29th Annual INCOSE international symposium
Orlando, FL, USA
July 20 - 25, 2019

Validating a System Architecture Model (SAM) for a Department of Defense (DoD) Acquisition Program Using a Phased Approach

Gene Rosenthal
Systems Engineer Staff – LM Space

Agenda

- System Safety Program Background
- Project Objectives & Deliverables
- Safety Critical Functions
- Model-Based Environment Transition Results
- Model Validation – Phased Approach
- Model Validation Highlights & Tracking
- Challenges & Lessons Learned
- Conclusion
- References & Q&A

System Safety Program Background

- The DoD System Safety Program consists of systems owners (contractors) and independent assessors
- Safety assessments are performed on systems architecture using a systems engineering approach
  - Established/Rigorous controlled process used to vet changes
  - Uses a defined set of safety critical functions for each subsystem
  - Assessments captured numerous documents, spreadsheets, and charts
- Safety Critical functions are managed in a text based document and distributed among various appendices by subsystem
  - Contractors responsible for each subsystem manage their critical functions
  - Each critical function lists associated safety standard(s) and additional attributes used for categorization
- Final assessments are Out-briefed at Systems Engineering milestones and Independent Safety Reviews
- The DoD realized to accommodate schedule pressure, fixed price contracts, it needs to improve how systems engineering is performed on programs by being faster and more agile
- Model-Based Systems Engineering (MBSE) addresses these concerns, and can lead to significant cost savings
- A systems engineering transformation initiative was chartered by the DoD
  - The DoD System Safety program was identified as a pilot project to exercise a model-based transition

Project Objectives & Deliverables

- The System Safety Pilot project included a mix of contractor critical function owners and customers
- Two Main Objectives:
  1. Duplicate safety document content in a system model to enable ongoing model-based management of safety data
  2. Establish a design-agnostic safety framework for use in future architecture trades
- Deliverables:
  1. Functional model capturing critical and derived functions that traces to subsystem physical architecture
  2. Modeling approach
  3. CONOPs to support safety assessments using the model

Safety Critical Function Breakdown

- Each subsystem critical function lists associated safety standard(s) and additional attributes used for categorization

