Agile Systems Engineering Life Cycle Model for Mixed Discipline Engineering

Western States Regional Conference

13-15 September 2019, Los Angeles, CA

Rick Dove

Paradigm Shift International

Chair, INCOSE Agile Systems & Systems Engineering Working Group

Abstract: This presentation exposes eight key findings from INCOSE's Agile Systems Engineering Life Cycle Model (ASELCM) discovery project: problem-space characterization, life cycle model framework, operating principles, a pattern of three concurrent behavioral systems, the concept of information debt, general response requirements, stakeholder engagement, and continuous integration platforms. These findings will be preceded by earlier discoveries that establish necessary architecture to enable operational agility.

Speaker Bio: Rick Dove is a researcher, practitioner, and educator of fundamental principles for agile enterprise, agile systems, and agile development processes. In 1991 he initiated the global interest in agility as co-PI on the seminal 21st Century Manufacturing Enterprise Strategy project at Lehigh University. Subsequently he organized and led collaborative research at the DARPA-funded Agility Forum, involving 250 organizations and 1000 participants in workshop discovery of fundamental enabling principles for agile systems and processes.

Rick is CEO of Paradigm Shift International, specializing in agile systems research, engineering, and education; and is 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 current INCOSE Agile Systems Engineering Life Cycle Model Discovery Project. He is an INCOSE Fellow, and the author of *Response Ability – the Language, Structure, and Culture of the Agile Enterprise*.

Agile Systems Engineering Life Cycle Model (ASELCM)

An INCOSE Project to…

- ❑ **Discover generic principles/patterns that are necessary for effective agile systems engineering of SW/FW/HW projects**
- ❑ **Publish informative case studies**
- ❑ **Build evidence-based generic agile-SE life cycle model to inform effective implementation**

And …

- ❑ **Provide material for next INCOSE Handbook revision**
- ❑ **Influence published standards**

Value Proposition for Agile Systems Engineering

Faster, lower cost system development?

An appealing argument, at the business level.

But to achieve this,
a different value proposition is needed at the engineering level:

Minimization of project risk and rework.

Defining Agile Systems Engineering

Need:

Effective system engineering in the face of uncontrolled change.

Intent:

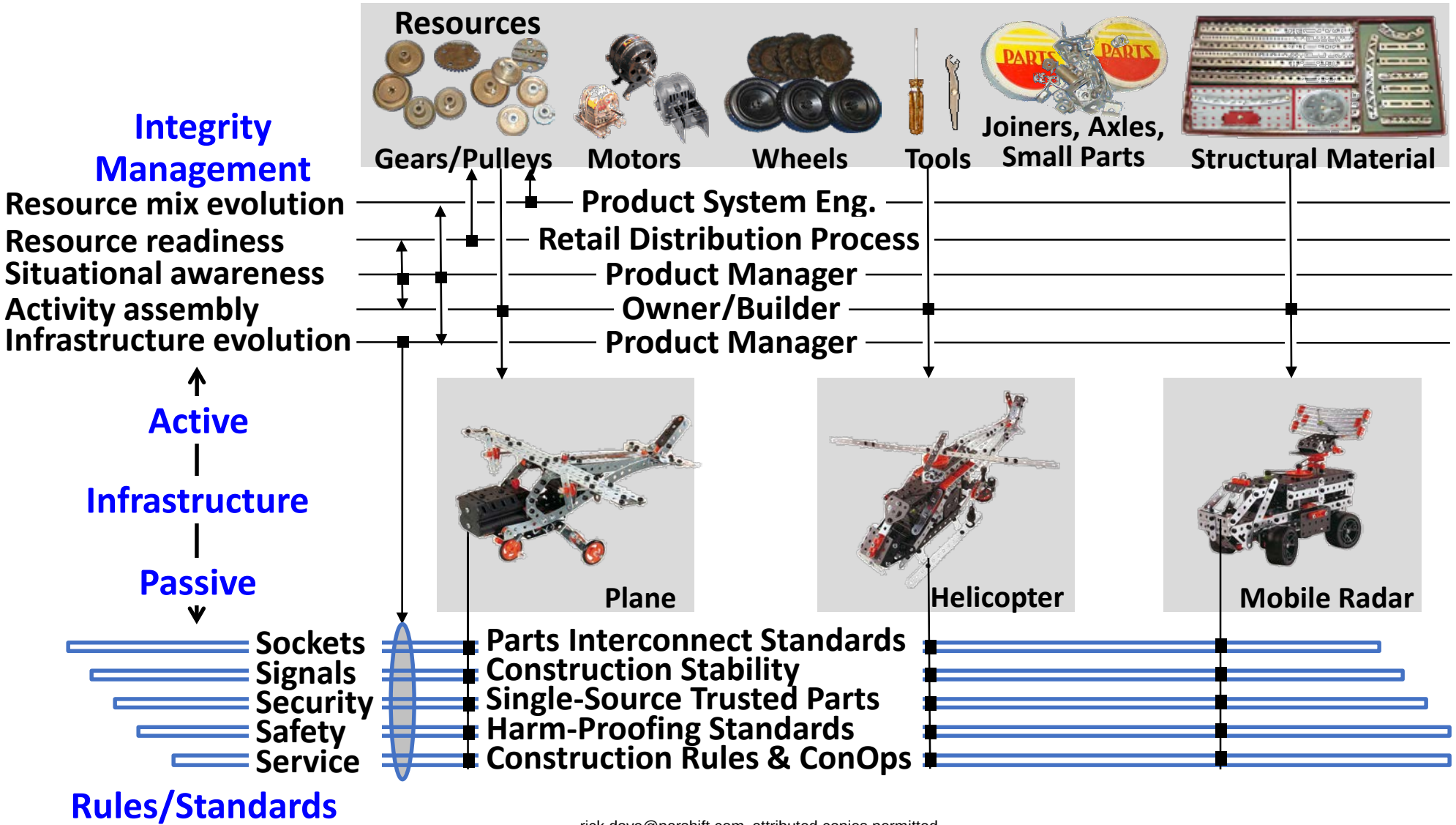
Effective response to a systems engineering operational environment that is capricious, uncertain, risky, variable, and evolving. This intent defines agile systems engineering.

