25th anniversary
annual INCOSE
international workshop
Los Angeles, CA
January 24 - 27, 2015

INCOSE 2015 TWENTY-FIFTH ANNIVERSARY

ATTACHMENT 1:
LIMITED SAMPLE FROM
DRAFT SESSION SLIDES

The Agile Systems Pattern

An MBSE-Based System Pattern, with Implications for Agile Modeling

Bill Schindel, ICTT System Sciences

schindel@ictt.com

IW2015 MBSE Workshop Breakout Session:
Agile Modeling and Modeling Agile Systems, Jan 24, 2015

Objectives of Overall Breakout Session on Agile Modeling and Modeling Agile Systems

- Gain a broader understanding of Agile Systems-engineering and the Engineering of Agile-Systems, and the common fundamental Agile Architecture Pattern of both that enables effective response in uncertain, unpredictable, and evolving SE and operational environments.
- Learn how formal model-based System Patterns can expand the fundamental Agile Architecture Pattern with necessary agile-enabling details for fleshing out an agile SE process and agile system design.
- Understand the role and impact of accumulated system patterns within Agile Systems.
- Learn about Pattern-Based Systems Engineering (PBSE), and how Agile Modeling is facilitated by PBSE.
- Learn how S*Patterns express model-based system patterns.
- Find out about the 2015/16 traveling workshop Agile System Engineering Life Cycle Model (ASELCM) Fundamentals project, to occur in the US and UK, along with how and why you and your organization might want to participate

<u>Assumption</u>: Some awareness of the general ideas of Agile Systems from this breakout's preceding sub-session, or the listed References

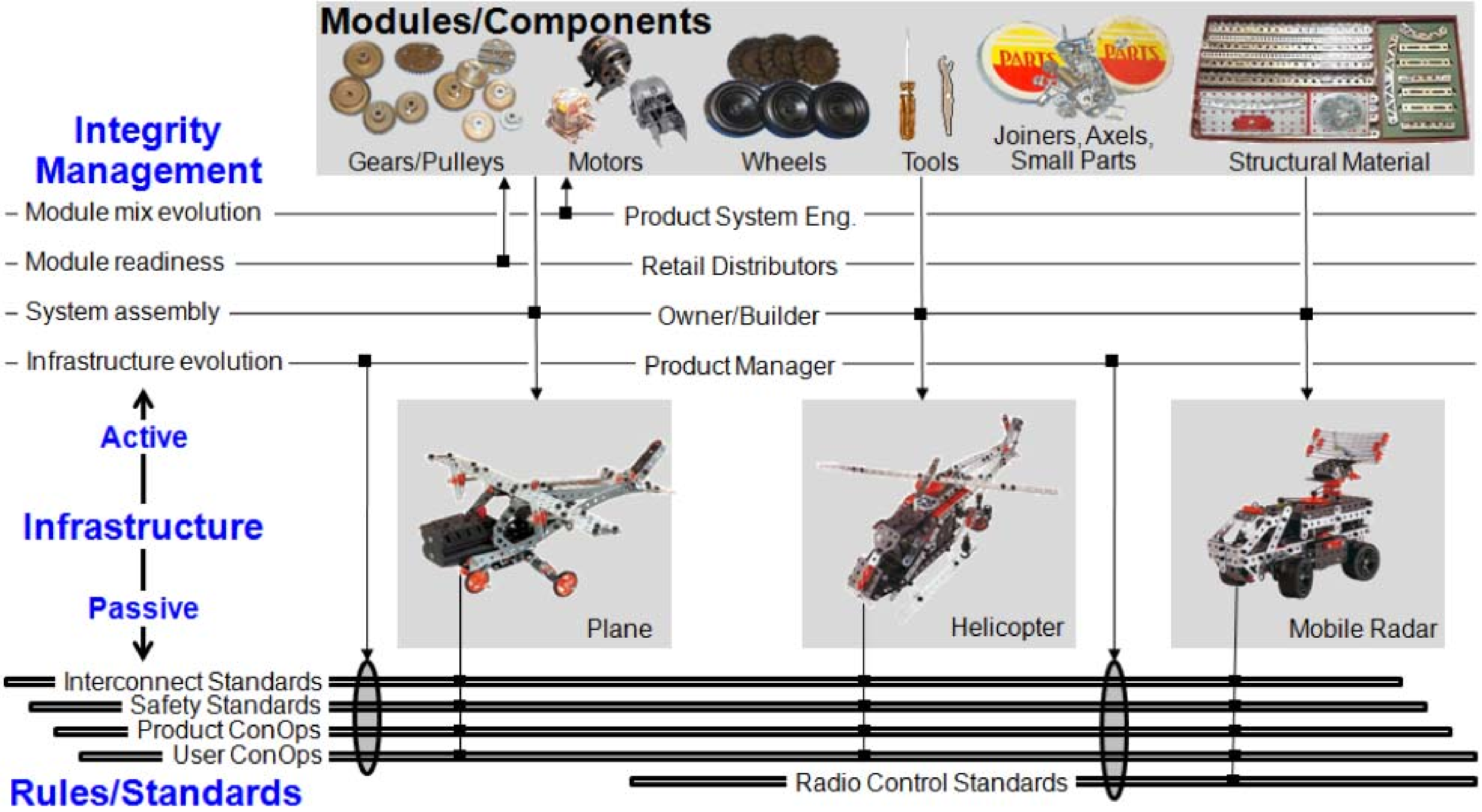
Goals of this sub-session

- Review a summary of major segments of the Agile Systems Pattern.
- From this example, learn about model-based, reusable, configurable representations of system patterns using the S*Metamodel.
- Understand the implications of the "Experience Accumulation" subsystem of the Agile Systems Pattern, for Agile Modeling, as well as other implications.
- Find out how to learn more.

