Fundamentals of Agile Systems Engineering – Part 1

Rick Dove
Paradigm Shift International
Taos County, New Mexico, USA
dove@parshift.com

Ralph LaBarge
Johns Hopkins University/APL
Laurel, Maryland, USA
ralph.labarge@jhuapl.edu

Revision 20-May-2018 made corrections and updates, changing UURV framework to CURVE, and modules to resources.

Abstract

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 is produced by an engineering process. The word agile refers to the adaptability and the sustainment of adaptability in both types of systems. Sustained adaptability is enabled by an architectural pattern and a set of system design principles that are fundamental and common to both types of systems. Research that identified this architectural pattern and design principles is reported, updated, and applied here in two Parts. Part 1 focuses on agile-systems engineering, reviewing the origins, values, and core concepts that define and enable domain independent agility in any type of system. Part 2 focuses on agile systems-engineering, identifying core agility-enabling concepts in the software-development domain-specific practice known as Scrum, reviewing an agile hardware/software satellite-development systems-engineering case for its source of agility, and then suggesting the development of an agile systems-engineering life cycle model as a natural next step.

Introduction

The value proposition of an agile system is rooted in risk management, providing options when system mission or system survival is threatened. Some might say the purpose or objective of an agile system is risk management, but natural agile systems exist without that purpose/objective, just that benefit. Most natural systems have evolved sufficient agility to sustain existence in the inherently risky environments that surround them. But nature doesn't care. Agility is a byproduct of natural selection, an algorithm without an objective (Dennett 1995), based on replication with variation in a competitive environment; an algorithm that unwittingly experiments with expendable resources over long periods of time. This method is generally not suitable to systems designed and built by man for purposeful objective, if these systems are to remain effective in an uncertain and unpredictable environment for a reasonable period of time. But we can learn from nature's experiments, perhaps improve upon their results, for nature finds sufficient but not necessarily optimal solutions.

Natural systems analysis is not the only path. We can also learn from man-made systems that exhibit the ability to survive, even thrive, in uncertain and unpredictable environments, and analyze these systems for common and replicable patterns that provide this capability. Intensively in the nineties, and continuously thereafter, well over 100 man-made systems exhibiting agile characteristics have been studied in workshops conducted at a wide variety of host sites, which

examined systems in many domains including manufacturing processes, enterprise processes, hardware systems, software systems (Dove 1993a; Dove et al 1995; Dove, Hartman, Benson 1996; Dove 1998; Dove 2001; Dove 2005), and more recently, development systems (Dove and LaBarge 2014).

This article summarizes the findings of those empirical studies, with the purpose of presenting in one document what appear to be necessary and sufficient fundamental architecture and design guidance for the systems engineering practitioner. The engineering usefulness of the architecture and supporting design principles have been confirmed by one of the authors in twenty five years of evolution and deployed employment, with examples in (Dove, Pirtle, Wilczynski 1987; Dove 2005; Dove 2009; Dove 2011), and in nine years of design and analysis projects conducted by masters students, with examples in (Bose and Dove 2010, Papke and Dove 2013).

Understanding the fundamental enablers of systems agility is timely. The pace of technology is reducing the useful lifetime of deployed systems and increasing the risk of long development programs. The pace of social collaboration on a global scale changes the effectiveness of government processes and increases the pace of technological and social innovation. The pace of global network dependencies of all kinds brings both benefit and vulnerability.

In the military, agility is sought in agile command and control (Alberts 1996, 2011), US force transformation (Cebrowski 2003), in composable force projection (Sillitto 2013), and in rapid acquisition and quick reaction capability (DSB 2009, SAF 2011). In commercial sectors agility is sought to sustain growth, innovation, and market leadership. In organizational support, agility is sought in service oriented architecture, web services, and cross organizational collaboration. In security, agility has been employed by the adversary to great effect for some time, prompting a growing voice for agile security systems.

Agility has been confusingly defined in the literature as various and overlapping system characteristics. Updating timeless core concepts developed in the ‘90s, this article presents a succinct core definition of agility; its relationship to various literature definitions; and the nature of uncertain, unpredictable, risky, and variable system environments that agile systems-capability is meant to address.

Loosely coupled modular systems are generally considered the core enabler of systems adaptability and flexibility in the literature (Orton and Weick 1990), but sustainability embedded in architecture has been largely ignored (with a notable exception in Weick 1999), as has the necessary core nature of infrastructure and resource pools, design principles, and methods for developing agile-response requirements. This article offers the practitioner means to address these issues.

Agility

In the 1980s the world conceded that the Japanese lean manufacturing concepts led to superior competitive manufacturing capability. Major manufacturers world-wide were scrambling to catch up. Charles Kimsey, in the Office of the Secretary of Defense, thought differently. He thought while everybody struggled to catch up, some effort also ought to be spent trying to identify what would be next, especially since the Japanese were already working toward a next paradigm, attempting to start what is now called the Holonic Manufacturing Systems consortia, investigating systems composed of holons: intelligent, autonomous, cooperative agents (Christensen 1994). Kimsey arranged to fund this look-ahead project at Lehigh University through the US Navy

Mantech program. Thirteen companies were invited to send appropriate thinkers to a summer-long workshop at Lehigh University, with the purpose of identifying an emerging problem so common and key to competition that it would become the next differentiator once lean was broadly diffused.

That problem was identified as the ability to respond effectively and with competence, to operational environments with increasing uncertainty and unpredictability (Dove and Nagel 1991, Dove 1992). That ability was named agility, and that study spawned the Agility Forum (nee Agile Manufacturing Enterprise Forum) to explore the nature of agile enterprise and domain-independent agile systems throughout most of the '90s. The primary focus of the 1991 study was on the agile manufacturing enterprise, not on manufacturing floor systems or processes as is often thought. The work in process was socialized widely for feedback with groups such as NIST, DARPA, the Defense Science Board, the Aerospace Industries Association, and others recognized in (Dove and Nagel 1991); which likely sparked subsequent military "agile enterprise" interests such as force transformation (Cebrowski 2003). The 1991 study identified the problem and developed agile-enterprise conceptual visions in four different commercial domains: automotive, process, semiconductor, and telecommunication industries. A subsequent Agility Forum study broadened the focus to agile systems of all kinds, and began the search and development of agile-system enabling fundamentals.

