Introduction to Model-Based Systems Engineering (MBSE) and Innoslate®

Steven H. Dam, Ph.D., ESEP
President, SPEC Innovations
571-485-7805
steven.dam@specinnovations.com
Sign Up at Innoslate.com

- www.innoslate.com
- Activate with your email
Overview

- Introductions
  1. Overview of Model-Based Systems Engineering
  2. Introduction to the Lifecycle Modeling Language
  3. Overview of Innoslate’s Capabilities
INTRODUCTIONS
Purpose

• To introduce model-based systems engineering using Innoslate®

• Provide a starting point for learning how to use Innoslate®
Class Guidelines

• Discussion…plus group participation
• Avoid interrupting others
• Ask questions as they arise
Introductions

• Steve Dam, Ph.D.
  – 40+ years of software development and systems engineering experience
  – Certified INCOSE Expert Systems Engineering Professional
  – Extensive experience in developing broad reaching and detailed architectures and systems engineering design across DoD, DOE, and NASA
  – Co-author (with Dr. Dinesh Verma) of Chapter 3, Concept of Operations and System Operational Architecture, *Applied Space Systems Engineering*
Who’s Here?

- Name?
- Organization?
- Familiar with Systems Engineering?
- Familiar with MBSE?
1. OVERVIEW OF MODEL-BASED SYSTEMS ENGINEERING
Views of Systems Engineering

Technical Orientation
A comprehensive, iterative problem-solving process that is used to:
– Transform customer requirements into a solution set
– Generate information for decision-makers
– Provide information for the next phase

Management Orientation
“… the management function which controls the total system development effort for the purpose of achieving an optimum balance of all system elements…”

— DSMC Handbook
Systems Engineering [INCOSE]

- Systems Engineering is an interdisciplinary approach and means to enable the realization of successful systems. It focuses on defining customer needs and required functionality early in the development cycle, documenting requirements, then proceeding with design synthesis and system validation while considering the complete problem:
  - Operations
  - Performance
  - Test
  - Manufacturing
  - Cost & Schedule
  - Training & Support
  - Disposal

- Systems Engineering integrates all the disciplines and specialty groups into a team effort forming a structured development process that proceeds from concept to production to operation. Systems Engineering considers both the business and the technical needs of all customers with the goal of providing a quality product that meets the user needs.
The Lifecycle

Current Operations and Maintenance
- Architecture Development
- System Design
- Hardware/Software Acquisition

Future Operations and Maintenance
- Operational T&E and Transition
- Integration and Test

Demolition and Disposal

Program Management

Integration & Verification
Systems Engineering During Design Phase

Adapted from EIA-632

Note: On the cloud the customer participate in the requirements development process and as such may want to focus on a scenario-based approach
What Is a Model?

• A model:
  – represents reality
    • By only “representing reality” it means that we are simplifying reality as it would take the Universe to model the Universe completely
    • “Essentially, all models are wrong, some models are useful”
  
• By useful, a model must meet the needs of both the developers of the model and their audience


George Edward Pelham Box (1919-2013)
What a Model Means to Me

• As a SE I Need:
  – A way to describe the system and its environment as simply as possible to make it understandable
    • In words AND
    • In pictures
  – AND verifiable through computable representations

• This model becomes a tool for me to describe to other stakeholders the system: what, why, where, when, how, how much, etc.
  – Ultimately, I need to be able to create a specification that can be used to buy or build the system and demonstrate to the owners and users of the system that it meets their needs
Drawings and Computable Models

• A drawing is a type of modeling approach

• A drawing consists of:
  – lines and boxes
  – text
  – pictures

• Drawings are static

• A computable model includes:
  – data that defines the system
  – it may include drawings to visualize the information
  – most “drawings” from a model result as visualizations of the data
Drawings Are Models, But …

• Drawings are useful visualizations of information

• But drawings are:
  – difficult to test
  – require extensive sets of rules
  – provide only 2-4 dimension of information including relationships effectively

• It would require a very large number of drawings to represent all the dimensions in a system model and extensive coordination to keep them consistent
Computable Model Features

• Models provide traceability in many dimensions
• Models can be automatically validated using simulation and rule checkers
• Models can be interrogated
• Models can be reused
• Models can be easily changed
Model-Based Systems Engineering