A Peek Ahead

The Agile System Pattern will capture (in S*Model) the key ideas associated with the pre-MBSE Agile System Architecture:

- As in (Dove and LaBarge, 2014)



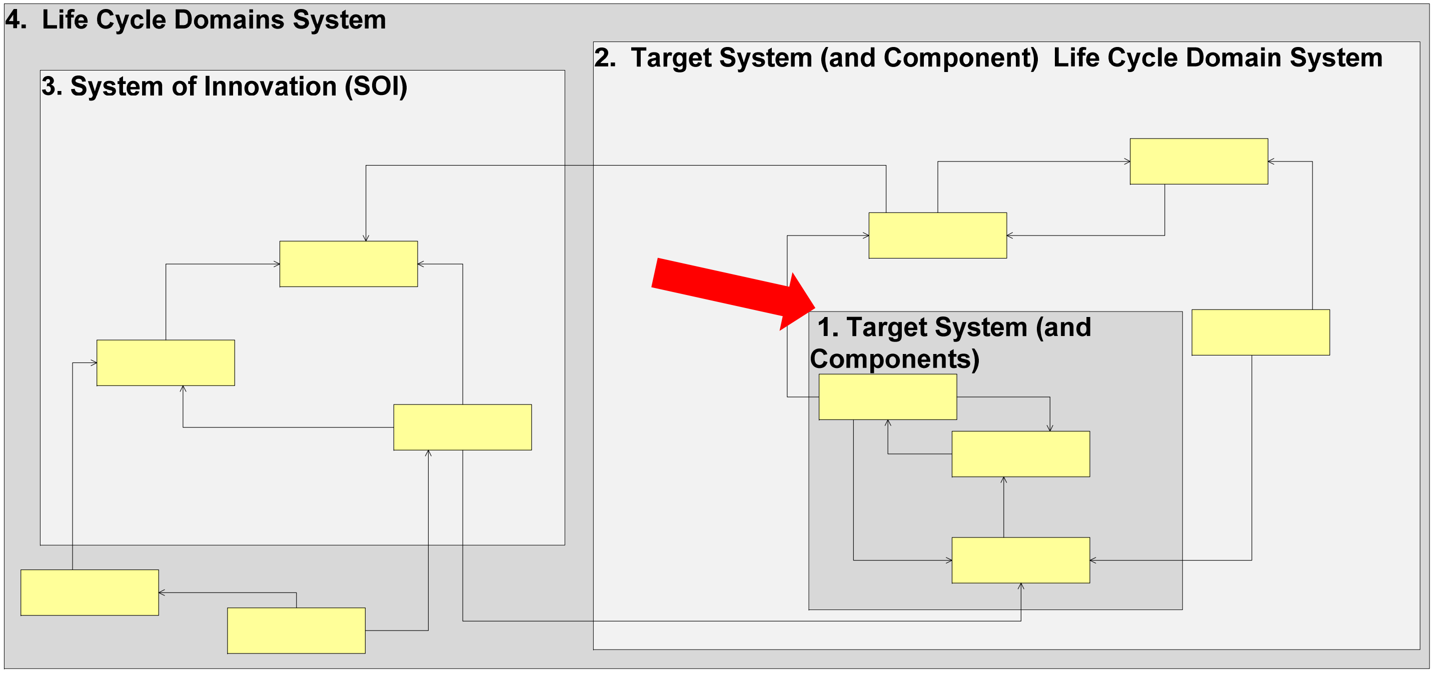
Agile System Pattern Content

- Summary (See Attachment for SysML version):
  - Domain Model
  - Logical Architecture Model
  - Physical Architecture Model
  - Input-Outputs, Interfaces, Systems of Access
  - Features Model
  - Attribute Coupling Model
  - Interactions Model
  - State Model
  - Requirements Model

The Agile System Domain Model

System 1: The **<u>Target System (and Components)</u>:** (Definition) The logical system of interest, which results from, or is subject to, innovation.

- Its behavior, characteristics, or performance are targets of the innovation (change, adaptation) process we'll introduce later.
- It is <u>potentially</u> agile.
- Examples include aircraft, satellites, the human immune system, restaurants, birds, and the health care delivery system.