With the agile label and concept in play, Hewlett Packard was the first to initiate a program to educate its customers (Dove 1993a) and subsequently bring to market IT support under the Agile Enterprise banner; DoD's Command and Control Research Program began an exploration of agile command and control (Alberts 1996) that continues today, and the Agile Manifesto for Software Development (Fowler and Highsmith 2001) adopted the agile label as appropriately descriptive and fundamentally consistent with their concepts[1].

So an enterprise focus came first, domain-independent agile fundamentals next, then an application to military force transformation, and then software development adopted the agile label with profound popular-awareness effect. Today the software development use of the label gets wide employment; perhaps because the software development community expressed a strong natural pull for a new development paradigm beyond waterfall on a large scale.

Tracking the history of the agile-systems fundamentals development effort, the 1991 publication of the 21st Century Manufacturing Enterprise Strategy (Dove and Nagel 1991) opened the door with strategic intent and vision, a call for action with little in the way of true guidance at that point. The next two years at the Agility Forum developed preliminary agile systems enabling frameworks, inspired by work in the late '80s on an object-oriented CAD-like product for developing factory-wide cellular control systems at a company called Flexis Controls (Dove, Pirtle, Wilczynski 1987). These frameworks were first published at the 1993 Defense Manufacturing Conference (Dove 1993b), fueling a subsequent five-year series of industry-collaborative workshop studies that involved some 1000 people and 250 organizations, who examined 100+ systems of all kinds (Dove 1994, 1998, Dove et al. 1995) which exhibited agile characteristics. During this same period an agile enterprise reference model was developed, which featured a capability maturity model and analysis process to measure how agile a company was in 24 different business practices (Dove, Hartman, and Benson 1996). The workshops and reference

[1] Personal communication with Jim Highsmith, a founder of the Agile Manifesto for Software Development (Fowler and Highsmith 2001).

model work refined and augmented the original frameworks, culminating in the publication of *Response Ability – the Language, Structure and Culture of the Agile Enterprise* (Dove 2001). Title notwithstanding, the book addresses systems within an enterprise, and was completed during the design and implementation of an enterprise-wide IT system that featured the first agile-ERP (enterprise resource planning) system; allowing each department to have ERP resources from any vendor, changeable at any time, all interacting as if from a single vendor. In 2001-2002 the development and implementation of this agile-ERP system was designed and managed as an agile development process (Dove 2005), with three successive three-month releases that each provided functional ERP operational capability to the company – on time and under budget.

In 2005 came a request to develop and teach two courses for an Agile Systems and Enterprises certificate at Stevens Institute of Technology – which has refined further the vocabulary and design processes which appear in collected form here. Working with practicing systems engineers pursuing graduate degrees and masters projects helps clarify the conceptual and operational stumbling blocks for the new initiate. In 2012 an INCOSE working group was chartered for Agile Systems and Systems Engineering, with a charter focus on applying and socializing the application of agile-system fundamentals to agile-systems design and agile systems-engineering, integrating these fundamentals with general Systems Engineering process concepts to explore the issues beyond Agile Software Development practices. It is anticipated that this working group effort will also return to the characteristics of agile systems beyond fundamentals, inspired by Alberts work and the recent work in resilient systems engineering. This present article is motivated by a need to provide a foundation of fundamentals for the working group activity; and to update the articulation, understanding and application of fundamentals arising from some eight years to date of teaching the architecture, enabling principles, and concepts of operations for agile systems creation.

Defining Agility

Words and phrases as labels for distinguishing system concepts have the ability to identify the core essence of the concept, and provide a valuable service in doing so. Example labels applied to system concepts of interest include lean, agile, resilient, composable, and many others. But as concepts become popular, their proponents often attempt to expand what they encompass to include related concepts, in what appears to be an attempt to unify everything of current interest under a single label of some personal or program interest. To be sure, there are many best practices shared among many legitimate labels, but they are applied specifically to augment and support the core essence of what entitles the label to represent a uniquely distinguishable concept.

In an invited synopsis paper of the 1991 Lehigh study (Dove 1992) defined agility as "that characteristic which allows an organization to thrive in an environment of constant and unpredictable change." Similarly, the most extensive and thoughtful ongoing effort to operationalize agility started in 1996[2] with a military command and control focus, that has since matured into a broader focus on agile enterprise of any kind, military or otherwise (Alberts 2011), affirms that definition: "Agility is a capability that enables an entity to succeed in changing circumstances." These overarching definitions are echoed equivalently in a variety of wordings for different domains, but consistently pinpoint the core definition of agile systems.

[2] The US DoD Command and Control Research Program (CCRP) has published a series of books about agile Command and Control beginning in 1996 with *Information Age Transformation* (revised in 2002), all of which are available for free download at www.dodccrp.org/html4/books_downloads.html

There is not a prescribed single way to express the definition of system agility, but however it is expressed, it should reflect the core concept as offered at the 1993 Defense Manufacturing Conference (Dove 1993b):

> "We can adopt a working definition of agility as: The ability to thrive in an environment of continuous and unpredictable change. The focal point here is 'change' - the ability to initiate it, and the ability to respond to it. 'Thrive' is a key word because it implies both long term success, as opposed to a lucky response, and because it implies wielding agility both as an offensive as well as a defensive capability. 'Continuous and unpredictable' underscores the new long-term picture but, most importantly, distinguishes agility from mere flexibility, enabling successful change even when there is little advance notice and no prior expectation."

A compatible version taught to systems engineering students emphasizes response effectiveness:

> Agility is 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).

As to the unpredictably evolving environment, emerging requirements are one typical and pervasive example in program and project management. In a more general sense, however, emerging requirements at both development and operational time are the only factor of interest, as all new response needs can be reduced to new requirements that should be addressed.

Core agreement on the agile definition notwithstanding, there is still confusion with other labels that appear to address some or all of the same capability: nimble, sense and respond, survivable, resilient, sustainable, autonomic, holonic, robust, and composable come quickly to mind. And of course there is the use of the agile label in Agile Software Development, which to many in the software and software-dependent fields is the exclusive understanding of what the agile label refers to and encompasses. Part 2 of this article will focus on agile systems-engineering, with both respect and perspective for the relevance of domain-specific agile software development practices to domain-independent agile systems-engineering.

