Lifecycle Modeling – An Approach to Simplified, Rapid Development, Operations, and Support

Steven H. Dam, Ph.D., ESEP
President, SPEC Innovations
571-485-7805
Steven.dam@specinnovations.com

October 24, 2011

Overview

- Why a New Language?
- Lifecycle Modeling Language Overview
- Suggested LML Processes
- LML Tool Needs
- Use of LML for Architecture
- Use of LML for Systems Design
- Use of LML for Software Development
- Use of LML in Test and Evaluation
- Use of LML in Operations and Support
- Summary

WHY A NEW LANGUAGE?

We already have SysML … what else do you need!

Complexity

Complexity is a Major Issue

- Integration of systems create a major problem with complexity
  - Within in a system, interactions grow as N squared or worse
  - Ability to understand and test becomes less certain
  - As more systems are added, the interfaces grow in a non-linear fashion
  - Many of the existing systems are old and not built for these interfaces
  - Conflicting or missing interface standards make it hard to define interface interactions
  - Hardware and software may be re-purposed and "heritage" compromised
  - Future systems will be integrated from multi-organizational, multi-national contributions, adding additional layers of complexity.
- Systems engineering must deal with this complexity
  - End-to-end systems engineering is needed, including "reengineering" of old systems
  - Robust M&S, verification and validation testing are A must

Page 8

From a presentation by Dr. Michael Ryschkewitsch, NASA Chief Engineer, at CSER Conference 15 April 2011

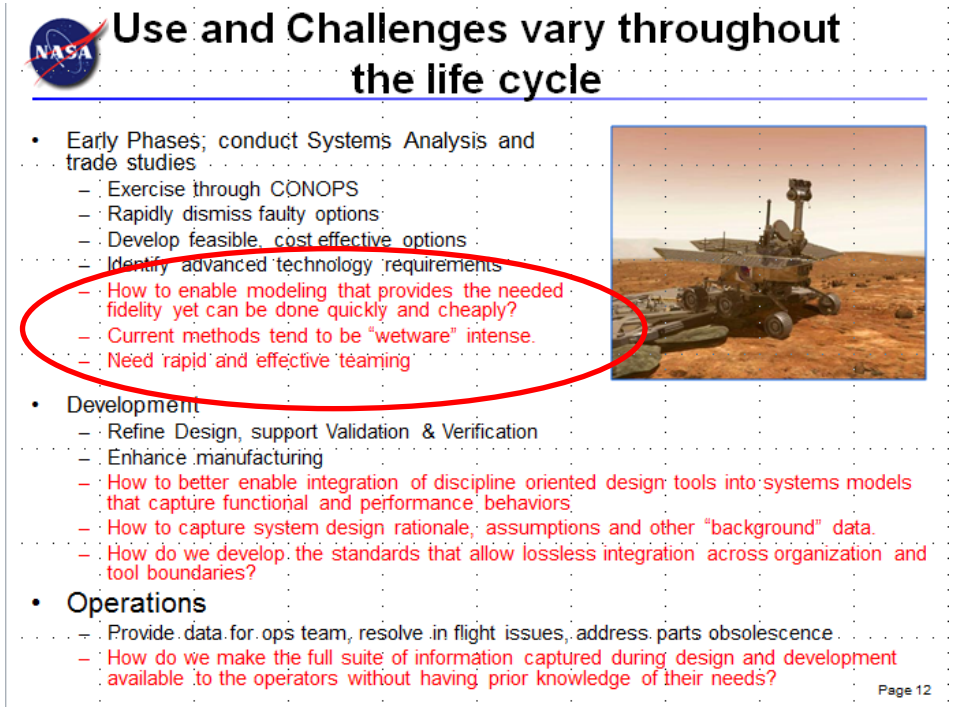
- With the growth of the Internet and daily changes in IT, systems have become more complex and change more rapid than ever before
- Systems engineering methods have not kept up with these changes
- SE has been relegated to the beginning of the lifecycle

How Does SE Typically Respond to Complexity?

- Focus on “architecture”
- More complex languages
- More complex procedures
- More layers of abstraction
  - “Systems of Systems”
  - “Family of Systems”
  - “Portfolio Management”
- Need more time and money!

SPEC INNOVATIONS

More Money Is a Problem



From a presentation by Dr. Michael Ryschkewitsch,
NASA Chief Engineer, at CSER Conference 15 April 2011

- Calls for doing more with less continue
- Need for lower labor and tool costs essential for acceptance of SE across the lifecycle

State of Current "Languages"

- In the past decade, the Unified Modeling Language (UML) and now the profile Systems Modeling Language (SySML) have dominated the discussion
- Why?
  - Perception that software is "the problem"
  - Hence need for an "object" approach
- SysML was designed to relate systems thinking to software development, thus improving communication between systems engineers (SE) and software developers

SPEC INNOVATIONS

Why Objects Are Not the Answer

- Although SysML *may* improve the communication of design between SEs and the software developers it does not communicate well to anyone else
  - No other discipline in the lifecycle uses object oriented design and analysis extensively
  - Users in particular have little interest/acceptance of this technique
  - Software developers who have adopted Agile programming techniques want functional requirements (and resent SEs trying to write software)

SPEC INNOVATIONS

So What Do We Do?

- Recognize that our primary job as SEs is to communicate between all stakeholders in the lifecycle
- Be prepared to translate between all the disciplines
- Reduce complexity in our language to facilitate communication

SPEC INNOVATIONS

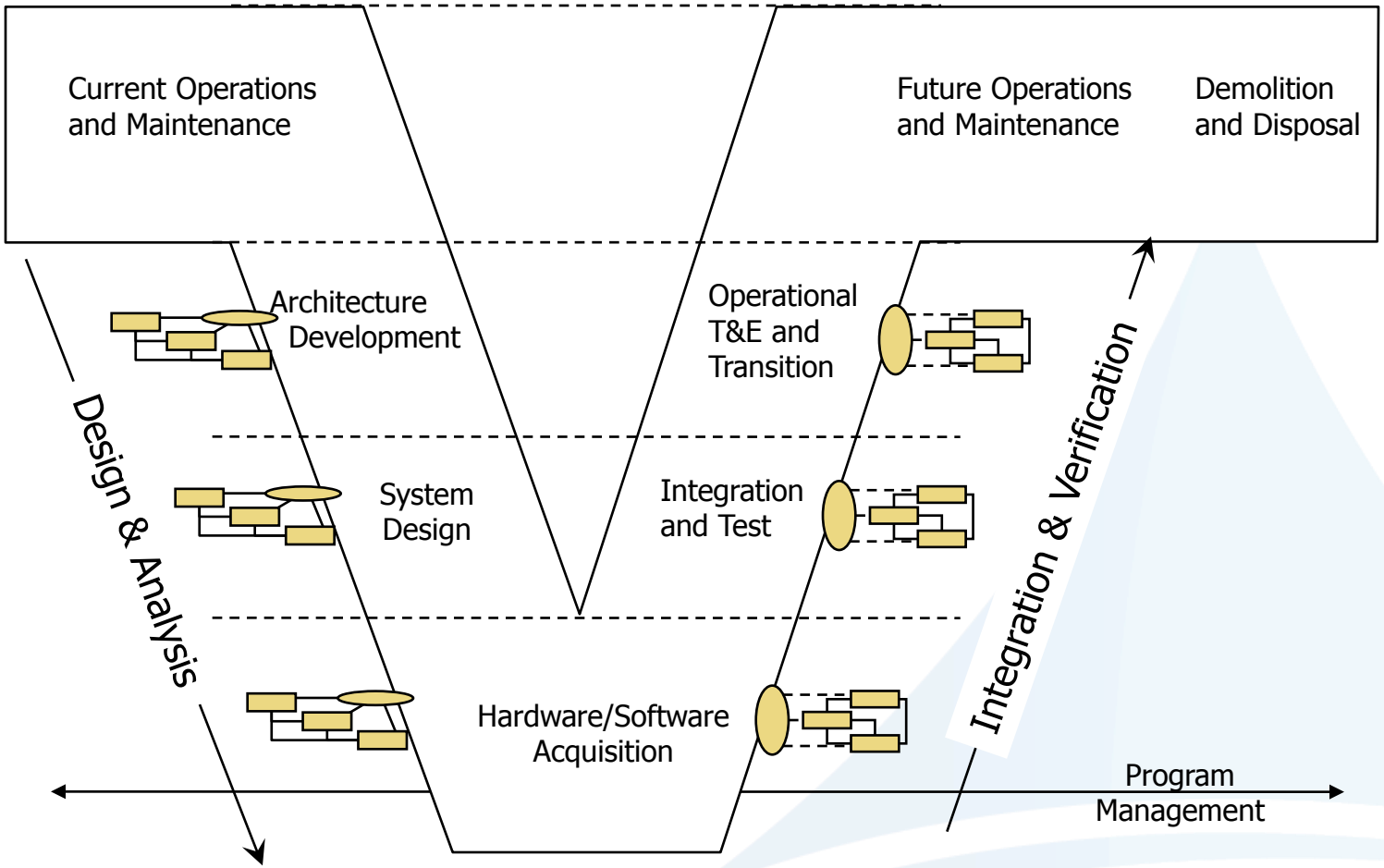
What We Did

- Researched the variety of languages (ontologies) in common use (DM2, SysML, BPMN, IDEF, SREM, etc.)
- Researched the variety of representations (FFBDs, N2, Behavior Diagrams, Class Diagrams, Electrical Engineering Diagrams, etc.)
- Took the best of each of these languages and representations and distilled them down to the essential elements, relationships, attributes, and diagrams

The result: Lifecycle Modeling Language

LIFECYCLE MODELING LANGUAGE (LML) OVERVIEW

The Lifecycle



Lifecycle Modeling Language (LML)

- LML combines the logical constructs with an ontology to capture information
  - SysML – mainly constructs – limited ontology
  - DoDAF Metamodel 2.0 (DM2) ontology only
- LML simplifies both the "constructs" and ontology to make them more complete, yet easier to use
- Goal: A language that works across the full lifecycle

SPEC INNOVATIONS

LML Ontology* Overview

*Ontology = Taxonomy + relationships among terms and concepts
** Taxonomy = Collection of standardized, defined terms or concepts

- Taxonomy**:
  - 12 primary element classes
  - Many types of each element class
    - Action (types = Function, Activity, Task, etc.)
- Relationships: almost all classes related to each other and themselves with consistent words
  - Asset performs Action/Action performed by Asset
  - Hierarchies: decomposed by/decomposes
  - Peer-to-Peer: related to/relates

