INCOSE
International Symposium
Las Vegas, NV
June 30 - July 3, 2014

Fundamentals of Agile Systems Engineering – Part 2 (Agile Systems-Engineering)

Rick Dove
Paradigm Shift International, and
Stevens Institute of technology

Ralph LaBarge
Johns Hopkins University/APL

INCOSE IS14

Las Vegas, NV, USA

30 June – 3 July, 2014

Agile?

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.

To the INCOSE Agile Systems and Systems Engineering WG, 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.

Operational adaptability may be needed
in the systems engineering process
and
in the systems engineering product

Reminding Part 1…

A system's "bone structure" is depicted in the Agile Architecture Pattern. All truly agile systems have the same basic structure and strategy. Knowing this will change the way you "see" and evaluate a system.

What i'm about to tell you is gonna change your life forever. Are you really sure you want to know it?

http://awespendo.us/animemangacomics/kermit-at-the-doctor/

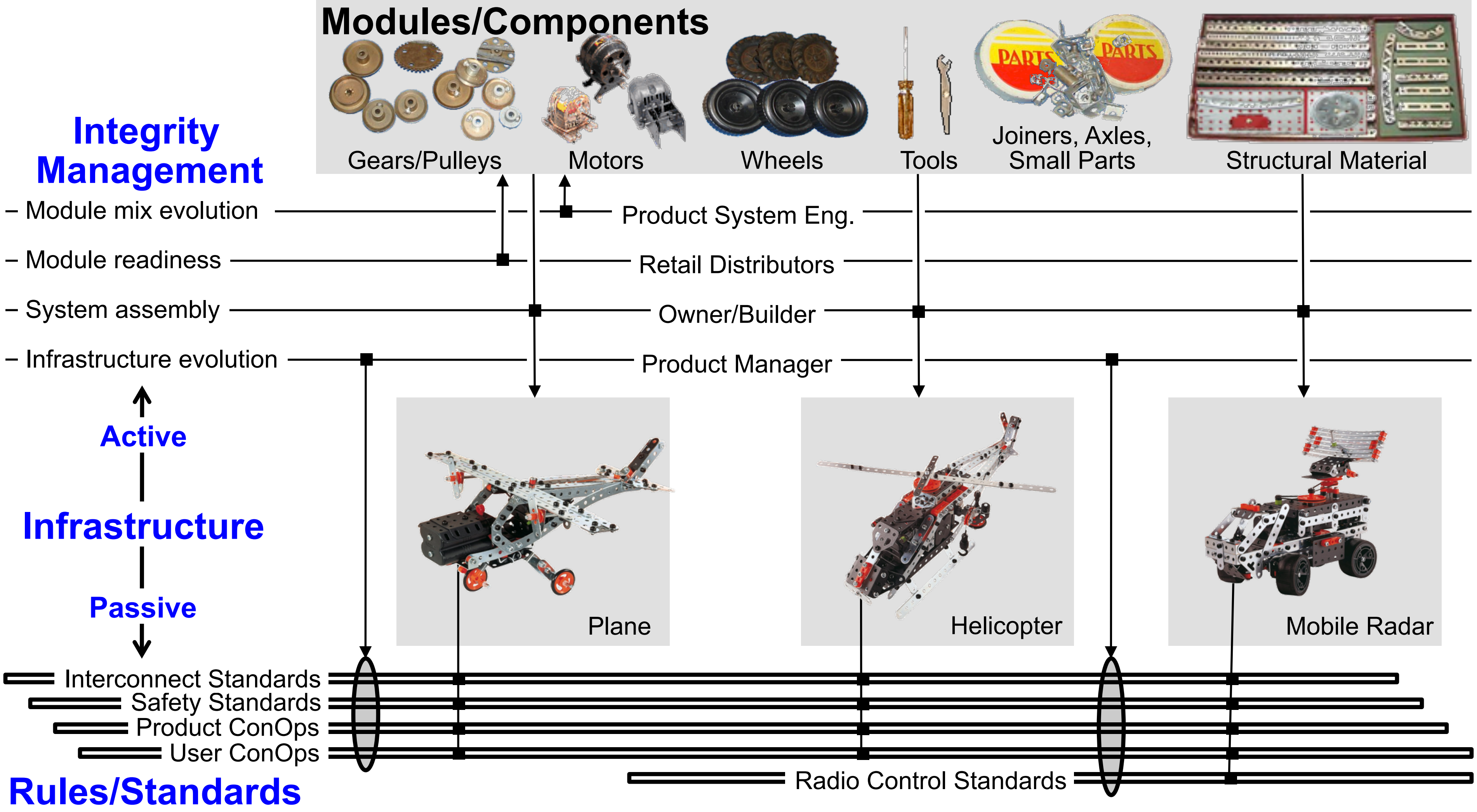
Here’s a Box of Bones

http://www.vintagetoys.com/toys/classified/962
No. 7½ Engineer's Set
Sensational New
ERECTOR
DEVELOPED AT THE GILBERT HALL OF SCIENCE
PARTS
PARTS

ERECTOR=MECCANO

Here is a System-Construction-Kit System

this *agile architecture pattern* provides adaptable structure (Agile 101)



Modules/Components

Gears/Pulleys

Motors

Wheels

Tools

Joiners, Axles, Small Parts

Structural Material

Integrity Management

– Module mix evolution

– Module readiness

– System assembly

– Infrastructure evolution

Product System Eng.

