Lifecycle Modeling Language (LML) – A Technique for Enhancing Reliability, Availability, and Maintainability (RAM) throughout the Lifecycle

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

October 2013
Overview

- Why is RAM often overlooked until late in the lifecycle?
- What is LML?
- How does LML help enhance RAM?
- What processes and tools work with LML to enhance RAM?
- Summary
Why is RAM often overlooked until late in the lifecycle?

• RAM analysis requires details to estimate uncertainties in estimated values and requirements, which takes time and money
• As such, it often is not addressed at all until the detailed design phase
• However, RAM should be part of the overall scenario analysis at the very beginning of the concept development phase

So what happens? Someone arbitrarily assigns the number of “9’s” needed.
Example “Requirements” for FireSAT

From Applied Space Systems Engineering, p. 113

• Reliability: “The FireSAT spacecraft shall have an on-orbit lifetime of at least five years”
• Availability: “The FireSAT spacecraft shall have an operational availability of 98%, excluding outages due to weather, with a maximum continuous outage of no more than 72 hours”
• Maintainability: “The FireSAT spacecraft shall require the removal (or opening) of no more than ten fasteners (panels) to replace any Line Replaceable Unit (LRU) ... during pre-launch ground operations”

Where did these come from? Were they the result of analyses or are they just best guesses?
Lots of metrics to take into account

• It’s not just the RAM metrics – it’s all the “illities”

• How can we capture and trace all these metrics?

From Applied Space Systems Engineering, p. 189
What Is LML?
Lifecycle Modeling Language (LML)

• LML combines the logical constructs with an ontology to capture information
  – SysML – mainly constructs – limited ontology
  – DoDAF Metamodell 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 Models

Documentation Entities
- Artifact
- Statement/Requirements

Functional Model
Primary Entities
- Action
- Input/Output

Physical Model
Primary Entities
- Asset/Resource
- Connection

Parametric and Program Entities
- Risk
- Characteristic/Measure
- Time
- Cost
- Location
LML Taxonomy Simplifies and Enhances the Semantic Schema Elements

- Action
- Artifact
- Asset
  - Resource
- Characteristic
  - Measure
- Connection
  - Logical
  - Conduit
- Cost
- Input/Output
- Location
  - Physical
  - Orbital
  - Virtual
- Risk
- Software Interface
- Statement
  - Requirement
  - Decision
- Time
### 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, Logical)</th>
<th>Risk</th>
<th>Statement (Requirement)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action</td>
<td>decomposed by* related to*</td>
<td>references</td>
<td>(consumes) performed by (produces) (seizes)</td>
<td>specified by</td>
<td>-</td>
<td>incurs</td>
<td>enables results in</td>
<td>generates resolves</td>
<td>located at</td>
<td>causes mitigates resolves</td>
</tr>
<tr>
<td>Artifact</td>
<td>referenced by</td>
<td>decomposed by* related to*</td>
<td>referenced by</td>
<td>defines protocol for referenced by</td>
<td>incurs</td>
<td>referenced by</td>
<td>enables referenced by results in</td>
<td>referenced by</td>
<td>located at</td>
<td>causes mitigates resolves</td>
</tr>
<tr>
<td>Asset (Resource)</td>
<td>(consumed by)</td>
<td>performs (produced by)</td>
<td>(seized by)</td>
<td>reference decomposed by* orbited by* related to*</td>
<td>specified by</td>
<td>connected by</td>
<td>incurs</td>
<td>made responds to results in</td>
<td>-</td>
<td>located at</td>
</tr>
<tr>
<td>Characteristic (Measure)</td>
<td>specifies</td>
<td>references specifies</td>
<td>specifies decomposed by* related to*</td>
<td>specified by</td>
<td>incurs</td>
<td>specifies</td>
<td>enables results in specifies</td>
<td>specified at specifies</td>
<td>causes mitigates resolves</td>
<td>(satisfies) specifications</td>
</tr>
<tr>
<td>Connection (Conduit, Logical)</td>
<td>-</td>
<td>defined protocol by references</td>
<td>connects to</td>
<td>specified by decomposed by* orbited by* related to*</td>
<td>incurs</td>
<td>enables results in</td>
<td>transfers located at</td>
<td>causes mitigates resolves</td>
<td>(satisfies) traced from (verifies)</td>
<td>occurs</td>
</tr>
<tr>
<td>Cost</td>
<td>incurred by</td>
<td>incurred by references</td>
<td>incurred by</td>
<td>incurred by</td>
<td>decomposed by* related to*</td>
<td>enables incurred by</td>
<td>results in</td>
<td>incurred by</td>
<td>located at</td>
<td>causes mitigates resolves</td>
</tr>
<tr>
<td>Decision</td>
<td>enabled by result of</td>
<td>enabled by references result of</td>
<td>enabled by made by result of</td>
<td>enabled by result of</td>
<td>enabled by result of</td>
<td>decomposed by* related to*</td>
<td>enabled by result of</td>
<td>enabled by result of</td>
<td>located at</td>
<td>causes mitigates resolves</td>
</tr>
<tr>
<td>Input/Output</td>
<td>generated by</td>
<td>received by references</td>
<td>-</td>
<td>specified by</td>
<td>transferred by</td>
<td>incurs</td>
<td>enables results in decomposed by* related to*</td>
<td>located at</td>
<td>causes mitigates resolves</td>
<td>(satisfies) traced from (verifies)</td>
</tr>
<tr>
<td>Location (Orbital, Physical, Logical)</td>
<td>locates</td>
<td>locates</td>
<td>locates</td>
<td>locates</td>
<td>locates</td>
<td>locates</td>
<td>locates</td>
<td>decomposed by* related to*</td>
<td>locates</td>
<td>locates mitigates</td>
</tr>
<tr>
<td>Risk</td>
<td>caused by mitigated by resolved by</td>
<td>caused by mitigated by references resolved by</td>
<td>caused by mitigated by by resolved by</td>
<td>caused by mitigated by</td>
<td>caused by</td>
<td>incurs</td>
<td>mitigated by</td>
<td>resolved by</td>
<td>caused by</td>
<td>enabled by</td>
</tr>
<tr>
<td>Statement (Requirement)</td>
<td>(satisfied by) traced to (verified by)</td>
<td>references (satisfied by) traced to (verified by)</td>
<td>(satisfied by) traced to (verified by)</td>
<td>(satisfied by) traced to (verified by)</td>
<td>(satisfied by) traced to (verified by)</td>
<td>(satisfied by) traced to (verified by)</td>
<td>alternative of</td>
<td>enabled by</td>
<td>traced to (verified by)</td>
<td>located at (satisfied by) traced to (verified by)</td>
</tr>
<tr>
<td>Time</td>
<td>occurred by</td>
<td>occurred by</td>
<td>occurred by</td>
<td>occurred by</td>
<td>occurred by</td>
<td>date resolves occurred by</td>
<td>occurred by</td>
<td>occurred by</td>
<td>occurred by</td>
<td>occurred by</td>
</tr>
</tbody>
</table>
LML Action Diagram Captures Functional and Data Flow