LML Taxonomy Simplifies Classes

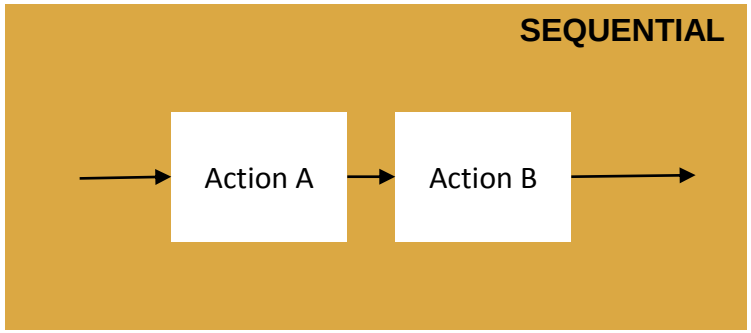
- Technical
  - Action
  - Artifact
  - Asset
  - Characteristic
  - Input/Output
  - Link
  - Statement
- Programmatic/Technical
  - Cost
  - Issue
  - Location
    - Physical, Orbital, Virtual
  - Risk
  - Time
    - Duration, Timeframe, Point-in-Time

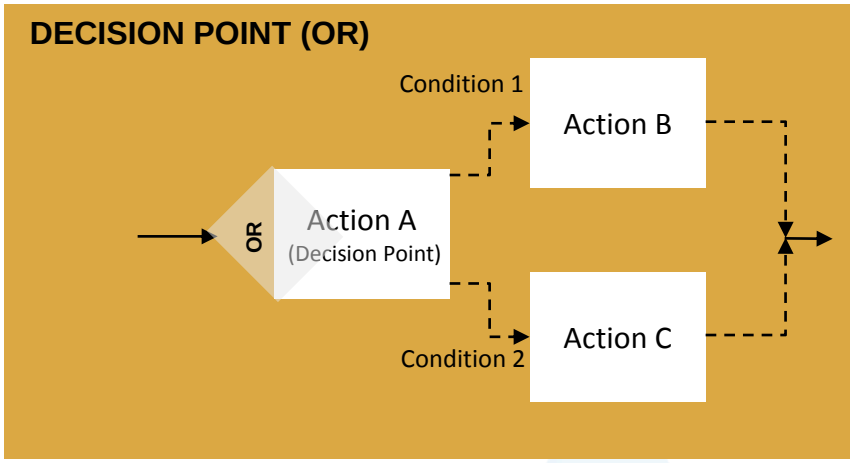
LML Relationships Provide Linkage Needed Between the Classes

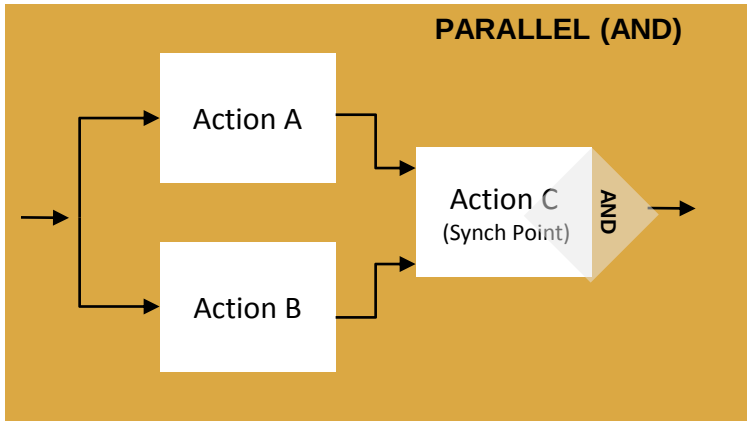
	ACTION	ARTIFACT	ASSET	CHARACTERISTIC	COST	INPUT/OUTPUT	ISSUE	LINK	LOCATION	RISK	STATEMENT	TIME	
ACTION	decomposed by related to	references	captures consumes preformed by produces	specified by	incurs	generates receives	causes resolves	-	located at	causes mitigates resolves	based on	takes occurs	ACTION
ARTIFACT	referenced by	decomposed by related to	referenced by	specified by referenced by	incurs referenced by	referenced by	causes referenced by	defines protocol for referenced by	located at	causes mitigates	based on referenced by	occurs	ARTIFACT
ASSET	captured by consumed by performs produced by	references	decomposed by orbited by related to	specified by	incurs	-	causes resolves responds to	connected by	located at	causes mitigates resolves	based on	occurs	ASSET
CHARACTERISTIC	specifies	references specifies	specifies	decomposed by related to	incurs specifies	specifies	causes resolves	specifies	located at	causes mitigates resolves	based on specifies	occurs	CHARACTERISTIC
COST	incurred by	incurred by references	incurred by	incurred by specified by	decomposed by related to	incurred by	causes incurred by resolves	incurred by	located at	causes incurred by resolves mitigates	cased on incurred by	occurs	COST
INPUT/OUTPUT	generated by received by	references	-	specified by	incurs	decomposed by related to	causes resolves	transferred by	located at	causes mitigates resolves	based on	occurs	INPUT/OUTPUT
ISSUE	caused by resolved by	caused by references resolved by	caused by resolved by responded by	caused by resolved by	caused by incurs resolved by	caused by resolved by	causes decomposed by related to resolved by	caused by resolved by	located at	caused by mitigates causes	caused by resolved by	date resolved by decision due occurs	ISSUE
LINK	-	defined protocol by references	connects to	specified by	incurs	transfers	causes resolves	decomposed by related to	located at	causes mitigates resolves	based on	delayed by occurs	LINK
LOCATION	locates	locates	locates	locates	locates	locates	locates	locates	decomposed by related to	locates mitigates	based on locates	occurs	LOCATION
RISK	caused by mitigated by resolved by	caused by mitigated by references resolved by	caused by mitigated by resolved by	caused by mitigated by resolved by	caused by incurs mitigated by resolved by	caused by mitigated by resolved by	caused by causes resolved by	caused by mitigated by resolved by	located at mitigated by	causes decomposed by related to resolved by	caused by mitigated by resolved by	occurs	RISK
STATEMENT	basis of	basis of references sourced by	basis of	basis of specified	basis of incurs	basis of	causes resolves	-	basis of located at	causes located at mitigates resolves	decomposed by related to	occurs	STATEMENT
TIME	taken by occurred by	occurred by	occurred by	occurred by	occurred by	occurred by	date resolves decided by occurred by	delays occurred by	occurred by	occurred by mitigates	occurred by	decomposed by related to	TIME
	ACTION	ARTIFACT	ASSET	CHARACTERISTIC	COST	INPUT/OUTPUT	ISSUE	LINK	LOCATION	RISK	STATEMENT	TIME	

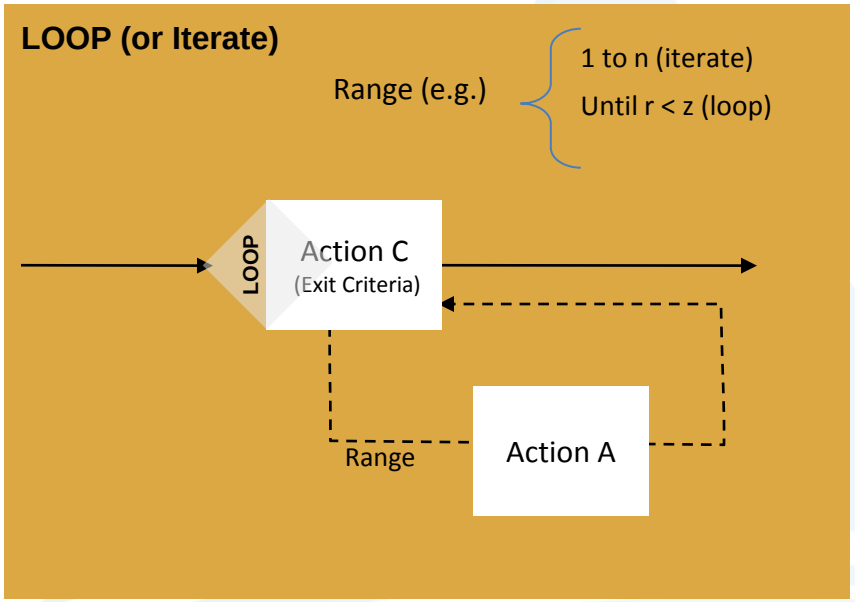
- decomposed by/decomposes
- orbited by/orbits
- related to/relates

