO/IEC 15288 Standard for Systems and Software Engineering, and move toward identifying fundamental principle-based activities and processes that provide agility, independently as well as collectively, across all stages that warrant an agile approach. This model might justify the application of these principles, activities, and processes by identifying common systems-engineering environmental situations in need of agile response capability. Ideally, the model support comes from case studies in a variety of systems engineering domains.

Nevertheless, how to do this? Developing the model as just another systems engineering project might be an obvious thought. Would this systems engineering occur in a stable engineering environment, or is it more likely that the operational environment of the systems-engineering system, as well as the operational environment of the engineered system (the life cycle model) continue to evolve? In the latter case, the very act of seeking an agile systems-engineering life cycle model testifies to the continued evolution of systems engineering. In the former case, the engineering of an agile systems-engineering life cycle model will occur wittingly or unwittingly in an operational environment of multiple stakeholders, entrenched cultures, competing political pressures, and all the other environmental realities that require a learning, accommodation, and evolutionary approach.

A method called Realsearch (Dove 1998), so called because it employs real people solving real problems in real time, refined, and socialized the original agile systems fundamentals discovered and organized in the nineties. It is a process of traveling, structured, collaborative workshops where investigators visit host sights by invitation. The process engages first in a collaborative exercise of situation analysis on local examples of agile process, then engages in collaborative identification of principles employed locally that enable agile capability, and finally engages in an exercise that applies learnings to an open problem in need of an agile process solution. A series of such workshops, planned by the AS&SE working group begins in 2015, designed to converge on a fundamental agile systems-engineering life cycle model applicable to the INCOSE community. Those interested in hosting workshops and participation as traveling members, please contact rick.dove@incose.org.

Agile-System Thinking

The articles in this issue of INSIGHT are intended to spur understanding and adoption of agile systems engineering concepts that can deal with uncertain and unpredictable development and system deployment environments. To many, the fields of agile software development practices are the only understanding of agile system and development concepts. These articles broaden that understanding. Rather than order these articles in some artificial progression of thought, we synopsize them in author alphabetical order.

Systems Agility Quotient (AQ)
Dave Alberts, PhD, explores the difference between a system's kinetic and potential agility, dealing with the issues of measurement and the ability to predict a system's agile response capability under future situations not yet encountered. Dave is Senior Fellow at the Institute for Defense Analysis, CEO of Agility Advantage, former director for the US Office of the Secretary of Defense Command and Control Research Program, and recipient of the US Secretary of Defense Outstanding Public Service Award; with the recent book: *The Agility Advantage*.

Quantifying Agility
Barry Boehm, PhD and INCOSE Fellow, and Dan Ingold discuss factors affecting success when attempting to accelerate software development time with agile development methods, and provide a tool that can help predict and manage schedule compression potential. Barry is Chief Scientist at the Systems Engineering Research Center, Director Emeritus at the University of Southern California's (USC) Center for Systems and Software Engineering, and Member of the National Academy of Engineering; with the recent book: *Balancing Agility and Discipline.* Dan Ingold is a late career PhD student and graduate research assistant at USC.

Program Agility and Adaptability
Tyson Browning, PhD, notes that most research on agile systems focuses on adaptations in the product and process systems, and argues that we need to include the organization, tools, and goal systems to achieve holistic program agility. Tyson is an Associate Professor at Texas Christian University, and the Associate Editor of *Systems Engineering*; with the recent book: *Design Structure Matrix Methods and Applications*.

Agile Systems-Engineering AND Agile-Systems Engineering
Rick Dove, INCOSE Fellow, provides context for the articles in this INSIGHT theme issue. Rick is President of Paradigm Shift International, adjunct professor at Stevens Institute of Technology, and chairs the Agile Systems and Systems Engineering working group and Systems Security Engineering working group; with the recent book: *Response Ability – the Language, Structure and Culture of the Agile Enterprise*.

Agile Project Governance: The Evolution of Phase/Gate
Jim Highsmith recognizes the seeming paradox between management's need for a deterministic control mechanism and engineering's need to build systems under conditions of uncertainty, and outlines a method of

compatible satisfaction. Jim is Executive Consultant at ThoughtWorks and a co-founder of the Agile Manifesto; with recent book: *Adaptive Leadership: Accelerating Enterprise Agility*.

