26th annual INCOSE international symposium
Edinburgh, UK
July 18 - 21, 2016

Agile Systems Engineering Process Features Collective Culture, Consciousness, and Conscience at SSC Pacific Unmanned Systems Group

INCOSE IS16
Edinburgh, Scotland, 18-21 July 2016

Rick Dove, Paradigm Shift International
Bill Schindel, ICTT System Sciences
Chris Scrapper, US Navy, SSC Pacific Unmanned Systems Group

Need and Objectives: INCOSE ASELCM Project

The engineering environment is unpredictable, uncertain, evolving: customer objectives, programmatic changes, requirements understandings, technology evolution, available resources, and corporate priorities.

Effectively-adaptable SE processes are needed to address this reality.

Agile software development practices appear to offer promise, but haven't had impact outside of the software domain:

- **their "principles" ignore fundamental purpose and intent**
- **their "practices" are insufficient for effective SE guidance**
- **hardware and firmware development have different issues and constraints**
- **acquisition policies have incompatible requirements**
- **SE standards (e.g. 15288) don't provide an effective agility foundation**
- **waterfall and V dominate cultural understanding and behavior**

How can this situation be rectified?

- **with SE principles based on clear fundamental purpose and intent**
- **with SE principles undeniably identified as necessary and sufficient**
- **with SE principles compatible with reality, culture, standards, and practices**
- **with SE principles universally applicable in all SE environments**
- **with people who have insightful understanding of core concepts**

How can this be done?

Examine what works and identify why it works, fundamentally

How We Know What We Are Talking About

Darwin didn't have a model of evolution that he tried to prove or force fit.
He observed, and asked, "What's going on here and how does it work?"

From that he iterated on model refinement
until he could find no exceptions
and could make effective predictions.

That's science, not conjecture, not a kinda good idea, not opinion.

Similarly…

in The '90s we analyzed hundreds of real-world systems that exhibited agility,
asked how they did that, and converged on a fundamental model that fit the facts.
No conjecture, no kinda good idea, no opinion.

We are doing it again, now,
analyzing real-world processes that exhibit agility,
asking how they do that, and converging on fundamental behavior principles,
that fit the facts, everywhere.

No conjecture, no kinda good idea, no opinion.

The UURVE Environment Drives Need for Agility
for both agile systems and agile systems engineering

Agile systems/processes have effective situational response options, under:

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

But agility doesn't occur unless someone actively:

- **is aware that a situation warrants a response**
- **has effective options appropriate for a response**
- **selects and affects an appropriate response**

Minds-on hands-on full and timely engagement.

Iconic Agile Architecture Pattern (AAP)

System Response-Construction Kit

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

Integrity Management

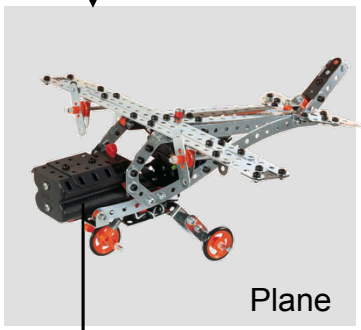
- Resource mix evolution
- Resource readiness
- Situational awareness
- Activity assembly
- Infrastructure evolution

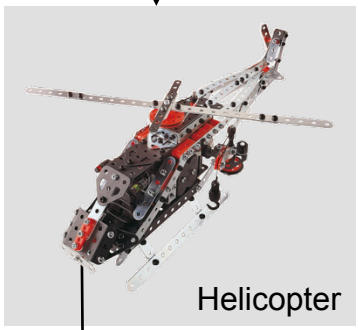
Product System Eng.
Retail Distribution Process
Product Manager
Owner/Builder
Product Manager

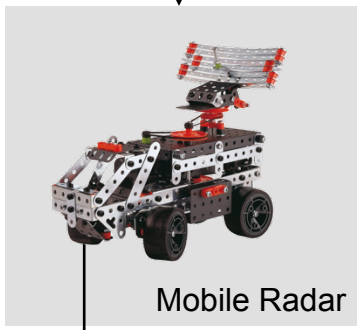
Active

Infrastructure

Passive







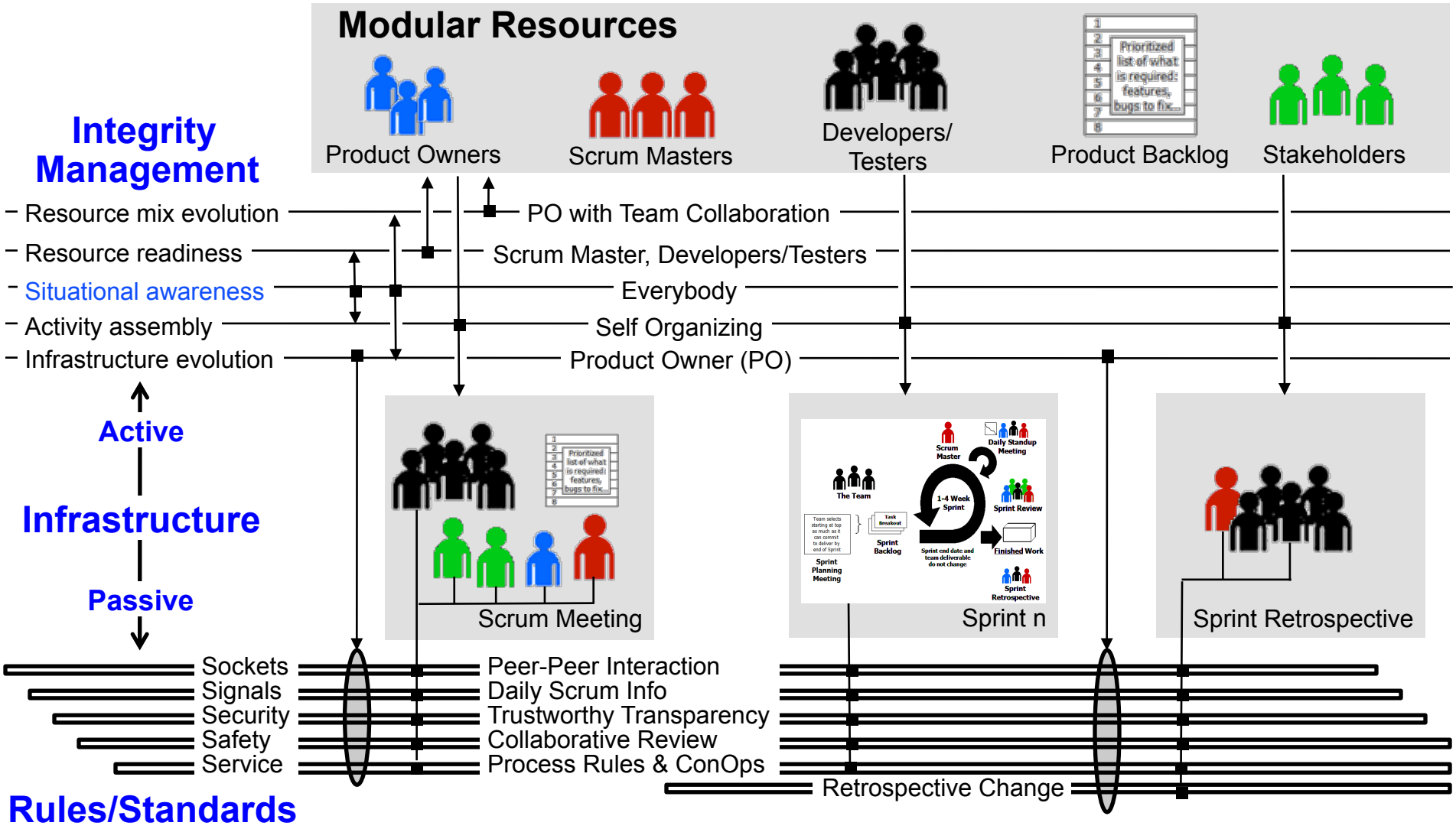
- Sockets — Parts Interconnect Standards
- Signals — Control Protocol
- Security — (None)
- Safety — Harm-Proofing Standards
- Service — Process Rules & ConOps

Radio Control Standards

Rules/Standards

Example: Scrum Agile Architecture Pattern

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



Systems Engineering Process – Architecture Elements

SE Process Drag and Drop Resource Pools

- Resources are self-contained encapsulated units which conform to the plug-and-play passive infrastructure. They can be dragged-and-dropped into the SE process. 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.