LML Logic



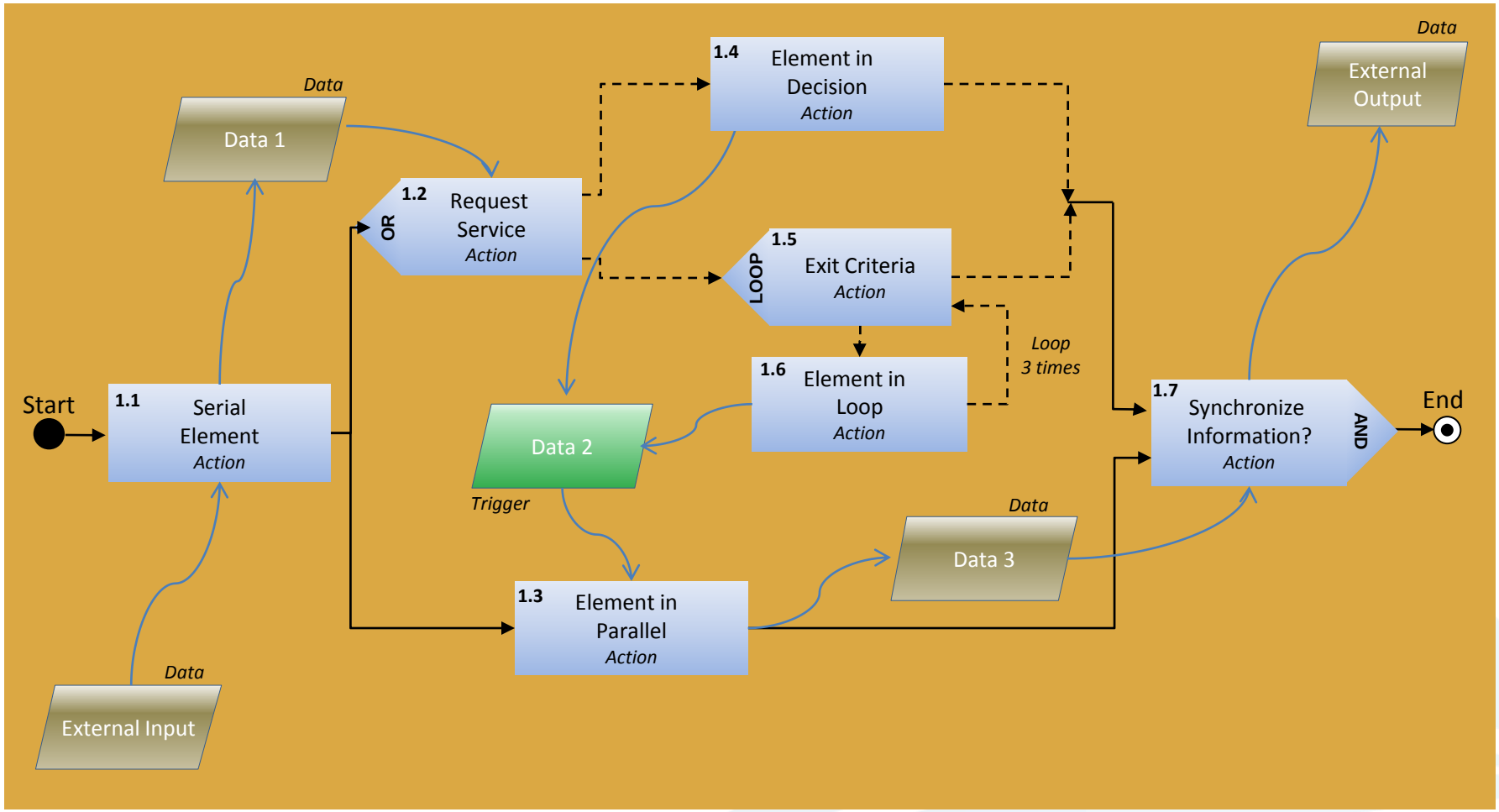




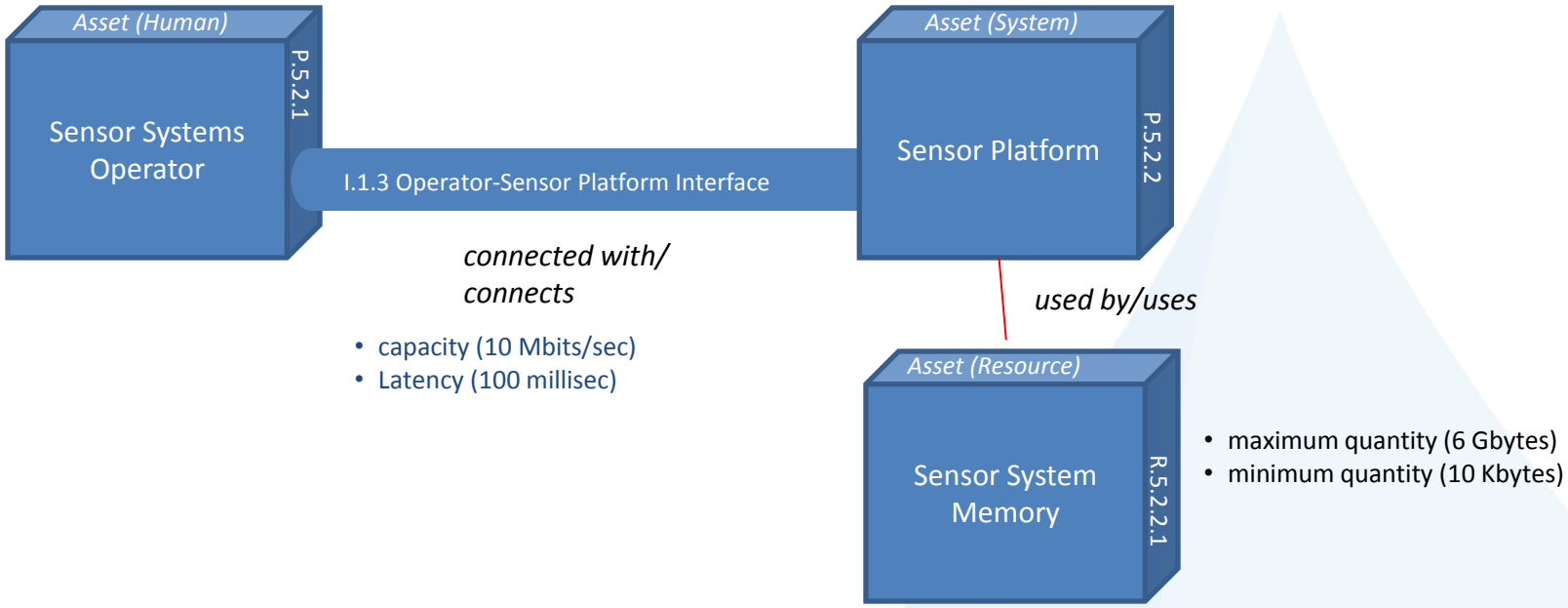


No constructs – only special types of Actions

LML Action Diagram Captures Behavior

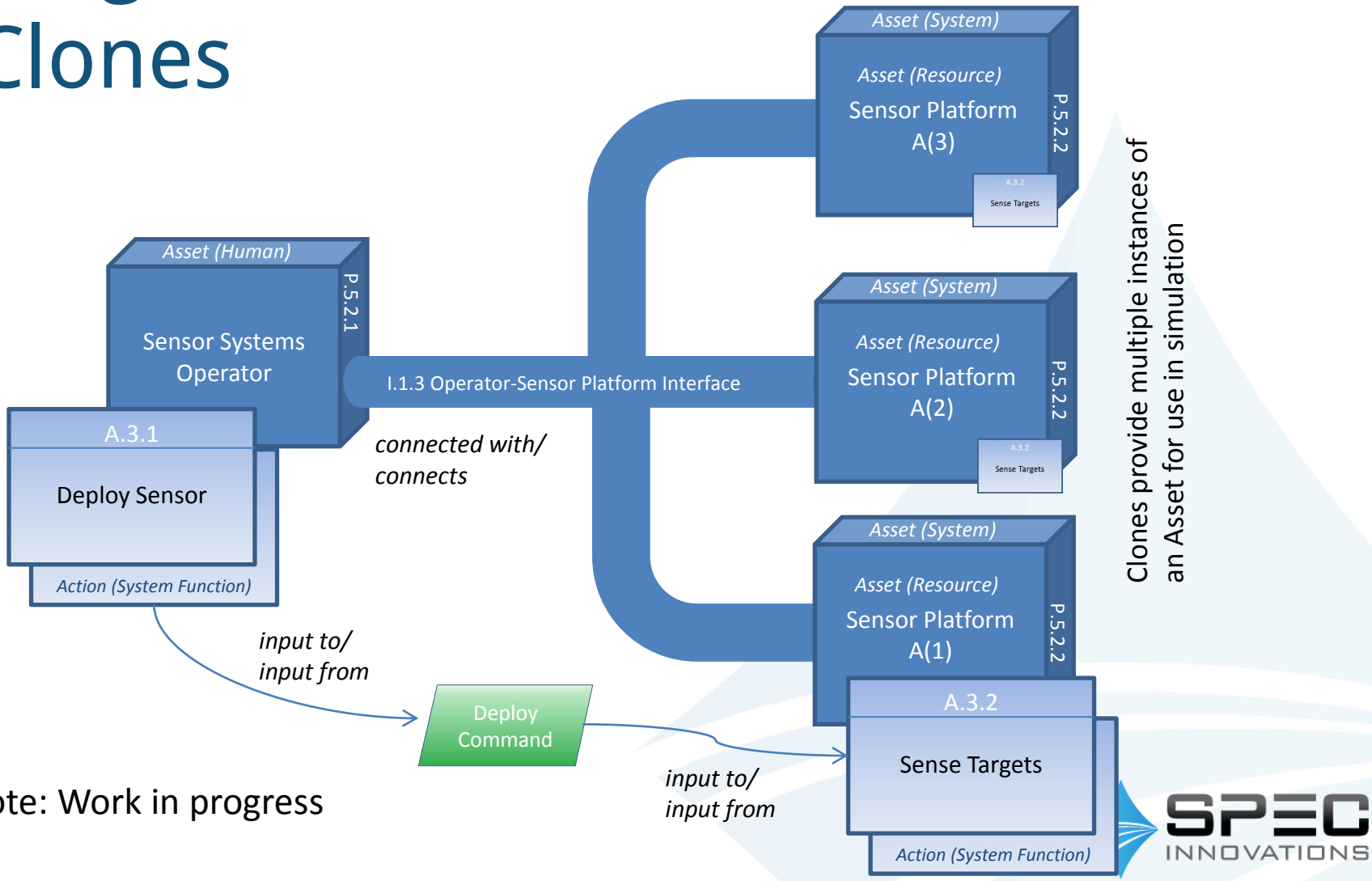


LML Physical Block Diagram*



*Note: Work in progress

LML Combined Physical Behavior Diagram* Enables Instances and Clones



*Note: Work in progress

LML Translation

- Two types of mapping for tailoring:
  - Map names of classes to enable other "schema" models to be used
  - Map symbols used (e.g., change from LML Logic to Electrical Engineering symbols)

LML Class	DM2	SysML	…
Action	Activity	Activity	
Asset	Performer	Actor	

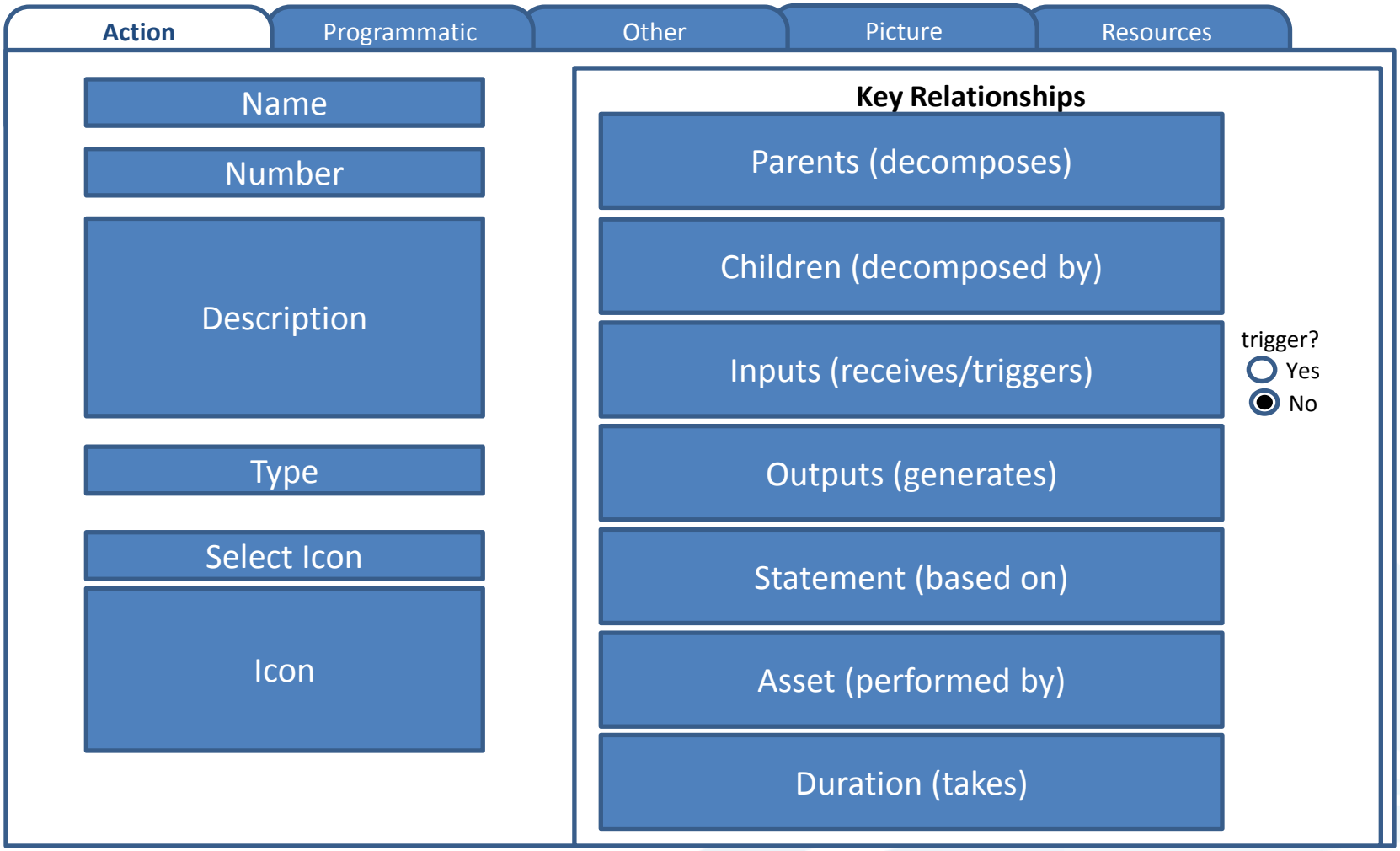
LML Symbol	Electrical Engineering	BPMN	…
			

