Object Based Systems Engineering (OBSE)

TECHNOLOGY DRIVEN. WARFIGHTER FOCUSED.

Pradeep Mendonza
Team Lead (Acting)
Policy, Process & Tools Development
Systems Engineering Group
US Army - TARDEC
(586) 282-4835
pradeep.mendonza.civ@mail.mil

John A. Fitch
Senior Systems Engineer
Science Applications International Corporation
(586) 943-6323
john.a.fitch.ctr@mail.mil

27 October 2011

OBJECTIVES

OBJECTIVES:

1. Communicate the background (past & current state) & motivation for Object Based Systems Engineering (OBSE)
2. Communicate the evolution of Systems Engineering practice
3. Communicate the concepts of OBSE
4. Communicate the roadblocks to realizing OBSE
5. Communicate the principles of Object Based Systems Engineering (OBSE)
6. Illustrate each OBSE principle
7. Communicate the benefits of OBSE

Background

Background

- Systems Engineering is knowledge-based process. Its success depends on timely, efficient and effective knowledge capture and sharing among a diverse set of system stakeholders, contributors and implementers
- Historically Systems Engineering (SE) practitioners have focused on producing document-based artifacts to relay SE knowledge to stakeholders and developers
- More recently a large part of the community has moved towards using view-based artifacts; e.g. DODAF to visually communicate SE knowledge

Limitations of document-based artifacts

Limitations of document-based artifacts

- Holding emerging system knowledge hostage until the next document review cycle
- Triggering the replication of much potentially-common data between documents
- Capturing information in large, free-form text paragraphs that fail to separate and singularize important system data elements.
- Focusing engineers on writing tasks, not thinking tasks (e.g. writing requirements instead of defining them).
- Conflating document structure with system decomposition hierarchies.
- Forcing users to create, maintain and synchronize redundant copies of data elements.
- Creating unnecessary manual effort to maintain unique requirement identifiers and traceability matrices.

Limitations of view-based artifacts

Limitations of view-based artifacts

- Views don’t offer full coverage of all classes of SE knowledge.

  They don’t capture the full decision and derivation trace that enables proactive impact/change analysis and reuse of knowledge across system life cycle phases.

- Views focus engineers on drawing diagrams or populating tables.

  While this is much better than a document-authoring paradigm, the views may still become an end in themselves rather than the natural byproduct of continuous and effective Systems Thinking.

- Views are often populated (e.g. drawn, compiled) after-the-fact from other sources.

  While they contain objects and relationships, this data is often a copy of the original/master that is stored elsewhere. This increases the effort/cost/time required to maintain a consistent and traceable Systems Engineering knowledge-base.

Motivation for OBSE

Motivation for OBSE

- Documents and views are simply containers that hold objects. SE knowledge is comprised of sets (classes) of objects that are related to each other. By directly managing these objects, containers can be reproduced as desired.