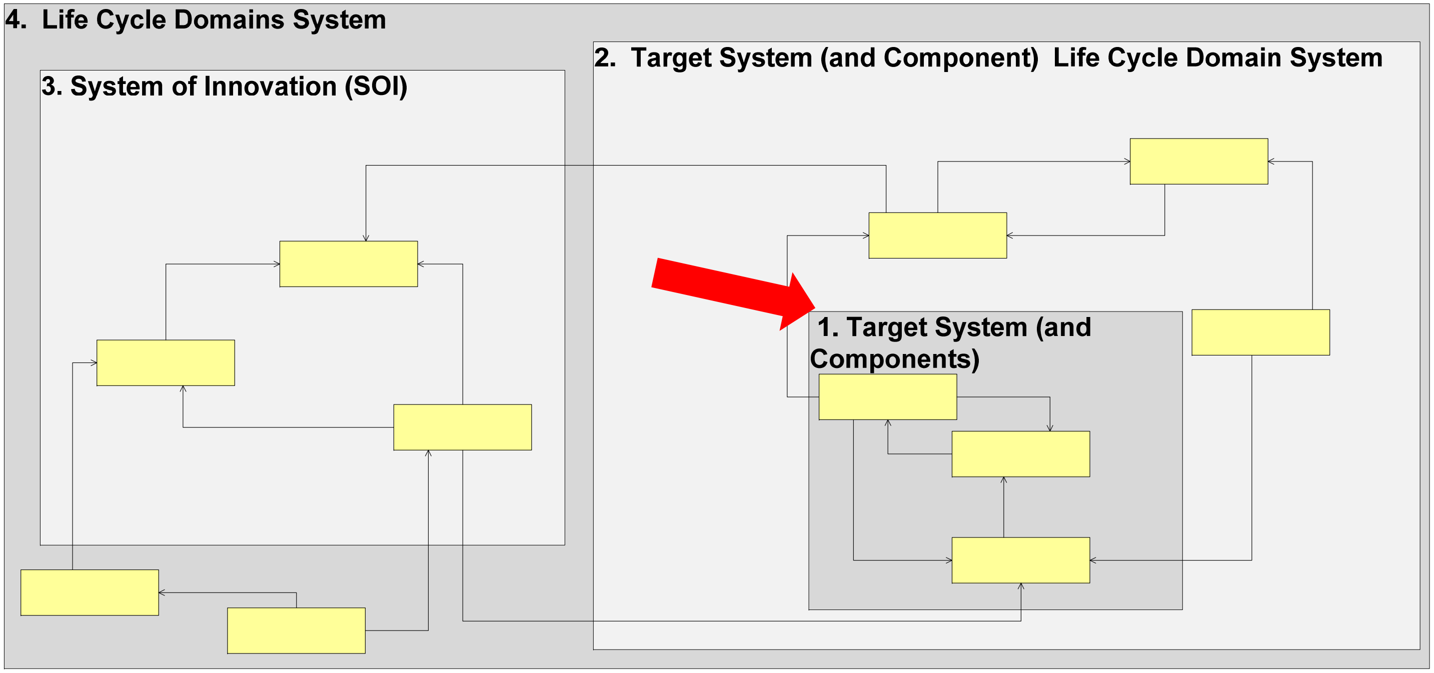

4. Life Cycle Domains System
3. System of Innovation (SOI)
2. Target System (and Component) Life Cycle Domain System
1. Target System (and Components)

Internal component roles shown (yellow shapes) are notional, and will be identified and defined later.

The Agile System Domain Model

System 1: The **Target System (and Components)**: (Definition) The logical system of interest, which results from, or is subject to, innovation.

- The Components maintained for integration into a Target System, but not yet integrated, are included in this domain.
- Notice that this idea can apply at multiple additional levels (e.g., SOS, System, Component, etc.)

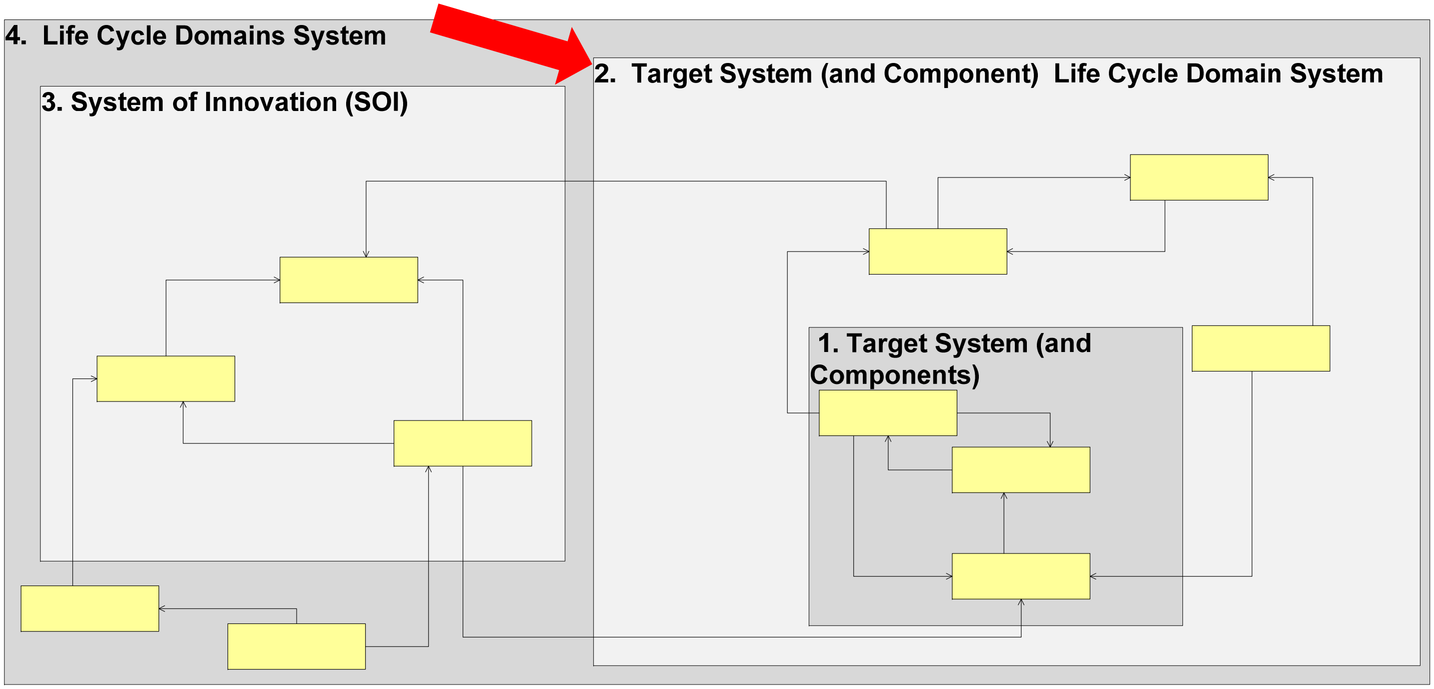

4. Life Cycle Domains System
3. System of Innovation (SOI)
2. Target System (and Component) Life Cycle Domain System
1. Target System (and Components)

Internal component roles shown (yellow shapes) are notional, and will be identified and defined later.