The definition of agile systems engineering is rooted in what it does, not how it does it.

There are many ways to accomplish the how at the project and engineering discipline level.

Iconic Agile Architecture Pattern (AAP)
Notional Concept: System Response-Construction Kit

Details in www.parshift.com/s/140630IS14-AgileSystemsEngineering-Part1&2.pdf



Sustaining Agility Requires …

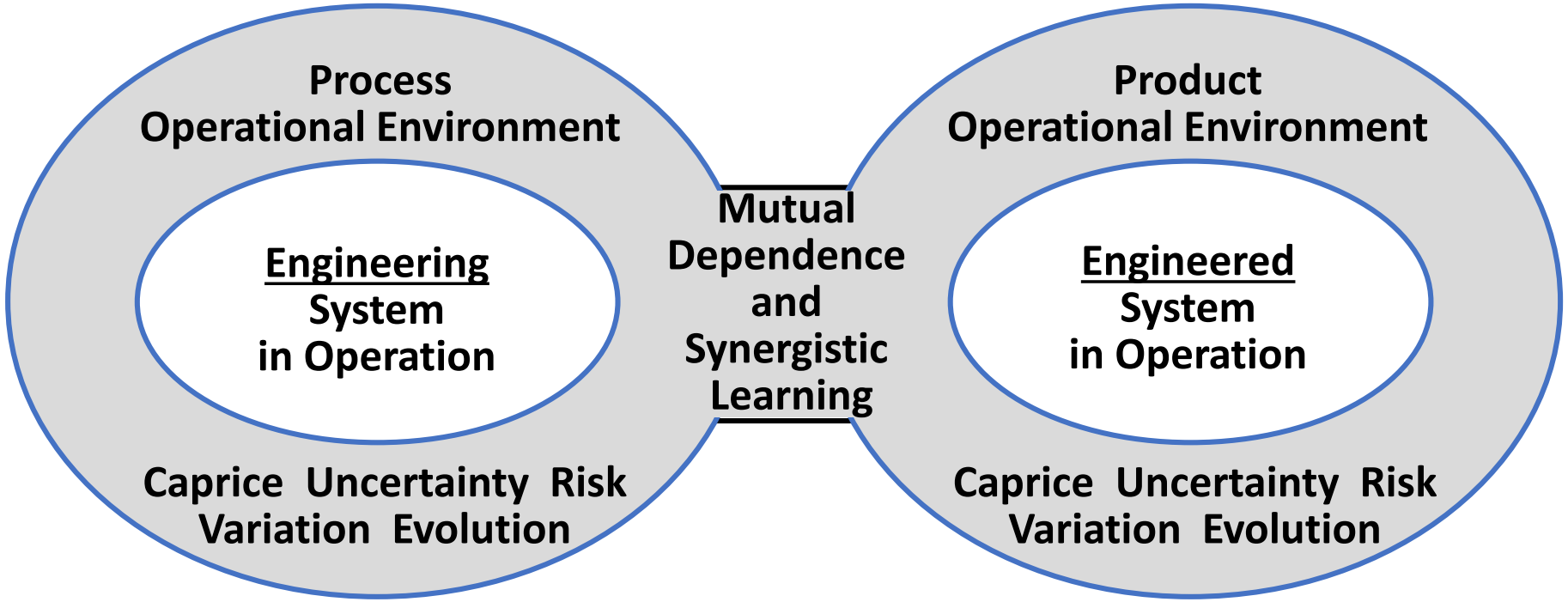
- Proactive awareness of situations needing responses
- Effective options appropriate for responses
- Assembly of timely responses

Five Agility-Sustaining Responsibilities:

1. Resource Mix Evolution
2. Resource Readiness
3. Situational Awareness
4. Response Assembly
5. Infrastructure Evolution

Two different systems with synergistic dependencies

(a first principle)



You can't have
an agile engineering process
if it doesn't engineer an agile product
(and vice versa)

ASELCM Project Findings

An IS19 paper discusses:

1. **Agile SE Life Cycle Model Framework**
2. **ASELCM Pattern of Three Concurrent Systems**
3. **CURVE Framework Characterizing the Problem Space**
4. **Operational Principles**
5. **Concept of Information Debt**
6. **General Agile SE Response Requirements**