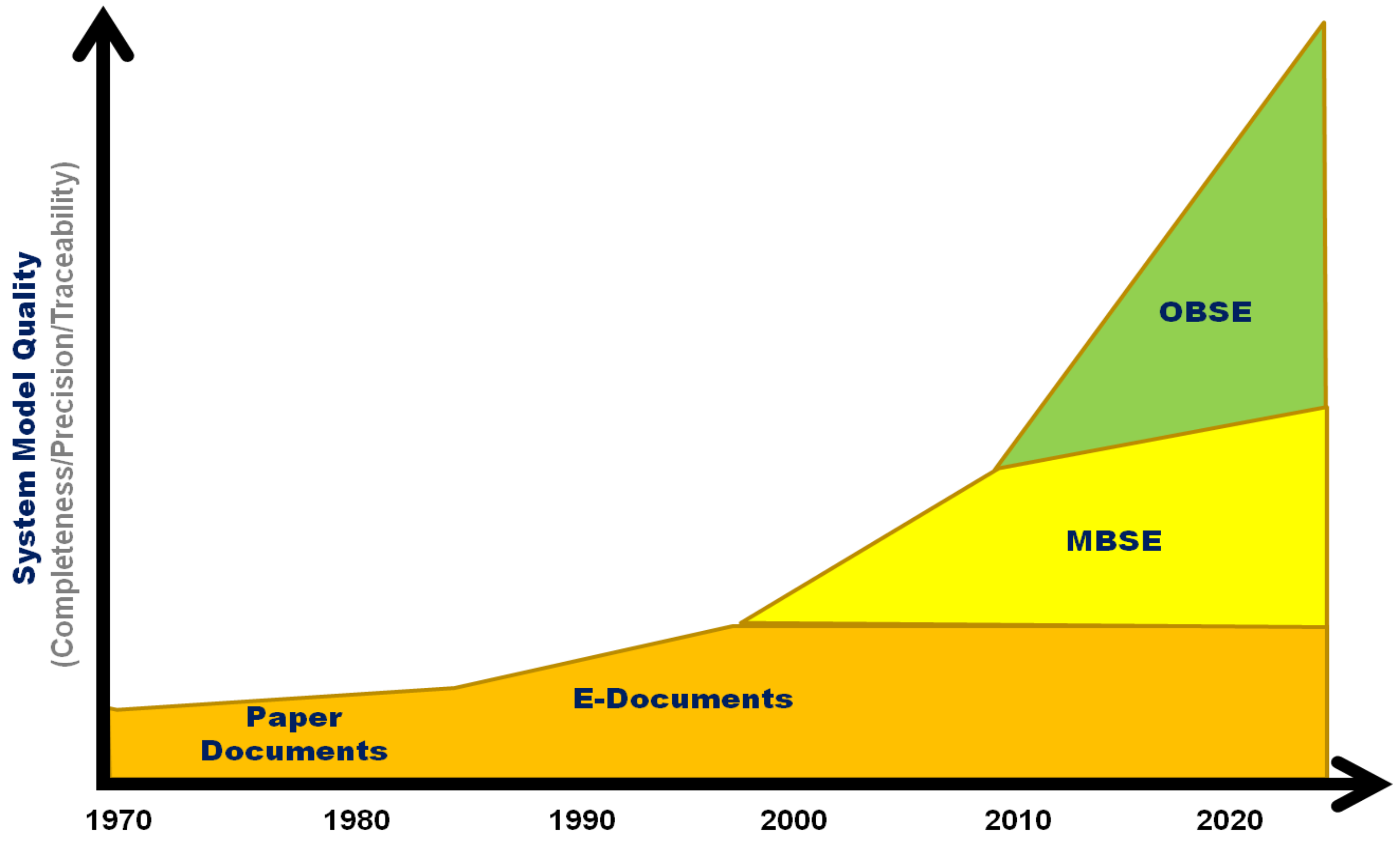
- MBSE proponents and initiatives clearly have Object Based System Engineering as their end goal, but the state of the practice lags the vision.

The goal of this paper is to help accelerate the realization of Object Based System Engineering in the everyday practices of the defense community.

RDECOM

Evolution of Systems Engineering Practice

TARDEC



This paper proposes a shift in the focus of SE from documents and views to objects by proposing a set of principles to integrate the information created by multiple, diverse SE methods.

Object Based SE Concepts

Object Based SE Concepts

- Object Based Systems Engineering is based on the simple concept that SE knowledge is comprised of sets (classes) of objects that are related to each other.
- The essential elements of Systems Engineering can be represented by ***objects*** that are comprised of and defined by ***attributes*** and associated through ***relationships***. These objects can be grouped into logical classes using an affinity process.
- It is certainly possible to create a comprehensive information architecture that captures Systems Engineering knowledge, but "possible" does not imply "easy".

Information Architecture Benefits

Information Architecture Benefits

- Reproduce artifacts (documents, paragraphs, diagrams and tables) by automated rule-based assembly of sets of objects.
- Populate paragraphs by the concatenation of object attributes and relationships.
- Populate diagrams from objects (nodes) and their relationships (lines).
- Populate table rows (objects) and columns (attributes, linked objects).
- Shift the focus from artifact (document, view) reviews to object quality.
- Analyze diagrams and tables using rule-based exception reports.
- Eliminate the variability between the actual system model (requirements, design, architecture) and the views used to communicate the model.