1.1 Action

1.2 Which path?

1.3 Action in Parallel Action

1.4 Optional Action 1

1.5 Exit Criteria

1.6 Optional Action 2 in Loop

1.7 Synchronize Information

External Input

Start

OR

Trigger

External Output

End

Input/Output 1

Input/Output 2

Input/Output 3
Uses Common Diagrams for Every Class

- Physical Block (Asset) 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

- 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
How Does LML Help Enhance RAM?
How does LML help enhance RAM?

• LML was designed with all aspects of systems engineering across the lifecycle
• LML provides:
  – Asset/Resource entities, Asset Diagrams, and Characteristics/Measures entities to capture physical information about the system
  – Action entities, Action Diagrams, and Input/Output to capture and model processes
  – Action Diagrams can be simulated to include Resource use
• As such, LML can support the analyses needed to derive key RAM metrics, such as mean time between failures (MTFB)
Example: Modeling for Reliability

- Use of redundancy to enhance reliability
- Modeling multiple computers that “vote” on a value
Example continued

- Functional model equivalent using Action Diagram
- Timing provided for each computer can be a random distribution, as can failure modes
Simulation of Example

- Discrete Event Simulation of the Action Diagram can show the random nature of timing
- Sensitivity to failure modes can then be identified and mitigated
Monte Carlo simulation of Action Diagram supports reliability analysis

- Executing the model with random time distributions provides way to derive key metric requirements
FireSAT Failure Mode Hierarchy

• This hierarchy comes from a series of Action Diagrams that model the failure processes.
Action Modeling for FMECA

- Modeling failure modes with Action Diagram
Execution of FMECA Model

- Monte Carlo simulation shows notional failure distribution for mission
- Realistic probability can now be used to assess the potential impacts of these failure on the systems
What Processes and Tools Work with LML

• We use a “middle-out” process that begins with functional analysis (scenarios) and derives the functional and performance requirements via simulation.

• Tools require both discrete event and Monte Carlo simulations of the LML Action Diagram.
This implementation of the middle-out approach has been proven on a variety of architecture projects.
Action Modeling with Innoslate

Action Diagrams for functional modeling can be simulated using discrete event and Monte Carlo techniques.
Summary

• LML provides the necessary language entities to capture the RAM-related information
• The accompanying tool must implement the language and have the capability to extend it to meet any specific needs
• The process used should emphasize all the “ilities” including RAM