The Agile System Domain Model

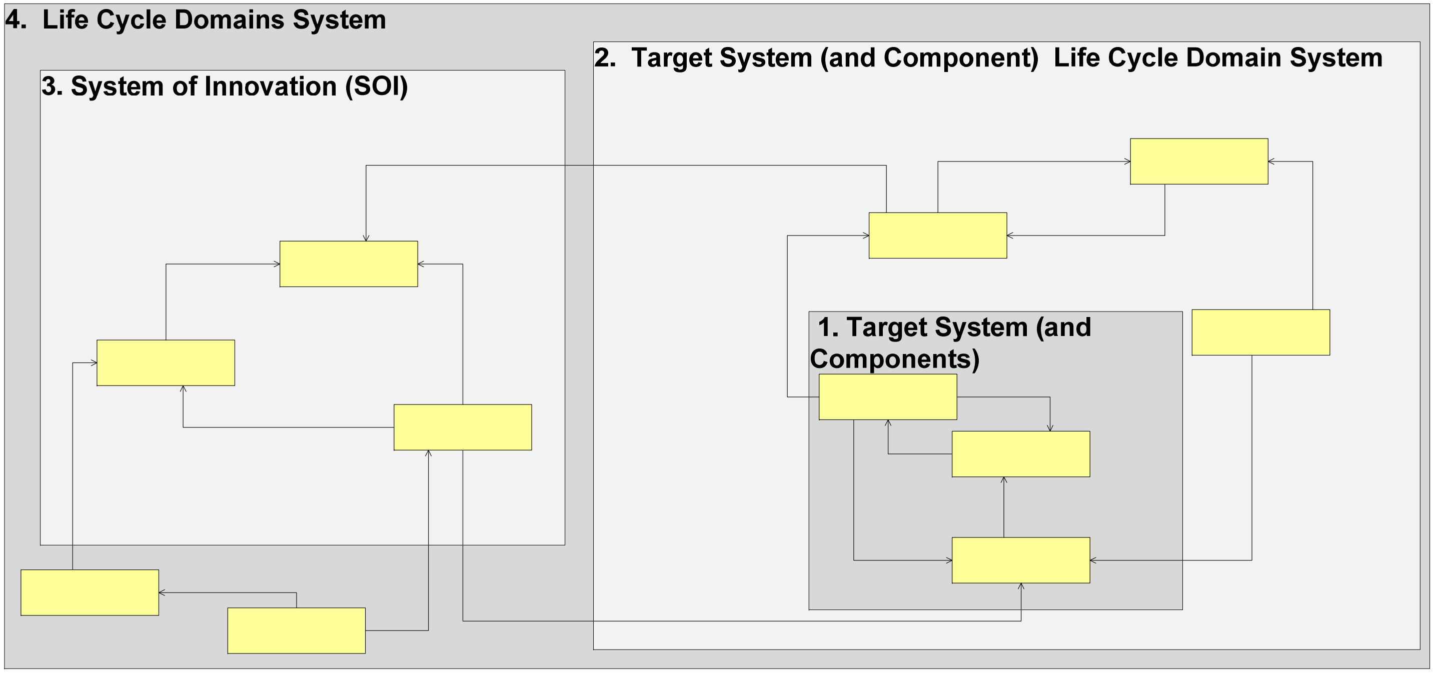
System 2: The **<u>Target System (and Component) Life Cycle Domain System</u>:** (Definition) The logical system within which the Target System will exist during its life cycle, when “in service” or otherwise. This domain includes all actors with which the Target System will directly interact during its life cycle:

– This includes any system that directly manages the life cycle of an instance of a Target System (or a Component)—production and integration systems, maintenance and operations systems, and others.


4. Life Cycle Domains System
3. System of Innovation (SOI)
2. Target System (and Component) Life Cycle Domain System
1. Target System (and Components)

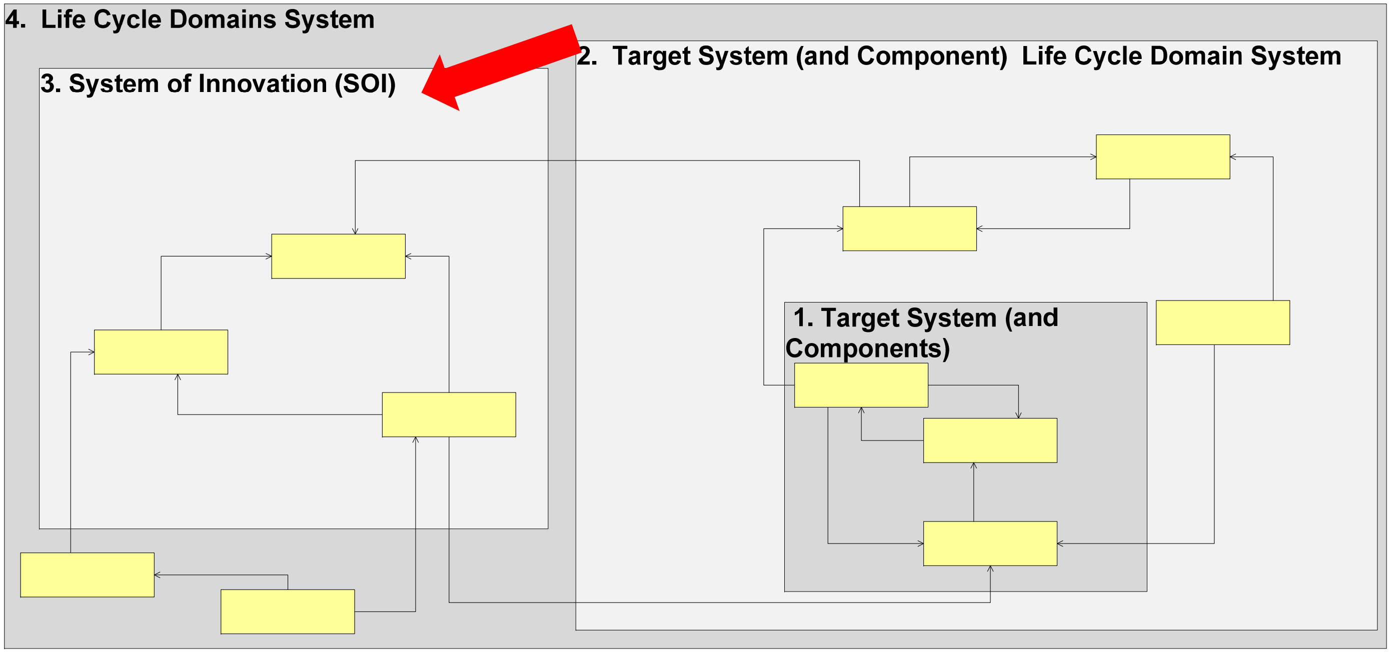
Internal component roles shown (yellow shapes) are notional, and will be identified and defined later.