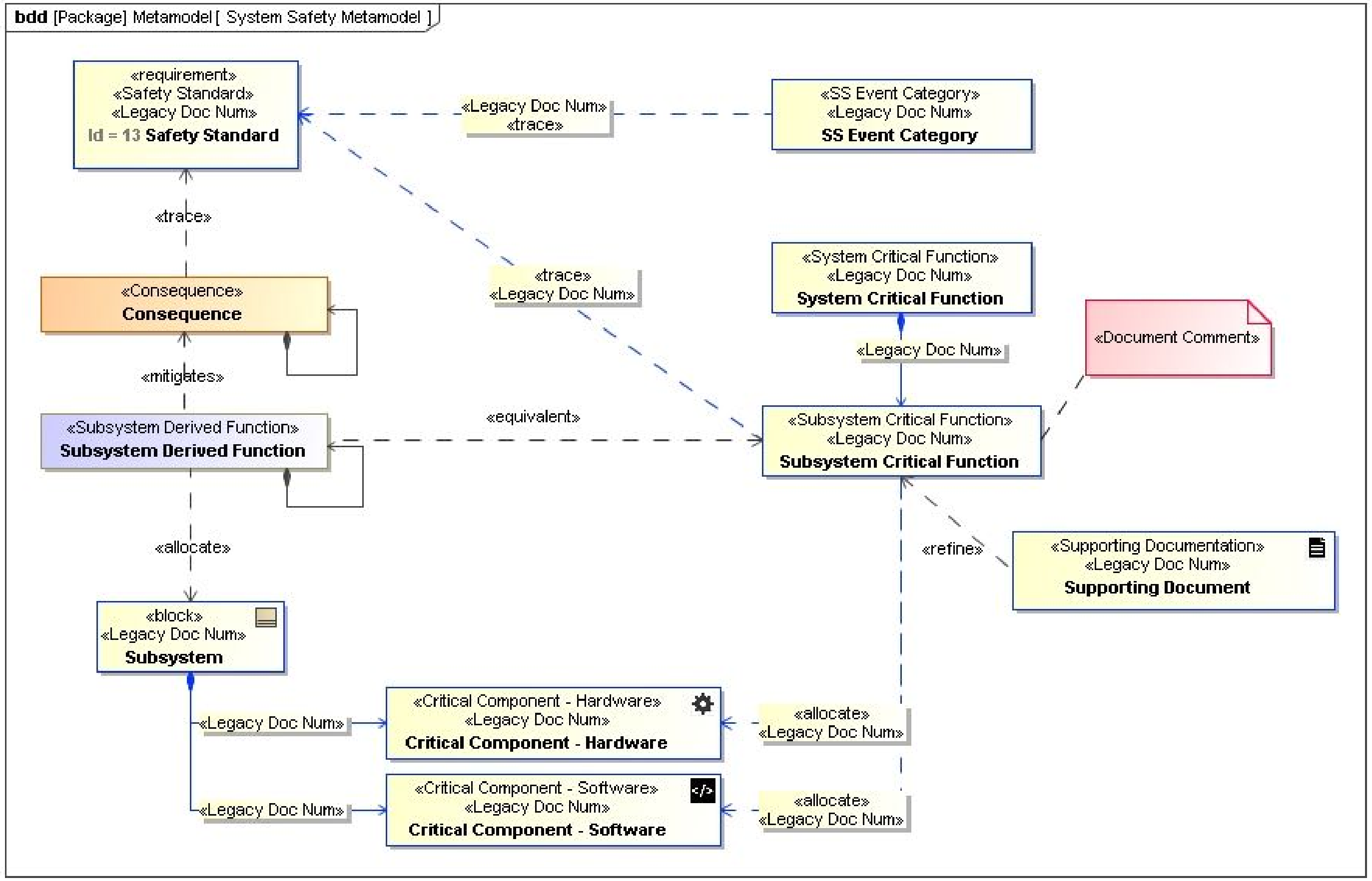
Safety Standard	Applicable safety standard the subsystem critical function satisfies. Satisfying the standard(s) is a requirement on the subsystem to prevent hazardous condition(s).
Event Category	Event categories were also used to classify subsystem critical functions. These event categories are closely related to the safety standards and allows critical functions related to a particular event easily identified (e.g. security).
System Critical Function	System critical functions are used to “bucket” subsystem system critical functions. One or more subsystem critical functions may fall under one system-level function.
Subsystem Critical Function	Name of the subsystem critical function
Definition	Definition of the subsystem critical function
Hazard/Rationale	In addition to the definition, each subsystem critical function lists the potential hazards that may result if the function does not meet the intent; related to the safety standards/event categories.
Critical Component(s)	Each subsystem critical functions list the affected component(s) related to the critical function
Supporting Documentation	Documentation related to critical function for reference

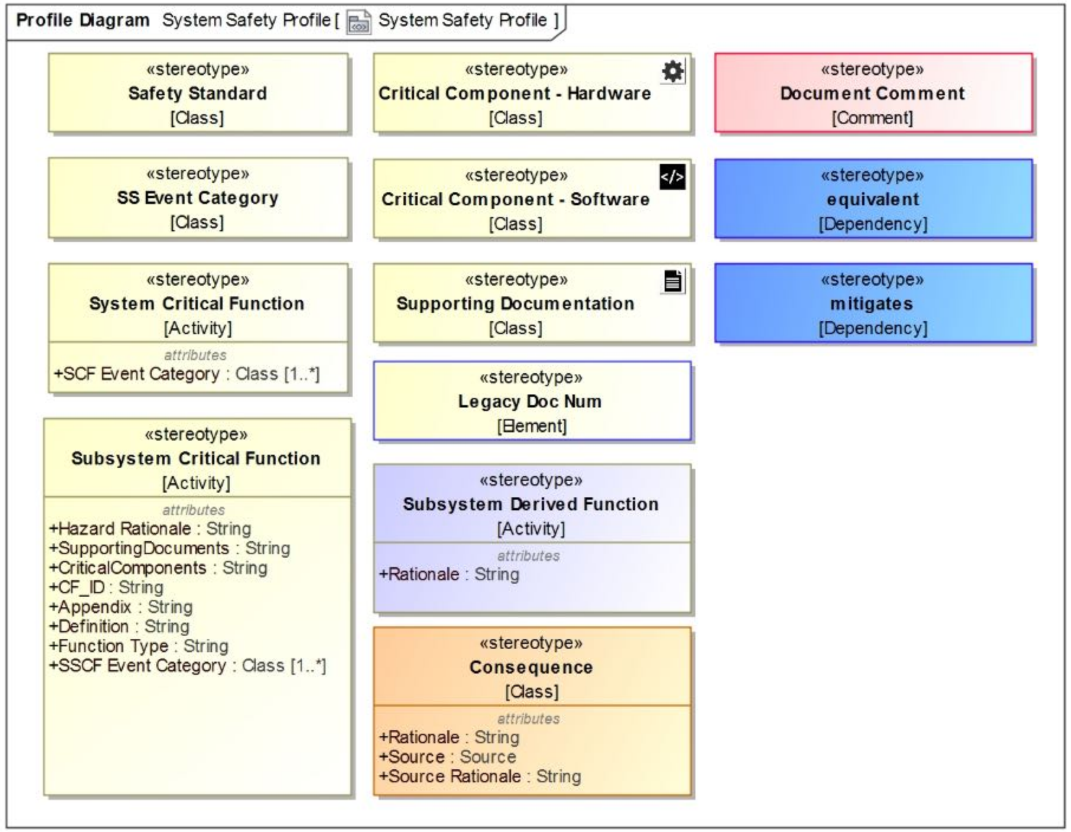
Understanding Document Structure Key to Model Organization