Roadblocks to OBSE realization

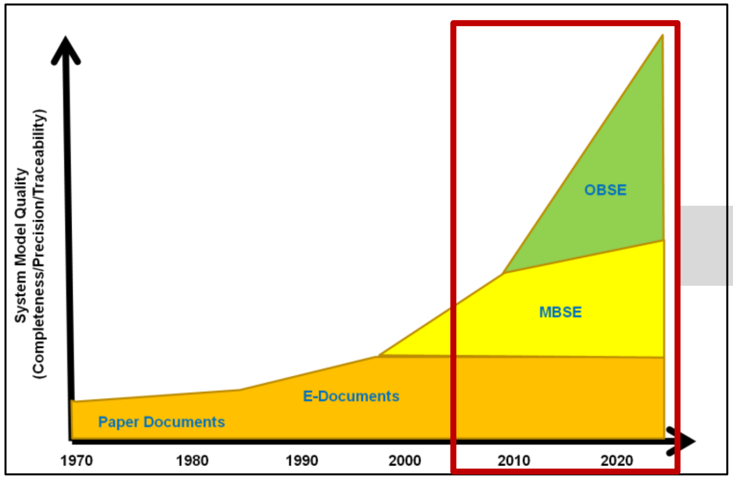
Tool limitations

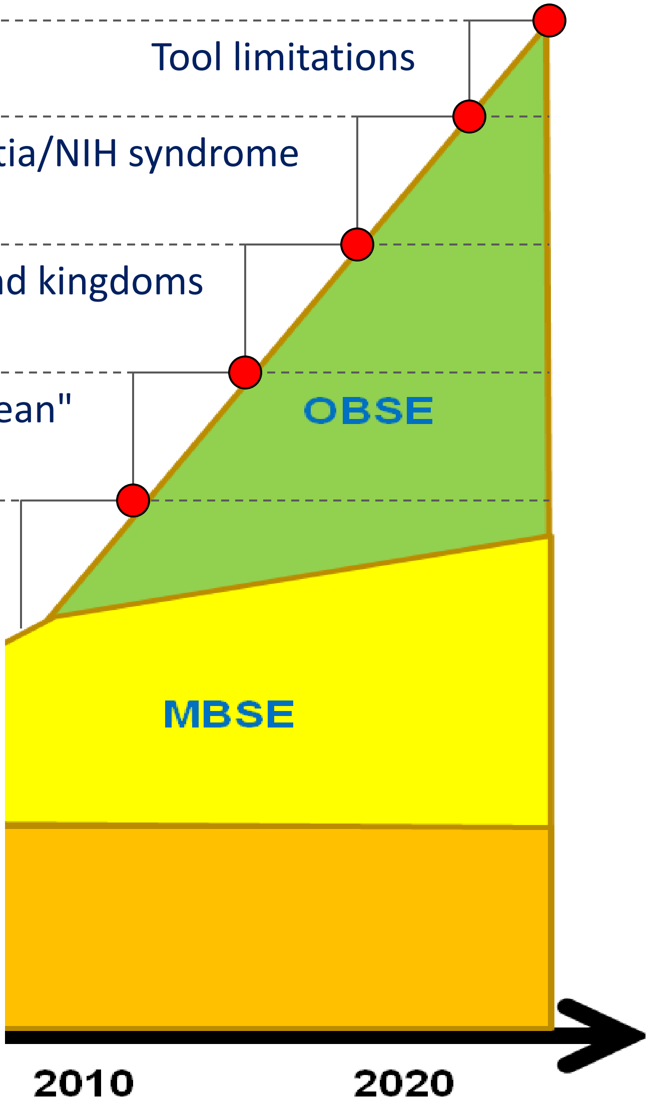
Process inertia/NIH syndrome

Information silos and kingdoms

Tight budgets - fear of "boiling the ocean"

Paradigm blinders - imbalanced emphasis on specific methods





RDECOM

OBSE Principles

TARDEC

Object based systems engineering principles

1. Map all SE knowledge to object classes and subclasses
2. Refine this information architecture against multiple SE methods to make it as lean as possible (maximize cohesion, minimize coupling).
3. Create all objects in context
4. Define each object as a set of lean attributes and relationships
5. Strive for zero redundancy
6. Maintain continuous traceability as knowledge is derived.
7. Capture the precious and transient logic behind this knowledge derivation.
8. Leverage the relationships between objects to proactively manage change.
9. Maintain continuity of objects across system/product life cycles and phases.
10. Harvest and reuse knowledge patterns for each class of object