Retail Distributors

Owner/Builder

Product Manager

Active

Infrastructure

Passive

Plane

Helicopter

Mobile Radar

Interconnect Standards

Safety Standards

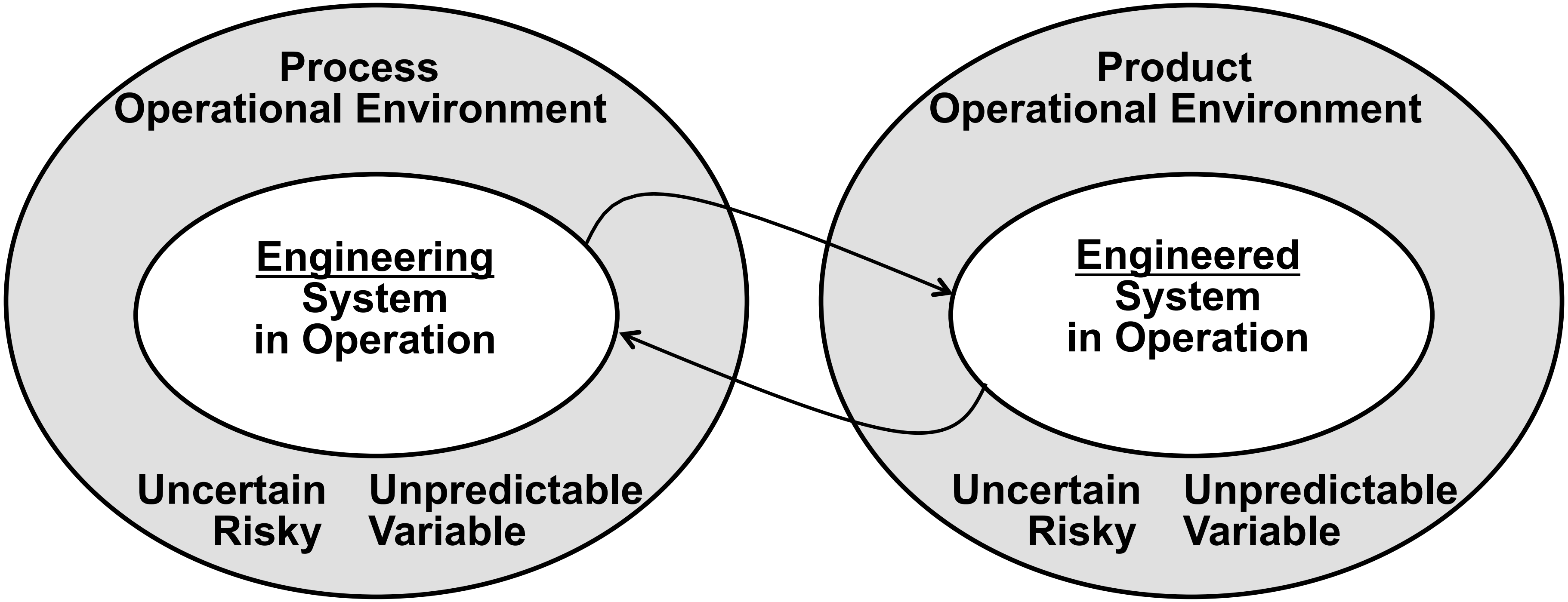
Product ConOps

User ConOps

Radio Control Standards

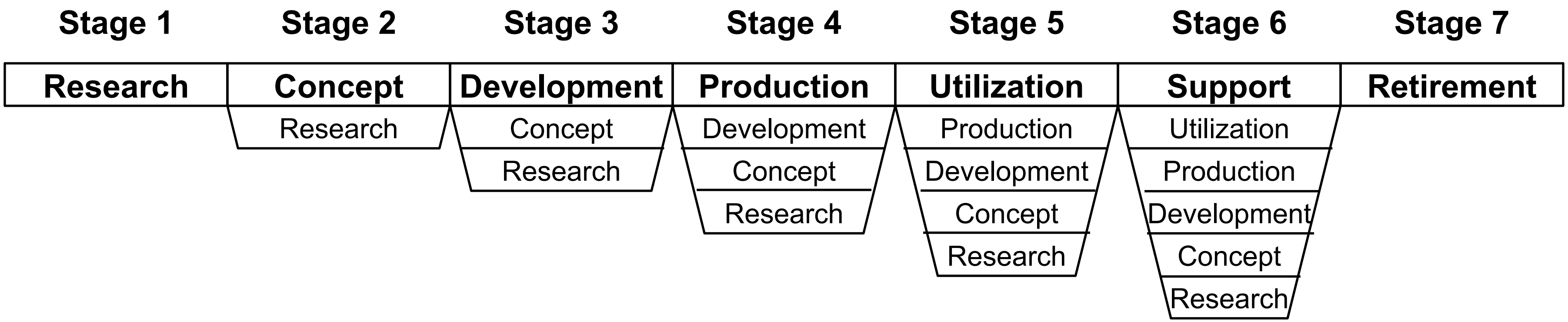
Rules/Standards

Two different operational environments defining necessary agile counterpoint for the systems they encompass