SE Process Active Infrastructure Sustainment Responsibilities

1. 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?
2. Resource Readiness – Who (or what process) is responsible for ensuring that sufficient resources are ready for deployment at unpredictable times?
3. Situational Awareness: Who (or what process) is responsible for monitoring, evaluating, and anticipating the operational environment in relationship to situational response capability.
4. Activity Assembly – Who (or what process) assembles new response configurations when new situations require something different in capability?
5. 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?

SE Process Plug and Play 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. At least five categories of standards and rules should be considered:

1. Sockets – physical interconnect
2. Signals – data interconnect
3. Security – trust interconnect
4. Safety – of process user, process, and environment
5. Service – response assembly and sustainment ConOps

Prior Work: 10 Agility-Enabling System Design Principles for fleshing out the architecture

Reusable

- Encapsulated resources (loosely coupled black-box units)
- Facilitated interfacing (easy resource insertion/removal)
- Facilitated re-use (support for finding/deploying appropriate resources)

Reconfigurable

- Peer-peer interaction (direct communication w/o intermediaries)
- Deferred commitment (decisions & fixed bindings at last-responsible-moment)
- Distributed control and information (decisions at point of maximum knowledge)
- Self organization (relationships and interactions negotiable)

Scalable

- Evolving infrastructure standards (resource interface and interaction change)
- Redundancy and diversity (duplicate and diverse resource populations)
- Elastic capacity (resource populations and functional capacity is variable)

Current Work: Process Operational-Behavior Principles
(WIP Hypothesis based on analytical workshops in process)

Monitoring (observe, orient)

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

Mitigating (decide, act)

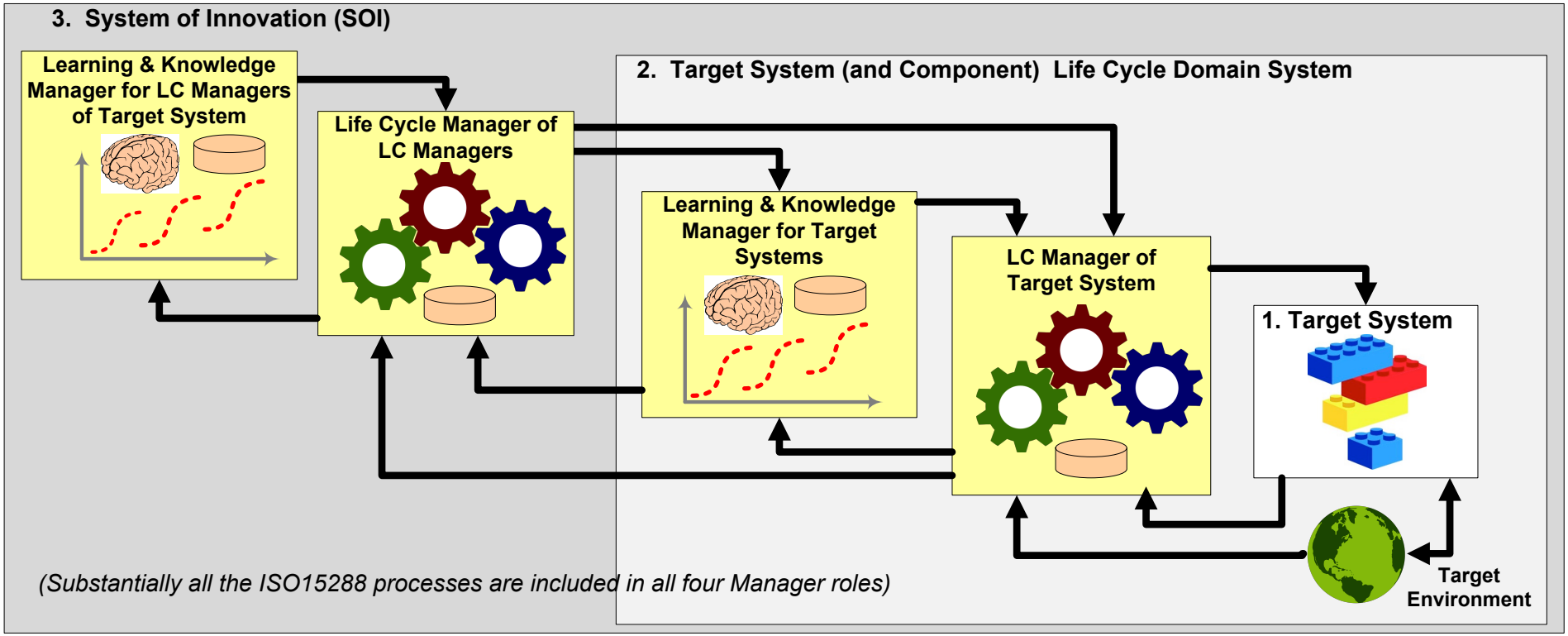
- **Decision making (timely, informed)**
- **Action making (invoke/configure process activity to address the situation)**
- **Action evaluation (V&V)**

Evolving (improves above with more knowledge and better capability)

- **Experimentation (variations on process ConOps)**
- **Evaluation (internal and external judgement)**
- **Memory (current process ConOps)**

Natural selection: replication with variation in competition.
(A ubiquitous algorithm, the essence of learning and evolution)