- Again, remember that these are logical (behavioral) roles. In realized physical systems, a single physical system may behave as both a Target System and a system that produces, modifies, reconfigures, or otherwise manages a Target System, by having roles from each allocated to it.
- For purposes of this logical roles description, they have been identified separately.
- We will add the physical components to the model shortly.


4. Life Cycle Domains System
3. System of Innovation (SOI)
2. Target System (and Component) Life Cycle Domain System
1. Target System (and Components)

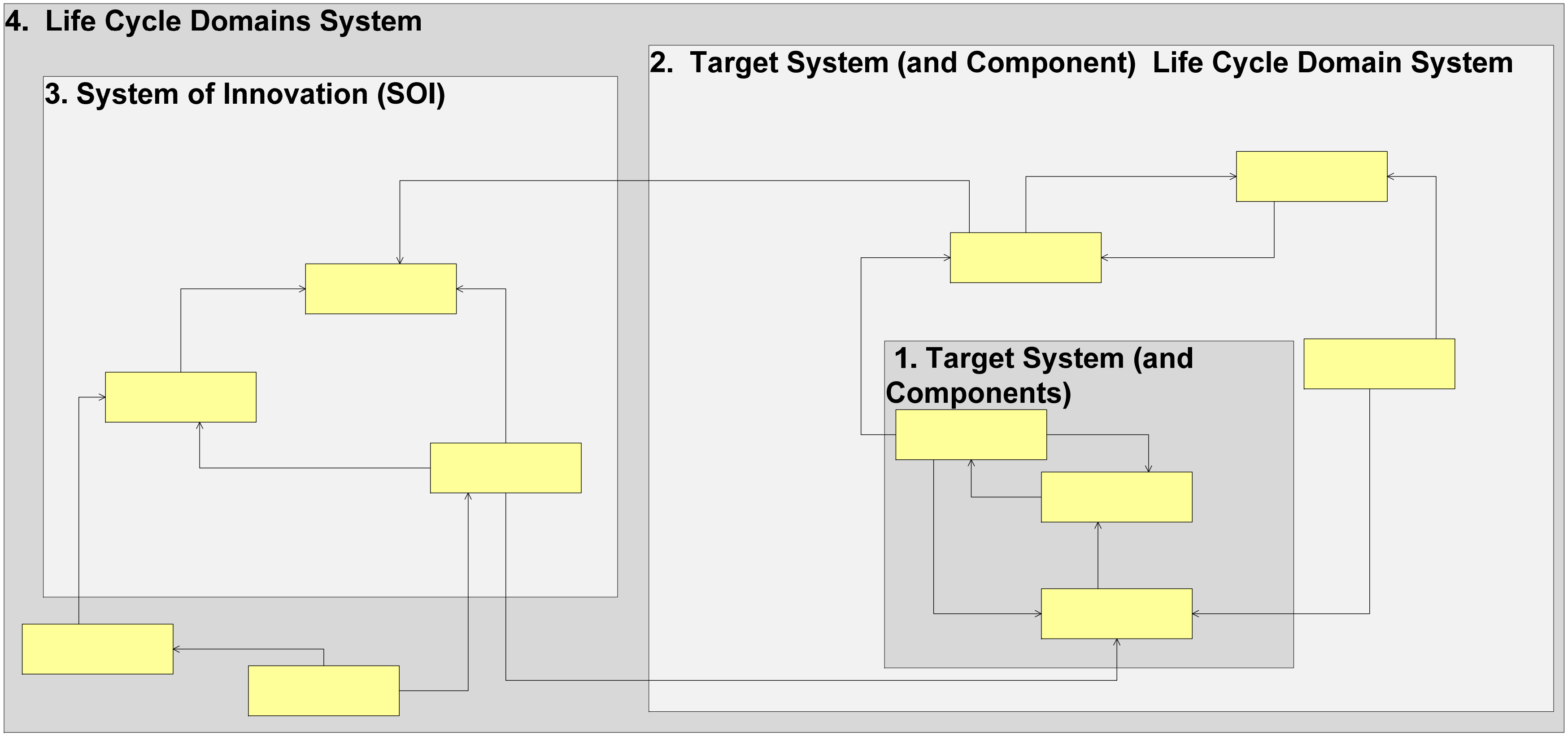
System 3: The **<u>System of Innovation</u>:** The logical system responsible for creating the possibility of (not production of) instances of Target System(s) with new or modified capabilities:

- Includes distillation of new knowledge (by observation) about Target Systems, their life cycle management, and their environmental domains, for future use.
- Also includes creation of instances of new production or other life cycle management capabilities for Target Systems, but not new instances of Target Systems.
- Engineers might think of this as the Engineering Process or the Development Process, but we have given it a more general name--to remind us that an innovation "competitor" may be operating from a cave or kitchen table, lacking a "recognized" engineering process; or, it might be a biological process that did not attend engineering school; or it might be some other type of innovation process, which we will study here.