**You can't have
an agile development process
if you don't build an agile product**

Framework of agile system engineering life cycle mode



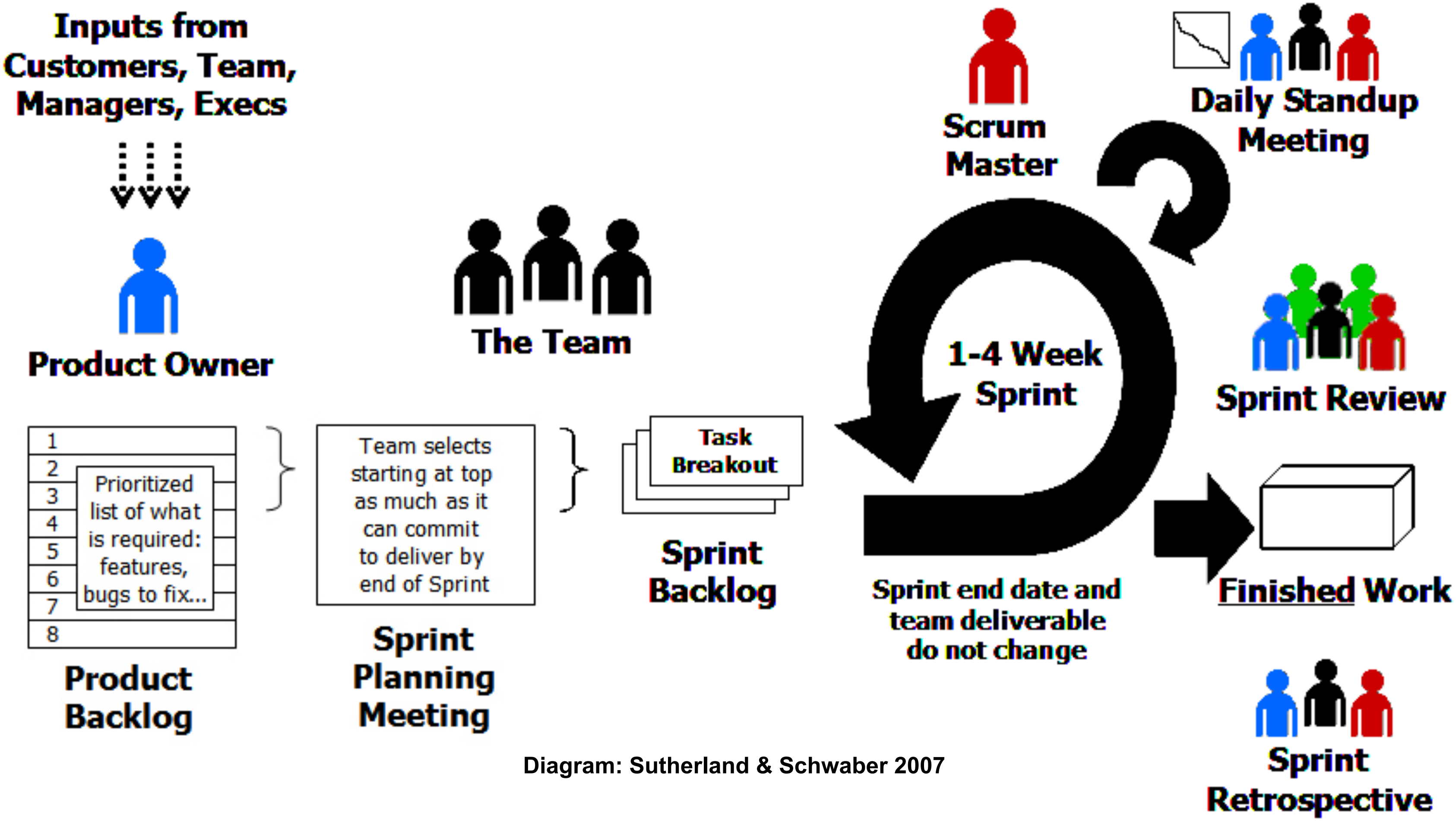
Stage 1	Stage 2	Stage 3	Stage 4	Stage 5	Stage 6	Stage 7
Research	**Concept**	**Development**	**Production**	**Utilization**	**Support**	**Retirement**
	Research	Concept	Development	Production	Utilization	
		Research	Concept	Development	Production	
			Research	Concept	Development	
				Research	Concept	
					Research	

Depicts constant evolution
of all prior ISO/IEC 15288 life cycle stages
as the life cycle progresses through maturity

“Classic” Scrum

Ken Schwaber, Jeff Sutherland. 2011. The Scrum Guide. www.scrum.org/

Jeff Sutherland, Ken Schwaber. 2007. The Scrum Papers: Nuts, Bolts and Origins of an Agile Process. Scrum Foundation. http://scrumfoundation.com



Inputs from Customers, Team, Managers, Execs

Product Owner

The Team

Scrum Master

Daily Standup Meeting

1-4 Week Sprint

Sprint Review

Finished Work

Sprint Retrospective

1

2

3

4

5

6

7

8

Prioritized list of what is required: features, bugs to fix...

Product Backlog

Team selects starting at top as much as it can commit to deliver by end of Sprint

Sprint Planning Meeting

Task Breakout

Sprint Backlog

Sprint end date and team deliverable do not change

Diagram: Sutherland & Schwaber 2007

“Scrum’s roles, artifacts, events, and rules are immutable, and although implementing only parts of Scrum is possible, the result is not Scrum.

Scrum exists only in its entirety, and functions well as a container for other techniques, methodologies, and practices.” (Schwaber and Sutherland 2013)

Essence

The explicit essence of successful Scrum is *effortful* learning through active collaborative communication.