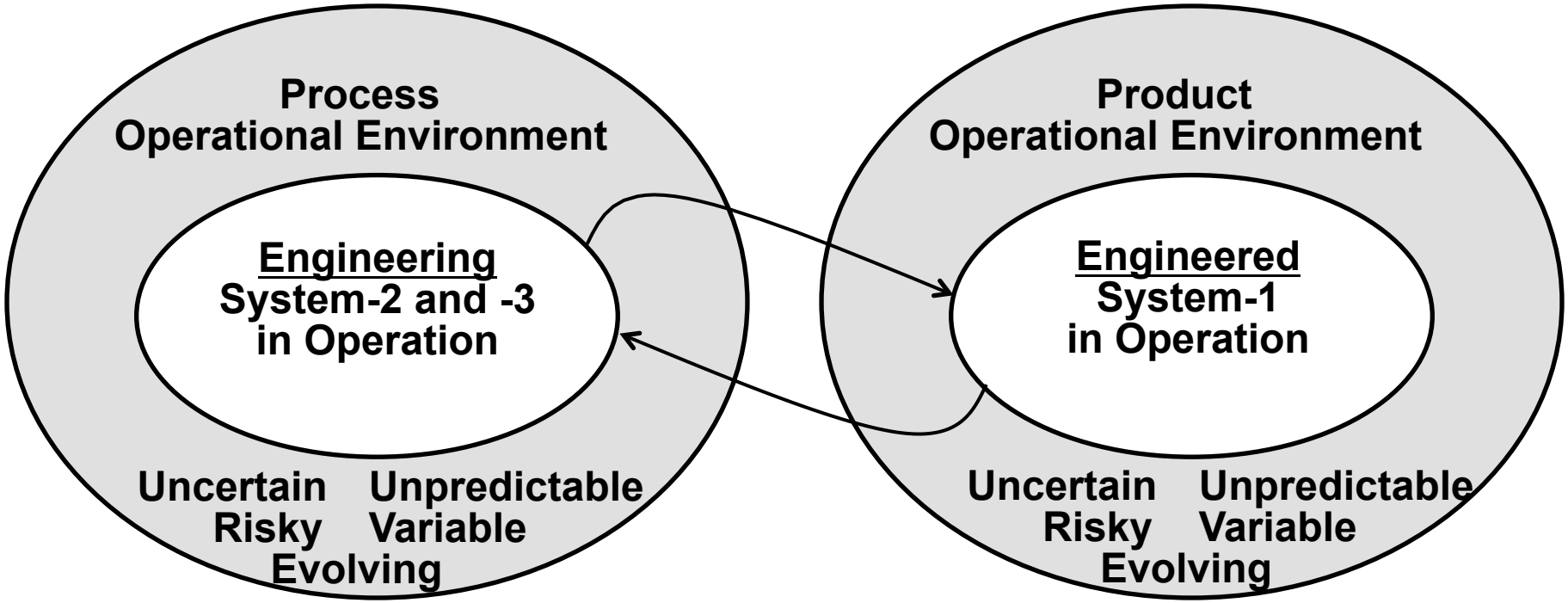
Agile Systems Engineering Life Cycle Pattern
Encompassing Systems 1, 2, and 3



- 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.

slide credit: Bill Schindel

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



It is counterproductive to have an agile development process if you don't have an agile product architecture

SSC-Pac Case Study

IS16 paper: www.parshift.com/s/ASELCM-01SSCPac.pdf

This case study reveals concepts with broad application in many domains.

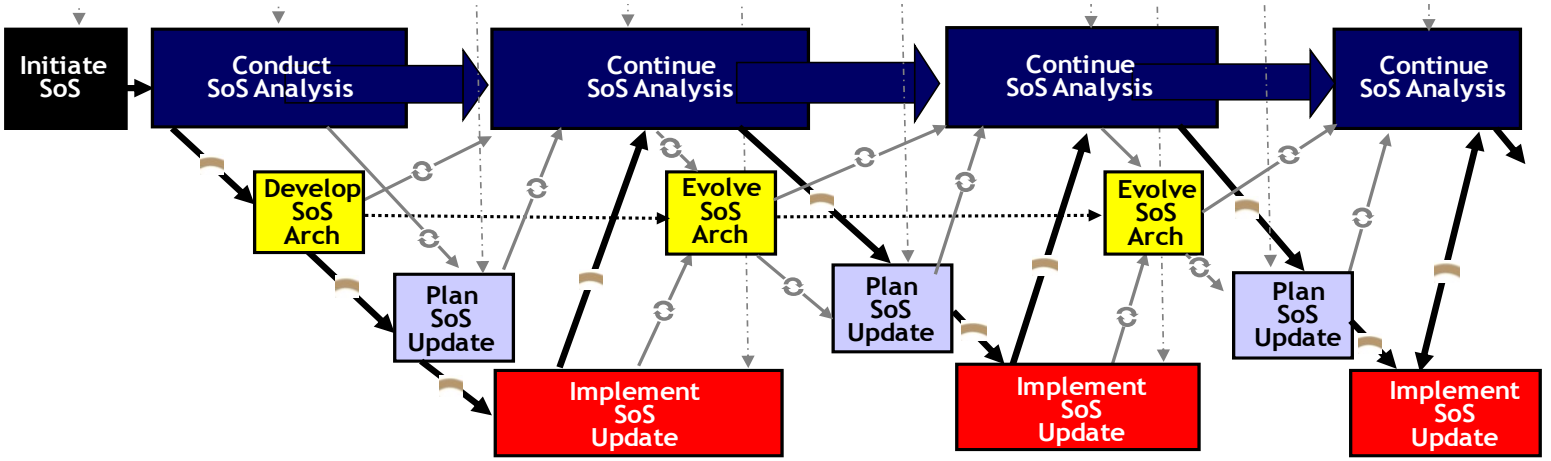
A systems engineering process with 6-month, 4-phase, overlapping "waves":

1. **System component development**
2. **System architecture evolution**
3. **Capability integration**
4. **Validation testing**

The process capability supports a portfolio of projects, with three years of respected and effective results.

Classic Wave Model, subsequently tailored for the analyzed program

(Scrapper and Dahmann, 2016)



The Process is Successful

…replaced a waterfall process plagued by cost overruns, missed schedules, inadequate development achievement, uncooperative teaming, and poor status visibility.

…orchestrates the interaction of the 60-some engineers and managers on the project, including six external organizations of 4-5 engineers each working on development of functional capabilities to be integrated into a federated system.

… encompasses research, development, integration, test, and evaluation of deployable system and component technologies that can provide new capabilities.

… demonstrated effectiveness over three years in lower and predictable costs, on-time capability deliveries, and continual advancements on the overall performance of the systems under development.

… will be migrated to other programs.

UURVE that Prompted an Agile SE Approach

**Systems Engineering (SE) process for HW/SW/WW*
for evolutionary development of innovative-edge technology**

Unpredictability (unknowable situations):
- Strategic realignment of project-sponsor priority.
- Changes in and/or availability of key personnel and development contractors.

Uncertainty (randomness with unknowable probabilities):
- Feasibility of technical approach and initial designs.
- Contracting issues, funding gaps, and budget short falls.

Risk (randomness with knowable probabilities):
- Failure to meet technical performance measures.
- Maturation and integration of required component technologies.

Variation (knowable variables and variance ranges):
- Availability of test environment and test support
- Time to obtain requisite approvals.
- Reliability, Availability, Maintainability of test-beds.

Evolution (gradual successive developments):
- Changes in technical landscape and insertion of emerging technology.
- Changes in programmatic objectives & stakeholder requirements (scope creep).

*** WW: Wet Ware (people)**

On Choosing the Agile Wave Model Approach

The Scrum software development process does learning in two-to-four week sequential development increments, with retrospective analyses of outcomes and process-behavior effectiveness.

Sequential Spiral approaches include more than software development, necessitating longer learning cycles, with risk reduction as a central cycle-driving theme.

The Wave Model approach has overlapping learning cycles, decoupling the development effort from the subsequent integration, test, and evaluation efforts.

This decoupling affords back-to-back development increments that don't have to wait for integration, test, and evaluation before starting the next increment of new-capability development.