Model-Based Environment Transition Results

- Model-Based Transition was done using a Phased Approach
  1. Setting up the Model - Establish Profile, Metamodel, and Package Structure
  2. Reproduce document content within the model
     - Generated table views that resemble tables within the document appendices
  3. Capture design-agnostic functions and rationale
     - Safety critical functions and rationale were very design-specific
     - Requires significant revision to apply to future system upgrades
     - Two levels of derived content
       - Subsystem Derived Functions: derived from Critical Function definitions
       - Consequences: derived from either the Safety Standard, Hazard/Rationale, or through Subject Matter Experts
  4. Developed Critical Function “Threads” - Block Definition Diagram (BDD)
  5. Identify approaches and benefits to use the model for safety activities (Use Cases)
- Issues uncovered:
  - Naming conventions/classification were reused
  - Compound functions/names
- Results:
  - Safety impacts related to architecture changes are now easily visible
  - Views established allow for quicker assessments to be made on critical functions or physical architecture changes
  - Design-agnostic safety model lays the foundation for future trade studies and safety assessments

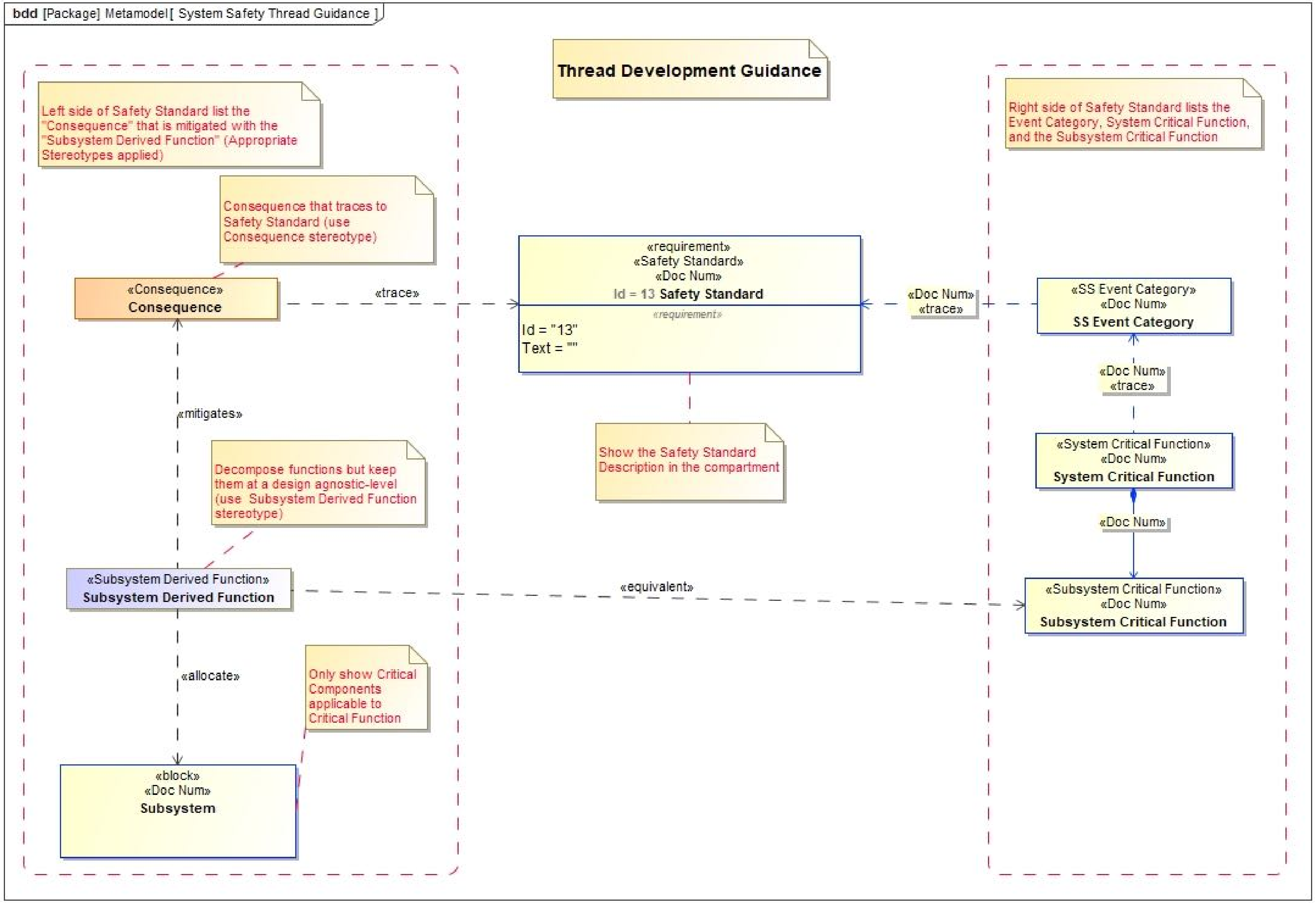
Metamodel and Profile





Metamodel & Profile Enable a Consistent Modeling Approach