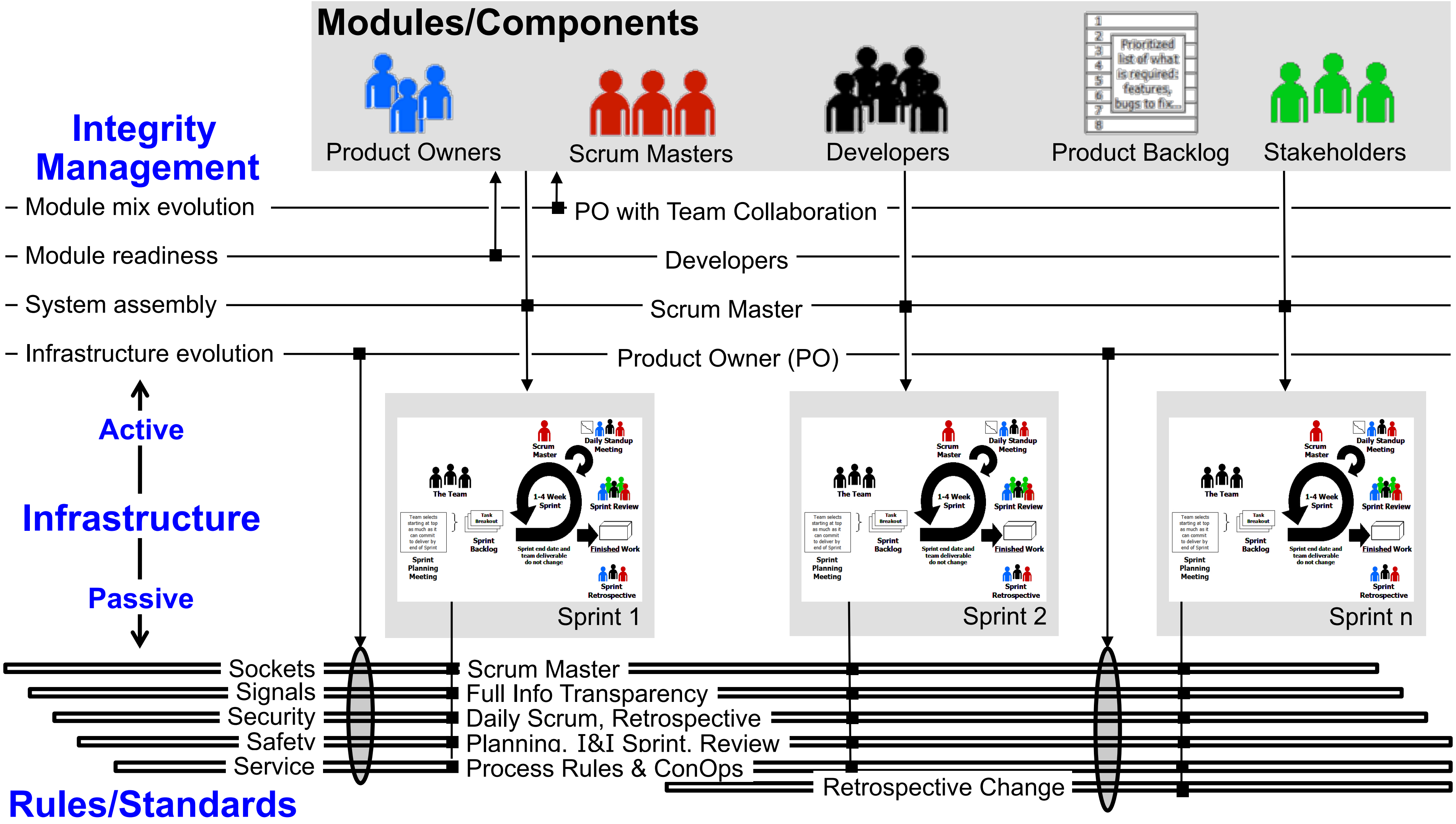
Effortful learning is a self-motivated process that continuously identifies the next thing to learn after successfully accomplishing the last learning objective.

The implicit essence of successful Scrum, however, is the ability to effectively adapt the process and the product to what has been learned.

This means changing what is being done in product development and changing how it is being done in the team's working process – which requires an agile (adaptable) architecture of both product and process to be effective.

Classic Scrum: an Agile Architecture Pattern (AAP) *Structure*

suitable for agile SW development, but not for agile systems-engineering …



Modules/Components

Product Owners

Scrum Masters

Developers

Product Backlog

Stakeholders

Integrity Management

– Module mix evolution

– Module readiness

– System assembly

– Infrastructure evolution

PO with Team Collaboration

Developers

Scrum Master

Product Owner (PO)

Active

Infrastructure

Passive

Sprint 1

Sprint 2

Sprint n

Sockets

Signals

Security

Safety

Service

Scrum Master

Full Info Transparency

Daily Scrum, Retrospective

Planning, I&I Sprint, Review

Process Rules & ConOps

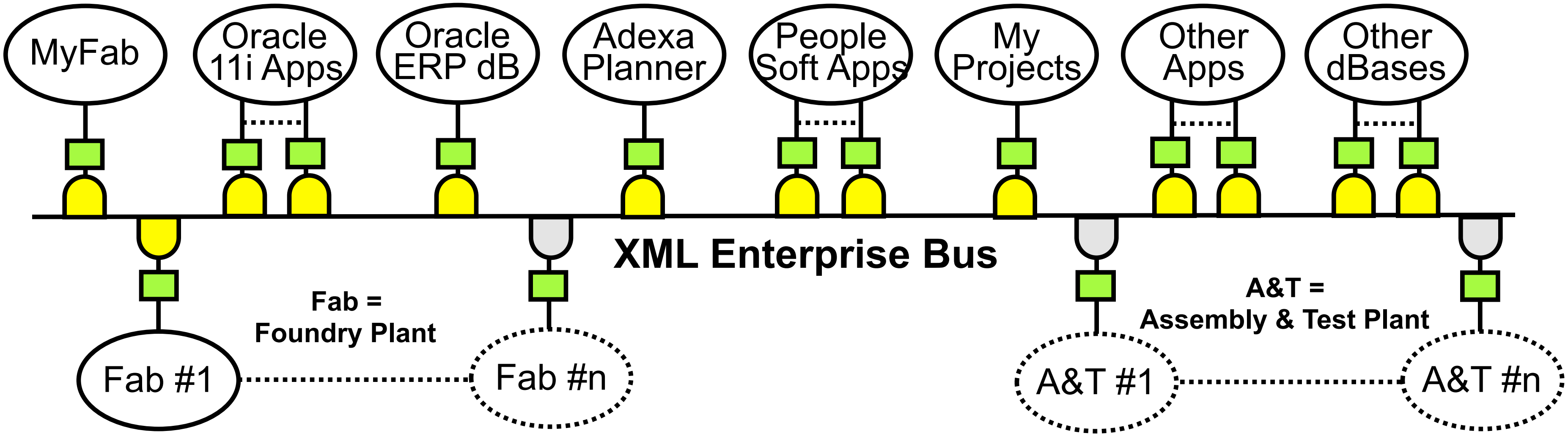
Retrospective Change

Rules/Standards

1
2
3
4
5
6
7
8
Prioritized list of what is required: features, bugs to fix...
Scrum Master
Daily Standup Meeting
The Team
1-4 Week Sprint
Sprint Review
Team selects starting at top as much as it can commit to deliver by end of Sprint
Task Breakout
Sprint Backlog
Sprint Planning Meeting
Sprint end date and team deliverable do not change
Finished Work
Sprint Retrospective