Outside the scope of this article is an examination of each of these concept labels for core differences, overlaps, and identical meanings. But two labels warrant some attention: resilient and composable. Since the early agile systems work in the '90s, variations on the quad graphic of Figure 1 have been used to make the point that agility is composed of both reactive and proactive change proficiency. Since then, resilient systems have become a strong focus of interest and study, and more recently a call for composable systems is being heard. In both cases an ability to reconfigure system resources effectively to deal with new environmental situations is called for. As will be shown later, this ability to change effectively is enabled by a fundamental architecture common to both.

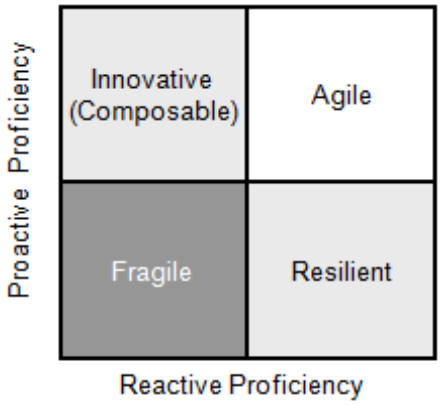

Figure 1. Two dimensions of response proficiency

Relating resilience to agility: Consolidating some 15 years of agile command and control

investigation for the US DoD, David Alberts recognizes resilience as one of six components (his word) of agile systems; and juxtaposes the definition with a need to respond to a "Change in Circumstances: The destruction, interruption, or degradation of an entity capability. … Resilience provides an entity with the ability to repair, replace, patch, or otherwise reconstitute lost capability or performance (and hence effectiveness), at least in part and over time, from misfortune, damage, or a destabilizing perturbation in the environment (Alberts 2011: 217-218)."

Relating composability to agility: In a recent paper addressing military "composable capability", Hillary Sillitto proposed: "…improved operational readiness, performance and interoperability can be achieved by applying a systems engineering methodology in which the 'system focus' is the force element, not the individual equipment; it is possible to identify a finite set of stable, well characterised building blocks (Force Elements) from which a wide variety of task force structures can be put together, providing almost infinite variety of capability solutions; …" Sillitto suggests that the "System Coupling Model" (Lawson, 2010: 23) sets the context of "composability," reproduced in Figure 2 as a condensed version of the agile architectural pattern shown later in Figure 3.

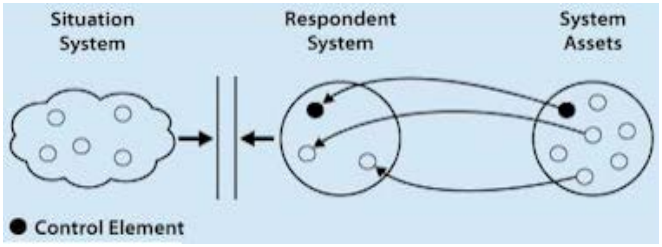

Figure 2. System-Coupling Diagram (Lawson 2010: 23) illustrating composability of a response system appropriate to a situation.

Effective response to both opportunity and threat is depicted in Figure 1 as response proficiency in two dimensions: proactive and reactive. As will be shown in the next section, an agile system's response to a change in the environment, whether to take advantage of an opportunity or to respond to a threat, is achieved, metaphorically, by reshaping the system so that it is compatible or synergistic with the changed shape of the environment. A reactive response is a compulsory systemic counter to a threatening change in the environment, with purpose to maintain or restore competitive functional performance. A proactive response is a non-compulsory systemic initiation enabled by a change in the environment, with purpose to improve competitive functional performance.

Metrics and Measures

How agile does a system have to be? Agility does not have a practical absolute measure, but is rather one of comparison to the dynamics of the environment, which includes competing systems that redefine acceptable performance requirements as they appear for the first time, and as they evolve.

There are four fundamental metrics for response proficiency: time, cost, predictability, and scope (Dove 2001: 70-87):

- Time to respond, measured in both the time to decide (after knowing) that a response is necessary, and the time to accomplish the response.
- Cost to respond, measured in both the cost of accomplishing the response and the cost incurred elsewhere as a result of the response.

- Predictability of response capability, measured before the fact in architectural preparedness[3] for response, confirmed after the fact in repeatable accuracy of effective response.
- Scope of response capability, measured before the fact in architectural preparedness for comprehensive response capability within mission, confirmed after the fact in repeatable evidence of broad response accommodation.

These metrics do not stand alone, but work together. Having the capability to respond quickly, even instantly, does little good if the cost of response precludes the ability to respond again, unpredictably and as often as necessary. Predictability in effective response is the third metric, and the mark of a systemically repeatable response process. Finally there is scope, which differentiates agility from flexibility, and should encompass the ability to respond to anything within the system's mission space. A method for measuring an organization's response proficiency is explained and employed in an agile enterprise case study (Dove 1996).

The Environment Drives the Need

Agile systems are defined in counterpoint to their operating environments. Words used to describe the general nature of the target environment often include and combine dynamic, unpredictable, uncertain, risky, variable, and changing, with little attention to clear distinction among them. To design and develop a system that can deal effectively with changing environments it is useful to articulate the nature of changes that should be considered. A practice employed in classes and workshops on design methods for agile systems considers five types of environmental dynamics: caprice (unpredictability), uncertainty, risk, variation, and evolution. This categorization originated from a desire to explain why it felt natural to talk about agile systems as ones that can deal with uncertain and unpredictable environments. Is there a meaningful difference between uncertain and unpredictable – or was this just a lazy tendency to use two words when one can do?

Research yielded the wisdom of Frank Knight, who very carefully and logically separated the meaning of risk from the meaning of uncertainty in his 1921 doctoral thesis, subsequently published and still available as a delightfully readable classic economics book (Knight 1921).

Knight's work argues that random events come in two varieties, those with knowable probability and those with unknowable probability; and that this distinction separates risk and uncertainty. His knowable/unknowable distinction can also be a key differentiator for unpredictability and variation, though these do not have the symmetrical relationship of Knight's risk vs. uncertainty.

Our objective is a tool that directs the designers mind to a multidimensional exploration of response needs, consistent with the expectations of an agile system.

Agile systems have effective situational response options, within mission, under a CURVE framework:

- Caprice: 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 developments.

[3] Architectural preparedness does not refer to a system's functional architecture, but rather to an underlying architecture which enables and sustains system's agility, discussed in the next section.