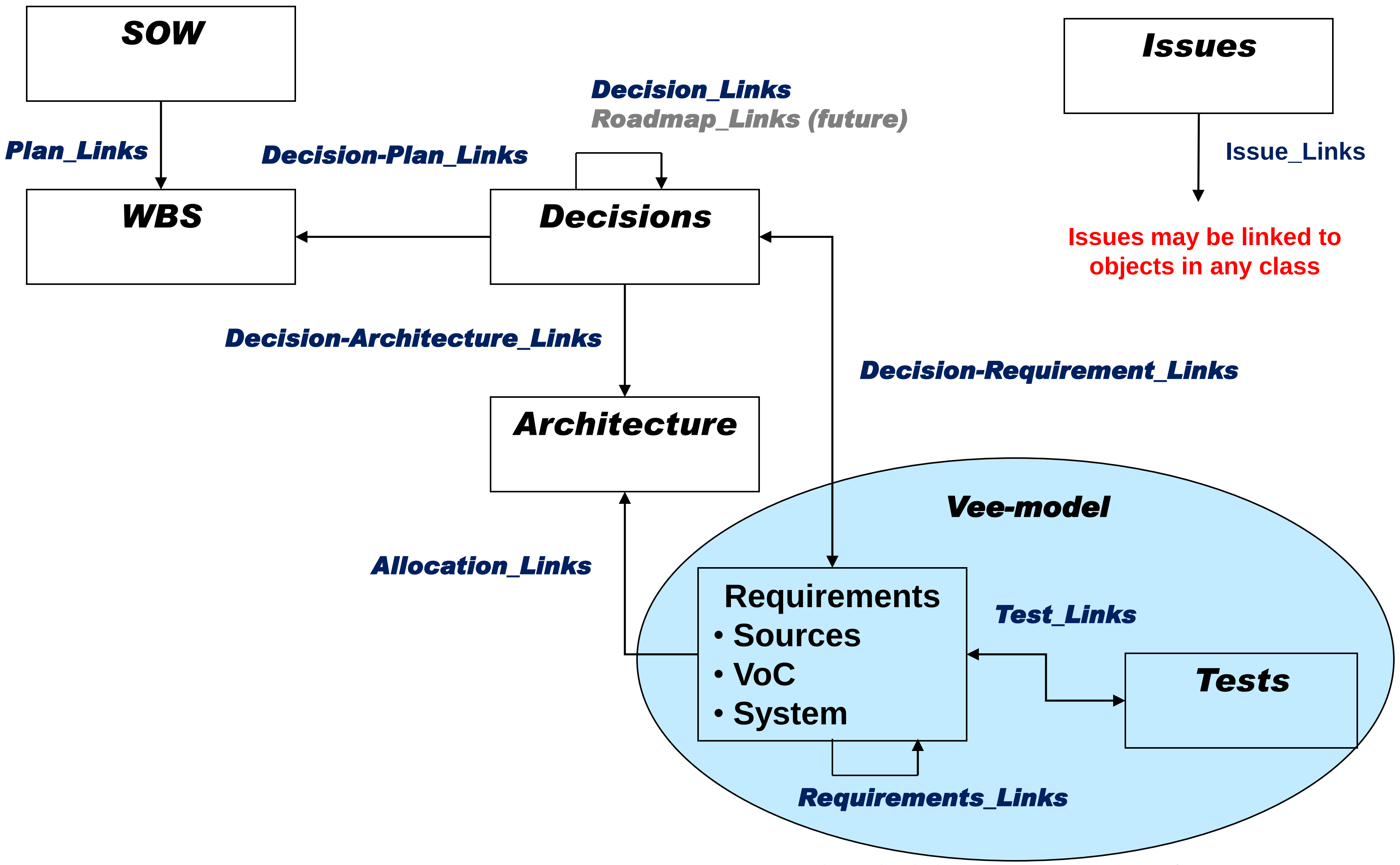
OBSE Principle 1

1. Map all SE knowledge to object classes and subclasses

Create an initial information architecture model by answering the following questions:

- What are the primary types of SE knowledge required to support the SE process?
- Which classes and subclasses support the anticipated set of SE process use cases (types of systems/products to be developed; life cycle phase, project size, system context)?
- How do these classes of information relate to one another? What classes of relationships connect various types of SE data?
- What are the most vital and volatile classes of objects/relationships to preserve and maintain?

OBSE Information Architecture



TECHNOLOGY DRIVEN. WARFIGHTER FOCUSED.

Questions traceability can answer

SOW What is our scope & charter?	How will we accomplish our charter? Is our plan adequate?			**N-Squared Diagram Legend**	
How will work flow down to others?	**WBS** What's our plan? Who's responsible? Is plan adequate?				
	How will we analyze or implement this decision?	**DECISIONS** Top N decisions? Status? Rationale? Consequences?	Why does this component exist? What role does it play?	Where did this requirement originate? Change impact?	
			ARCHITECTURE Components in our solution? Interfaces?		
		What decisions did this req't drive? Budget allocation? Change impact?	Allocated requirements? Budget flow-down?	**REQUIREMENTS** Success = ? Clear? Complete? Source?	Requirements per test? Verification coverage?
				Requirements met? Priority gaps to fix?	**TESTS** Test events/cases? Test enablers? Results/findings?

N-Squared Diagram Legend

Node A	A-to-B interaction
B-to-A interaction	**Node B**

Read the interactions clockwise

TECHNOLOGY DRIVEN. WARFIGHTER FOCUSED.