Key Take Away:

- **Let an understanding of the problem pull an agile solution that fits.**
- **Don't push a favored agile process just because.**

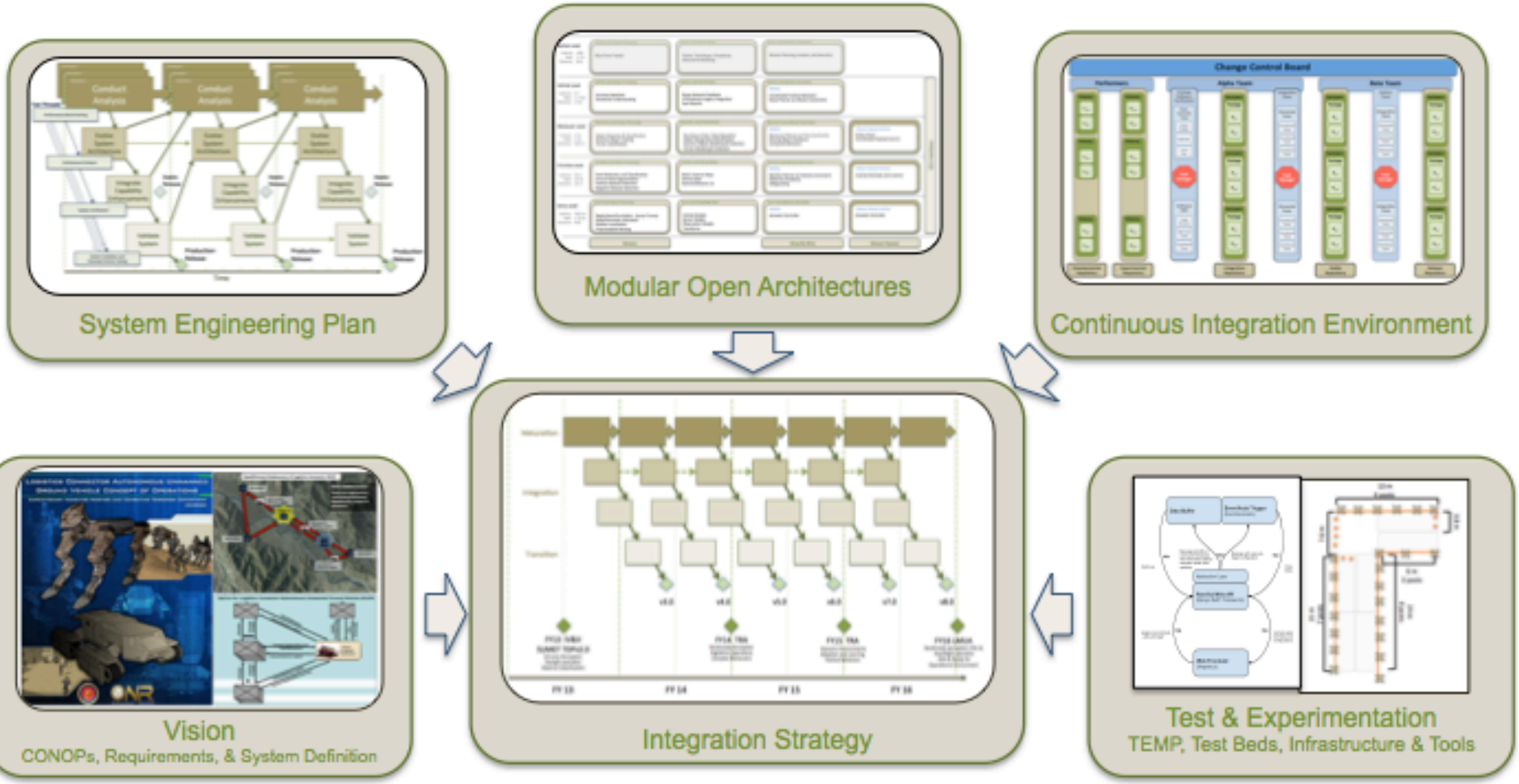
Wave Benefits to this Program

The Wave Model offered meaningful progress feedback in project-appropriate 6-month cycles, long enough to accommodate incremental new-capability development time, and short enough to demonstrate frequent progress to sponsors and allow learning and affordable re-planning and corrective action when needed.

There is nothing about the Wave Model that precludes a Scrum approach in the software-development activity, if software developers wish.

The Wave Model approach accommodates tailoring based on size of project, funding levels, and overall project goals.

Wave, using a modular-component architecture, lowers costs to all sponsors with re-usable modules across projects.

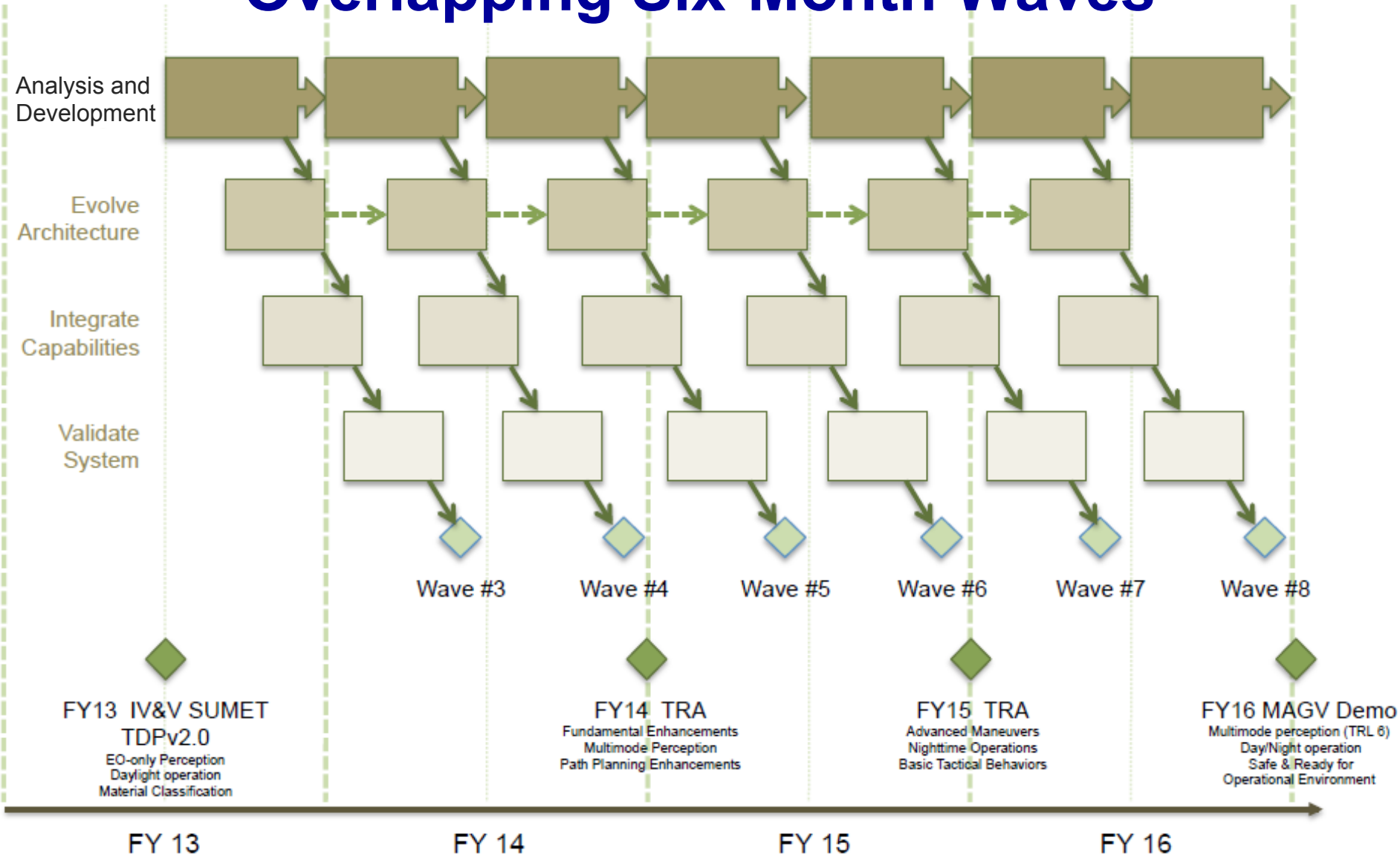


Five elements of the Integration Strategy

- Vision
- Systems Engineering Plan
- Modular Open Product-System Architecture
- Integration Test and Experimentation Master Plan
- Continuous Integration Environment