Other Diagrams

- Physical Block Diagrams
  - With option for icon substitution
- Interface Diagrams
  - N2 (Assets or Actions)
- Hierarchy Diagrams
  - Automatically color coded by class
- Time Diagrams
  - Gantt Charts
  - Timeline Diagram
- Location Diagrams
  - Maps for Earth
  - Orbital charts
- Risk Chart
  - Standard risk/opportunity chart
- Organization Charts
  - Showing lines of communication, as well as lines of authority
- Pie/Bar/Line Charts
  - For cost and performance
- Combined Physical and Functional Diagram (?)

SPEC INNOVATIONS

Reducing Complexity by Association



Reducing Complexity by Association

Action | **Programmatic** | Other | Picture | Resources

Name

Number

incurs: Cost

List of Costs or None

located at: Location

Map It

List of Locations or None

occurs: Time (TimeFrame or PointInTime)

List of Times or None

Relations for Risk Analysis

causes: Issue or Risk

List of Issues or Risks or None

resolves: Issue or Risk

List of Issues or Risks or None

mitigates: Risk

List of Risks or None

SPEC INNOVATIONS

Reducing Complexity by Association

Action | Programmatic | **Other** | Picture | Resources

Name

Number

references: Artifact

List of Artifacts or None

specified by: Characteristic

List of Characteristics or None

Peer-to-Peer Relationships

related to: Action

List of Actions or None

Context

relates: Action

List of Actions or None

Context

SPEC INNOVATIONS

Reducing Complexity by Association

Action | Programmatic | Other | Picture | **Resources**

Name

Number

Relations for Resource Modeling

captures: Resource (Asset subclass)

List of Resources or None

consumed by: Resource (Asset subclass)

List of Resources or None

produces: Resource (Asset subclass)

List of Resources or None

Amount

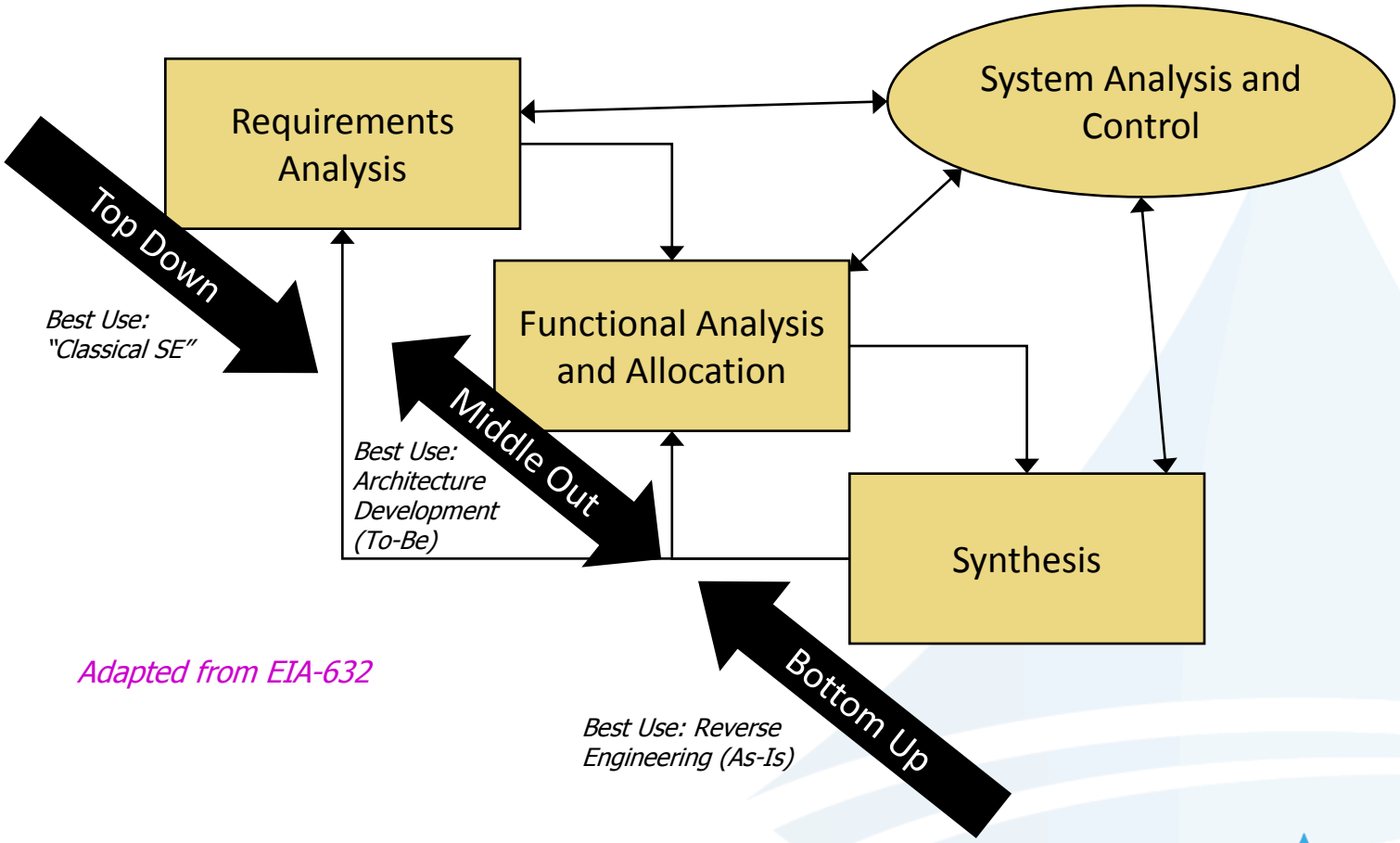
LML Summary

- LML contains the basic technical and programmatic classes needed for the lifecycle
- LML defines the Action Diagram to enable better definition of logic as functional requirements
- LML uses Physical Diagram to provide for abstraction, instances, and clones, thus simplifying physical models
- LML provides the “80% solution”
  - It can be extended to meet specific needs (e.g. adding Question and Answer classes for a survey tool that feeds information into the modeling)