… because the RSA is different for an agile systems-engineering process, and the Scrum AAP *strategy* is inadequate for systems engineering

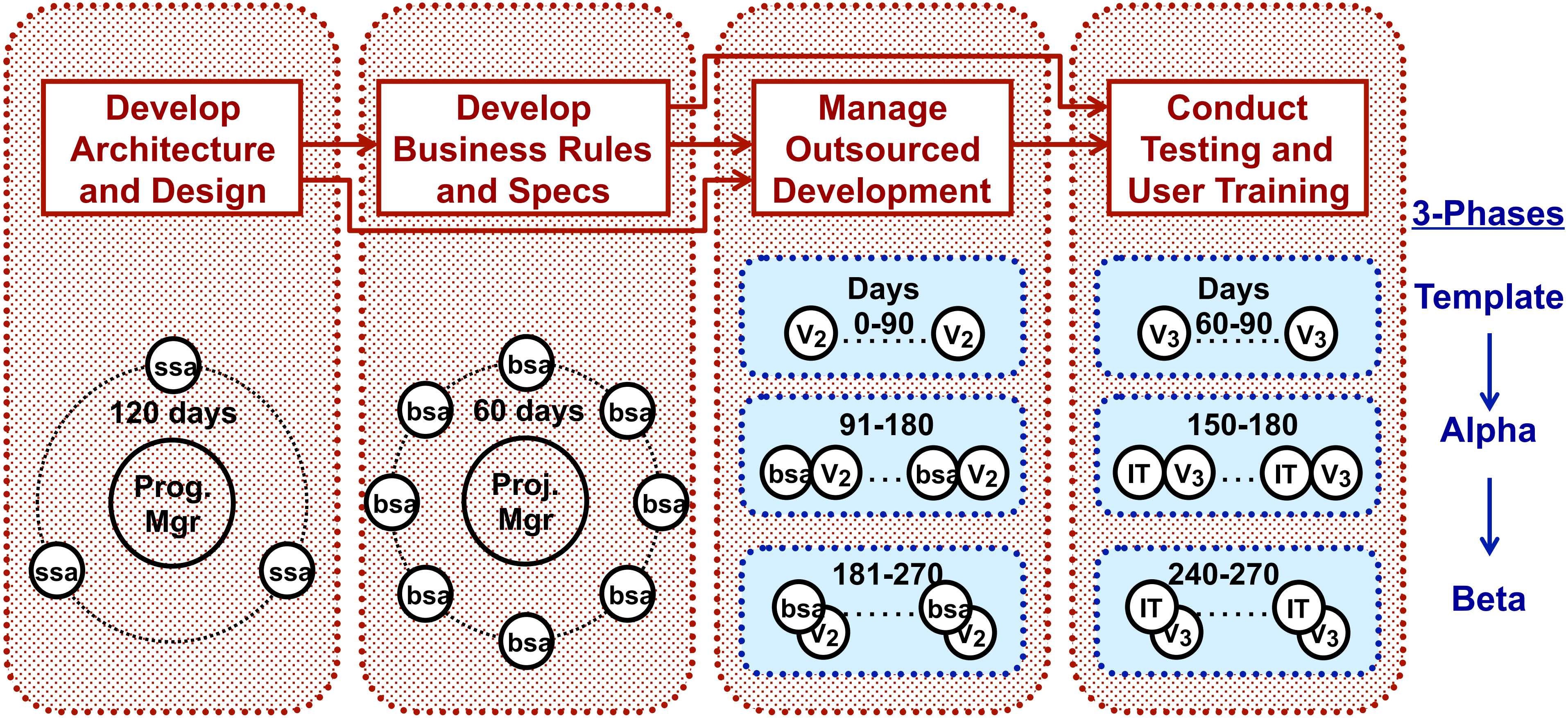
Enterprise IT-Infrastructure Design



- 🟨 = Bus Interface Module (BIM)
- 🟩 = Extract/Transfer/Load (ETL) Interface Modules
- MyProjects = Web-accessible strategic-project portfolio manager
- MyFab = Web-accessible operations transparency