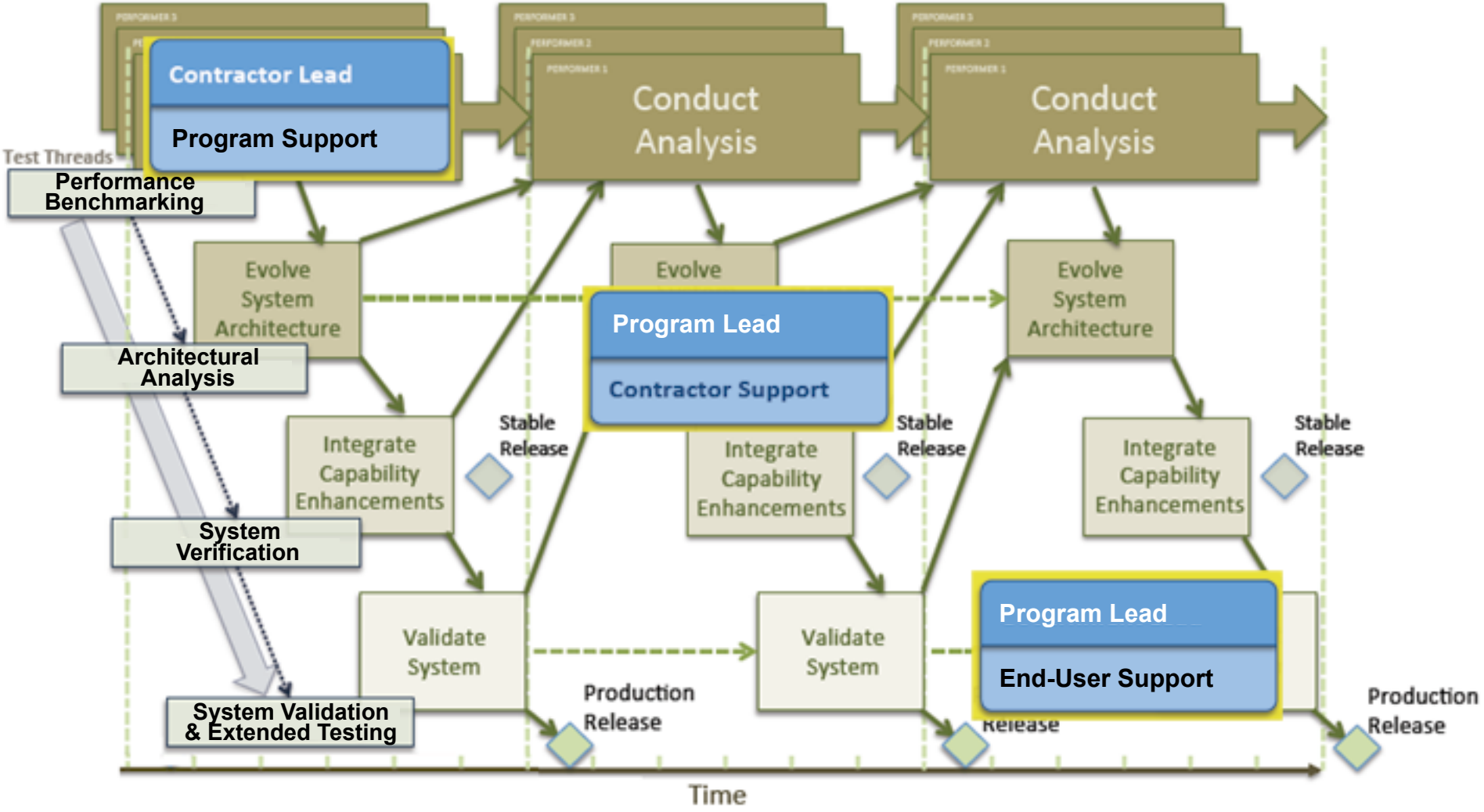
slide credit: Chris Scrapper

Integration Strategy
Overlapping Six-Month Waves



slide credit: Chris Scrapper

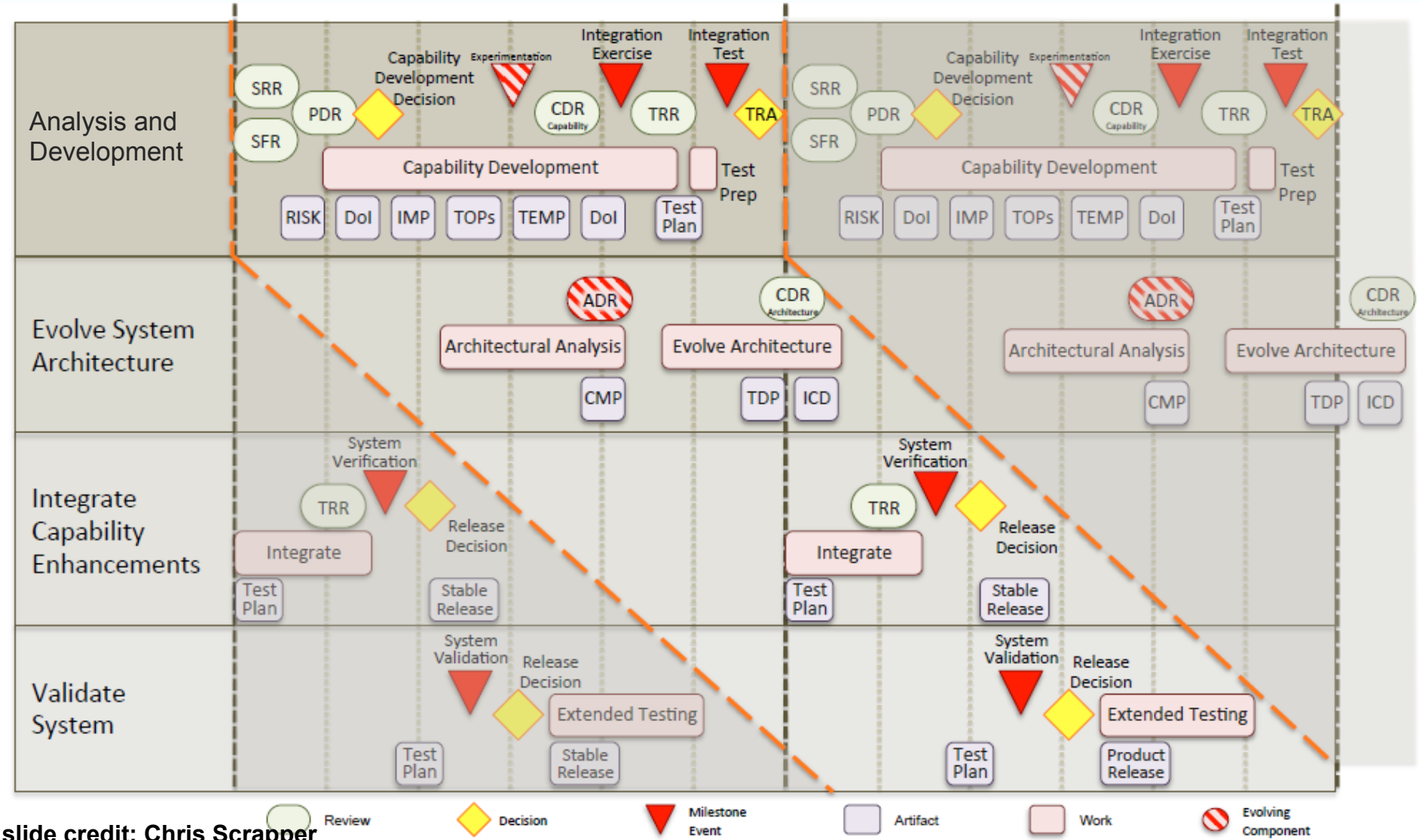
Engaged Integrated Team: Alternate Leads and End-Users