4. Life Cycle Domains System
3. System of Innovation (SOI)
2. Target System (and Component) Life Cycle Domain System
1. Target System (and Components)

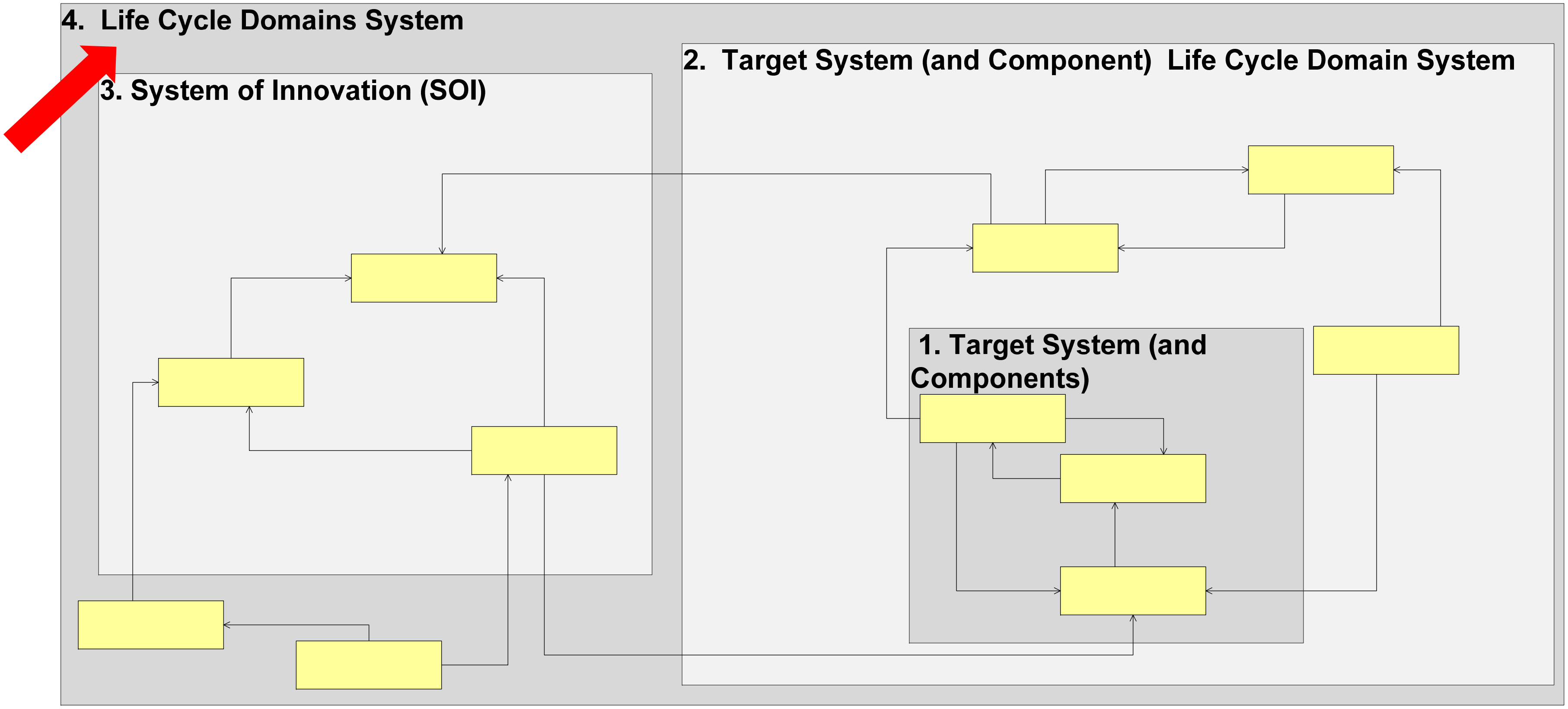
Internal component roles shown (yellow shapes) are notional, and will be identified and defined later.

- Summary so far:
  - System 2, the Target System Life Cycle Domain System produces and modifies instances of System 1, the Target Systems (and Components).
  - System 3, the System of Innovation, produces new abilities to do so, including knowledge.


4. Life Cycle Domains System
3. System of Innovation (SOI)
2. Target System (and Component) Life Cycle Domain System
1. Target System (and Components)

The Agile System Domain Model

System 4: The **Life Cycle Domains System**, consisting of the entire environment of the Target System, along with that Target System, across all of its life cycle stages, including innovation:

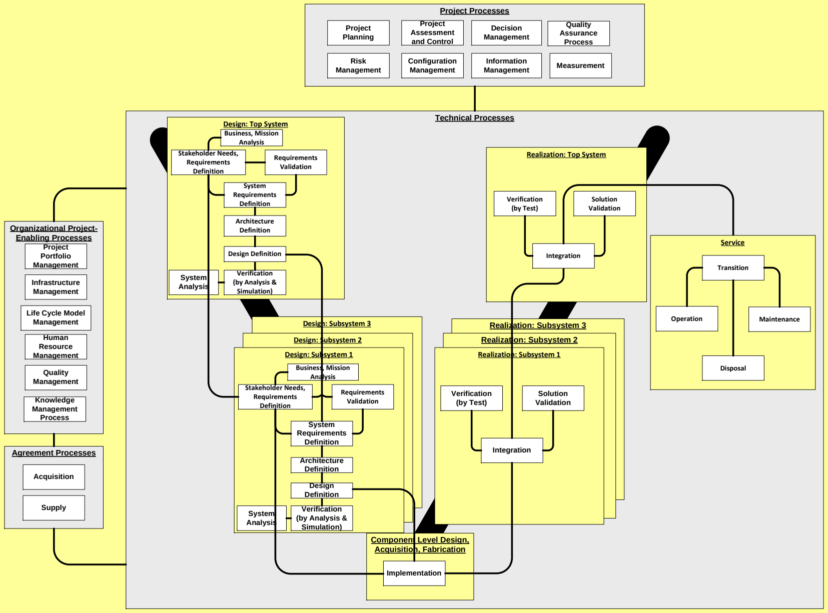

4. Life Cycle Domains System
3. System of Innovation (SOI)
2. Target System (and Component) Life Cycle Domain System
1. Target System (and Components)