Thread Development Guidance



Model Validation – Phased Approach

- Validation was done using a phased approach
  1. Model content accurately captures safety document content
  2. Derived functions – design agnostic? Consistent terminology used? Reuse?
  3. Consequences – accurately reflect rationale? Relationships to derived functions were correct
  4. Deep dive into derived functions that cross multiple subsystems (handshakes)
  5. Visual thread validation – element relationships correct

Validation Ensures Model Utility

Validation Phase 1 – Safety Document Accuracy

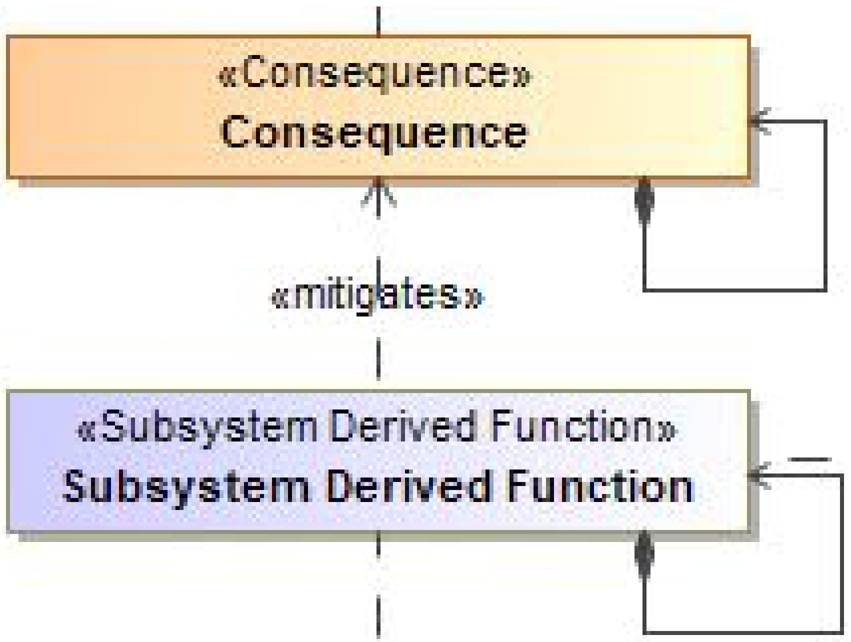
- Deep-dive review performed to ensure model accurately captured safety document content
- Systematic review by subsystem with the appropriate Subject Matter Expert(s)
- Compared against critical functions tables within each document appendix
- Methods used
  - Generated tables in the model to partition content into simpler views for review
    - Any updates made, were automatically propagated to any view using the same model elements
- Issued uncovered related to the metamodel
  - Initial metamodel element relationships were defined to match the document structure
  - Highlighted issues between the Document Ontology and the Model
  - Ambiguous relationships
    - Possible to trace to the wrong safety standard using the event category element
  - Metamodel revised after evalutating proposed solutions
  - Led to a better model/product