UNCLASSIFIED

OBSE Principle 2

2. *Refine this information architecture against multiple SE methods to make it as lean as possible*

Information architecture is driven by the set of SE methods engines that have been selected to power the SE process. Refine the information architecture to make it scalable across a broader range of methods and ask the following:

- ➢ What methods could be used to create each class of object? What methods create the primary relationships among object classes?
- ➢ How would different methods engines change the information model (e.g. add new classes, attributes or relationships?
- ➢ Can an information model be created that captures the superset of all the classes/attributes/relationships required by the full range of methods engines under consideration?
- ➢ How can this model be made more lean; simplified to reduce the number of object classes and/or relationships?

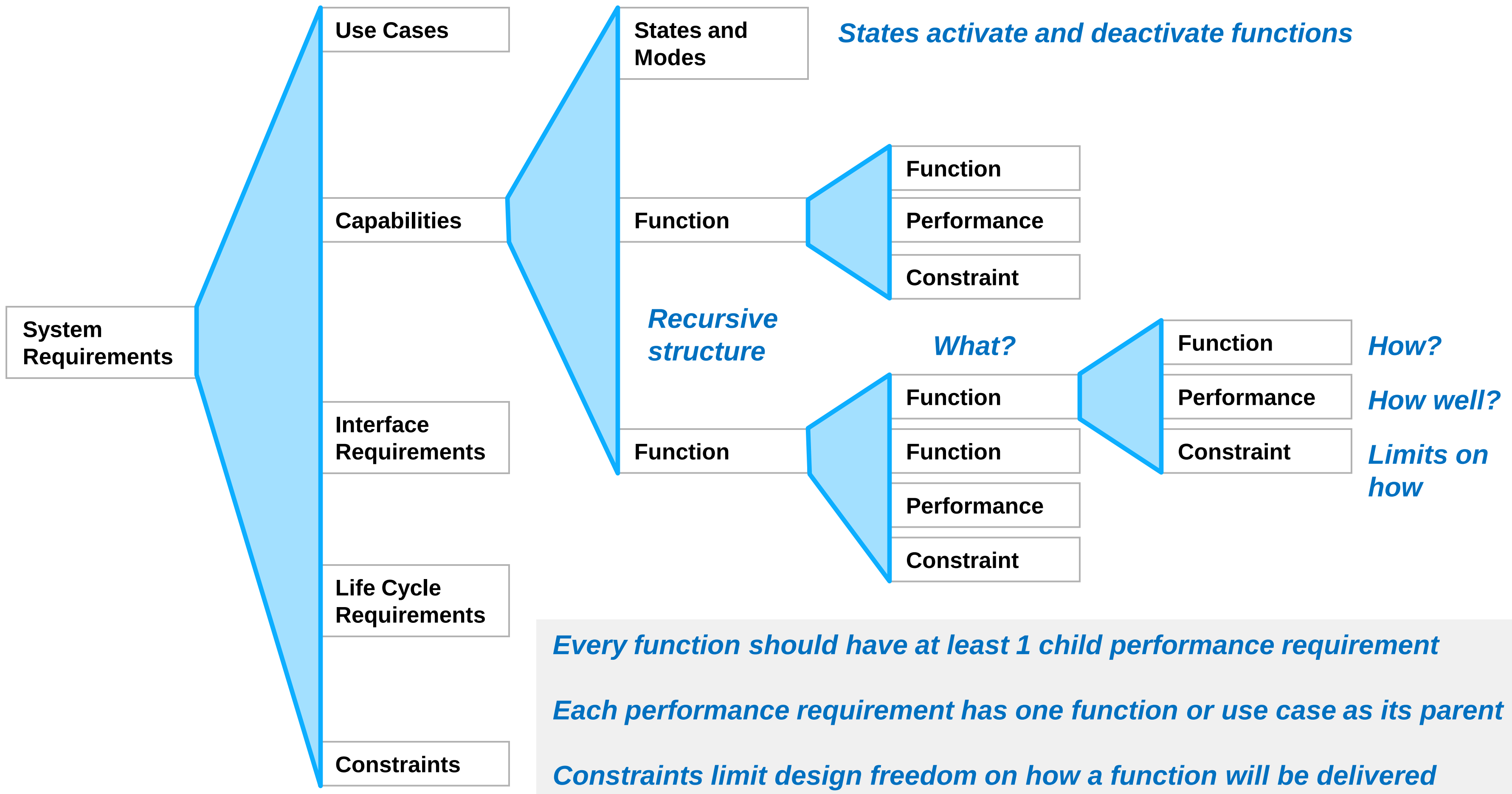
OBSE Principle 3

3. *Create all objects in context (within a hierarchy appropriate to its class)*

- SE knowledge represents a network of associated objects.
- Within each primary class of objects, a hierarchical structure (taxonomy) provides an efficient structure.
- These class hierarchies typically include subclasses arranged in a recursive pattern.
- Class hierarchies are valuable knowledge patterns.
  - Jump-start new projects by seeding the SE knowledge-base with a proven set of relevant objects.
  - They highlight missing (but valuable) data as holes in the recursive structure. For example:
    - Every functional requirement should have at least 1 performance requirement that specifies "How well?" the function must be performed.

Requirements Hierarchy

Requirements Hierarchy



Every function should have at least 1 child performance requirement

Each performance requirement has one function or use case as its parent

Constraints limit design freedom on how a function will be delivered

Use cases may be modeled as a thread or flow of functions

OBSE Principle 4

4. *Define each object as a set of lean attributes and relationships (avoid free-form text)*

The document model encourages free-form text paragraphs.

- Leads to jumbled object, attribute and relationship data.
- Individualized and situation-driven writing paradigm
- Leaves precise translation as an exercise for the individual reader.
- Ad hoc, non-repeatable process contributes to system model ambiguity

Growing system complexity demands precise capture of Systems Engineering knowledge as objects.

Fundamental enemies of project success = uncertainty and ambiguity.

- Combine to produce overwhelming complexity -> program failures.