Above covered in the IS19 paper:

www.parshift.com/s/ASELCM-05Findings.pdf

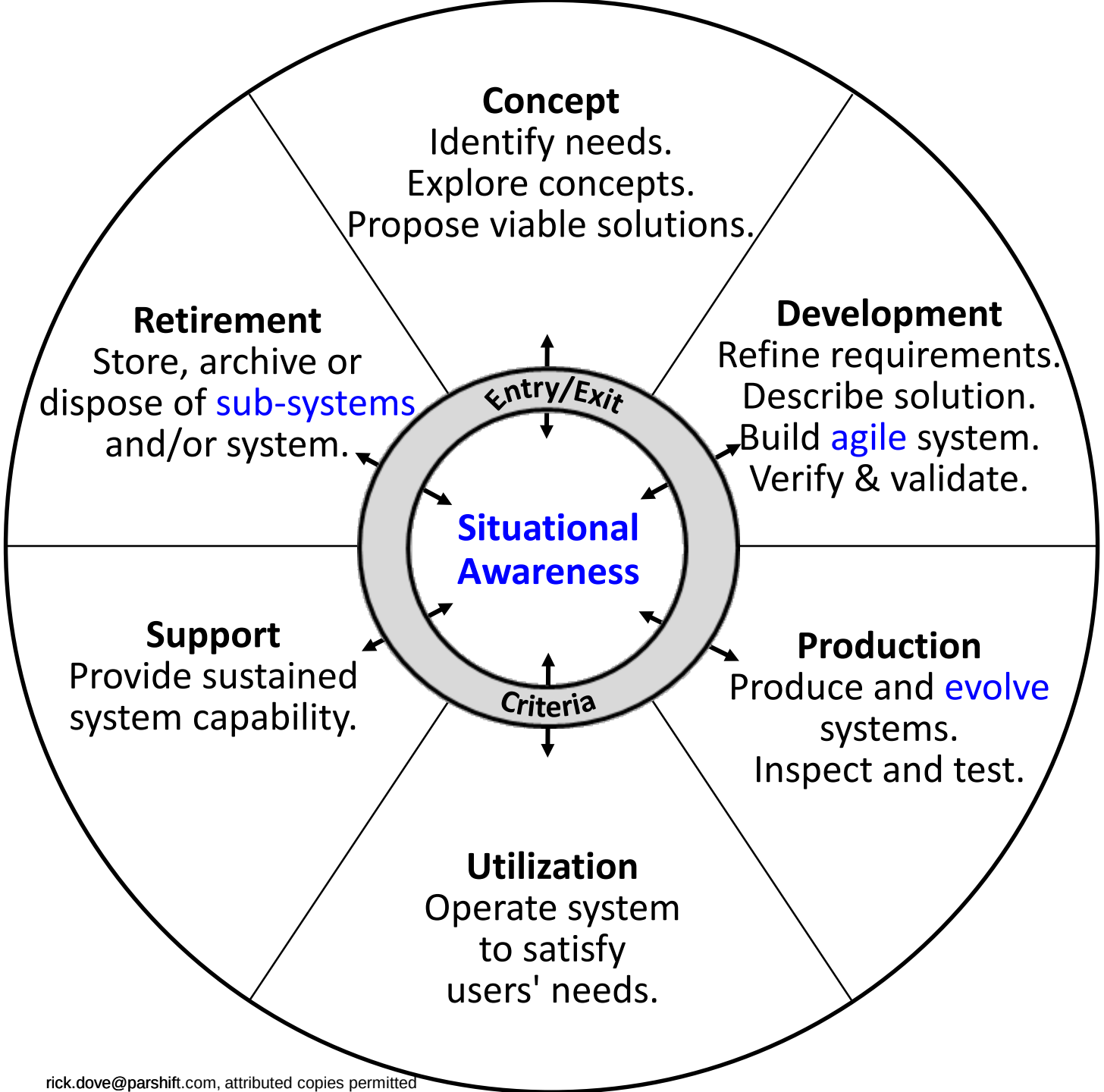
Here we add a 7th and 8th finding, with some focus:

7. **Stakeholder Engagement**
8. **Continuous Integration Platform**

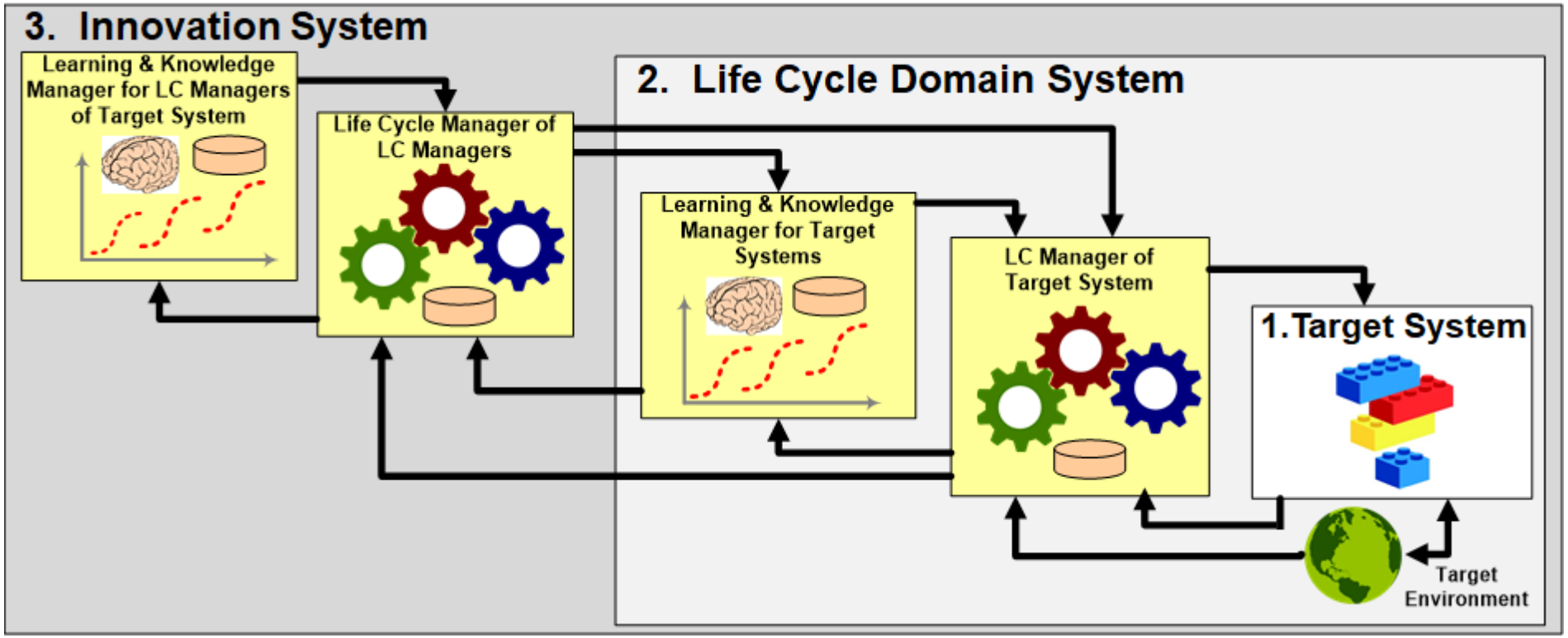
1. Agile SE Life Cycle Model Framework

Asynchronous/Concurrent Stages.
Consistent with
ISO/IEC/IEEE 24748-1:2018

Situational Awareness Engages System Evolution Stages/Tasks



2. ASELCM Pattern of Three Concurrent Systems



- System-1 is the target system under development.
- System-2 includes the basic systems engineering development and maintenance processes, and their operational domain that produces System-1.
- System-3 is the process improvement system, called the system of innovation that learns, configures, and matures System-2.

The Innovation System is responsible for situational awareness and evolution, the provider of operational agility. Intent is continuous, not episodic, info flow.

3. CURVE Framework for Characterizing the Problem Space

Internal and external environmental forces that impact process and product as systems

Caprice: unanticipated system-environment change
(randomness among unknowable possibilities)

Uncertainty: kinetic and potential forces present in the system
(randomness among known possibilities with unknowable probabilities)

Risk: relevance of current system-dynamics understanding
(randomness among known possibilities with knowable probabilities)

Variation: temporal excursions on existing behavior attractor
(randomness among knowable variables and knowable variance ranges)

Evolution: experimentation and natural selection at work
(relatively gradual successive developments)

The goal of agile systems engineering is S2 and S1 compatibility with their CURVEd environments.

The general CURVE shown here is applicable to both S2 and S1.

S2 and S1 have cyber-physical-social dimensions.

General SE CURVE
Caprice
• Survivability (i.e., current order compatibility) • Occurrence and nature of emergent behavior • Game-changing technologies • Availability of symbiotic social relationships
Uncertainty
• Relevance (i.e, appropriate to current desires) • Cohesion in the greater SoSs (multiple) • Integrity and symbiosis of social relationships.
Risk
• Viability (i.e., capable of working successfully) • Cohesion among constituent parts
Variation
• Operational environments • Social compatibility
Evolution
• Toward more operating environment complexity • Toward more SoI complexity • Toward shorter SoI static viability • Toward new technology options • Toward new malevolent threats to viability • Toward greater social involvement.

4. Operational Principles

Sensing (observe, orient)

- External awareness (proactive alertness)
- Internal awareness (proactive alertness)
- Sense making (risk & opportunity analysis, trade space analysis)