www.parshift.com/Files/PsiDocs/Rkd050324CserPaper.pdf

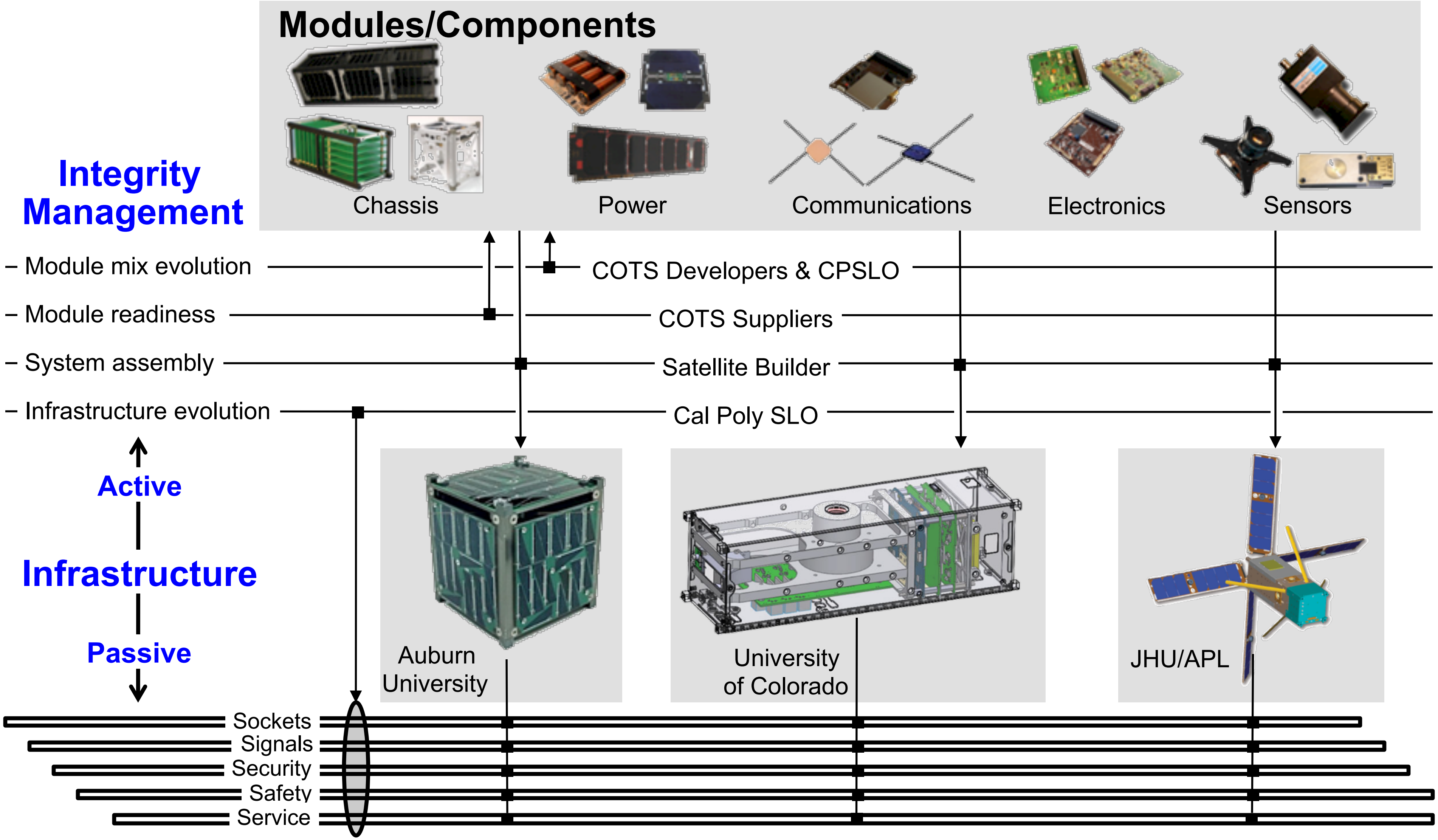
Encapsulated Development Process



- Designed to Accommodate Requirements Evolution

www.parshift.com/Files/PsiDocs/Rkd050324CserPaper.pdf

CubeSat Agile Architectural Pattern



Modules/Components

Chassis

Power

Communications

Electronics

Sensors

Integrity Management

– Module mix evolution

– Module readiness

– System assembly

– Infrastructure evolution

COTS Developers & CPSLO

COTS Suppliers

Satellite Builder

Cal Poly SLO

Active

Infrastructure

Passive

Auburn University

University of Colorado

JHU/APL

Sockets

Signals

Security

Safety

Service

Rules/Standards

System Examples of Increasing Complexity and Chronological Order →

CP SLO: ***Cal Poly San Luis Obispo***

JHU/APL Multi-Mission Bus Demonstration (MBD)

CubeSat specifications say nothing about the system-engineering process that will develop and assemble a mission specific CubeSat at any of the many organizations that do these projects.

Given the financial and window-of-opportunity risks associated with terminal deployment of a system that cannot be returned for replacement, correction, or design update, a common systems-engineering process would likely be some variation of a traditional waterfall engineering process, with big upfront planning and design.

A group at Johns Hopkins University Applied Physics Laboratory (JHU/APL) thought differently. Maybe because they had to.

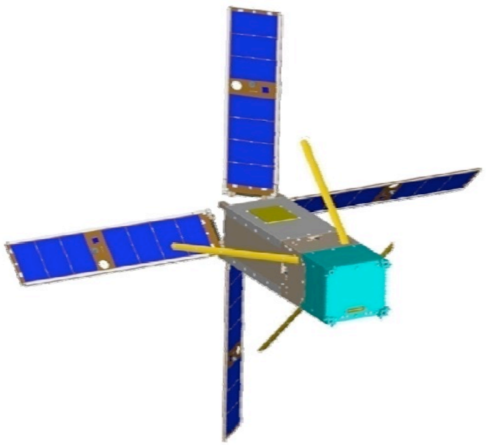
"The Multi-Mission Bus Demonstration (MBD) is a Johns Hopkins University Applied Physics Laboratory program to demonstrate a government sponsored mission in the standardized 3U (10 x 10 x 30 cm) CubeSat form factor. …

With vicious cost and schedule control, the MBD project is providing a classified DoD payload that will revolutionize the mission area and provide an operationally relevant capability to the war fighter. …

The MBD space vehicles will cater to mission operation versatility and rapid response launch capabilities."

An Agile Systems-Engineering Example

Project uncertainty was rooted in the combination of a small physical envelope constraint, high technical capability requirements, an unprecedented low budget, and an unprecedented short program duration.



This was recognized by the MBD sponsor, who was willing to make compromises and accept more risks than the typical NASA mission “in order to balance cost, schedule, and reliability while still meeting all mission requirements.