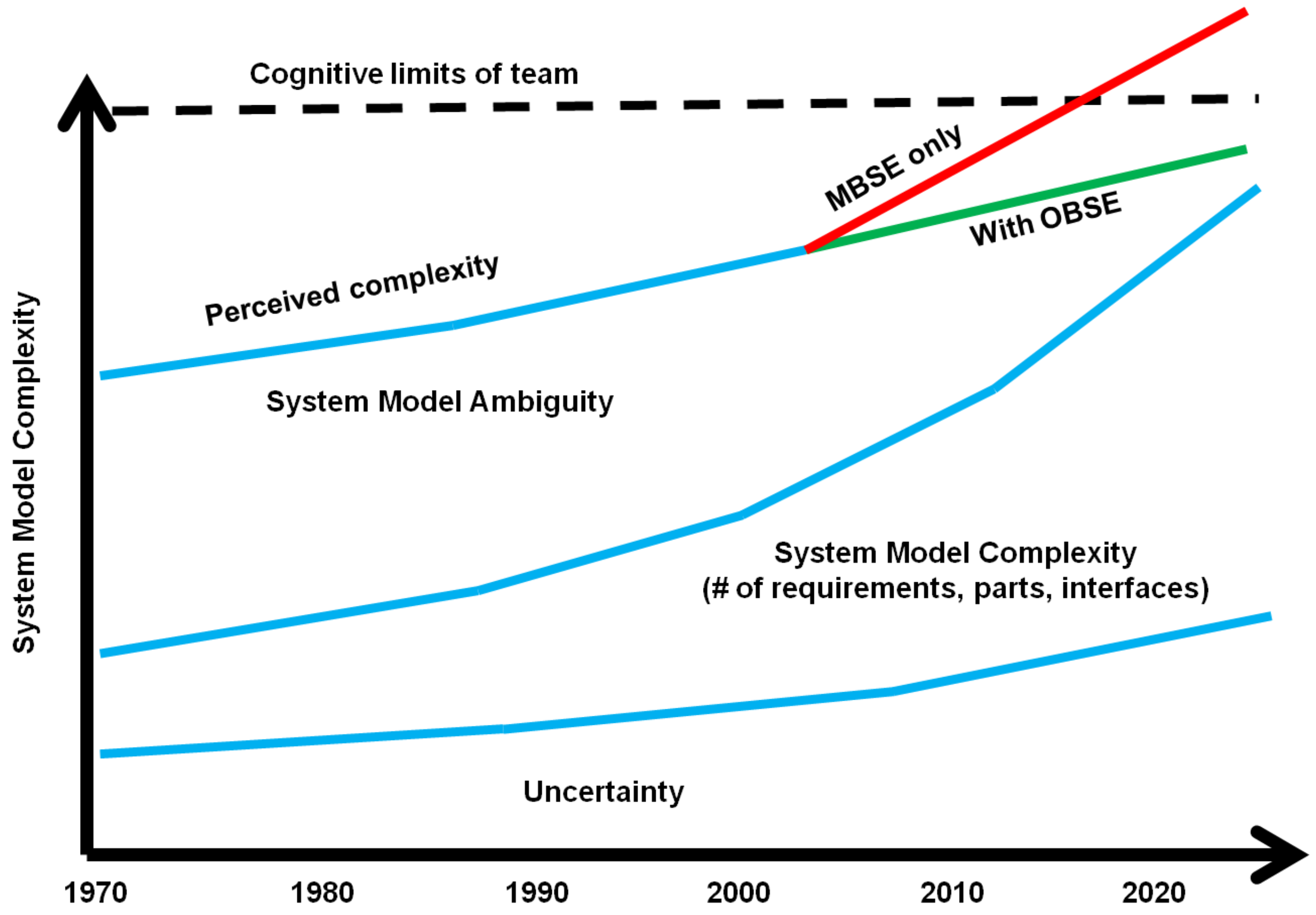
Ambiguity vs. Uncertainty

- **Uncertainty**
  - A product of the real-world unknowns and unknowable’s.
  - Increasing with the pace of technology change/turnover
  - Reduced through investments in knowledge-creating tasks (e.g. simulation and prototypes) but not driven to zero.
  - May be managed, but can't be eliminated
    - There are no facts about the future!

- **Ambiguity**
  - A self-inflicted wound;
  - Results from fuzzy and ad hoc methods that create high variance in the definition, context, derivation and interpretation of SE knowledge.
  - The goal of OBSE is to drive ambiguity toward zero.
    - Keep the system model's perceived complexity within the cognitive limits of a team.