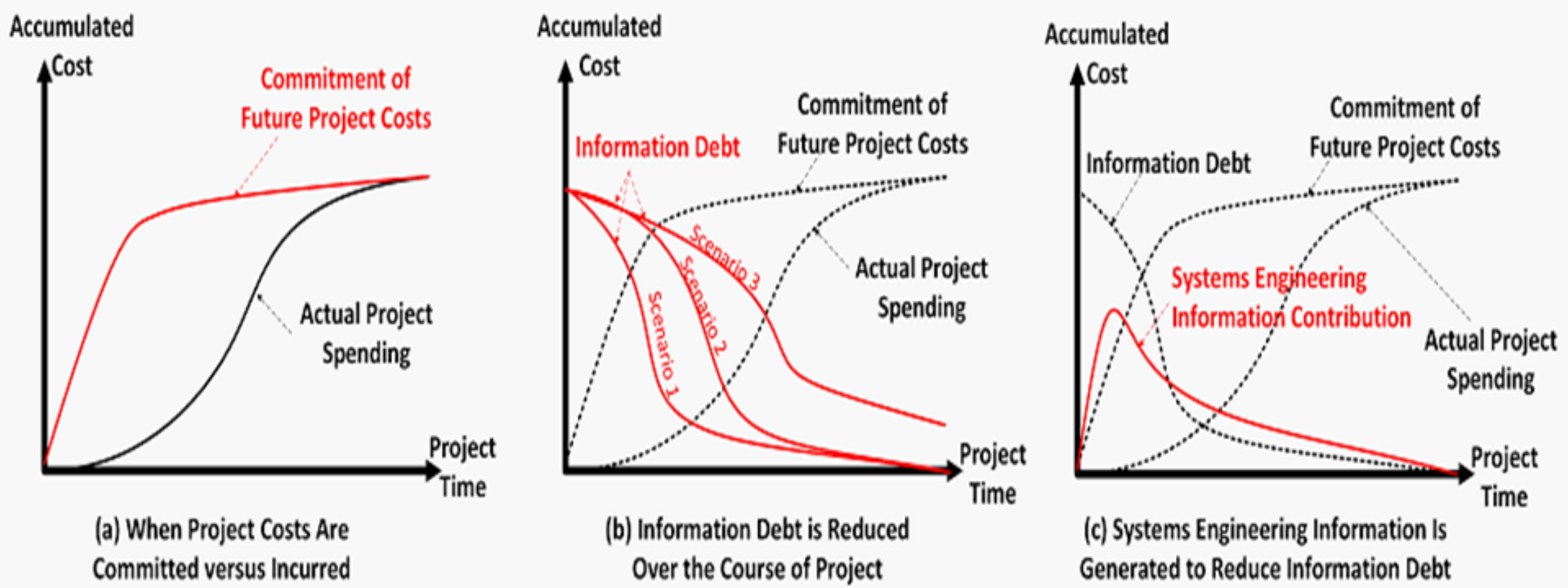
Responding (decide, act)

- Decision making (timely, informed)
- Action making (invoke/configure process activity for the situation)
- Action evaluation (validation & verification)

Evolving (improve above with more knowledge and better capability)

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

5. Concept of Information Debt



Future costs of a project become committed early by SE decisions. One of the traditional arguments for early stage SE investment.

Will project end with outstanding information debt: a "working system" but an interest penalty caused by shortage of needed information?

SE information must be generated (e.g., reqs, architectures, risk assessments, etc.) early enough in the project.

6. General Agile SE Response Requirements

Domain		Response Requirements
Proactive	Creation	•Opportunity & risk awareness •Response actions/options • Acculturated memory • Decisions to act
	Improvement	•Awareness/Sensing •Memory in culture, options, ConOps • Action/option effectiveness
	Migration	•New fundamentally-different types of opportunities and risks
	Modification (Capability)	•Actions appropriate for needs •Personnel appropriate for actions
Reactive	Correction	•Insufficient awareness •Ineffective actions/options • Wrong decisions
	Variation	•Effectiveness of actions/options •Effectiveness of evaluation
	Expansion (Capacity)	•Capacity to handle 1-? actions simultaneously
	Reconfigu-ration	•Elements of an action •Response managers/engineers

7. Stake Holder Engagement

Developers	Subcontractors	Security Engineers
Operators	Producers	Maintainers
Customers	End Users	Management

Three typical forms of stakeholder engagement:

Integrated product team (IPT) is a multidisciplinary group of people who are collectively responsible for delivering a defined product or process. The emphasis of the IPT is on involvement of all stakeholders (users, customers, management, developers, contractors) in a collaborative forum. (Wikipedia)

Concurrent engineering (CE) is a work methodology emphasizing the parallelization of tasks (i.e. performing tasks concurrently), which is sometimes called simultaneous engineering or integrated product development (IPD) using an integrated product team approach. It refers to an approach used in product development in which functions of design engineering, manufacturing engineering, and other functions are integrated to reduce the time required to bring a new product to market. (Wikipedia)

DevOps is a set of software development practices that combine software development (*Dev*) and information-technology operations (*Ops*) to shorten the systems-development life cycle while delivering features, fixes, and updates frequently in close alignment with business objectives. (Wikipedia)

(Wikipedia access 11-Sep-2019)

Engagement can’t be forced.

Engagement should not be perceived as a time-eating task.

Engagement must provide experiential and take-away value to all involved.

DevOps concepts offer a relevant discussion path.

What is DevOps?

Excerpts from The 7 Principles of DevOps and Cloud Applications. 2015. Gerardo Dade at SolarWinds
www.slideshare.net/SolarWinds/the-7-principles-of-devops-and-cloud-applications

"DevOps is [was born as] a software development practice where development and operation teams work together, taking the intelligence of how an application runs to inform and improve how the application is being built, in a rapid iterative process.

There is no official definition of DevOps, there are many.

DevOps allows to respond faster to customers, fix things faster, produce with more quality. Quality includes performance, security and less bugs.

However, you don't 'do' DevOps, it's not a process, it is a business practice, an approach."

Thus – a look at underlying principles will establish a foundation for an approach.

Seven Principles of DevOps

(slightly modified from Dade's software-focused text)

DevOps is a development practice where development and operation teams work together, taking the intelligence of how a product runs to inform and improve how the product is being built, in a rapid iterative process.

1: End User Focus – It's all about the total end user experience.

2: Collaboration – Develop, Test, and Demo/Run – an integrated process.

3: Performance Orientation – Performance is measured all the time, everywhere.

4: Development Speed – Short, asynchronuous iterative processes accelerate innovation and learning.