slide credit: Chris Scrapper

CDR: Critical Design Review
DoI: Declaration of Intent
PDR: Preliminary Design Review
SDR: System Design Review
SFR: System Functional Review
SRR: System Requirements Review
TEMP: Test and Experimentation Master Plan
TOP: Test Operating Procedures
TRR: Test Readiness Review

Integrated Strategy Chart



slide credit: Chris Scrapper

Systems Engineering Process AAP

for evolving autonomous off-road-vehicle robotic military technology

SE-Process Reusable/Reconfigurable Resources

- IL Integration Leads
- FL Functional Leads
- TL Technical Leads
- CP Contract Performers
- WF Users (War Fighters)
- RC Reusable Components
- CD CIE Data
- TM Test Methods

Integrity Management

- Resource mix evolution
- Resource readiness
- Situational awareness
- Activity assembly
- Infrastructure evolution

PM+CIT.
PM+CIT (Core Integration Team)
PM+CIT+Leads
Leads
PM (Process Manager)

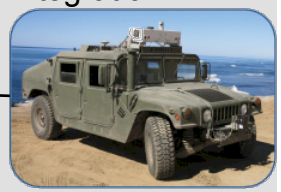
Active Facilitating

Infrastructure

Passive Enabling

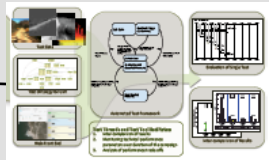
RaDER Integration

EV1 Integration



IPT Working-Group

Validation Testing



- Sockets
- Signals
- Security
- Safety
- Service

Rules/Standards

Sockets: CIE, System-1 modular architecture, roles, culture, test threads
Signals: Vision, Declarations of Intent, Config Mgmnt Plan, Integration Strategy, CIE data, decisions, engaged team feedback
Security: User agreement/NDA, Config Mgmnt Plan, CIE access controls
Safety: Open-process visibility, open communication, protected communication
Service (SE ConOps): Vision, Culture, Consciousness(CIE), Conscience, Wave, Integration Strategy/TEMP, Sys-1 and Sys-2 AAP

Resources Assembled in Process-Activity Configurations

Integration Lead – Develops the Vision for System-1 and oversees the technical execution and coordination of activities and processes in System-2.

Technical Lead – Oversees technical execution and mitigation of technical risk associated with a specific phase in System-2.

Functional Lead – Provides in-depth technical expertise in each designated functional area to support the research, design, implementation, operation, maintenance, and assessment of new capability enhancements.

Contract Performer – Leads the development of desired functional capability for System-1.

End-Users – Validates the operational concept for System-1 and provides feedback into System-2 regarding utility of current and planned capabilities.

Reusable Components – Functional capabilities and tools to support the integration and specification of System-1 capabilities for different vehicle types and mission sets.

CIE Data – Artifacts and evidentiary information produced by System-2 and shared across extended team to enable the rapid and agile development of System-1.

Test Methods – Tools, procedures, and metrics for quantifying the performance of System-1 to enable the rapid assessment, characterization, and inter-comparison of experimental results.

Collective Culture of Engagement

Most pronounced during the analysis activity was the pervasive nature of the culture, its thoughtful development, and its continual reinforcement. This is done with a combination of soft skills and supporting infrastructure.

Culture is a shared set of expectations for behavior, and an environment that enforces that behavior.

Here culture isn't written like a mission statement, but is rather practiced by leadership, shaped by consistent reinforcement, and enforced by dealing openly with infractions detrimental to the team and at odds with a pervasive collective agreement to work together toward total success.

Full and active engagement with the SE process intent and the SE project objectives is the expectation. All team members are on a shared mission, and all team members need to support and be supported by all other team members, at all times.

The nature of the SE process, its leadership, and the transparency of comprehensive real-time project status provide team-engagement sensitivity. Where the culture doesn't fit an individual (or vice-versa), that individual will either move on, or adjust. The culture will not tolerate in-action.

Collective Consciousness

The Continuous Integration Environment (CIE) is a data-driven repository of knowledge, with customized viewing templates for different needs. CIE provides user interfaces that separate internal representations of data (the *model*) from the ways that information is presented to users (the *view*), with custom views for different stakeholders.

This homegrown CIE is structured as a federation of independent capabilities, mostly off the shelf, and is being evolved to provide real-time relevant and comprehensive views of history and current status to all team members.

The CIE intent is to facilitate a real-time collective consciousness, where all team members are plugged in to all information associated with full project success, as well as to the information of relevance to their specific responsibilities and tasks.

New data, new decisions, new issues, new test results, ripple through the relevant federation of CIE components and CIE user views immediately.

This collective consciousness manifests for the team much like it does for musicians in a symphony orchestra, where off notes and bad timing are immediately sensed by all.

Collective Conscience

Meeting openings remind everyone that the customers are taxpayers and warfighters. These reminders don't stop with a simple statement. They are rooted in image and story that elevates them to personified walking needs with faces.

The warfighter needs tools that are effective, timely, and affordable for mission achievement and self preservation. Warfighter reality is obtained with their critical presence at testing events, and with structured workshops between waves.

The tax payer needs tools that are effective, timely, and affordable for national/ homeland security – capability that is affordably deployable, not costly technology that limits production quantities and threatens sustainable programs.

In these contexts (warfighter and taxpayer) the team accepts responsibility, and evaluates decisions with that critical internal customer voice.

The team develops and maintains a collective conscience to do what is responsibly right. This breaks the inertia of building upon favorite and comfortable technical approaches, to consider technologies that address the fundamental needs.

Notable Process Concepts

Common process spanning a portfolio of projects.

Government-retained architecture ownership.

Systems engineering structured as a Wave-Model-inspired evolutionary process.

Continuous integration with comprehensive regression testing.

Clear unambiguous roles and responsibilities.

Common culture embracing development contractors.

Ubiquitous real-time shared awareness of project progress and status.

A sense of collective mission.

Quality-of-engagement sensitivity.

Distributed test threads and continuous risk management.

Meaningful user involvement.

All are discussed in the paper.