Process vs. Information:

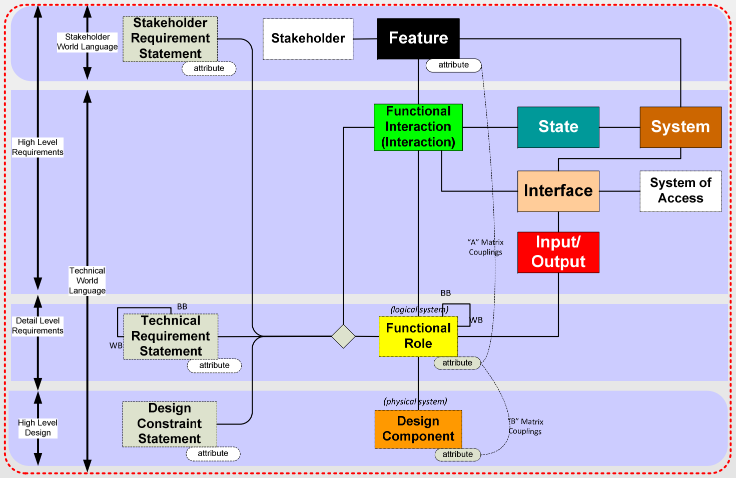
- The S*Metamodel describes the MBSE information that passes through life cycle processes, as S*Models.
- Using PBSE accelerates this process, by basing S*Model information on knowledge-managed S*Patterns.
- None of this requires any specific sequence or order of processes, which may be concurrent or otherwise, depending on strategy.
- What is the “agile trajectory” through S*Space?
- Agile Scrum strategy is to “sprint” short distances in S*Space, with fixed time and resource budgets.
- How are the “trajectory deltas” planned for each Agile Scrum?

System Life Cycle Manager: Logical Architecture

(Adapted from ISO/IEC 15288:2014)



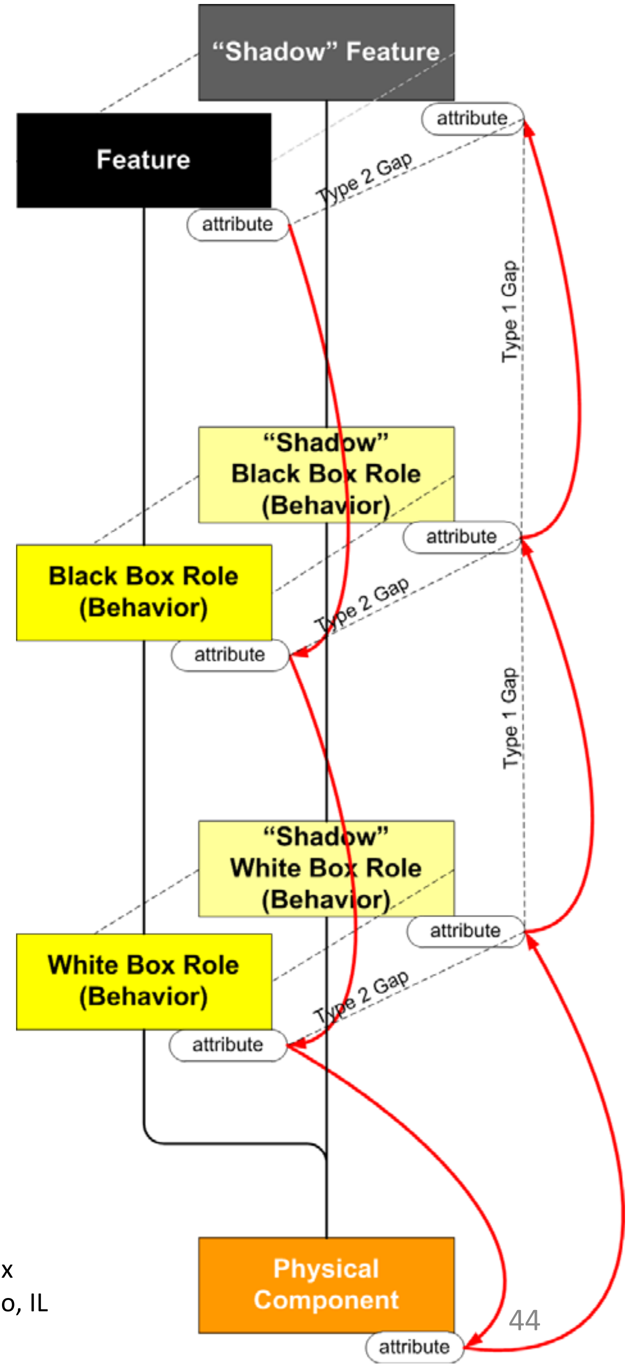
Information Passing Through Processes Above



(S*Metamodel Summary)

Agile Trajectories Are Driven By Gradient Field & Gaps:

- Type 1 and Type2 Gaps measure the gradient field used to establish incremental Agile Scrum “sprint” deltas.
- Driven by Stakeholder Feature Attribute Coupling metrics