5: Service Orientation – Producers and maintainers operate a service to the user.

6: Automation and Repitition – Software configured/controlled regression testing.

7: Monitor Everything – Measure and display what matters.

How do you put these principles into effective practice for mixed discipline systems engineering?

8. Continuous Integration Platforms

Agile SE processes deal with changing knowledge and environment

- **They learn and employ that learning during SE process operation**
- **They modify/augment product-development work-in-process, enabled by an SoI Agile Architecture Pattern (AAP)**

Agile SW development relies on AAP for SoI structure – commercially available

- **Program code development employs an object-oriented development platform (e.g., C++, Java, Eclipse)**
- **Web code development employs a loosely-coupled modularity inherent with hyperlinked web-pages**

Agile HW development doesn't have off-the-shelf integration platforms

- **Proprietary Product-Line-Engineering employs AAP**
- **Proprietary Open System Architecture (OSA) employs AAP**
- **Proprietary Live-Virtual-Constructive employs AAP**

Agile Systems Engineering Goals

produce an innovative result,

produce a “success-assured” result,

produce a sustainable result,

rapidly.

Rework is the bane of Rapid.

Continuous Integration Platform
for mixed-discipline, mixed-supplier projects

Need: Minimize rework.

Intent: An agile Continuous Integration Platform (CIP),
that enables and facilitates…

- An asynchronous continuous test capability (less rework).
- Early detection of integration issues (less rework).
- WIP feedback demos to users/customers/management/suppliers (less rework).
- DevOps/DevSecOps/IPT/CE collaborative development interaction (less rework).
- Alternative/prototype experimentation (less rework).
- A set-based knowledge-development test stand (less rework).

Less rework is a value common to all engineering disciplines.

Boeing's F-22 team accelerates avionics modernization with new approach

The F-22 Agile Integration Lab interlinks the 757 Flying Test Bed with a ground-based test and evaluation facility. This provides the capability to test several avionics software versions dynamically during a single six-hour flight.

"It's not a stretch to say we can accomplish in a day what used to take a month."

They upload multiple software versions into the flying test bed's workstations, test them during a single six-hour flight, land and park the airplane beside the ground-based lab, reconnect the umbilical, and certify the software updates in concert with F-22 systems that don't need to fly on the test bed: e.g. weapons, engines and flight controls.

They use existing assets to provide more robust test and evaluation capability at lower cost and shorter delivery time.

Cantwell, Doug. 2008. F-22 Team Accelerates Avionics Modernization with New Approach. Boeing Frontiers, Goin' Agile.
www.boeing.com/news/frontiers/archive/2008/july/i_ids05.pdf

SpaWar Continuous Integration Platforms

RaDER
Reconnaissance and Detection Expendable Rover

Two of the Integration Platforms

(for autonomous off-road vehicle technology)

Full system test and demo every 6 months, with next cycle adding new features.

Asynchronous testing of wip within the 6-month cycle frequently.

Platforms are instrumented to detect integration problems early (e.g., a wip device from a subcontractor hogging to much bandwidth or CPU cycles).

SE team evolves the platform architecture every cycle to accommodate new needs.

Both warfighters and sponsors witness end-of-cycle tests and demos, and often show up during a cycle for wip demos.

EV1
Expeditionary Vehicle 1

Rockwell Collins Continuous Integration Platform

Cedar Rapids military radio projects

The focus is on the evolution of firmware-containing circuit cards needed by software development for incremental testing during sprint iterations, and especially at three-month increment testing events.

Four elements are evolved, to accommodate this:

- **Integrated computing platform (ICP): a Rockwell-built scalable circuit card rack with supporting power and cabling that can accommodate multiple circuit cards, and interface with external devices and computers.**
- **Commercially available system-on-chip prototype boards.**
- **Product Line component inventory: Rockwell-built circuit cards are readily available as either actual end-product reusable cards or sufficiently similar to act as proxies for early software interface testing.**
- **Hardware chassis are either drawn from the product line inventory or developed new, with employment of an inventoried LRU favored for early physical form.**

ICP hardware evolves continuously and asynchronously with software sprints.

Lockheed IFG Continuous Integration Platform

In 2015 IFG was in early experimentation with a CIP concept, called the Agile Non-Target Environment (ANTE).

ANTE systems consist of simulated components, previous re-usable components, wip components, finished components, low-fidelity COTS proxies, IFG software work-in-process, and operators.

Of note: ANTE employs lower-fidelity open-market proxy devices with similar capability but lower performance than what is eventually expected.

Subcontractors are required to provide device simulations to ANTE specs.

ANTE concept was self funded for values they expected and realized.