CubeSat – An Agile System Architecture?
Ralph LaBarge examines a case study of the CubeSat small satellite program, asking if the CubeSat program specifies a generic agile system capability, and answering that question by examining the programs specified architecture for agile enabling principles. Ralph is Principal Professional Staff at Johns Hopkins University Applied Physics Laboratory, and a systems engineering PhD student at George Washington University.

The Role of Systems Engineering in Large Scale Agile Projects
Phyllis Marbach, Larri Rosser, Gundars Osvalds, and David Lempia tackle issues of software intensive projects employing agile software development concepts within a traditional systems engineering environment, suggesting methods for integrating systems engineers and software engineers working in cross-functional teams. Phyllis is Senior Software Manager at Boeing, Larri is Systems Engineering Integration and Test Lead at Raytheon, Gundars is CSEP and Principal Systems Engineer at Praxis Engineering, and David Lempia is Principal Systems Engineer at Rockwell Collins.

Response to Cyber Security Demands for Agility
Perri Najib and Dawn Beyer outline the Lockheed comprehensive initiative to place responsibility for systems security on the systems engineering community, and notably have implemented an agile continuous learning process in response to the agile nature of the growing cyber threat. Perri is Sr. Fellow and IS&GS Chief Technology Officer at Lockheed Martin, and Dawn is Fellow and IS&GS Security Engineering Domain Advocate at Lockheed Martin.

"Composable Capability" for Agile SoS
Hillary Sillitto, INCOSE Fellow and ESEP, discusses needs and methods for composable systems that can be assembled from available assets into a just-in-time response to a real-time situational need, employing a military force projection example based on a generic approach applicable in other domains. Hillary is a partner at Sillitto Enterprises, visiting professor at the University of Bristol, and Fellow of the Institute of Physics, and Chartered Engineer; with a book in press: *Architecting systems – A Primer on Purpose, Concepts and Methods*.

Rediscovering Systems Engineering
Richard Turner, DSc, provides a high level view of fundamental systems engineering objectives, and outlines a spiral, concurrent environment that doesn't negate the power of systems engineering discipline, yet maintains the opportunity for continuous systems engineering. Richard is distinguished service professor at Stevens Institute of Technology; with the recent book: *Balancing Agility and Discipline.*

The Challenge of Reflection in Agile Development Systems
John Young steps outside the area of agile development to look at the environment surrounding development, suggesting that reflective agile work cannot happen outside the context of the wider organizational system, particularly the power structures within an organization. John is Managing Director, Red Elephant Technology.

In Closing

The AS&SE working group views agility as a sustainable capability, enabled and constrained fundamentally by a common high-level system architecture. This architecture delivers agile capability as reconfiguration, augmentation, and evolution of system and process functionality in the operational process and product environment; enabling both the process and product to respond to new and immediate situational requirements effectively.

References

Blank, Greg. 2014. DCMA Study of Agile and DoD Software Acquisition. Defense Contract Management Agency. Web presentation made to the NDIA Systems Engineering and Agile Working Group, 7 May. Request a copy from Geoff Draper at gdraper@harris.com.

Carson, Ron. 2013. Can Systems Engineering be Agile? Development Lifecycles for Systems, Hardware, and Software. Symposium of the International Council on Systems Engineering, Philadelphia, US-PA, 24-27 June.

Dove, Rick. 1998. Realsearch: A Framework for Knowledge Management and Continuing Education. In proceedings IEEE Aerospace Conference. Aspen, US-CO. 28 March. www.parshift.com/Files/PsiDocs/RealsearchIEEE.pdf.

———. 2001. *Response Ability – the Language, Structure, and Culture of the Agile Enterprise*. Malden, US-MA: John Wiley & Sons.

Dove, Rick and Ralph LaBarge. 2014a. Fundamentals of Agile Systems Engineering – Part 1. Symposium of the International Council on Systems Engineering, Los Angeles, US-CA,, 30 June - 03 July. www.parshift.com/s/140630IS14-AgileSystemsEngineering-Part1.pdf

———. 2014b. Agile Systems Engineering – Part 2. Symposium of the International Council on Systems Engineering, Los Angeles, US-CA, 30 June - 03 July. www.parshift.com/s/140630IS14-AgileSystemsEngineering-Part2.pdf

Fowler, Martin and Jim Highsmith. 2001. The Agile Manifesto. Dr. Dobb's Journal, August. www.drdobbs.com/open-source/the-agile-manifesto/184414755.

Sheard, Sarah A. 2000. Three Types of Systems Engineering Implementation. Symposium of the International Council on Systems Engineering, Minneapolis, US-MN, July.