Agility-Enabling S2-S1 Design Principles

Reusable

- **Encapsulated resources: black-box components, people with individual styles.**
- **Facilitated interfacing: strict S2-process and S1-component interface rules.**
- **Facilitated re-use: engaged full-knowledge-team can/will pitch in as needed.**

Reconfigurable

- **Peer-peer interaction: full project transparency and open communications.**
- **Deferred commitment: working groups configured at time of need.**
- **Distributed control & info: Individual responsibility for activity & CIE data.**
- **Self organization: open planning (relationships and interactions negotiable)**

Scalable

- **Evolving infrastructure standards: architecture and CIE evolve per wave.**
- **Redundancy and diversity: multiple resources for any activity**
- **Elastic capacity: scalable process accommodates multiple projects.**

Simple examples, not comprehensive

Process Operational-Behavior Principles

(WIP Hypothesis based on analytical workshops in process)

Monitoring (observe, orient)

- **External awareness: Warfighter workshops & testing, technology monitoring,**
- **Internal awareness: Resource quality-of-engagement sensitivity/monitoring.**
- **Sense making: Pervasive risk analysis distributed in all activities.**

Mitigating (decide, act)

- **Decision making: data and risk driven, open, inclusive, immediate, fearless.**
- **Action making: IPT working groups configured for resolution.**
- **Action evaluation: data-driven retrospectives and corrections.**

Evolving (improves above with more knowledge and better capability)

- **Experimentation: encouraged technical and process ConOps experiments.**
- **Evaluation: constant data-driven open evaluation with full team.**
- **Memory: CIE and wikis updated daily**

Simple examples, not comprehensive

Push vs Pull

A very key lesson we learned at the SSC-Pac workshop was the power of a "pull" approach. Chris Scrapper designed his process to fit his problem. He didn't come to the party with Scrum or Spiral or Wave in his mind.

Agile SE concepts should be pulled into practice by a need to solve recognized SE problems, rather than pushed into practice by a belief that they must be better than current practice. One thing this means: don't start with Scrum in mind as a solution, ready to force fit it to the engineering and management environments.

Instead, understand your problem environment, relative to UURVE issues, in terms your engineering and management people can relate to. Then identify the intent and nature of solution concepts needed to address the issues. Then and only then examine the ready-made practices for conceptual bits and pieces of usefulness.

In other words, develop your agility-requirements before choosing a solution. With a clear understanding of the fundamental and true requirements, an agile SE approach can be incrementally introduced and evolved to fit the culture, the business, and the engineering environment.

Bottom Line: SE as Active Learning Process

Engagement.

Awareness.

Collaboration.

Experimentation.

Evaluation.

Evolution.

What you've done?

What you are doing?

What you should have done?

Evolve accordingly.

How you've done it?

How you are doing it?

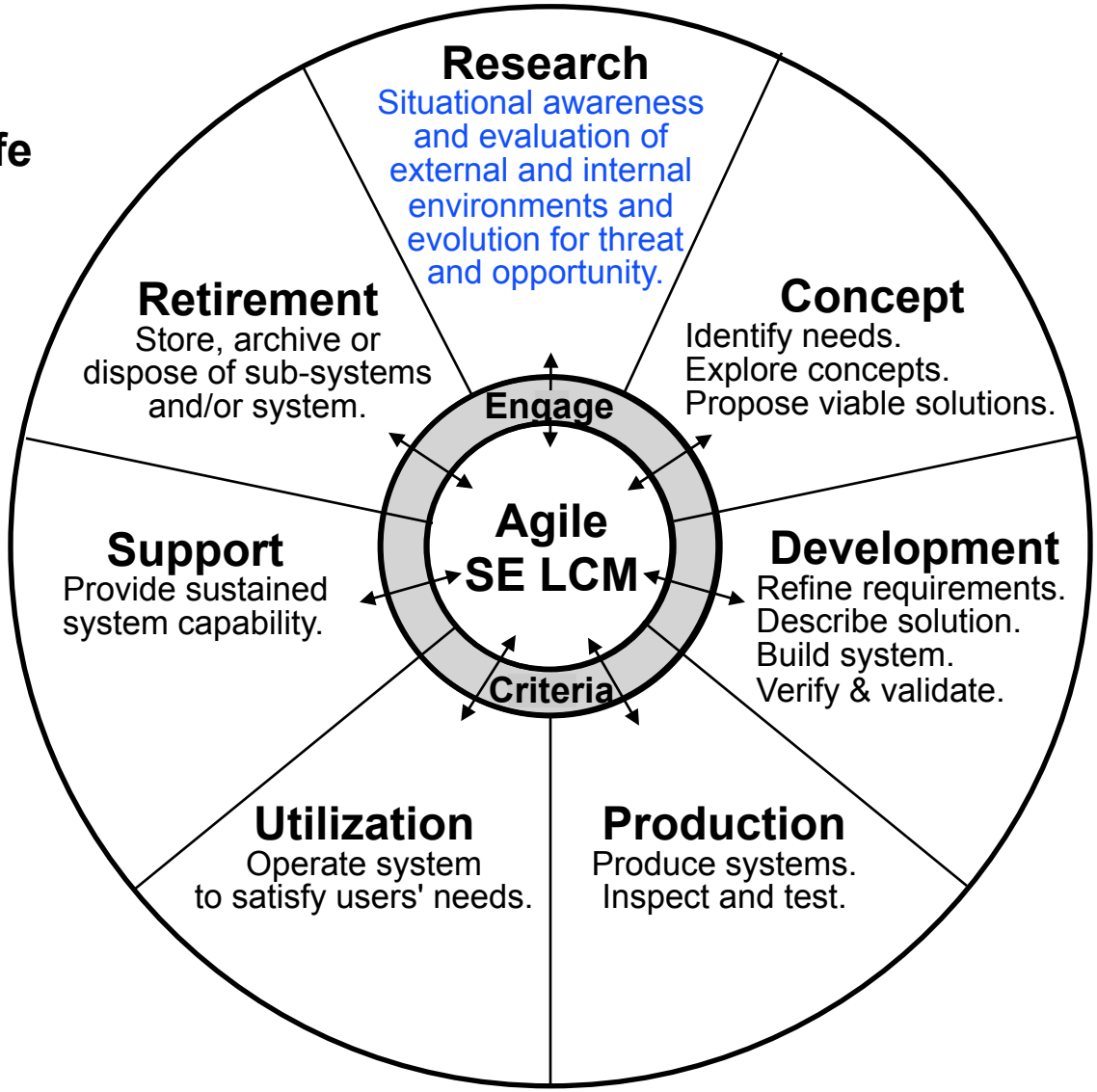
How you should have done it?

Evolve accordingly.

Asynchronous/Simultaneous Agile SE-LCM Framework

Systems and software engineering — Life cycle management — Part 1: Guide for life cycle management ISO/IEC TR 24748-1:2010(E)

This Agile SE Life Cycle Model is consistent with ISO/IEC/IEEE standards



Observed in all workshops to date

Seven asynchronously-invoked stages are engaged repetitively and simultaneously when engagement criteria are met

2015 ASELCM Workshop Characterizations

SSC-Pac (Autonomous vehicle technology innovation)

- Three years proven process success
- Product portfolio architecture leveraging reusable resources in a Wave process
- Driven by quality of results over process dogma/conformance (process evolves)
- Awareness and management of cooperative engagement behavior (team culture)
- Real-time knowledge management and assimilation (team consciousness)
- Focus on delivering meaningful value to customers (team conscience)

Northrop Grumman (Systems-of-systems information hub capability-evolution)

- Six years proven process success using a Scrum/Waterfall/Wave hybrid process
- Mitigation of uncontrolled System-of-System environment (22 independent systems)
- Intimate stakeholder involvement in the SE process
- Asynchronous/simultaneous life cycle stages, in never-ending system growth/ evolution
- CMMI level 5 procedure discipline, providing seamless-release stability, and more
- Awareness and mitigation of external environment evolution
- Real-time optimal process-control model, re-prioritizes development-increment activity