By mid-2017 ANTE was declared a successful experiment, and had achieved eventual applause in customer feedback that values:

- **Early and incremental demonstration of working concepts.**
- **Early exposure to difficulties in need of attention.**

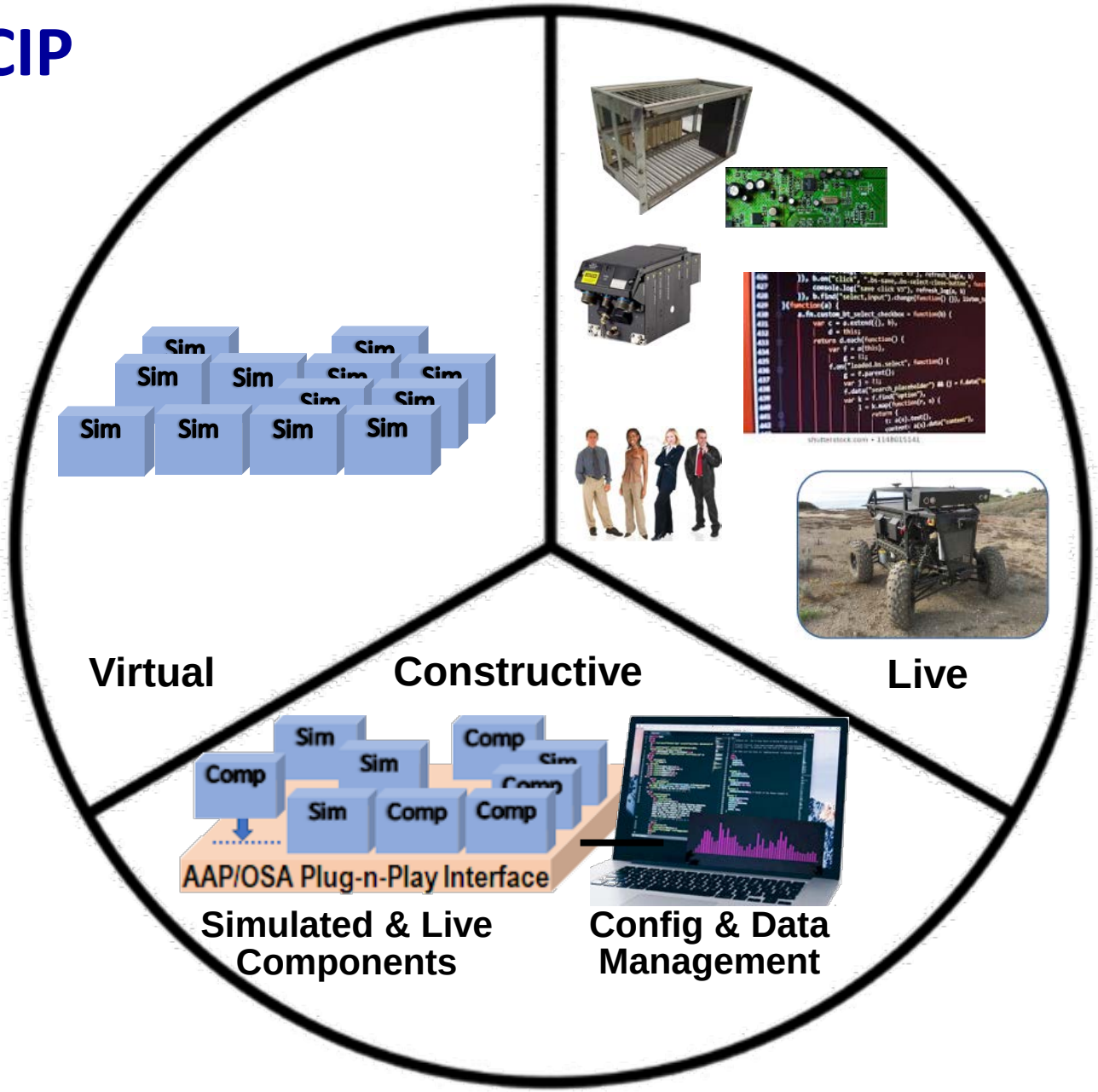
Live Virtual Constructive CIP

Live components
(people, things)
Virtual components
(simulations)
Constructive capabilities
(configuration and data management)

L&V components are functional system elements; configured, challenged and monitored by C elements for performance and anomalies.

LVC/CIP… demonstration/test/experimental events can occur at any time with the latest instantiation of simulations & components.

Caveat: LVC internet search is dominated by military training applications.



CIP: Continuous Integration Platform
AAP: Agile Architecture Pattern
OSA: Open Systems Architecture

You Can't Buy It – You Have to Design/Build/Evolve It

Start Affordably and Evolve Incrementally

Establish preliminary needs and intents (purposes and goals)

Publish preliminary Agile Architectural Pattern: define/evolve plug-and-play 5s infrastructure interface specs

- **Sockets: physical interconnect**
- **Signals: data/stuff interconnect**
- **Security: trust interconnect**
- **Safety: environment interconnect (relative to internal and external safe operation)**
- **Service: user interconnect (ConOps and OpsCon)**

Consider preliminary simulation stubs (accept requests & data, reply with temporary pro-forma responses)

Consider CIP-operation configuration management (for tests and demos)

Consider performance instrumentation (conflict detection, operational results)

Consider visual stakeholder interface (for collaborative DevOps and wip Demos)

Consider re-use of available SIL equipment, previously developed components, COTS temporary proxies

Consider automated regression testing (retest everything tested before and add new tests cumulatively)

Consider physical as well as cyber interfaces

Consider set-based data accumulation over time

Consider ghosting – comparing outputs of one component with a potential replacement

Early & frequent wip stakeholder feedback should illuminate rework cost avoidance/reduction