SPEC INNOVATIONS

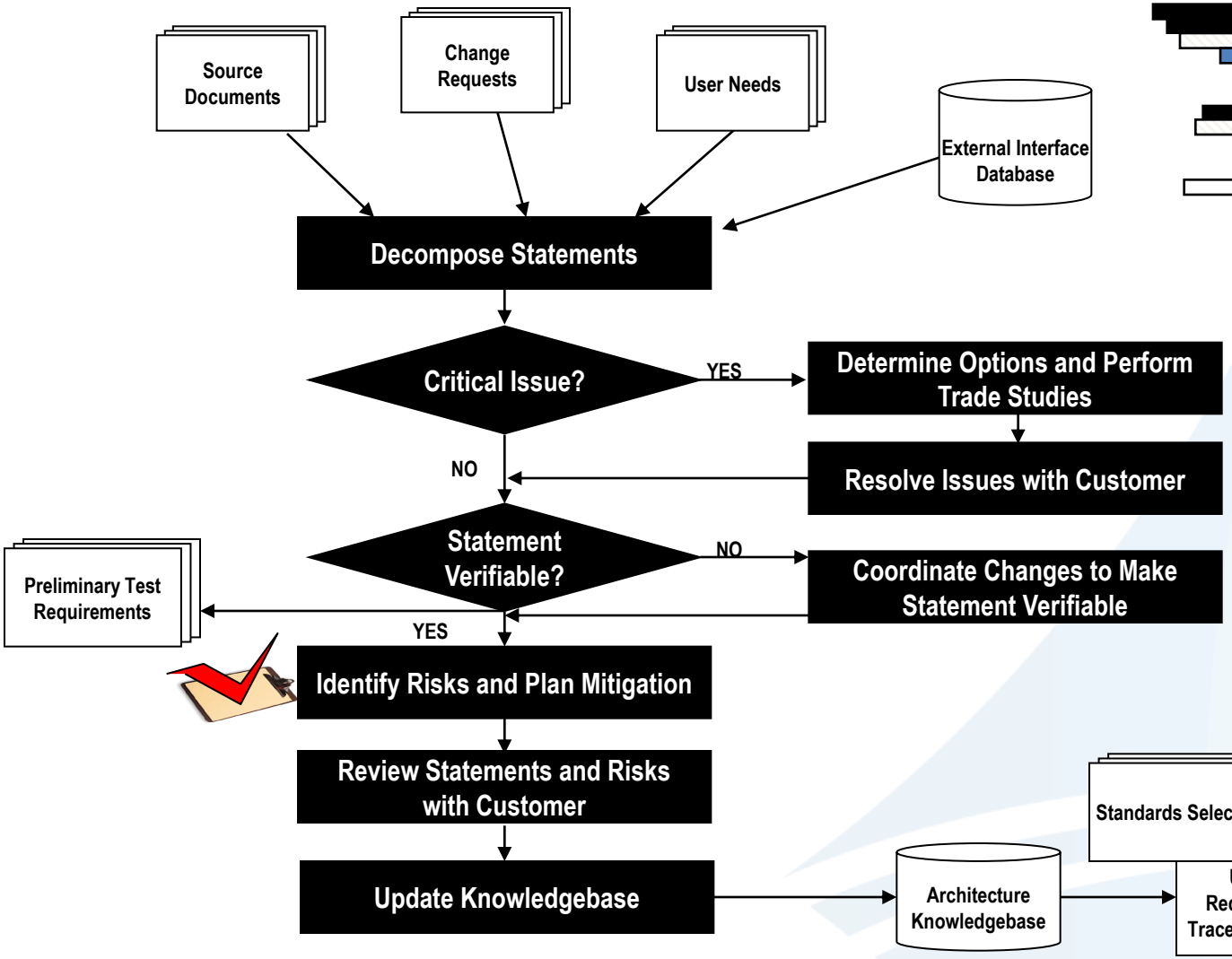
SUGGESTED LML PROCESSES

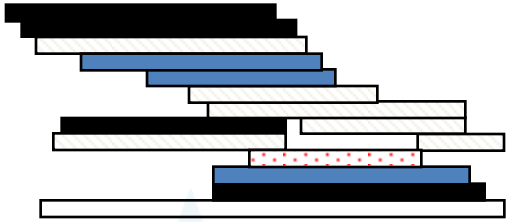
Systems Engineering During Design Phase



Adapted from EIA-632

Requirements Analysis



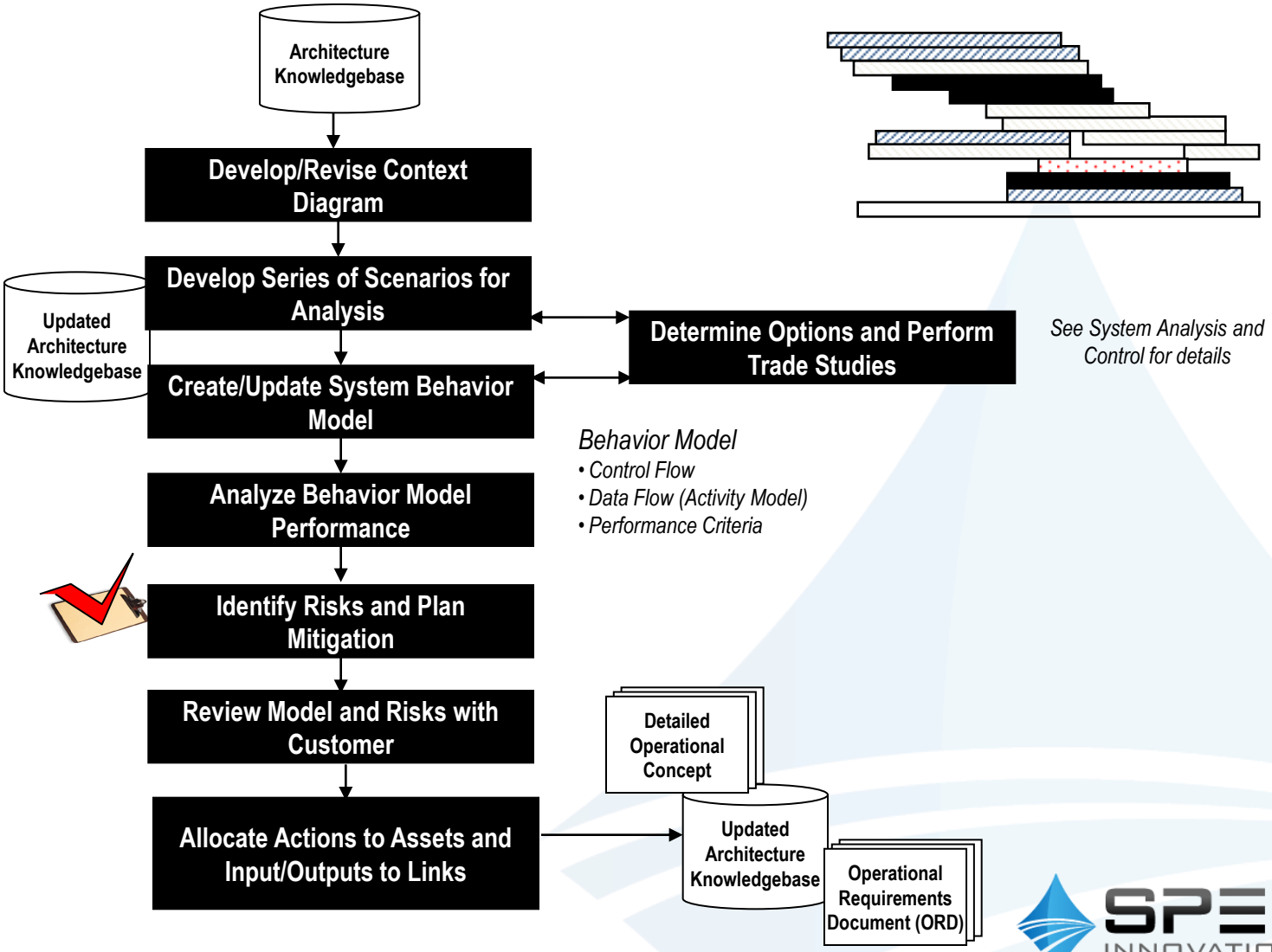


See System Analysis and Control for details

Key LML Elements for Requirements Analysis Process

- Artifact
  - Types: Briefing, Change Notice, Change Request, Concept of Operations, Directive, Doctrine, Document, Guidance, Instruction, Interface Control Document, *Policy, Procedure, Requirements Document*, Trade Study, …
- Statement
  - Types: Acronym, Composite, Constraint, Definition, Directive, Doctrine, Functional, Goal, Objective, Performance, Plan, Policy, Question, *Requirement*, Rule, Scope, Standard, Statement, Vision
- Risk
- Issue

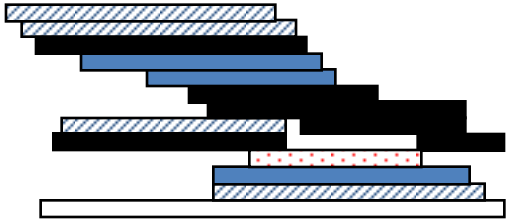
Functional Analysis and Allocation

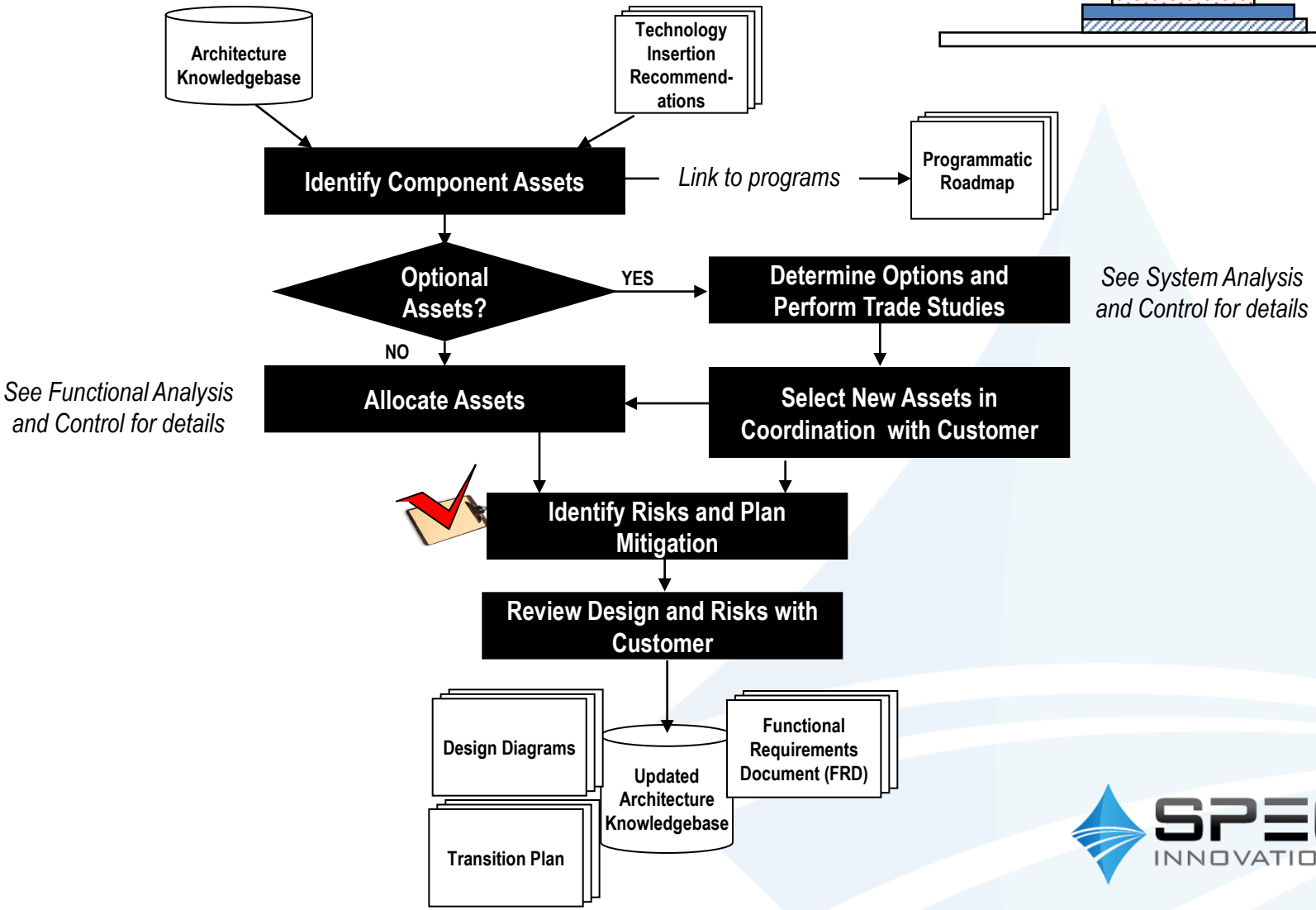


Key LML Elements for Functional Analysis Process

- Action
  - Types: Activity, Capability, Function, Mission, Operational Activity, Program, Service Orchestration, Simulation Workflow, Subprocess, System Function, Task, Training, Work Process, Workflow
- Input/Output
  - Types: Analog, Digital, Event, Mixed, Physical, Product, Verbal
- Time: Duration