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

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

Agile Systems

Agile-systems engineering and agile systems-engineering are two different concepts (Haberfellner and de Weck 2005), but both rely on a common architecture to enable the agility in each. The architecture will be recognized in a simple sense as drag-and-drop, plug-and-play, loosely coupled modularity, with some critical aspects not often called to mind with the general thoughts of a loosely coupled modular architecture.

Agile systems 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.

The focus in this article is on architecture, metaphorically the design of an instrument, and not on practice, the playing of that instrument. The second part of this discussion (Dove and LaBarge 2014) focuses on practice-enabling capabilities, while (Weike, Sutcliffe, and Obstfeld 1999) deal well with the operational practice aspects of awareness and action to employ these capabilities.

We are all very familiar with architectures that accommodate and facilitate structural change. Think of the construction sets we grew up with: Erector set, 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, they all share a common architectural pattern.

Some, like Erector set with motors, can be employed to build active constructions such as Ferris wheels, helicopters, race cars, or simple robots. A Ferris wheel has a functional architecture, an Erector set has an agile architecture. The agile architecture enables the building and changing of the functional architecture. One could argue that the agile architecture is also a functional architecture, just in a different domain.

A system engineer tasked to design an agile system in some functional domain starts with the design of the erector set architecture for that domain. This agile architectural pattern is depicted in Figure 3 as applied to an Erector set, and explained subsequently in its general pattern sense. The standard graphic depiction pattern typically shows three sample system assemblies to indicate a range of configuration change.

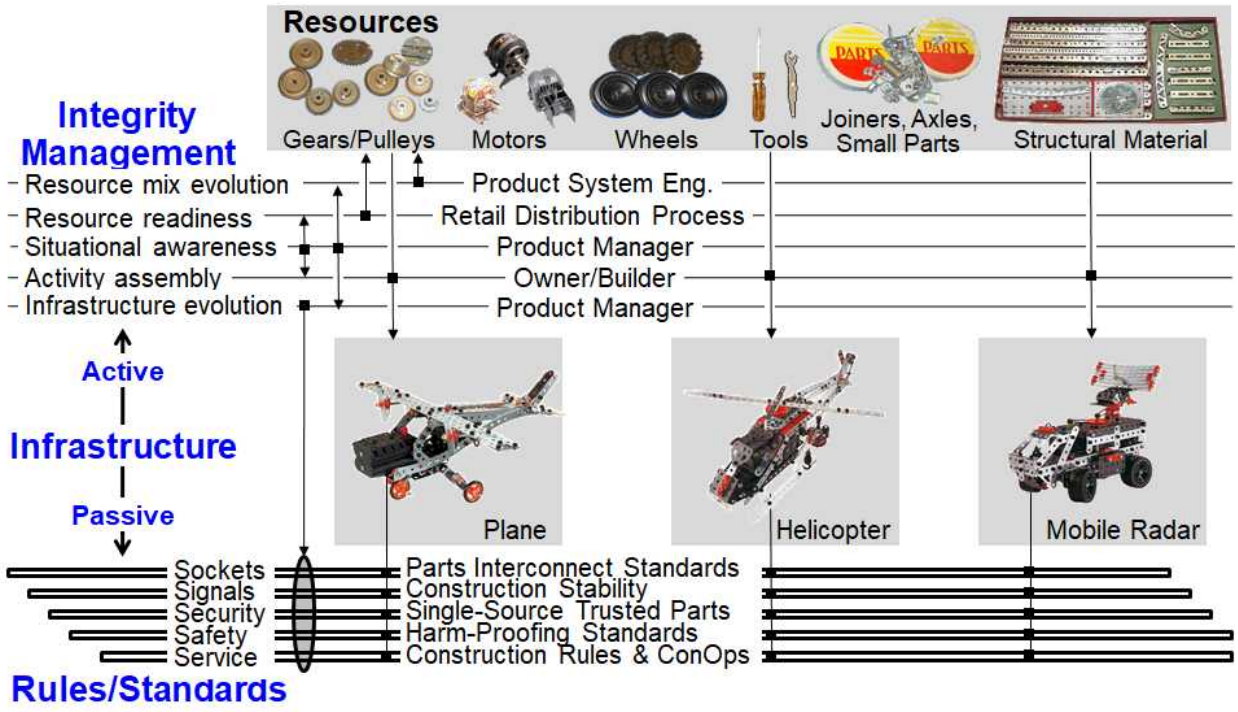

Figure 3. Agile architecture pattern depicting an Erector set construction kit example.

Agile Architecture Fundamentals

There are three critical elements in the agile architectural pattern: a roster of drag-and-drop encapsulated 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 four specific responsibilities for sustaining agile operational capability. The coverage here of these elements is necessarily brief.

Here the word resource is generally used as a generic term for system functional assets, which can be human or inanimate.

- Resources—Resources are self-contained encapsulated units complete with well-defined interfaces which conform to the plug-and-play passive infrastructure. They can be dragged-and-dropped into a system of response capability with relationship to other resources determined by the passive infrastructure. Resources are encapsulated so that their methods of functionality are not dependent on the functional methods of other resources, except perhaps as the passive infrastructure may dictate.
- Passive Infrastructure—The passive infrastructure provides drag-and-drop connectivity between resources. Its value is in isolating the encapsulated resources so that unexpected side effects are minimized and new operational functionality is rapid. Selecting passive infrastructure elements is a critical balance between requisite variety and parsimony – just enough in standards and rules to facilitate resource connectivity, but not so much to overly constrain useful innovative system configurations. At least five categories of standards and rules should be considered: sockets (physical interconnect), signals (data interconnect), security (trust interconnect), safety (of user, system, and environment), and service (system assembly ConOps 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 which may happen very frequently, even daily in some cases. In order for new configurations to be enabled when needed, four responsibilities are required: the roster of available resources must evolve to be always what is needed, the resources that are available must always be in deployable condition, the assembly of new configurations must be accomplished, and both the passive and active infrastructures must have evolved when new configurations and new resources require new standards and rules. Responsibilities for these four activities must be designated and embedded within the system to ensure that effective response capability is possible at unpredictable times. The "how" processes of dispatching responsibility should be articulated in the service element of the passive infrastructure.
    - Resource Mix Evolution—Who (or what process) is responsible for ensuring that existing resources are upgraded, new resources are added, and inadequate resources are removed, in time to satisfy response needs?
    - Resource Readiness—Who (or what process) is responsible for ensuring that sufficient resources 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 Design Principles

Ten common Reusable-Reconfigurable-Scalable (RRS) design principles were discovered in workshop analysis of existing agile systems, and now are used to guide architectural design strategy. These principles are split into three categories, with the understanding that a principle in one category often provides benefit in the other categories. Need and intent are briefly outlined for each principle, with the "intent" providing a general strategy for meeting the need, and the understanding that an augmented or related approach may be a better fit to a specific-system need. Entire papers could be written on the variations and nuances of each of these principles. It is left to a designer's creative insight to adapt the essence of the principle to the system of interest.