Validation Phase 2 – Derived Functions

- Derived functions extrapolate critical function definition to describe the intent
  - Knowledge transfer for future workforce
- Focus of this phase was to ensure derived functions are
  - Design-agnostic, but not at a point where they offer no value
  - Consistent terminology used
  - Look for opportunities for reuse
  - Look for opportunities to consolidate
- Derived functions with dependency relationships and the “equivalent’ stereotype applied to Subsystem Critical Function were reviewed
  - Assess if relationships were correct or if another derived function was required
- Additional tables and views were established to aid this process (BDDs, Tables, Relationship Maps)

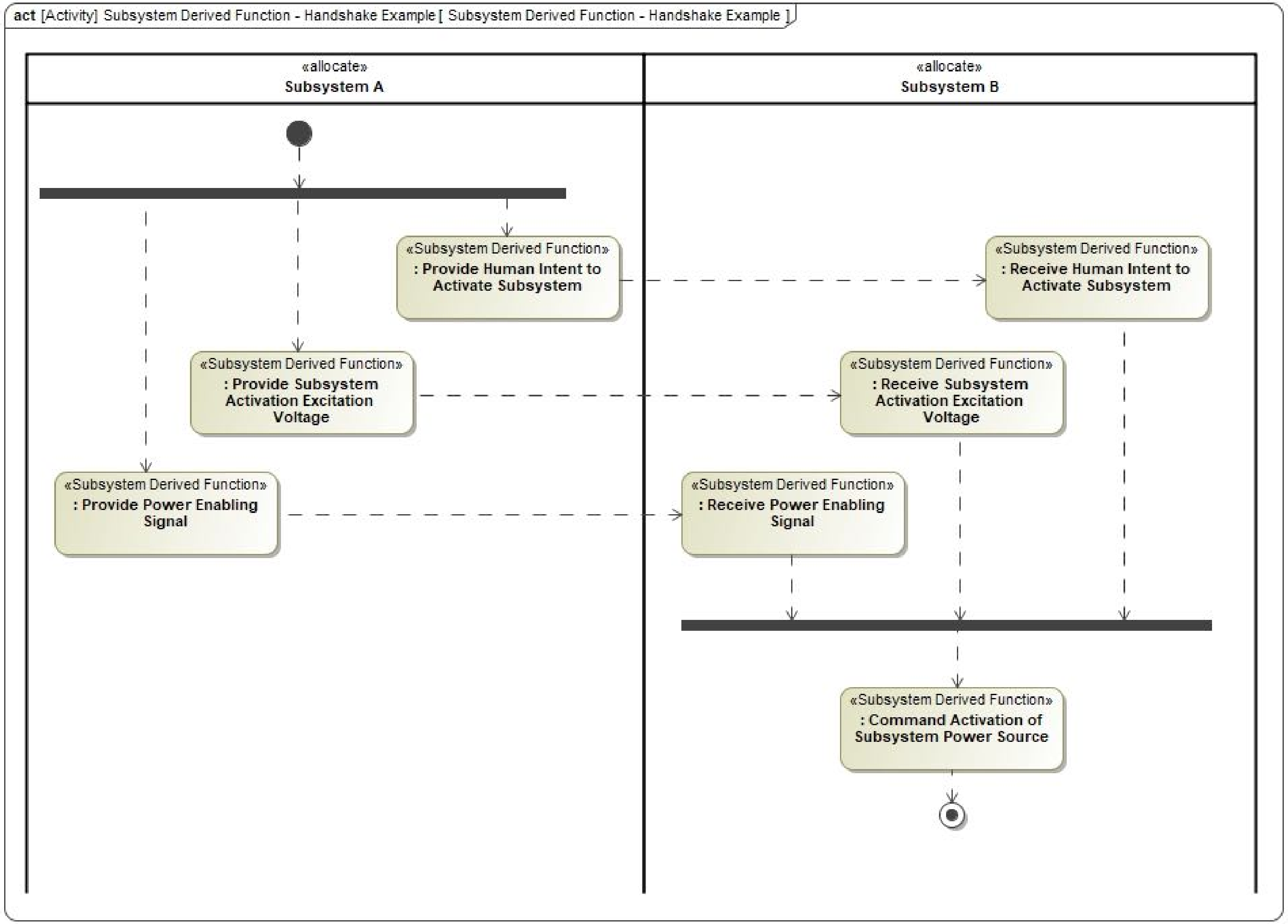
Validation Phase 3 – Consequences

- Similar to derived functions, consequences were reviewed using similar criteria
  - Consistent terminology used
  - Look for opportunities for reuse
  - Look for opportunities to consolidate
- Consequence relationships to Derived Functions also validated
  - Equivalent derived functions mitigates the consequences
- Some cases, the relationship was not correct and required a different derived function
- Safety subject matter expert review played a key role



Validation Phase 4 – "Handshake" Derived Functions

- Deep dive into derived functions that cross multiple subsystems (handshakes)
- Initial attempts to perform review involved spreadsheets and different model views
- Activity Diagrams used as a "tool" to perform the review
- Valuable approach, easy to determine if gaps existed and if consistent terminology was used



Validation Phase 5 – Visual Thread Validation

- Focus was to validate each critical function thread was visually consistent
  - Aligns with metamodel and development guidance
- Readability is important so consistency is required
- Allowed the team to “spot check” if appropriate stereotypes were applied
- Validation scripts were used to aid in the review
  - Metamodel & Profile used by validation scripts
  - Issues highlighted in red
  - Generic Tables used to report results
- Majority of the updates centered on diagram clean up and organization

Model Validation Highlights & Tracking

- Key items used during the Validation Effort
  - Safety Profile – Key when validating content matched the safety document
  - Metamodel – validates if model relationships were properly established
  - Validation Scripts – scripted used to auto-validate relationships, check for redundancy, and look for orphan elements