Synthesis

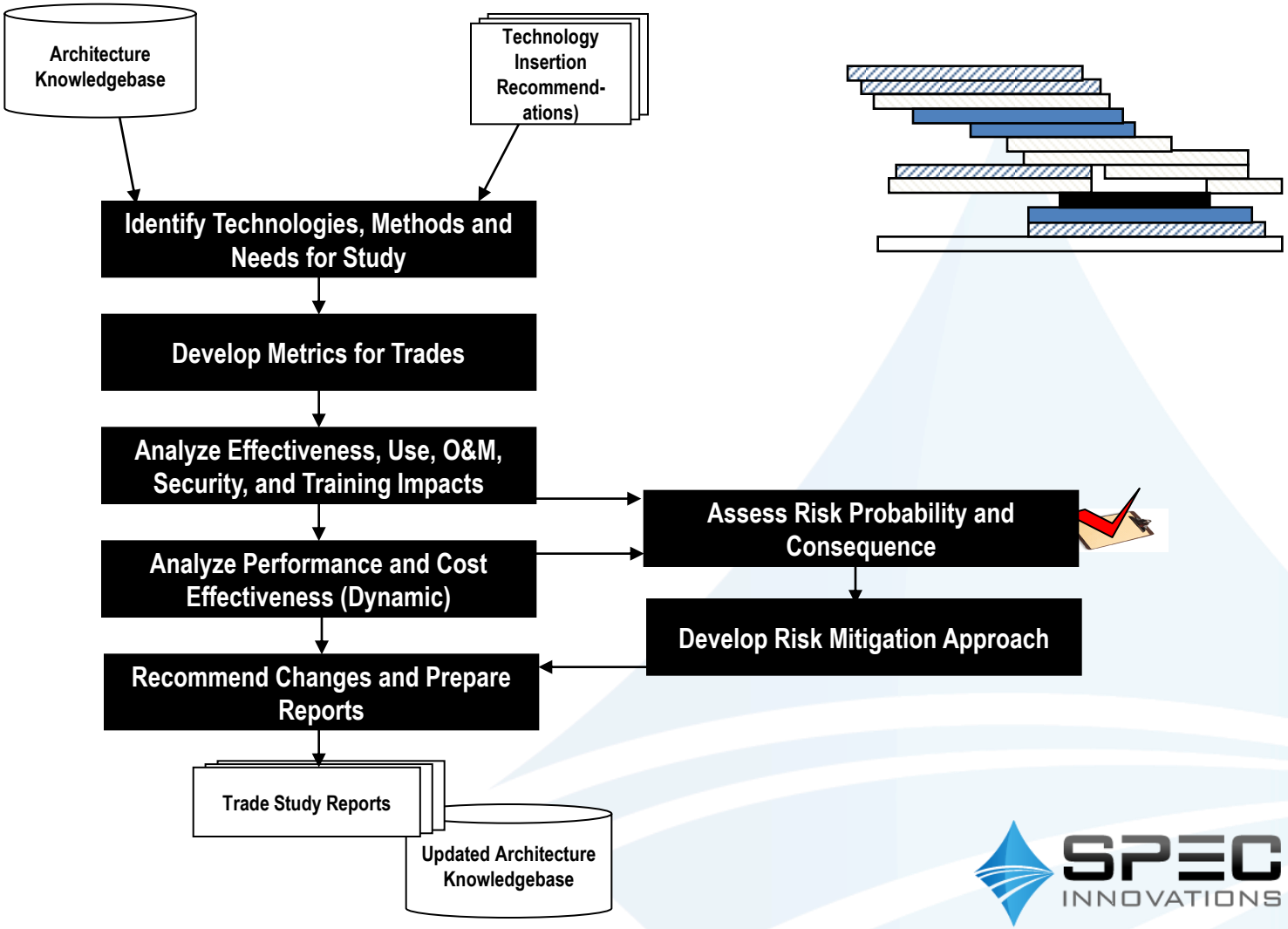




Key LML Elements for Synthesis Process

- Asset
  - Types: Architecture, Assembly, Component, Context, CSC, CSCI, CSU, Element, Environment, External System, Facility, Hardware, Human, HW Element, HWCI, Infrastructure, LRU, Materiel, Operational Element, Organization, Part, Performer, Personnel, Segment, Service, Software, Subassembly, Subsystem, System, System Instantiation, Test Equipment, Test Software, Unit
- Link
  - Types: Cable, Downlink, Human-in-the-Loop, Human Machine Interface, Interface, Landline, Link, Needline, Network, Pipe, Relationship, RF - Satcom, RF - Terrestrial, Roadway, Service Interface, Uplink, Wireless
- Location
- Cost
- Characteristic

Systems Analysis and Control



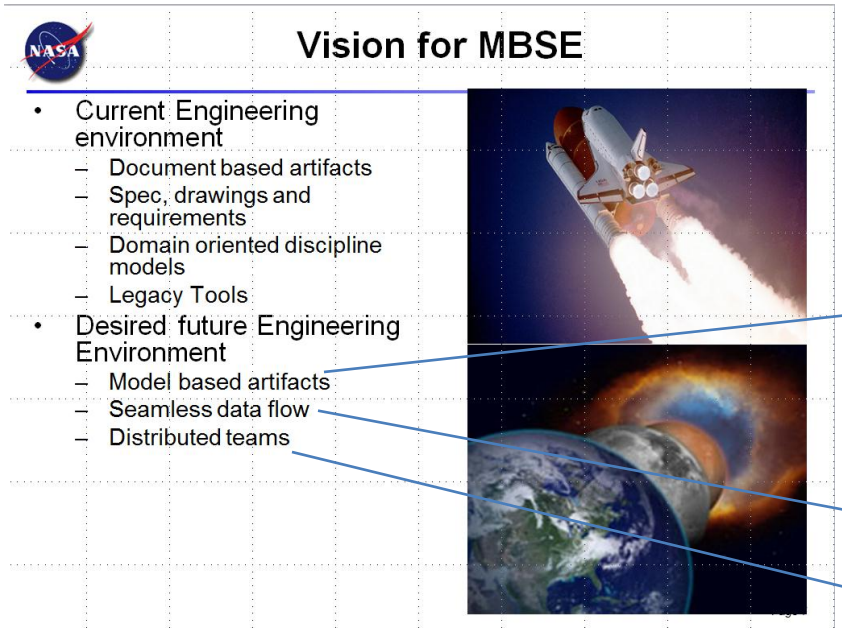
Key LML Elements for Systems Analysis and Control Process

- Characteristic
  - Types: Condition, Data Element, Environmental, Functional, Performance, Physical, Scenario, Security, Verification Category
- Time: subclass Timeframe
- Risk

Break

LML TOOL NEEDS

Tool for Vision for MBSE



From a presentation by Dr. Michael Ryschkewitsch, NASA Chief Engineer, at CSER Conference 15 April 2011

- Tool must be designed as a MBSE tool using Lifecycle Modeling Language (LML)
  - All artifacts must be produced from the tool using standard reports
- Tool must enable both seamless data flow and distributive, collaborative teams working on the same model
- Implies need for ability to scale "infinitely" on demand

Tool for Complexity

Complexity is a Major Issue

- Integration of systems create a major problem with complexity
  - Within in a system, interactions grow as N squared or worse
  - Ability to understand and test becomes less certain
  - As more systems are added, the interfaces grow in a non-linear fashion
  - Many of the existing systems are old and not built for these interfaces
  - Conflicting or missing interface standards make it hard to define interface interactions
  - Hardware and software may be re-purposed and "heritage" compromised
  - Future systems will be integrated from multi-organizational, multi-national contributions, adding additional layers of complexity.
- Systems engineering must deal with this complexity
  - End-to-end systems engineering is needed, including "reengineering" of old systems
  - Robust M&S, verification and validation testing are A must

From a presentation by Dr. Michael Ryschkewitsch, NASA Chief Engineer, at CSER Conference 15 April 2011

Page 8

- Tool must also reduce complexity by use of special input screens, which enhance configuration management as well

- Tool must be designed for use throughout the lifecycle, thus enabling end-to-end systems engineering
  - Legacy systems are a key part of any modeling environment – LML has specific information classes for capturing designs at different times in one knowledgebase
- Tool must embed simulation, which enables real time simulations as well as the ability to scale "infinitely" to hundreds of server instances

Tool for Multi-Decadal/Generation

Supporting multi-decadal, multi – generation activities is a major challenge

- The International Space Station will have a lifespan of at least 20 years with evolving uses and constant changes
- Systems analyses show that as we explore beyond low earth orbit, launch costs will remain a driver and thus put a huge value in re-using systems already moved up the gravity well
  - We will need to track systems health and status against predictions and threshold
  - Systems will be modified, updated and re-purposed multiple times
  - Operating environments and conditions may change from those used for design
  - Likely to want to use systems well beyond initial life objectives

Page 9

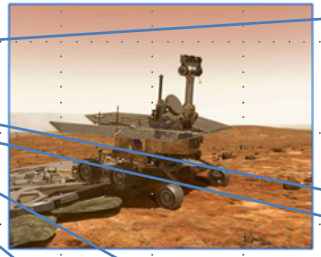
From a presentation by Dr. Michael Ryschkewitsch, NASA Chief Engineer, at CSER Conference 15 April 2011

- Tool must enable tracking of performance over time as Characteristics of the Assets
- Environments and conditions can also be captured as evolved over time and location (including orbital locations)
- Tool must evolve of the lifecycle of systems, but the data must be captured and reused throughout the lifecycle

Tool for Uses and Challenges

Use and Challenges vary throughout the life cycle

- Early Phases; conduct Systems Analysis and trade studies
  - Exercise through CONOPS
  - Rapidly dismiss faulty options
  - Develop feasible, cost effective options
  - Identify advanced technology requirements
  - How to enable modeling that provides the needed fidelity yet can be done quickly and cheaply?
  - Current methods tend to be "wetware" intense.
  - Need rapid and effective teaming



- Development
  - Refine Design, support Validation & Verification
  - Enhance manufacturing
  - How to better enable integration of discipline oriented design tools into systems models that capture functional and performance behaviors
  - How to capture system design rationale, assumptions and other "background" data.
  - How do we develop the standards that allow lossless integration across organization and tool boundaries?
- Operations
  - Provide data for ops team, resolve in flight issues, address parts obsolescence
  - How do we make the full suite of information captured during design and development available to the operators without having prior knowledge of their needs?

Page 12

From a presentation by Dr. Michael Ryschkewitsch, NASA Chief Engineer, at CSER Conference 15 April 2011

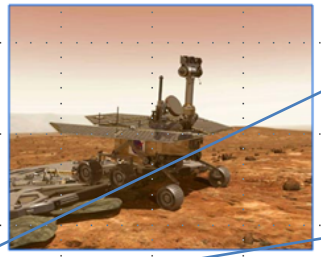
- Tool should automate what can be automated to reduce "wetware" requirements and enhance teaming

- Tool needs to create a CONOPS report based on scenario modeling
- Capture Issues and Decisions, as well as Cost for cost, schedule and performance tradeoffs
- TRL levels must be easily captured in the tool
- Software as a service provides inexpensive tool, while scalability enables fidelity level required; LML method enables rapid SE design and analysis

Tool for Uses and Challenges *(continued)*

Use and Challenges vary throughout the life cycle

- Early Phases; conduct Systems Analysis and trade studies
  - Exercise through CONOPS
  - Rapidly dismiss faulty options
  - Develop feasible, cost effective options
  - Identify advanced technology requirements
  - How to enable modeling that provides the needed fidelity yet can be done quickly and cheaply?
  - Current methods tend to be "wetware" intense.
  - Need rapid and effective teaming



- Development
  - Refine Design, support Validation & Verification
  - Enhance manufacturing
  - How to better enable integration of discipline oriented design tools into systems models that capture functional and performance behaviors
  - How to capture system design rationale, assumptions and other "background" data.
  - How do we develop the standards that allow lossless integration across organization and tool boundaries?
- Operations
  - Provide data for ops team, resolve in flight issues, address parts obsolescence
  - How do we make the full suite of information captured during design and development available to the operators without having prior knowledge of their needs?