Reusable Principles:

- Encapsulated Resources (Modularity)—Need: System assemblers want effective resource replacement and internal change without side effects. Intent: Resources physically encompass a complete capability, and have no dependencies on how other resources deliver their capabilities.
- Facilitated Interfacing (Plug Compatibility)—Need: System assemblers want effective interfacing that facilitates integration and replacement of resources. Intent: Resources share minimal interface standards, and are readily inserted and removed.
- Facilitated Reuse—Need: System assemblers want effective resource selection and acquisition that facilitates reuse. Intent: Available resources are identified by capability and requirements, and can be readily discovered and acquired for deployment.

Reconfigurable Principles:

- Peer-Peer Interaction—Need: System assemblers want effective communication among resources. Intent: Resources communicate directly on a peer-to-peer basis to avoid intermediary relay failure, content filtering, and time delay.
- Distributed Control and Information—Need: System assemblers want effective information-based operational decisions. Intent: Decisions are made where maximal situational knowledge exists, and relevant information is maintained local to decision making resources while accessible globally.
- Deferred Commitment—Need: System assemblers want to maintain effective response ability. Intent: Conserve the commitment and consumption of limited resources to the last responsible moment, in anticipation of future unpredictable events and uncertain response needs.
- Self-Organization—Need: Systems assemblers want effective adaptation of interacting resources. Intent: Resource relationships are self-determined where possible, and resource interactions are self-adjusting or self-negotiated.

Scalable Principles

- Evolving Infrastructure—Need: System assemblers want effective acquisition and deployment of new resource capabilities. Intent: Passive infrastructure standards and rules are monitored for current relevance, and evolve to accommodate new and beneficial resource types in anticipation of need.
- Redundancy and Diversity—Need: System assemblers want effective resilience under quantitative (need more of something) and qualitative (need something different) situational variance. Intent: Duplicate or replicable resources provides quantitative capacity options and fault tolerance options; diversity among similar resources provides situational fit options.
- Elastic Capacity—Need: System assemblers want to incrementally match committed system resources to situational capacity needs of unpredictable or uncertain range. Intent: Resources may be combined in unbounded quantities, where possible, to increase or decrease deliverable functional capacity within the current architecture.

Response Requirements Guide Architectural Design

In addition to the system functional requirements, response situation analysis (RSA) identifies response requirements that inform the design and implementation of the agile architecture pattern. RSA indicates the necessary nature of resources and resource pools, which in turn identify the necessary nature of both passive and active infrastructure.

Requirements arising from RSA may not be directly present in customer requirements. Unlike functional requirements, typically captured in all-encompassing shall-statements, response requirements need only enumerate sufficient situational diversity to result in a capability that can respond to un-enumerated situations.

An effective framework for guiding RSA exercises drives analytical thinking in four reactive and four proactive domains. Note that response requirements should be stated as situations which arise during operation (the problem) that require a system response, independent of possible ways the

response might be satisfied (the solution). Solution strategies will change over time as new technology and knowledge become available.

Proactive responses are generally triggered internally by the application of new knowledge to generate new value. They are proactive responses even if the values generated are not positive and even if the knowledge applied is not new – self initiation is the distinguishing feature here. A proactive change is usually one that has effect rather than mere potential; thus, it is an application of knowledge rather than the invention or possession of unapplied knowledge. Proactive change proficiency is the wellspring of leadership and innovation in system capability.

Proactive domains:

- Creation/Elimination—What range of opportunistic situations will need resources assembled into responsive system configurations; what elements must the system create during operation that can be facilitated by resources and resource pools; what situational evolution will cause obsolesce of resources which should be removed? The distinguishing feature is the creation of something new or reincarnated that is not currently present. To note, this is not about the situation that calls for the original creation of an agile system, but rather about the evolution of the agile system during its operational period. Situations to identify are those that require system configuration assemblies during operation, and those that require new resources for employment in those assemblies.
- Improvement—What improvements in system response performance will be expected over the system's operational life? The distinguishing feature is performance of existing response capability, not the addition of new capability. Situations to identify are generally those involving competencies and performance factors, and are often the focus of continual, open-ended campaigns.
- Migration—What evolving technologies and opportunities might require future changes to the infrastructure? The distinguishing feature is a need to change the nature of the plug-and-play infrastructure, not the addition of new resources. Situations to identify are generally those that enable the transition to possible and potential next generation capabilities.
- Modification (of capability)—What evolving technologies and opportunities might require modification of the available resources and roster of resource pools? The distinguishing feature is a necessary change in available resource capabilities. Situations are generally those that require something unlike anything already present, or the upgrade or change to something that does exist.

Reactive responses are generally triggered by events which demand a response: problems that must be attended to or fixed, opportunities that must be addressed. The distinguishing feature is little choice in the matter – a reaction is required. Reactive responses are often addressing threatening competitive or environmental dynamics. They may also be responses to new customer demands, agility deterioration/failure, legal and regulatory disasters, product failures, market restructuring, and other non-competitor generated events. Reactive change proficiency is the foundation of resilience and sustainability in system capability.

Reactive domains

- Correction—What types of response activities might fail in operation and need correction? The distinguishing feature is a dysfunction or inadequacy during attempted response. Situations to identify are those that require a recovery from response malfunction, recovery from unacceptable side effects of a response, and inability to assemble an effective response.
- Variation—What aspects of operational conditions and resources vary over what range when response capabilities must be assembled? The distinguishing feature is predictable but uncertain variance. Situations to identify are those that manifest as variances in resource availability, resource performance, and resource interactions.
- Expansion/Contraction (of capacity)—What are the upper and lower bounds of response capacity needs? The distinguishing feature is capacity scalability. Situations to identify are those that can be satisfied with planned capacity bounds, as well as those that have indeterminate and unbounded capacity needs.
- Reconfiguration—What types of situations will require system reconfiguration in order to respond effectively? The distinguishing feature is the configuration and employment of available resources for new or reincarnated response needs. Situations to identify are those that are within the system mission boundaries, and that may require a reconfiguration of an existing system assembly, perhaps augment with removal of resources or addition of available resources.