Key Characteristics for Case Studies Below are Work-in-Process (more before November)

Rockwell Collins (avionics product-line engineering)

- Proven software-process success, still evolving integrated HW/FW process agility
- Product Line architecture with Scrum software process
- Infrastructure with reusable resources for combined HW/SW/FW development
- Focus on stake-holder relationship facilitation & management
- Continuous market alertness and awareness

Lockheed (warplane capability evolution)

- Early process success, still evolving
- SAFe process tailored for Lockheed project portfolio and gov't contract nature
- Agile SE assimilation in a cooperative defense acquisition environment
- Appreciation of constraints imposed on SAFe proprietary IP utilization
- Appreciation of information debt in addition to technical debt

References

Dove, R. and W. Schindel. 2016. Agile Systems Engineering Process Features Collective Culture, Consciousness, and Conscience at SSC Pacific Unmanned Systems Group. INCOSE International Symposium (IS 2016), Edinburgh, Scotland, UK, July 18-21. www.parshift.com/s/ASELCM-01SSCPac.pdf

Schindel W. and R. Dove. 2016. Introduction to the Agile Systems Engineering Life Cycle MBSE Pattern. INCOSE International Symposium (IS 2016), Edinburgh, Scotland, UK, July 18-21.

Scrapper, C., R. Halterman, and J. Dahmann. 2016. An Implementers View of the Evolutionary Systems Engineering for Autonomous Unmanned Systems. IEEE Systems Conference, Orlando FL, 18-21 April.

INCOSE Webinars:

Agile 101: Architecture Pattern,
www.parshift.com/s/AgileSystems-101.pdf, www.parshift.com/s/AgileSystems-101.wmv

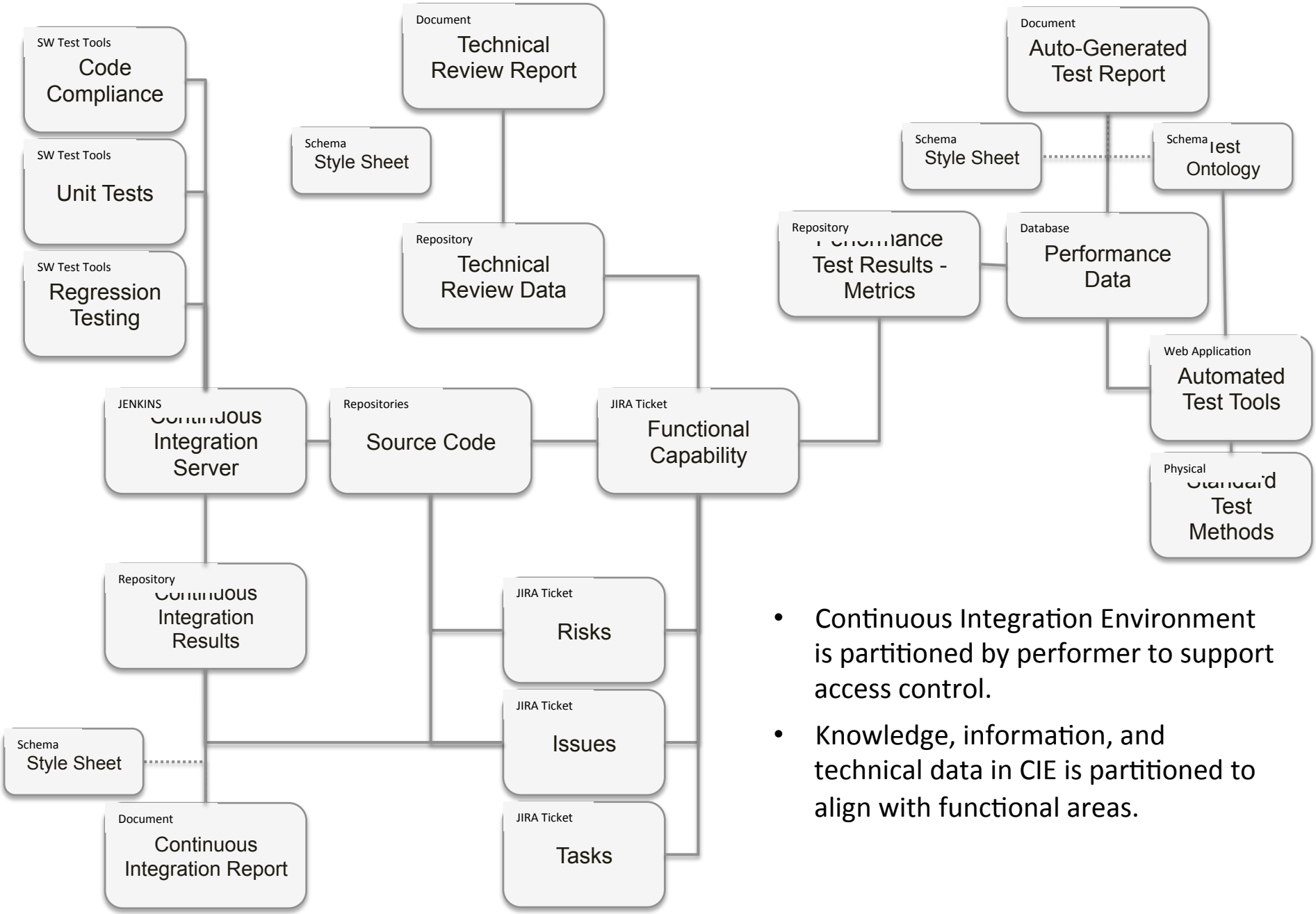
Agile 102: Design Requirements,
www.parshift.com/s/AgileSystems-102.pdf, www.parshift.com/s/AgileSystems-102.swf

Agile 103: Design Principles,
www.parshift.com/s/AgileSystems-103.pdf, www.parshift.com/s/AgileSystems-103.mp4

Agile 104: Engagement Quality,
www.parshift.com/s/AgileSystems-104.pdf, www.parshift.com/s/AgileSystems-104.mp4

ASELCM project and workshop Host information/details:
www.parshift.com/ASELCM/Home.html

Backup and Unused



- Continuous Integration Environment is partitioned by performer to support access control.
- Knowledge, information, and technical data in CIE is partitioned to align with functional areas.

How data is connected in CIE

Agile System History Perspective

Agile manufacturing systems - 1991

Agile enterprise Systems - 1992

Agile CCRP C2 - 1996

Software development – 2001 (with predecessor work, e.g., Spiral, etc)

Military as agile enterprise - 2013

Systems engineering becomes a focus - 2015

American Football is Agility in Action

Operational Environment

- **Unpredictability (injury)**
- **Uncertainty (composition of opposing team on game day)**
- **Risk (impaired team-work day)**
- **Variation (weather)**
- **Evolution (team competencies)**

Dynamic game situations require certain response capabilities, e.g.

- **Creating: e.g., a tailored game plan for each game**
- **Improving: e.g., opponent-evaluation accuracy**
- **Migrating: e.g., pre to post salary cap rule, now concussion concerns**
- **Modifying: e.g., game plan strategy with game-time learning**
- **Correcting: e.g., on-field competitive mismatch in specific position**
- **Varying: e.g., defense-offence competitive strength balance**
- **Expansion/Contraction: e.g, range of player-position depth of 2-4**
- **Reconfiguring: e.g., mix of 11-on-field frequently**

Performance quality is determined
by degree of engagement of every team member at every moment

Example: Football Agile Architecture Pattern

Drag-and-drop modules in a plug-and-play infrastructure

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