Ambiguity vs. Uncertainty

Ambiguity Vs Uncertainty



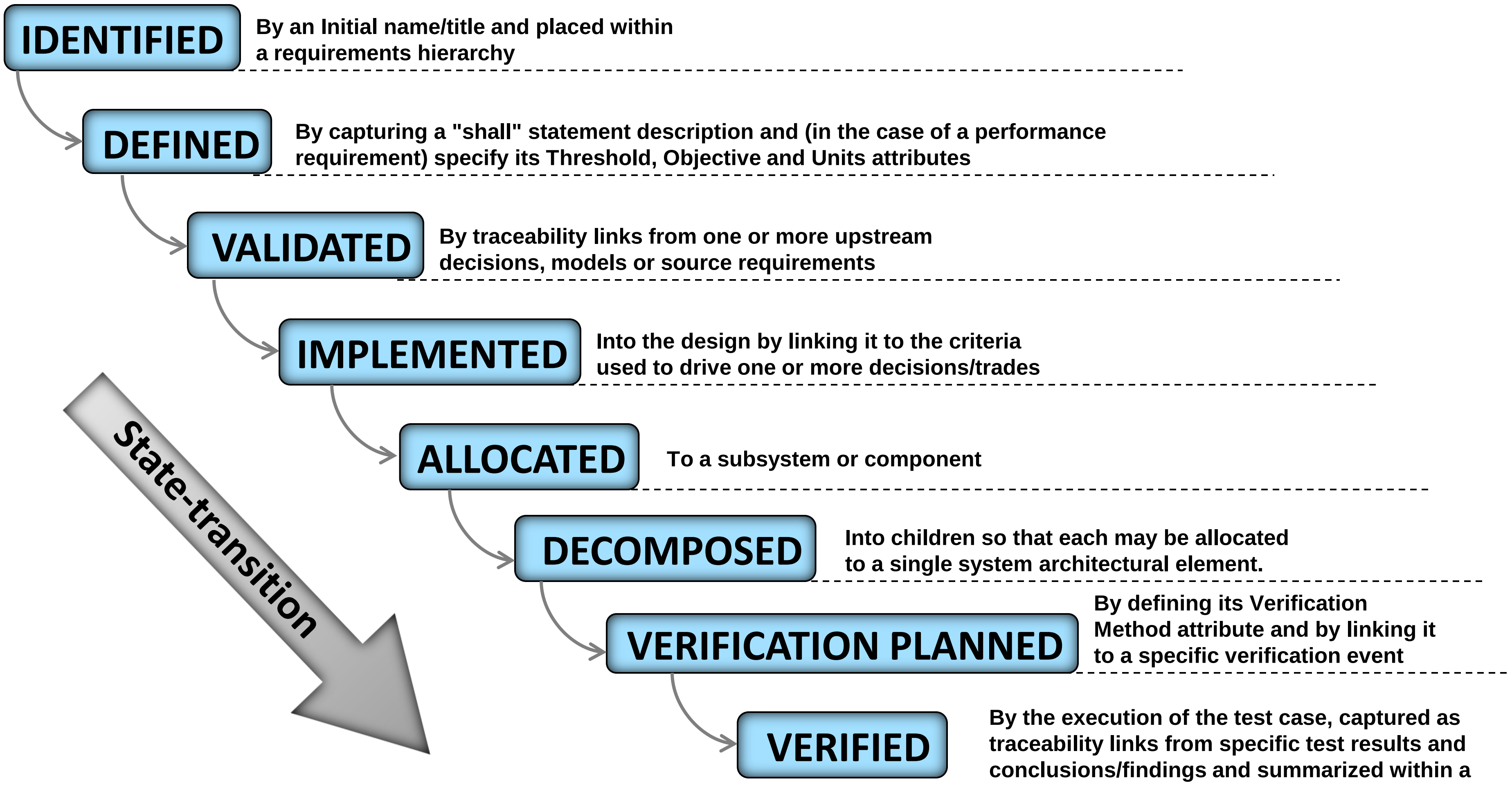
OBSE Principle 5

5. *Strive for zero redundancy (store a single instance of an object; visualize in many ways)*

- Avoid self-inflicted complexity.
  - Don't copy objects to populate documents or views
  - Maintain a single master instance of each object
  - Maintain the leanest possible information model to represent the problem and system
- Focus version control on object masters
  - Life cycles states of each object = object versions
  - Capture states as changes to the attributes and relationships associated with each object.

Requirement States

Requirement States



IDENTIFIED	By an Initial name/title and placed within a requirements hierarchy
DEFINED	By capturing a "shall" statement description and (in the case of a performance requirement) specify its Threshold, Objective and Units attributes
VALIDATED	By traceability links from one or more upstream decisions, models or source requirements
IMPLEMENTED	Into the design by linking it to the criteria used to drive one or more decisions/trades
ALLOCATED	To a subsystem or component
DECOMPOSED	Into children so that each may be allocated to a single system architectural element.
VERIFICATION PLANNED	By defining its Verification Method attribute and by linking it to a specific verification event
VERIFIED	By the execution of the test case, captured as traceability links from specific test results and conclusions/findings and summarized within a Verification Status or Compliance attribute

OBSE Principles 6 & 7

6. Maintain continuous traceability as knowledge is derived.

- Maintain derivation traceability continuously (simple, cheap)
  - Never have enough time to backfill it (expensive, impractical).
- Loss of this traceability:
  - Multiplies cost of proactive impact/change analysis (what-ifs) or makes it impossible without original SMEs with perfect recall.

7. Capture the precious and transient logic behind this knowledge derivation.