An Agile System Example

The CubeSat Project originally set out to provide a low cost and condensed development time approach for very small satellites, affordable and suitable for university educational and research programs. The first CubeSat specification was developed in 1999 by California Polytechnic State University and Stanford University. While its original purpose was to help universities develop and test small, cost-effective satellites, the specification has also been used by commercial organizations and Governments around the world. By the end of 2012 over 75 CubeSats had been launched using 24 different launch vehicles (CubeSat 2012).

Key to the effectiveness of this program is its conformance to an agile architecture pattern from the start. Though the specification has evolved from lessons learned and open collaborative workshops over the years (Heidtl et al. 2000, Nugent et al. 2008), critical plug-and-play infrastructure specifications have remained stable to ensure plug compatibility of the deployment package (Figure 4) with a variety of launch vehicles, and plug compatibility of satellites with the deployment package.

The agile architecture pattern for CubeSat is shown in Figure 5. CubeSat satellites can be designed and built using a variety of modular components. Off-the-shelf chassis, power systems, communications, electronics, propulsion and sensor components are available from a number of different commercial providers. CubeSats can be deployed using a variety of launch vehicles and deployment systems.

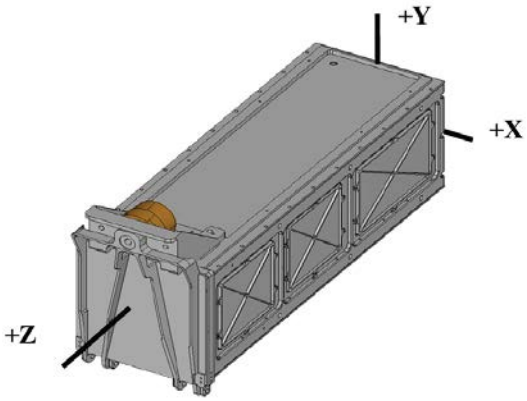

Figure 4. Poly Picosatellite Orbital Deployer (P-POD)

The CubeSat design specification (CubeSat 2013) defines a set of physical, mechanical, electrical, environmental, safety, operational, magnetic, and test requirements for satellites. CubeSats can be made in three form factors: 10 x 10 x 10 cm, 10 x 10 x 20 cm, and 10 x 10 x 30 cm sizes, with a total weight of less than 5.0 kg. The small size and limited weight of a CubeSat enable "piggy back" launches, also called rideshares, to make use of extra space and lift capacity on third party launch vehicles. CubeSats are deployed using a Poly Picosatellite Orbital Deployer (P-POD), a standard deployment system developed by the California Polytechnic State University shown in Figure 4.

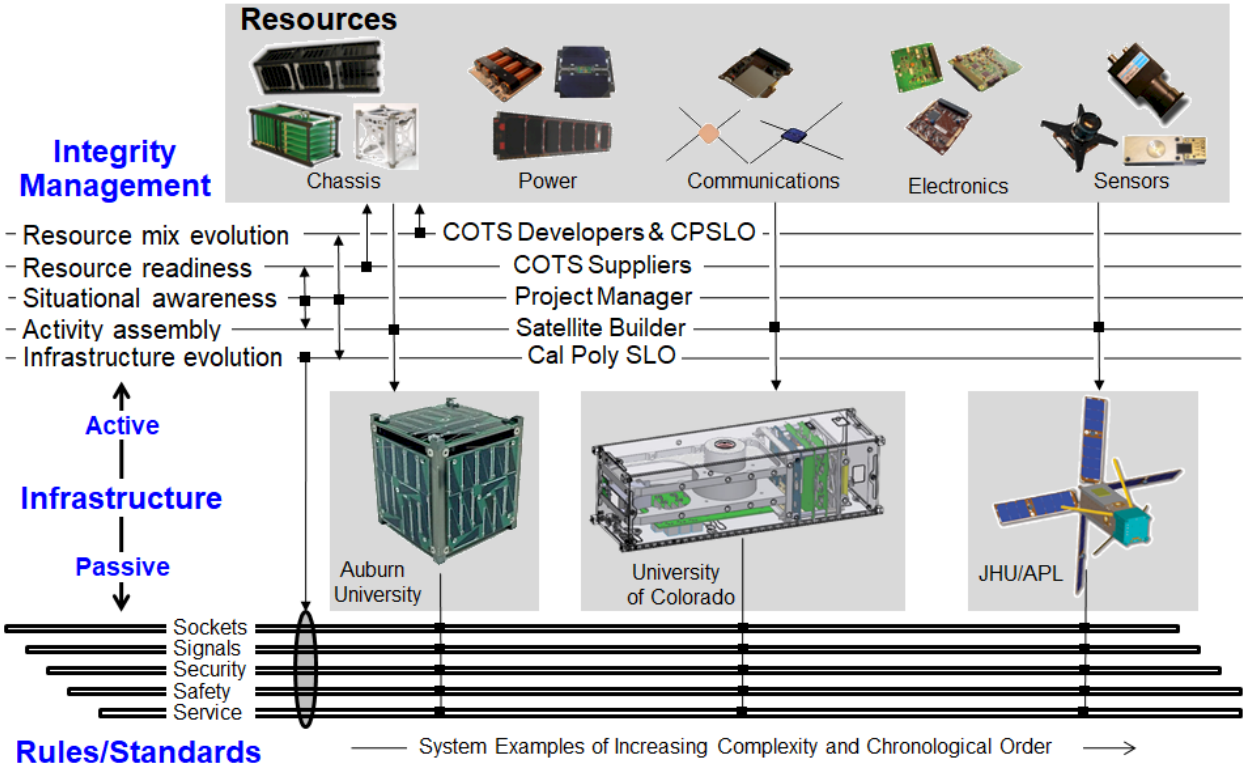

Figure 5. CubeSat Agile Architectural Pattern

The CubeSat specification includes a number of standards and requirements related to the P-POD deployment system, environmental standards for spacecraft launches, range safety, testing,

materials, and orbital debris. Launch vehicle operators may levy additional requirements on CubeSat developers in order to insure the safety of the launch vehicle, and other satellites that are to be deployed. In the past a variety of launch vehicles have been used to deploy CubeSats including Russian Kosmos-3M and Dnepr rockets, SpaceX Falcon 9 rockets, United Launch Alliance Delta II rockets, Orbital Science Minotaur IV and Antares rockets, and the International Space Station.