To meet the dramatically constrained volume, costs and schedule while increasing functionality more than ever seen in a CubeSat format, new designs and concepts needed to be created, developed, and manufactured. …

The MBD spacecraft is designed with all the complex and critical subsystems found within a typical earth observing multi-instrument satellite.”Couched in the UURV framework outlined in Part 1:

- **Unpredictability:** **Appetite for stakeholders to stay the course when things look uncoordinated, or when unresolved development issues are allowed to persist. Cultural adjustment of engineers working outside their standard procedures.**
- **Uncertainty:** **what requirements to use as technical drivers, what technical path to take, how changed subsystem dependencies will interrupt momentum, what untraditional decisions will have to be made, what SME expertise will be needed.**
- **Risk:** **it can’t be, or doesn’t get, done within the constraints.**
- **Variation:** **nothing relevant foreseen.**

APL’s Scrum-like development process

The MBD program used sprints with one-day durations, while classic Scrum typically uses sprints one to four weeks in duration.

Sprint Planning, Daily Scrum, Sprint Review and Sprint Retrospective meetings were combined into a single “Round-Up” meeting at the start of each work day.

The MBD Program Manager assumed the role of Product Owner. The program implemented six parallel agile development efforts for the Payload, Electrical, Software, Mechanical, Ground & Navigation Control, and Avionics subsystems of the MBD satellite.

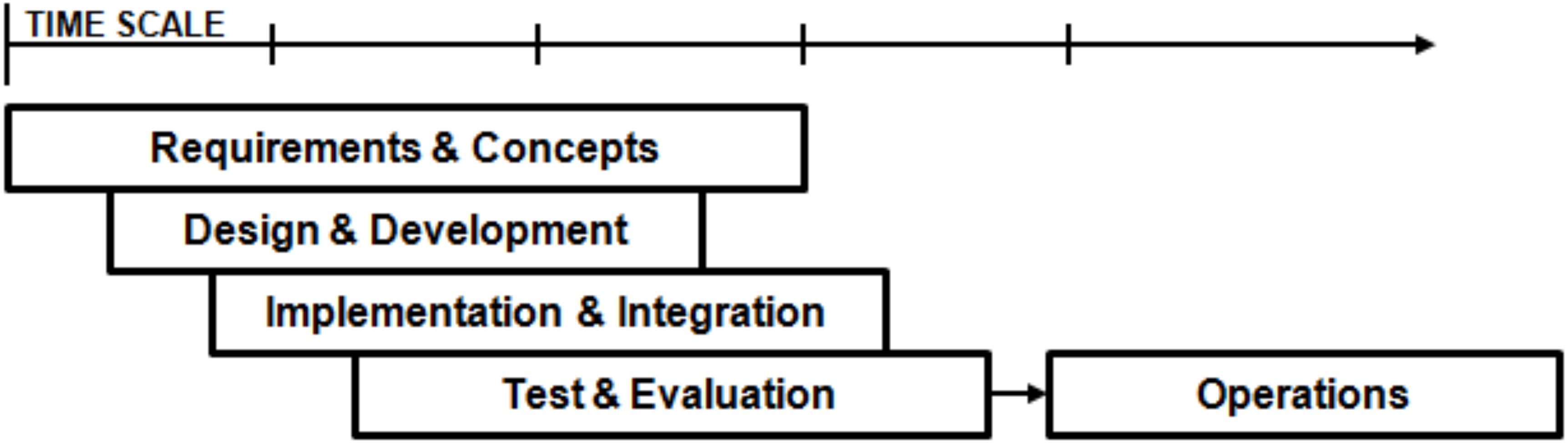
Each development team lead performed a role similar to a Scrum Master for their subsystem, but also acted as a Development Team member at the system level.

Thus the MBD agile process implementation was similar to a Scrum of Scrums with a sprint length of one day.

The need to design, develop and test custom hardware, such as the deployable solar array, required the MBD team to coordinate very short duration software development sprints (1 day) with longer duration hardware development sprints (1 month or more).

When custom hardware was required, the MBD team built two prototypes for each hardware element. Each of the hardware prototypes was integrated into the two satellites being built during the program. To the extent possible prototype hardware items were used as production items in the final satellites, even if that required rework such as cutting printed circuit board traces and adding wires to implement a design change.

APL’s Scrum-like satellite development process



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

Landmark Events

The first perspective came in 1981 with the publishing of *Systems Thinking, Systems Practice*, (Checkland 1981), questioning the application of rigid systems engineering practices to a class of systems that don't appear amenable to logical thinking, yet they are pervasive in the systems around us that have multiple stakeholders in various evolving states of satisficing for the moment. Checkland went beyond questioning, offering an alternative approach now known as soft systems methodology.

The second (chronologically) perspective came in a 1986 Harvard Business Review paper, *The New New Product Development Game* (Takeuchi and Nonaka 1986), acknowledged in (Sutherland and Schwaber 2007: 6) as sparking the thoughts that led to Scrum. That paper profiled a general process that engineered breakthrough innovative products composed more of hardware than of software, and exposed the role of what they called "subtle management", which affected product outcome by working behind the scenes to constantly rebalance diversity within the development teams. This concept is ignored by Scrum, yet crucial to the success of a rapid agile learning process.

The third perspective came in 1988 with the publication of *A Spiral Model for Software Development and Enhancement* (Boehm 1988). This marked a new turn of thought, offering an iterative, incremental alternative to the sequential waterfall approach, subsequently refined in a fundamental view (Boehm 2000).

Oversimplifying

Checkland put a focus on people.

Takeuchi and Nanoka put a focus on product.

Boehm put a focus on process.

All were concerned with uncertain and unpredictable engineering efforts. Each of these developments in the ‘80s gave legitimacy to, and spurred interest in, questioning the old ways and exploring new paths; paths meant to deal with uncertain and unpredictable operational environments.

Subsequent Events

At the turn of the millennium three more bodies of synergistic relevant thought emerged.

The first perspective came early in 2001 with the publishing of "*Response Ability – the Language, Structure, and Culture of the Agile Enterprise* (Dove 2001); which organized the agile systems research findings of the '90s into domain-independent enabling fundamentals for agility in engineered systems of any kind.