Definition

• “Model-based systems engineering (MBSE) is the formalized application of modeling to support system requirements, design, analysis, verification and validation, beginning in the conceptual design phase and continuing throughout development and later life cycle phases.” - INCOSE
What the Definition Tells Us

• By this definition, MBSE has been around for a long time – every since someone drew the first flowchart
• More recently it has come to mean a way to capture the essential elements of information in a database and then visualize that information in many different ways, including the production of the many SE documents needed (plans, specifications, risk reports, etc.)
• Many systems engineers equate SysML with MBSE – we disagree
Perhaps a Better Definition

• Model-Based Systems Engineering is the formalized application of modeling (static and dynamic) to support system design and analysis, throughout all phases of the system lifecycle, through the collection of modeling languages, structures, model-based processes, and presentation frameworks used to support the discipline of systems engineering in a model-based or model-driven context. The four tenets of this definition are:
  – Modeling Languages
  – Structure
  – Model-Based Processes
  – Presentation Frameworks


Hence, now MBSE isn’t just automated SE business as usual – it’s something new, improved, and valued by the community
2. INTRODUCTION TO THE LIFECYCLE MODELING LANGUAGE
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
LML Ontology* Overview

• 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

*Ontology = Taxonomy + relationships among terms and concepts  
** Taxonomy = Collection of standardized, defined terms or concepts
LML’s Simplified Schema

- Action
- Artifact
- Asset
  - Resource
- Characteristic
  - Measure
- Connection
  - Conduit
  - Logical
- Cost
- Decision
- Input/Output
- Location
  - Physical, Orbital, Virtual
- Risk
- Statement
  - Requirement
- Time

Supports capturing information throughout the lifecycle
Primary Entities
• Action
• Input/Output

Primary Entities
• Asset/Resource
• Connection

Documentation Entities
- Artifact
- Statement/Requirements

Functional Model

Physical Model

Parametric and Program Entities
- Characteristic/Measure
- Cost
- Decision
- Location
- Risk
- Time
LML Primary Entities and Relationships for SE Support

- **Artifact**
  - decomposed by/decomposes
  - source of/sourced by

- **Statement (Requirement)**
  - decomposed by/decomposes
  - traced to/traced from

- **Action**
  - decomposed by/decomposes
  - performed by/performs

- **Asset (Resources)**
  - decomposed by/decomposes
  - specified by/specifies

- **Characteristic (Measure)**
  - decomposed by/decomposes

- **Input/Output**
  - decomposed by/decomposes
  - generated by/generates
  - received by/receives
  - transferred by/transfers

- **Connection (Conduit)**
  - decomposed by/decomposes
  - connected by/connects
## LML Relationships Provide Linkage Needed Between the Classes