Page 12

From a presentation by Dr. Michael Ryschkewitsch, NASA Chief Engineer, at CSER Conference 15 April 2011

- Tool should enable design down to detailed level (if desired), as well as V&V support
- Manufacturing enhanced by linking design to CAD/CAM systems (export design information as required)
- Use of XML files to import/export data from other design tools
- Design rationale, assumption and other programmatic information captured as part of Issues, Risk and other program-related elements of LML
- Standards should be captured and developed using the tool; enforcement by tailoring "personal trainer" version

- Operations data and obsolescence part of LML elements (Assets, Characteristics and Time)
- Use of a scenario-based approach enables better information capture from operators

Tool for Managing Complexity

Managing Complexity

- Current approaches tend to reflect human thinking that complex systems are simple extrapolations of simpler systems yet we know this is not true – N2, difficulty to test completely, etc
  - We currently divide and conquer, attempting to limit interactions via simple interfaces
  - What are the optimal strategies for limiting the undesirable or unrecognized effects
  - Are there better test strategies (probabilistic methods???)
  - Are there modeling and simulation strategies that allow us to truly model probabilistic behaviors in all their details and exercise them against operational scenarios and yet do so quickly and efficiently so that they become design tools
  - move away "peeling the onion"

From a presentation by Dr. Michael Ryschkewitsch, NASA Chief Engineer, at CSER Conference 15 April 2011

Page 15

- Tool in conjunction with a process should capture a reasonable number of scenarios for modeling and simulation, using a test matrix approach (move away from "peeling the onion")
- Use "test matrix" approach to provide breadth of problem space, including failure modes, to capture the full functionality required
- Test must become even more dependent on simulation

Tool for Managing Complexity

Collaborative Systems Engineering

- Current systems processes inherently reflect the need to limit complexity and detail in models, simulation and analyses
  - Driven by time to assembly, compute power, lack of interface standards and techniques
  - Forced to extract behaviors by parameterization or simplifying models
  - Will this always be the case?
    - Just like CAD systems have libraries of detailed parts, can we have methodologies that allow integration of complex sub-models?
    - Can we have methodologies that allow systems models to evolve with initial simple parametric sub-models and evolve to design based sub-models?
    - Can we better enable rapid and diverse teaming unimpeded by technological barriers?

From a presentation by Dr. Michael Ryschkewitsch, NASA Chief Engineer, at CSER Conference 15 April 2011

Page 16

- Tool should enable sharing of models, thus creating a "library" of component models
- Contributions should be made from academia (tool should be free access to academia) to this library
- Government sponsored research can also be made available to the NASA family via website

SPEC INNOVATIONS

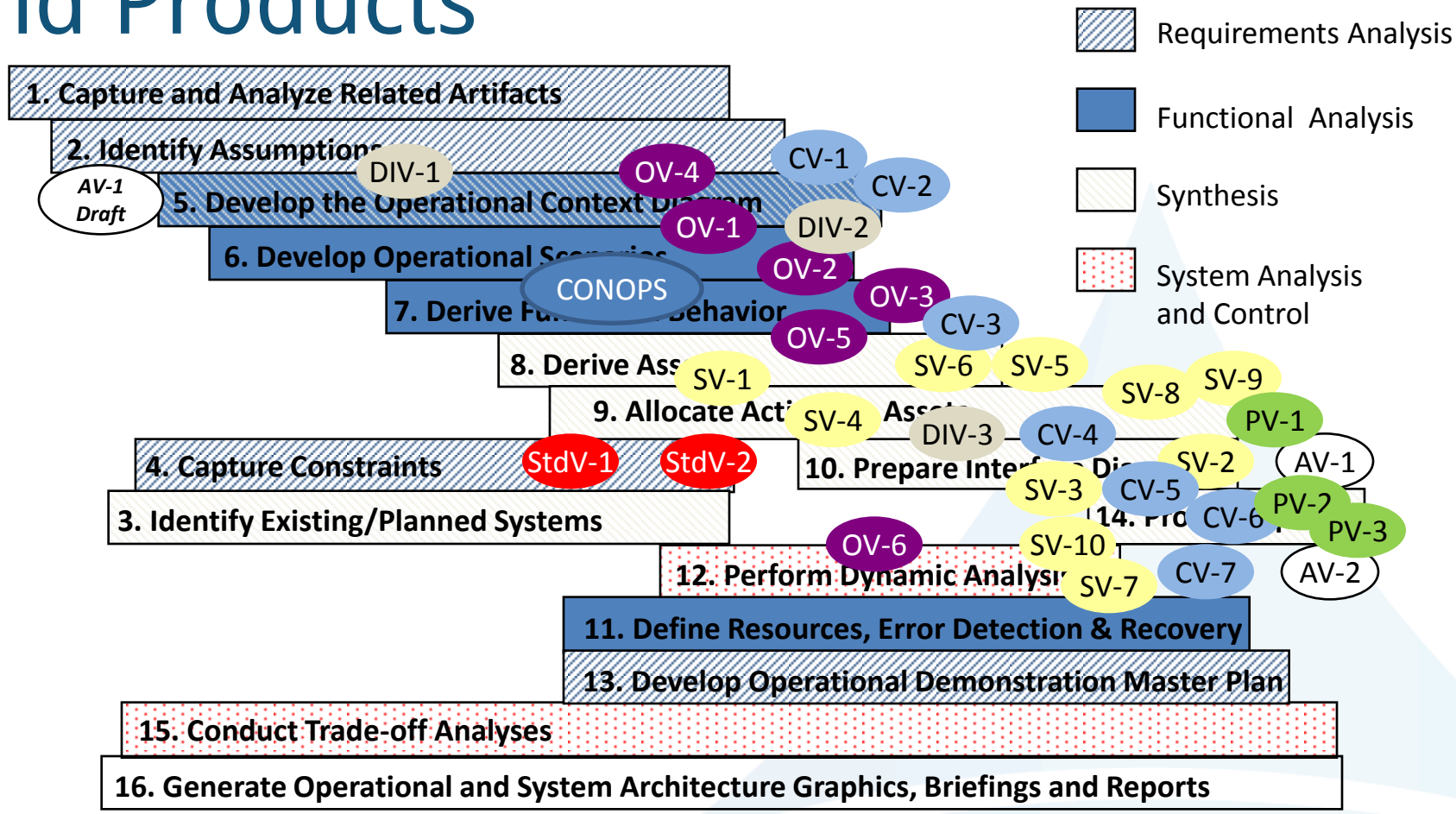
Key Tool Features Needed

- Scalable Modeling
- Scalable Discrete Event Simulation
- Functional Behavior Diagrams
- Physical Block Diagrams w/ Icons
- Element Extractor/Automated Parsing
- Splitting and Merging Requirements
- Merge databases from other tools
- Access to the web
- Automated report writing
- Enterprise data search
- Customization
- Collaboration
- Configuration Management

SPEC INNOVATIONS

USE OF LML FOR ARCHITECTURE DEVELOPMENT

Architecture Development Process and Products



This implementation of the middle-out approach has been proven on a variety of architecture projects

Key Architecture Products

- DoDAF Diagrams
- Concept of Operations (CONOPS)
- Functional Specifications of Hardware and Software
- Early Design Validation Through Modeling and Simulation
- Test and Evaluation Plans (for T&E)
- Processes and Procedures (for Operations and Support, as well as inputs to training plans)

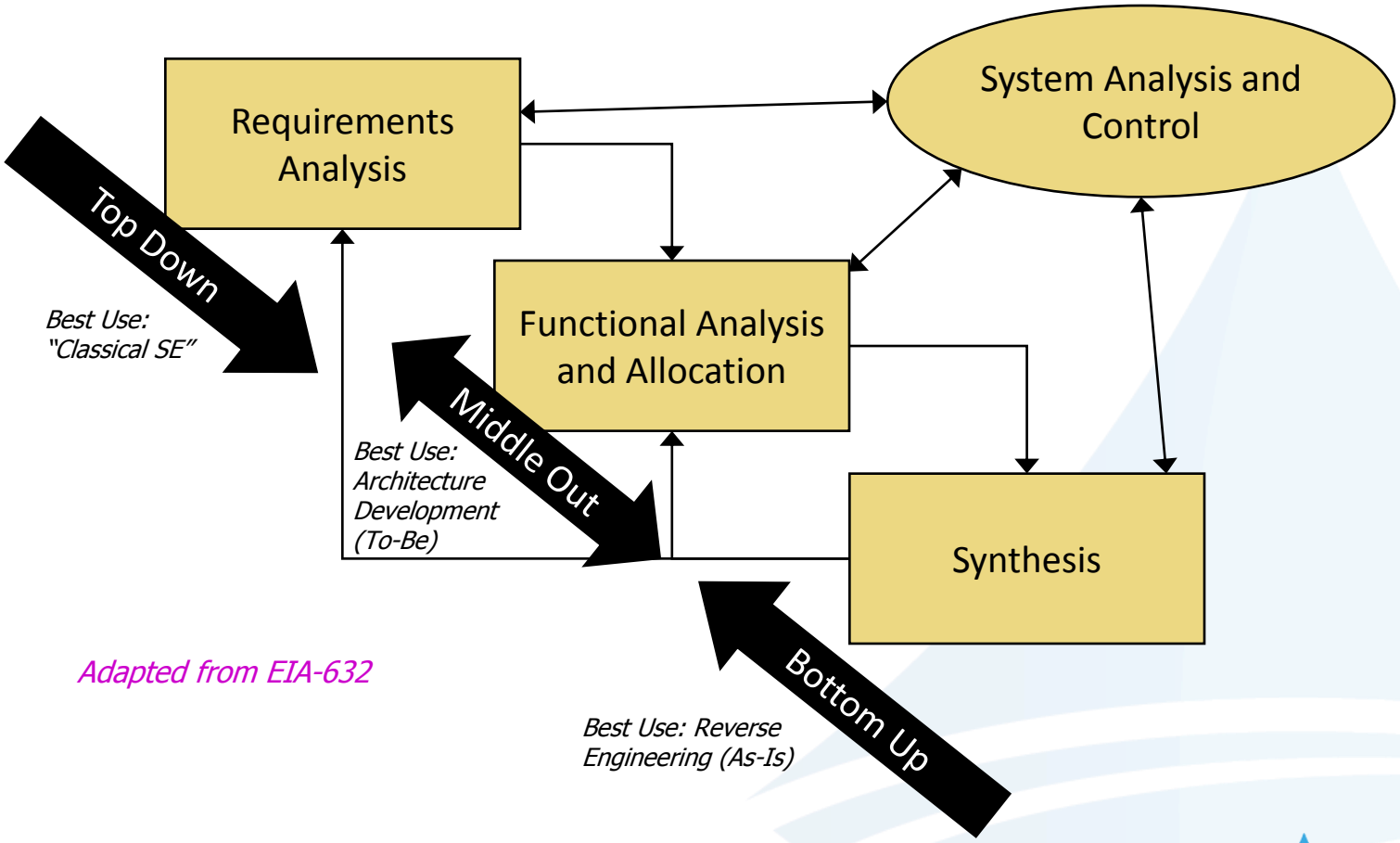
Key LML Elements and Use - Architecture

Elements	Use
Action	Scenario analysis
Asset	High level for gap analysis
Cost	High level cost estimates - lifecycle
Characteristic	Key metrics for performance/acceptance
Input/Output	Key data elements
Link	Critical interfaces (between systems of systems)
Risk	Key schedule, cost, performance risks
Time - Timeframe	Broad roadmap