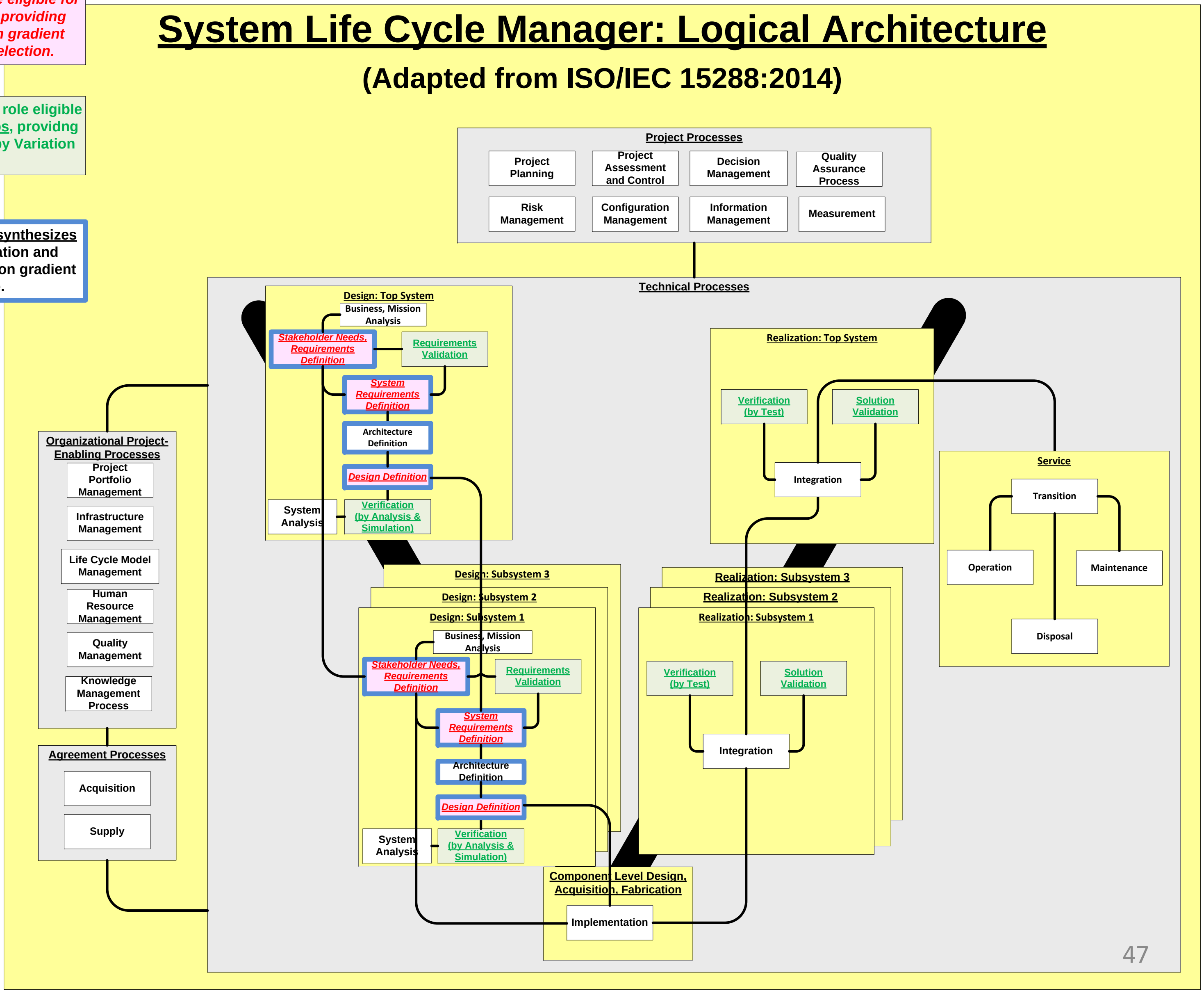
From: “Innovation as Emergence: Hybrid Agent Enablers for Evolutionary Competence”, W. Schindel, Complex Adaptive Systems, Volume 1, Cihan H. Dagli, Editor in Chief, *Procedia Computer Science*, Elsevia, 2011, Chicago, IL

Methodology for Selecting Agile Scrum Delta Vectors

***Red Italics Underline** means role eligible for measurement of **Type 1 gaps**, providing feedback to setting innovation gradient direction by Variation and Selection.*

Green Roman Underline means role eligible for measurement of **Type 2 gaps**, providng innovation gradient direction by Variation and Selection

Thick blue outline means role synthesizes information eligible for Variation and Selection, and setting innovation gradient direction in S*Space.



System Life Cycle Manager: Logical Architecture

(Adapted from ISO/IEC 15288:2014)

Project Processes

Project Planning

Project Assessment and Control

Decision Management

Quality Assurance Process

Risk Management

Configuration Management

Information Management

Measurement

Technical Processes

Organizational Project-Enabling Processes

Project Portfolio Management

Infrastructure Management

Life Cycle Model Management

Human Resource Management

Quality Management

Knowledge Management Process

Agreement Processes

Acquisition

Supply

Design: Top System

Business, Mission Analysis

Stakeholder Needs, Requirements Definition

Requirements Validation

System Requirements Definition

Architecture Definition

Design Definition

System Analysis

Verification (by Analysis & Simulation)

Design: Subsystem 3

Design: Subsystem 2

Design: Subsystem 1

Business, Mission Analysis

Stakeholder Needs, Requirements Definition

Requirements Validation

System Requirements Definition

Architecture Definition

Design Definition

System Analysis

Verification (by Analysis & Simulation)

Component Level Design, Acquisition, Fabrication

Implementation

Realization: Top System

Verification (by Test)

Solution Validation

Integration

Realization: Subsystem 3

Realization: Subsystem 2

Realization: Subsystem 1

Verification (by Test)

Solution Validation

Integration

Service

Transition

Operation

Maintenance

Disposal