<table>
<thead>
<tr>
<th>Action</th>
<th>Artifact</th>
<th>Asset (Resource)</th>
<th>Characteristic (Measure)</th>
<th>Connection (Conduit, Logical)</th>
<th>Cost</th>
<th>Decision</th>
<th>Input/Output</th>
<th>Location (Orbital, Physical, Virtual)</th>
<th>Risk</th>
<th>Statement (Requirement)</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>decomposed by* related to*</td>
<td>references</td>
<td>consumed by</td>
<td>performed by</td>
<td>specifies</td>
<td>incurred</td>
<td>enables</td>
<td>generates</td>
<td>located at</td>
<td>causal mitigates</td>
<td>resolves</td>
<td>occurs</td>
</tr>
<tr>
<td>Artifact</td>
<td>referenced by</td>
<td>referenced by</td>
<td>referenced by</td>
<td>defines protocol for</td>
<td>incurred</td>
<td>enables</td>
<td>connected by</td>
<td>located at</td>
<td>causal mitigates</td>
<td>resolves</td>
<td>occurs</td>
</tr>
<tr>
<td>Asset (Resource)</td>
<td>decomposed by* related to*</td>
<td>references</td>
<td>consumed by</td>
<td>performed by</td>
<td>specifies</td>
<td>incurred</td>
<td>decomposed by*</td>
<td>(satisfied by)</td>
<td>causes</td>
<td>mitigates</td>
<td>resolves</td>
</tr>
<tr>
<td>Characteristic (Measure)</td>
<td>specifies</td>
<td>references</td>
<td>specifies</td>
<td>decomposition by*</td>
<td>related to*</td>
<td>specifies</td>
<td>specified by</td>
<td>located at</td>
<td>causal mitigates</td>
<td>resolves</td>
<td>occurs</td>
</tr>
<tr>
<td>Connection (Conduit, Logical)</td>
<td>connects to</td>
<td>specified by</td>
<td>decomposed by*</td>
<td>joined by</td>
<td>related to*</td>
<td>incurred</td>
<td>enables</td>
<td>result of</td>
<td>located at</td>
<td>causal mitigates</td>
<td>resolves</td>
</tr>
<tr>
<td>Cost</td>
<td>incurred by</td>
<td>incurred by</td>
<td>incurred by</td>
<td>incurred by</td>
<td>decomposed by*</td>
<td>enabled</td>
<td>incurred by</td>
<td>result of</td>
<td>located at</td>
<td>causal mitigates</td>
<td>resolves</td>
</tr>
<tr>
<td>Decision</td>
<td>enabled by</td>
<td>enabled by</td>
<td>enabled by</td>
<td>enabled by</td>
<td>decomposed by*</td>
<td>enabled by</td>
<td>enabled by</td>
<td>result of</td>
<td>located at</td>
<td>causal mitigates</td>
<td>resolves</td>
</tr>
<tr>
<td>Input/Output</td>
<td>generated by</td>
<td>references</td>
<td>transferred by</td>
<td>incurred</td>
<td>decomposed by*</td>
<td>located at</td>
<td>causal mitigates</td>
<td>resolves</td>
<td>occurs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Location (Orbital, Physical, Logical)</td>
<td>locates</td>
<td>specified by</td>
<td>locates</td>
<td>locates</td>
<td>locates</td>
<td>decomposed by*</td>
<td>located at</td>
<td>causal mitigates</td>
<td>resolves</td>
<td>occurs</td>
<td></td>
</tr>
<tr>
<td>Risk</td>
<td>caused by</td>
<td>caused by</td>
<td>causes</td>
<td>caused by</td>
<td>enabled by</td>
<td>causes</td>
<td>enabled by</td>
<td>result of</td>
<td>located at</td>
<td>causal mitigates</td>
<td>resolves</td>
</tr>
<tr>
<td>Statement (Requirement)</td>
<td>(satisfied by)</td>
<td>traced to</td>
<td>(verified by)</td>
<td>(satisfied by)</td>
<td>traced to</td>
<td>(verified by)</td>
<td>alternative of</td>
<td>enabled by</td>
<td>located at</td>
<td>causal mitigates</td>
<td>decomposed by*</td>
</tr>
<tr>
<td>Time</td>
<td>occurred by</td>
<td>occurred by</td>
<td>occurred by</td>
<td>occurred by</td>
<td>date resolved</td>
<td>causal mitigates</td>
<td>resolves</td>
<td>occurs</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- decomposed by/decomposes
- orbited by/orbits
- related to/relates
Diagrams Are Needed for Every Class

- Action Diagram (Mandatory)
- Asset Diagram (Mandatory)
- Spider Diagram (Mandatory)
- 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

- Class/Block Definition Diagram
  - Data modeling
- 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
No constructs – only special types of Actions – ones that enable the modeling of command and control/ information assurance to capture the critical decisions in your model.
Asset Diagram (mandatory)

Block diagram general form

Block diagram using pictures
Spider Diagram (Mandatory for Traceability)

Shows entities and relationships in visual form
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)
  - Enable diagram translations (e.g., Action Diagram to IDEF 0)

<table>
<thead>
<tr>
<th>LML Class</th>
<th>DM2</th>
<th>SysML</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action</td>
<td>Activity</td>
<td>Activity</td>
<td></td>
</tr>
<tr>
<td>Asset</td>
<td>Performer</td>
<td>Actor</td>
<td></td>
</tr>
</tbody>
</table>
LML Summary

- LML provides the fundamental foundation for a tool to support SE
- 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)
3. OVERVIEW OF INNOSLATE’S CAPABILITIES
The Innoslate® Solution

**Simplicity**
- Built-in help and common options
- Primary tool language is easy to learn
- Web browser user interface

**Collaboration**
- Real-Time Collaboration
- Easy Communication to other engineers

**Accuracy**
- Full Discrete Event Simulator which simulates cost, schedule, and performance
- Full Monte Carlo Simulator to simulate variance
- Intelligence View and Requirements Quality Checker