SPEC INNOVATIONS

USE OF LML FOR SYSTEM DESIGN

System Design Process



Adapted from EIA-632

SPEC INNOVATIONS

Key Systems Design Products

- System/Segment Specifications
- Interface Control Documents
- Detailed Integration and Test Plans

Key LML Elements and Use – System Design

Elements	Use
Action	Functional analysis
Asset	Design packaging
Cost	Detailed cost estimates - lifecycle
Characteristic	Specific metrics for performance/acceptance
Input/Output	Data elements
Link	Interfaces (between components)
Location	Identifies differences due to location
Risk	Schedule, cost, performance risks
Statement	Key requirements
Time - Duration	Key time dependences

SPEC INNOVATIONS

USE OF LML FOR SOFTWARE DEVELOPMENT

Support Object Analysis (if desired)

- Assets = Objects
- Links = Relationships/Interfaces
- Characteristics = Attributes
- Actions = Methods
  - Can include key logic algorithms

Otherwise – Support Functional Requirements Specification

- Actions
- Input/Output
- Packaging using Assets and Links

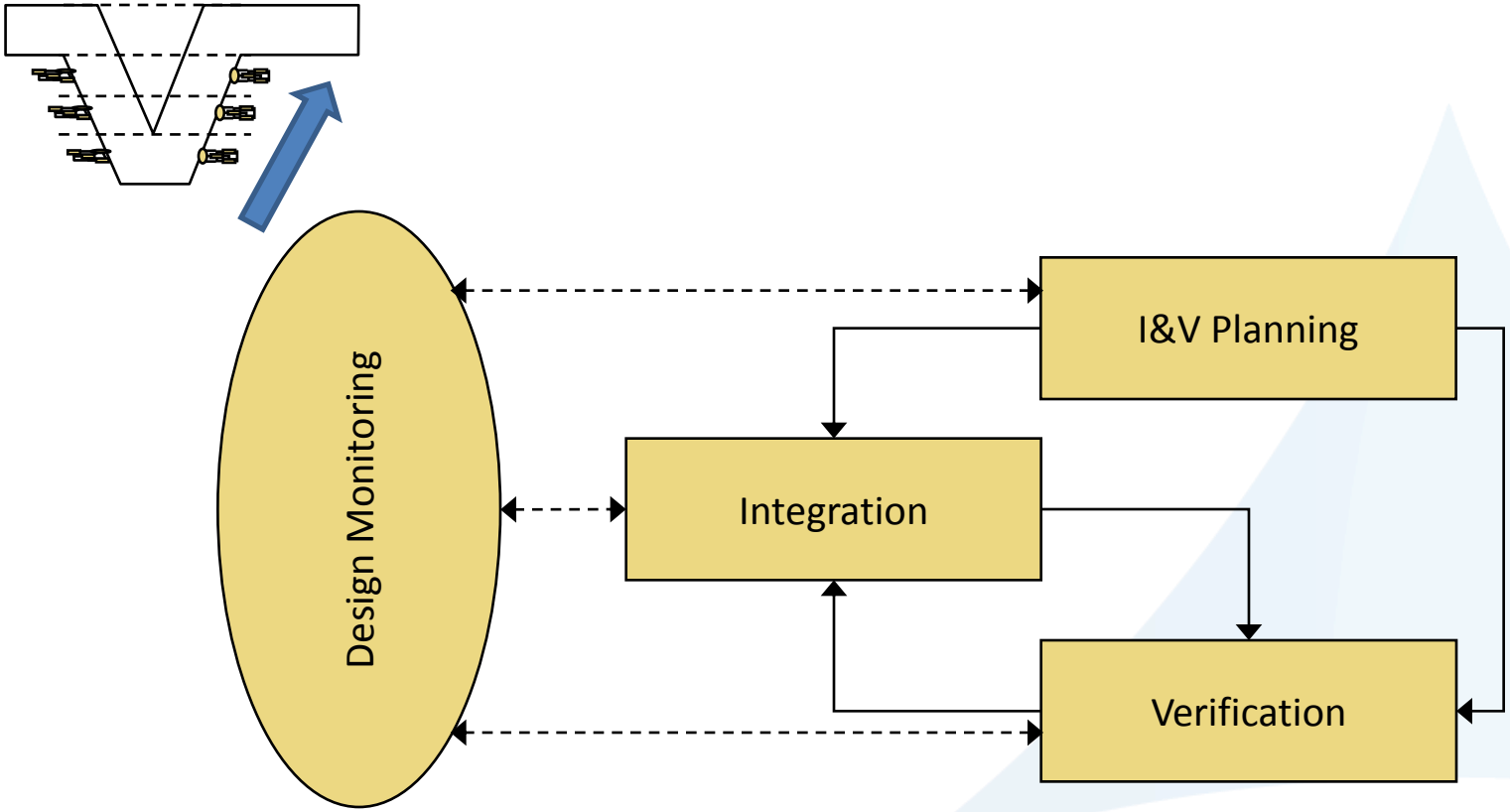
Key LML Elements and Use – Software Development

Elements	Use
Action	Functional requirements for methods
Asset	Classes and/or packages
Characteristic	Specific metrics for performance/acceptance
Input/Output	Data elements
Link	Interfaces (Inheritance between Classes)

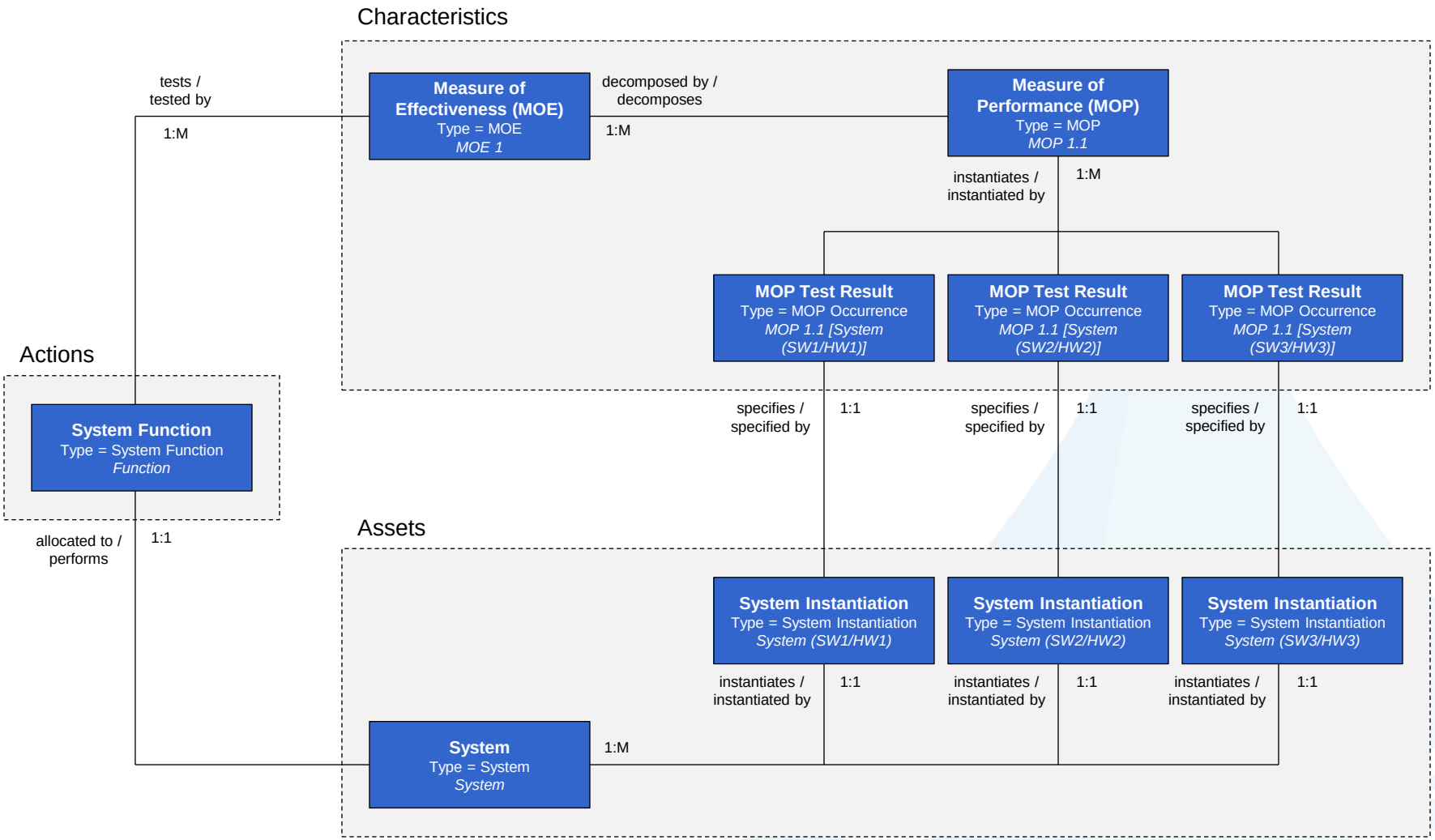
SPEC INNOVATIONS

USE OF LML IN TEST AND EVALUATION

Coming Up the Vee



Measure of Performance (MOP) View



Key LML Elements and Use – Test and Evaluation

Elements	Use
Action	Test processes and procedures/ functional requirements for testing
Asset	Test configurations
Cost	Test costs
Characteristic (Technical Measurement subclass)	Specific metrics for performance/acceptance (MOEs, MOPs)
Input/Output	Data elements for measurement
Link	Interfaces (between components)
Statement	Key requirements for testing

USE OF LML IN OPERATIONS AND SUPPORT

LML Support Operations and Support Analyses

- Process Modeling
- Simulation of Operations
- Training Processes and Procedures
- Operations Manuals
- Logistics Analysis

Key LML Elements and Use – Operations and Support

Elements	Use
Action	O&S processes and procedures
Asset	Configurations
Cost	O&S costs
Characteristic (Technical Measurement subclass)	Specific metrics for performance/acceptance (MOEs, MOPs)
Input/Output	Data elements for decision making
Location	Differences due to location

SPEC INNOVATIONS

SUMMARY

People Considerations for LML

- Large teams make organization and focus on a vision very difficult
  - Need better ways to collaborate
- You need people with a wide variety of skills and personalities
  - Someone with vision
  - Someone who can perform the detailed system engineering
  - Someone who understands the domain
  - Someone familiar with the technique and tools
  - Someone who understands the process
- They need to be trained as a team – including the government personnel

SPEC INNOVATIONS

LML Bottom-Line

- LML provides a simple, complete language for all stakeholders, not just software developers
  - SysML/UML focus on software developers only
- Use of Actions instead of constructs to capture command and control functions explicitly
- Translation from LML to other languages now feasible
- Support for entire lifecycle
- Reduces complexity

SPEC INNOVATIONS