The second perspective came later that same year with the publication of *Agile Software Development with Scrum* (Sutherland and Beedle); detailing a systems engineering management process for agile software development, and reviewed in this article for its agile enabling core.

The third perspective came in 2002 published as *Agile Software Development Ecosystems* (Highsmith 2002); notable for its sane and revealing coverage of the principle software development practices sharing the agile family name at that time.

There is no pretention that the six events referenced encompass all of the thought that needs consideration for

developing an agile systems-engineering life cycle model,

nor suggestion that seminal new thinking won't continue to emerge.

And Now the CAB Shows High Interest

Workshop held 3 days ago (29 June)

Purpose

To clearify the true nature of the Top 5 needs

1) SE Professional development
2) Agile/Expedited methods
3) Effective Trade Studies
4) Product lines, re-use
5) Better Value proposal for INCOSE and Systems Engineering

Status

No clear updated documentation of the needs – many thoughts but no identified way forward (SE professional development is an exception)

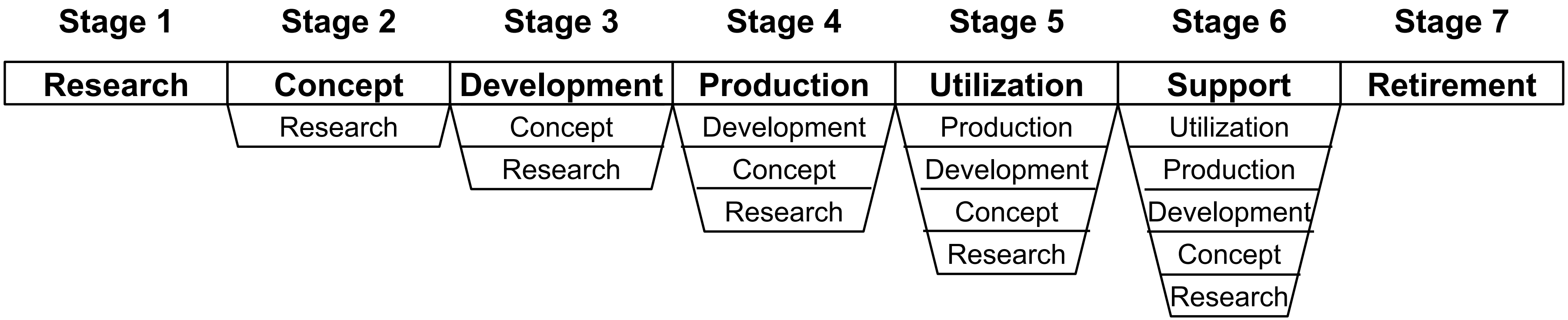
Result from the break-out

A better understanding of the need which shall be used in the future integration/interaction with the TechOps and IOB.

Clarifying the Issues in CAB Agile Priority (eight people)

- Short cycle constant evolution – i.e., counter-IED "systems"
- Long cycle constant evolution – i.e., 20-year design/build period for complex plants, with all factors continuously evolving.
- Integrating agile approach concepts with planned approach concepts.
- Meaningful WIP measures when an agile approach is employed.
- How to pivot a project effectively when feedback dictates a path change.
- Dealing with hardware design/build timeframes and sunk costs.
- Guidance on when/where to use an agile approach.
- Clarity/consistency on what agile means independent of the software practice.
- Systems as works in process after deployment (maintaining improvement "backlog")
- Case studies.

Agile SE Life Cycle Model – Domain Independent



Stage 1	Stage 2	Stage 3	Stage 4	Stage 5	Stage 6	Stage 7
Research	**Concept**	**Development**	**Production**	**Utilization**	**Support**	**Retirement**
	Research	Concept	Development	Production	Utilization	
		Research	Concept	Development	Production	
			Research	Concept	Development	
				Research	Concept	
					Research	

Framework for an agile system engineering life cycle model

www.parshift.com/s/140630IS14-AgileSystemsEngineering-Part2.pdf

The life cycle model must take into account at least three different types of systems engineering (Sheard 2000):

- **Discovery (very high complexity in problem space)**
- **Programmatic (complexity in solution space and possibly organizational)**
- **Approach (complexity in variation of applications, and possibly product lines)**

Developing an agile systems-engineering life cycle model might start with the framework displayed above, take guidance from ISO/IEC 15288 and 5000.02, 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 would be supported with case studies in a variety of systems engineering domains.

Life Cycle Model – Making it Happen

A method called Realsearch, so called because it employs real people solving real problems in real time, was employed to refine and socialize the original agile systems fundamentals discovered and organized in the nineties.

It is a process of traveling, structured, collaborative 2.5-day workshops that visit host sights by invitation, 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 what has been learned to an open problem in need of an agile process solution. Host companies will absorb costs.

A series of such workshops is being planned by the AS&SE working group beginning in 2015, designed to converge on a fundamental agile systems-engineering life cycle model applicable to the INCOSE community. Interest in hosting workshops and participation as traveling members should be referred to dove@parshift.com.

Project: Agile Life Cycle Model

Deliverable: Description of domain-independent core processes and activities.

This is discovery, not invention: what fundamentals fit all needs.

Focus is on fundamental and universal principles
regardless of the agile practice employed (spiral, Scrum, whatever).

On-site hosted 2-1/2 day structured workshops.

- Analyze two different development processes employed by host.
  - Identify fundamental principles, processes, activities that enabled effective SE under conditions of uncertainty and unpredictability.
  - Identify forces and barriers that have to be mitigated/accommodated.
- Apply abstracted knowledge to a host-presented area in need.

Visit a variety of maybe 10-12 host sites.

Traveling participants should attend at least three workshops.

Hosts absorb workshop costs.

Takes place in 2015 and maybe spills into 2016.

Value proposition for participants/hosts:
applied action learning – real people working on real problems in real time.

Express interest in hosting workshops and participation as traveling members
dove@parshift.com.