Appropriate Context

CIP is a

Cyber-Physical-Social system

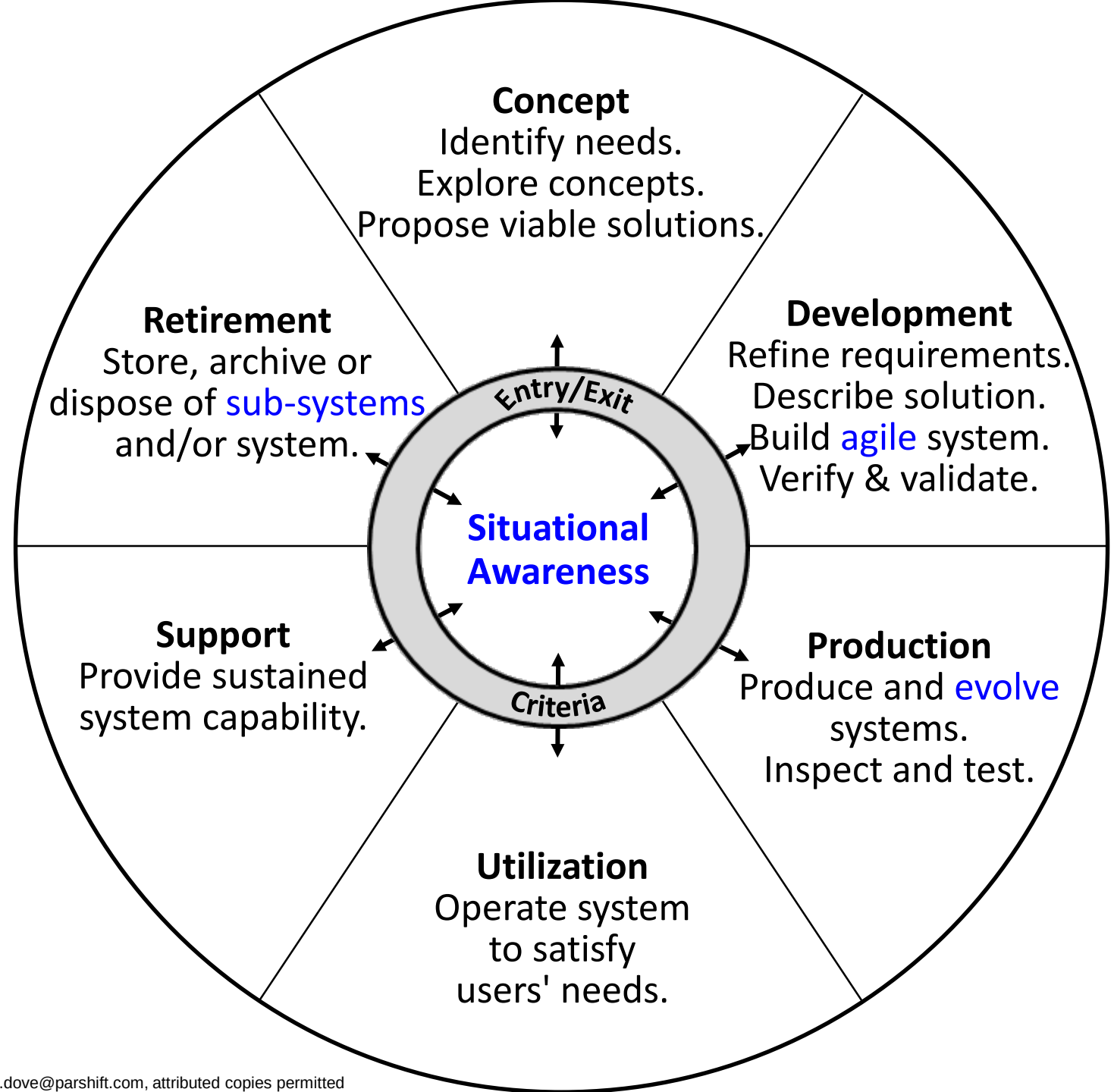
In general, the social dimension of all systems needs more appreciation and attention.

The social dimension includes:

1. **Stakeholder engagement.**
2. **Symbiosis with the greater SoS.**
3. **Compatibility with the operating environment.**

Another story for another time.

Agile SE Life Cycle Model Framework



Situational Awareness Engages System Evolution Stages/Tasks

Studies Supporting the Findings

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

Introduction to the Agile Systems Engineering Life Cycle MBSE Pattern. Schindel, W., R. Dove. International Council on Systems Engineering. International Symposium, Edinburgh, Scotland, July 18-21, 2016. www.parshift.com/s/160718IS16-IntroToTheAgileSystemsEngineeringLifeCycleMBSEPattern.pdf

Case Study: Agile systems engineering process features collective culture, consciousness, and conscience at SSC Pacific Unmanned Systems Group. Dove, R, W. Schindel, C. Scrapper. International Council on Systems Engineering. International Symposium, Edinburgh, Scotland, July 18-21, 2016. www.parshift.com/s/ASELCM-01SSCPac.pdf.

Case Study: Agile Hardware/Firmware/Software Product Line Engineering at Rockwell Collins. Dove, R., W. Schindel, R. Hartney. 11th Annual IEEE International Systems Conference. Montreal, Quebec, Canada, April 24-27, 2017. www.parshift.com/s/ASELCM-02RC.pdf

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

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

Agile Systems Engineering Life Cycle Model for Mixed Discipline Engineering. Dove, R., W. Schindel. International Council on Systems Engineering. International Symposium, Orlando, FL, July 20-25, 2019. www.parshift.com/s/ASELCM-05Findings.pdf.