- Derivation traceability is very precious, but transient knowledge.
  - More than a link; includes the derivation rationale, i.e. How? or Why?
- Capture minority viewpoints and discussion threads for each object of interest.
  - Lessons learned to drive process improvement.

OBSE Principles 8, 9 & 10

8. *Leverage the relationships between objects to proactively manage change*

- ➢ Walk the links between objects and assess the ripple effect
- ➢ Human-in-the-loop thought process, supplemented by simulation models.

9. *Maintain continuity of objects across system|product life cycles and phases*

- ➢ Documents and views pass little useful information between system life cycle phases.
  - Continuity of thought = continuity of team members.
- ➢ OBSE database enables maximum reuse through continuity of objects/states.

10. *Harvest and reuse knowledge patterns for each class of object*

- ➢ Harvest knowledge within class hierarchies and rule-based data structures.
- ➢ Enable knowledge reuse across many domains (project types, systems, technologies)
- ➢ Increases the ROI from investments in Systems Engineering discipline.

OBSE Benefits

Benefits of Object Based System Engineering (OBSE)

Applied OBSE requires skill, creativity and balanced judgment

Potential benefits:

- Simplify SE tasks. Leanest information model -> leanest value-focused task model.
- Reduce overlapping efforts and information silos.
- Foster the insight that leads to innovation.
- The leanest possible information model -> innovative insights. Efficient, focused brainstorming. Increased collaboration.
- Improve solution quality. Rule-based exception reports highlight missing, but needed knowledge
- Accelerate development. Maximize team's ability to do parallel and aligned thinking.

Contact Information

Pradeep Mendonza
Team Lead (Acting)
Policy, Process & Tools Development
TARDEC Systems Engineering Group
(586) 282-4835
pradeep.mendonza.civ@mail.mil

John A. Fitch
Senior Systems Engineer
Science Applications International Corporation
(586) 943-6323
john.a.fitch.ctr@mail.mil