The active infrastructure of the CubeSat Project is supported through the international collaboration of over 100 universities, private companies and Government agencies responsible for the development of satellites. The system assembly role in the active infrastructure is played by universities, commercial organizations and Government agencies that are designing and developing CubeSats. On occasion these organizations may also play the role of resource mix evolution and resource supplier as they leverage their past experiences in developing CubeSats for future projects. The principal role players of resource mix evolution are COTS device developers plus the California Polytechnic State University as the developer of the CubeSat and P-POD specifications. More indirectly, launch vehicle operators and Government agencies responsible for licensing communications bandwidth affect resource mix evolution. A number of commercial vendors fill the role of resource readiness by providing a wide variety of off-the-shelf items that can be used to build a CubeSat. Finally, the role of infrastructure evolution falls principally to a team at Cal Poly San Louis Obispo (CPSLO) that publishes the evolution of design specifications (CubeSat 2013).

The passive infrastructure of the CubeSat Project is supported by the various specifications and standards that have been published by the California Polytechnic State University and others. The CubeSat specification defines a set of physical, mechanical, electrical, magnetic, and operational requirements that a satellite must meet in order to be plug and play compatible with the P-POD delivery system and the various launch vehicles. These specifications form the basis of the “Sockets” and “Services” portions of the passive infrastructure. The “Safety” portion of the passive infrastructure is supported by the combination of the CubeSat specification, a number of Government standards, and additional requirements imposed by launch vehicle operators. For example the CubeSat specification requires that satellites incorporate battery charge/discharge protection to avoid hazardous cell conditions that might endanger the launch vehicle or other CubeSats in the same P-POD. The “Signals” portion of the passive infrastructure is supported by the combination of the CubeSat specification and a set of Government regulations on the transmission of data using RF bandwidth. For example the CubeSat specification requires that CubeSat operators obtain and provide documentation of proper licenses for use of radio frequencies prior to launch. The only portion of the passive infrastructure that is not currently supported by the CubeSat Project is “Security”. The security of each CubeSat is left up to the developer and operator of the satellite.

Conclusion

This article is Part 1 of a two part article on agile systems engineering. This part deals with agile-systems engineering, a necessary precursor for understanding agility in agile systems-engineering, as an agile systems-engineering process is itself an agile system.

Unique to this article is the historical review of agile system definition, research, and concept development; and the recognition of David Albert’s extensive work in Agile C4I and military enterprise as compatible. Also unique, but intended as the practitioner’s take-away, is the model

of agile-systems engineering as the engineering of a system construction kit; the introduction of the CURVE framework; the updated and augmented articulation of the agile architectural pattern, the ten agile system design principles, and the eight response situation analysis domains. The introduction of the CubeSat agile system example in this article will play a role in Part2, when the agile systems-engineering process at John Hopkins University Applied Physics Laboratory (JHU/APL) for developing CubeSats is examined in some detail.

Part 2 of this article will suggest a parallel between Peter Checkland's Soft Systems Methodology and the situation faced when an agile systems-engineering process is appropriate; introduce a life cycle framework for agile systems-engineering; employ the fundamental agile concepts of Part 1 to examine the source of agility in the process known as Scrum for managing agile software development; employ the Part 1 fundamentals to examine JHU/APL's agile systems-engineering process for developing CubeSat hardware/software systems; and finally, suggest a method for developing a domain-independent agile systems-engineering life cycle model.

Acknowledgements

The authors want to thank Jim Highsmith, Rock Angier. and INCOSE Fellow Ron Carson in particular, as well as JHU/APL and unknown submission reviewers, for meaningful critical comment and improvement advice. Some of these suggestions could not be addressed appropriately within the constraints of this publication, but they warrant, and will guide, attention in subsequent opportunities.

References

Alberts, David S. 1996 revised 2002. *Information Age Transformation – Getting to a 21st Century Military*. DoD Command and Control Research Program (CCRP). www.dodccrp.org/html4/books_downloads.html.

Alberts, David S. 2011. *The Agility Advantage: A Survival Guide for Complex Enterprises and Endeavors*. DoD Command and Control Research Program (CCRP). www.dodccrp.org/html4/books_downloads.html.

Aven, Terje and Bodil S. Krohn. 2014. A New Perspective on how to understand, assess and manage risk and the unforeseen. Reliability Engineering and System Safety. Reliability Engineering and System Safety, 121, January, pp 1–10.

Boss, Jason and Rick Dove. 2010. Agile Aircraft Installation Architecture In a Quick Reaction Capability Environment. INCOSE International Symposium, Chicago, IL, July 12-15.

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

Cebrowski, Arthur K. 2003. *Military Transformation: A Strategic Approach.* U.S. Department of Defense, Office of Force Transformation. *www.dau.mil/pubscats/pubscats/atl/2004_05_06/str-mj04.pdf.*

CubeSat. 2012. Past Launches. www.cubesat.org/index.php/missions/past-launches.

CubeSat. 2013. Revision 13 CubeSat Design Specification Provisional Release, August 19, 2013. www.cubesat.org/index.php/documents/developers.

Dennett, Daniel C. 1995, *Darwin's Dangerous Idea – Evolution and the Meanings of Life*. Simon & Schuster.

Dove, Rick, Mel Pirtle, and Dave Wilczynski. 1987. An Overview of FLEXIS - A Methodology for the Design of Flexible Control Systems. Tutorial, Autofact Conferance, Nov 1987, Detroit, MI.

Dove, Rick and Roger Nagel (Principle Investigators). 1991. *21st Century Manufacturing Enterprise Strategy – An Industry-Led View* (Volume 1) and – *Infrastructure* (Volume 2). Eds: S. Goldman and K. Preiss. Diane Publishing Company. www.parshift.com/s/21stCenturyManufacturingEnterpriseStrategy-Volume1.pdf, www.parshift.com/s/21stCenturyManufacturingEnterpriseStrategy-Volume2.pdf.