**Scalability**
- Tested to over 10 Million entities and 1,000 simultaneous users

**Full Lifecycle Management Support**
- Includes full requirements capability with Requirements View
- Includes full modeling capability (SysML/LML/IDEF0)
- Includes Test Center, CONOPS, Project Plan, and Test Plan document views

**Interoperability**
- CAD Integration
- Automatically generate and/or use other representations (SysML, DoDAF)
- Import from other RM and modeling tools (Word DOCX, DOORS CSV, Excel CSV, XMI)
- Integration with 700+ tools through Zapier (GitHub/Jira etc)
System Requirements

• Platform Independent
  – Works on Windows XP/7/8/RT, MAC OS X, Linux, iOS, Android

• Software
  – Any modern web browser
    (Google Chrome, Mozilla Firefox, Safari, IE 10 or 11)

• No downloads required
Innoslate Security

- All connections are SSL encrypted in transit
- New files uploaded are 256bit AES encrypted at rest
- All developers in Northern Virginia
- Public cloud provider has the following security certifications:
  - ISO 27001:2005
  - SAS70 Type II
  - SSAE 16 Type II
  - ISAE 3402 Type II
  - FedRAMP
- On-site version can be deployed locally behind your firewall
Cross View Real-time Collaboration

- Collaborate with your team members across multiple views
- Simultaneously shows real-time user status
- Group chat
- Project notifications
Innoslate Dashboard

• Setup “widgets” to see the information you want
  – Wiki
  – Tables
  – Bar and pie chart
  – Comment feed
• Activity Feed shows project changes
• Send Feedback
Innoslate Import Analyzer

- Import documents from spreadsheets, MS Word, and text/PDF documents
- Import XMI and XML from other Innoslate projects and other tools
Project Sharing

• You can share your project with your colleagues
  – Owner
  – Reviewer
  – Viewer
  – Collaborator
  – Or any other roll defined by the Organization
• Team, or user name/e-mail address
Documents View

- Author all your documents in Innoslate
- Create any document type
- Use pre-defined templates or develop your own
- Use search to find specific documents
- Add your own document types by creating a new label for the Artifact class with the word “Document” in the name
Requirements Documents

• Create Requirements hierarchies
• Track status
• Check quality
• Organize using labels
• Baseline
• Reports
Diagrams View

• Create or modify diagrams directly from this view
  – Select from 22 diagram types including all SysML
• Search for diagrams using complex searches
• Create reports in MS Word or MS PowerPoint
Key Diagram: Action Diagram

- Capture functional behavior with simple logic
- Include JavaScripts for decisions and complex calculations
- Allocate to Assets
- Use Inputs/Outputs to trigger events
- Resource modeling
Simulate Models

- Use Action Diagram or Activity Diagram
- Model functional behavior constrained by physical architecture
- Include resource modeling
- Develop cost and schedules from execution
Test Center

• Create test cases and suites
• Capture expected and actual results
• Track status
• Create Action Diagrams from Test Cases to model and cost test processes and procedures
Use Intelligence to Enhance Model Quality

• Provides 68 heuristics to improve modeling by applying NLP technology

• Select between warnings, errors, and ignoring heuristics

• Fix problems or ignore using buttons
Database View

- Filter and organize information
- Show and edit related entities
- Use complex search queries
- Save those queries
- Show information in hierarchical views
Edit Schema and Create Work Flows

- Add or modify classes, relationships, and attributes
- Hide schema elements
- Create work flows for any class using enumerated attributes
Use Cross-Project Relationships

• Cross-Project relationships enable the inclusion of information from one project into another

• If the person viewing the information does not have permission to see the other project then that information will be redacted in any view
Capture Product Design Information

- Capture and visualize CAD files
- Decompose into database objects
- Capture artifacts from other, more detailed analyses
Use Traceability Matrices to Connect Information

• Trace anything to anything else using the relationships
• Use NLP technology to help identify how information should be traced
• Identify suspect links too
Capture and Relate Program Management Information

- Model management processes using Action Diagram
- Create work breakdown structures
- Capture decisions
- Identify and manage risks and issues
- Get Gantt Charts and costs from simulations
- Get MS Project files from simulations
SUMMARY
Summary

The Full Lifecycle in One Tool