Dove, Rick. 1992. The 21st Century Manufacturing Enterprise Strategy or What is All This Talk about Agility? Invited paper originally published by Paradigm Shift International (December) and then translated into Japanese and published in a 1993 issue of *Prevision*, the Japan Management Association Research Institute.

Dove, Rick. 1993a. *Beginning the Agile Journey – A Guidebook.* Hewlett Packard. www.parshift.com/Files/PsiDocs/Pap930701Dove-BeginningTheAgileJourney-A Hewlett Packard Guidebook.pdf.

Dove, Rick. 1993b. Lean and Agile: Synergy, Contrast, and Emerging Structure. Defense Manufacturing Conference '93, San Francisco, CA, November 29 - December 2.

Dove, Rick. 1994. Best Agile Practice Reference Base - 1994: Challenge Models and Benchmarks. Proceedings: 4th Annual Agility Conference, Agility Forum, Bethlehem, PA., March. www.parshift.com/Files/PsiDocs/Rkd5Art1.pdf.

Dove, Rick, Steve Benson, William Drake, Anthony Fiore, David Goldman, H.T. Gorenson, Susan E. Hartman, H. Van Dyke Parunak, Sal Scaringella, Raja Seshadri, Brian J. Turner. 1995. Agile Practice Reference Base. Agility Forum Report AR95-02. Agility Forum, Bethlehem, PA.

Dove, Rick, Sue Hartman, and Steve Benson. 1996. An Agile Enterprise Reference Model, With a Case Study of Remmele Engineering. Agility Forum Report, December. www.parshift.com/Files/PsiDocs/AerModAll.pdf.

Dove, Rick. 1998. Realsearch: A Framework for Knowledge Management and Continuing Education, IEEE Aerospace Conference, March 1998. www.parshift.com/Files/PsiDocs/RealsearchIEEE.pdf.

Dove. Rick. 2001. *Response Ability – The Language, Structure, and Culture of the Agile Enterprise*. Wiley.

Dove, Rick. 2005. Fundamental Principles for Agile Systems Engineering. Conference on Systems Engineering Research (CSER), Stevens Institute of Technology, Hoboken, NJ, March. www.parshift.com/Files/PsiDocs/Rkd050324CserPaper.pdf.

Dove, Rick. 2009. Pattern recognition without tradeoffs: scalable accuracy with no impact on speed. Proceedings Cybersecurity Applications and Technology Conference for Homeland Security, IEEE Computer Society, March 3-4, 2009.

Dove, Rick, 2011, Self-Organizing Resilient Network Sensing (SornS) with Very Large Scale Anomaly Detection, IEEE International Conference on Technologies for Homeland Security, Waltham, MA, Nov. 15-17.

Dove, Rick and Ralph LaBarge. 2014. Agile Systems Engineering – Part 2. International Council on Systems Engineering IS14 Conference, Los Vegas, NV, 30-Jun-03Jul. www.parshift.com/s/140630IS14-AgileSystemsEngineering-Part2.pdf

DSB (Defense Science Board). 2009. Report of the Defense Science Board Task Force on Fulfillment of Urgent Operational Needs. Office of the Under Secretary of Defense for Acquisition, Technology, and Logistics. Washington, DC.

Fowler, Martin and Jim Highsmith. 2001. The Agile Manifesto. Dr. Dobb's Journal, August. 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.

Heidt1, Hank, Jordi Puig-Suari, Augustus S. Moore, Shinichi Nakasuka, Robert J. Twiggs. 2000. CubeSat: A new Generation of Picosatellite for Education and Industry Low-Cost Space Experimentation. 14TH Annual/USU Conference on Small Satellites. Utah State University, Logan, UT, 21-24 August.

Huang, Philip M., Andrew A. Knuth, Robert O. Krueger, and Margaret A. Garrison-Darrin. 2012. Agile hardware and software systems engineering for critical military space applications. In SPIE Defense, Security, and Sensing, pp. 83850F-83850F. International Society for Optics and Photonics.

Klinke, Andreas and Ortwin Renn. 2002. A New Approach to Risk Evaluation and Management: Risk-Based, Precaution-Based, and Discourse-Based Strategies. Risk Analysis, Vol. 22, No. 6.

Knight, Frank H. 1921. *Risk, Uncertainty and Profit.* Hart, Schaffner & Marx. Full text available at: www.econlib.org/library/Knight/knRUPCover.html.

Lawson, Harold 'Bud'. 2010. *A Journey Through the Systems Landscape*. College Publications.

Nugent, Ryan, Riki Munakata, Alexander Chin, Roland Coelho, and Dr. Jordi Puig-Suar. 2008. The CubeSat: The Picosatellite Standard for Research and Education. AIAA Space 2008 Conference and Exposition, 9-11 September 2008, San Diego, CA.

Orton, J. Douglas, and Karl E. Weick.1990. Loosely coupled systems: A reconceptualization. Academy of management review 15, no. 2: 203-223.

Papke, Barry and Rick Dove. 2013. Combating Uncertainty in the Work Flow of Systems Engineering Projects. INCOSE International Symposium, Philadelphia, PA, June 24-27. Best paper award.

SAF (Secretary of the Air Force). 2011. Air Force Instruction 63-114, Quick Reaction Capability Process. 4 January. Washington, DC.

Sillitto, Hillary G. 2013. Composable Capability – Principles, Strategies and Methods for Capability Systems Engineering. INCOSE International Symposium, Philadelphia, PA 24-27 June.

Weick, Karl E., Kathleen M. Sutcliffe, and David Obstfeld. 1999. Organizing for High Reliability: Processes of Collective Mindfulness. R.S. Sutton and B.M. Staw (eds), Research in Organizational Behavior, Volume 1. Stanford: Jai Press, pp. 81–123.

Biography

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

Ralph LaBarge is a Principal Professional Staff member of The Johns Hopkins University Applied Physics Laboratory where his experience spans systems engineering, digital signal processing and cyber security. He received master's degrees in Computer Science, Electrical Engineering, and Information Assurance from The Johns Hopkins University, and a bachelor's degree in Electrical Engineering from the University of Delaware. He is currently enrolled in a doctoral program at George Washington University in systems engineering.