  - Activity Diagrams – Used as a tool to validate “handshake” derived functions
- Validation Tracking
  - Few approaches taken
  - Excel spreadsheets used at first, but were inefficient; risk of multiple versions
  - Validation Profile setup to track validation level directly in the model
    - Relationships establish to model elements representing level of validation
    - Dependency Matrix use to manage relationships
  - Process worked well, but not efficient as content matured during reviews

Challenges and Lessoned Learned

- Phased Approach
  - Lack of an Agile tool made tracking actions & progress difficult
  - Using a tool, such as JIRA, would have helped capture work to be done (backlog) and assign tasks (sprints)
- Visualizing Handshake Interactions
  - Switching back and forth between critical functions threads was not efficient
  - Activity Diagrams great improved process
  - Levels of decomposition presented a challenge
  - Structured activity nodes allowed gaps to be filled
- Transition from a Document to a Model
  - Visual representation (threads) of Critical Functions highlight document inconsistencies
  - Document ontology flaws discovered
- Validation Tracking
  - Spreadsheets – difficult to track; risk of having multiple versions
  - Validation Profile & Dependency Matrix – tedious to maintain as the model matured
- SysML® Education
  - Several folks new to MBSE and SysML®
  - General concepts understood, but nuances of modeling language and tools were new concepts

Conclusion

- Using a phased approach for validation allowed for more agility
- Validation emphasizes the models utility
- Key items used, like scripts, can help automate the reviews
- Model updates made in earlier phases help resolve issues in later phases (time savings!)
- Beginning to collaborate with other projects to pilot the models use
  - Includes standing up a multi-contractor collaborative environment
- As more programs begin adopting MBSE and use this model, the more insight and feedback will further mature the model

Questions ?

Biography

Email:
gene.rosenthal@lmco.com
Phone: 408-756-5654

- Gene Rosenthal, Systems Engineer Staff
  - Lockheed Martin, Space ~15 years
  - Model-Based Systems Engineer (MBSE) practitioner for ~3 years
  - Leading multiple MBSE projects, including projects for the Navy and Missile Defense customers
  - Previously worked on other programs, including the International Space Station and NASA's Near Infa-Red Camera (NIRCam) programs

29th Annual INCOSE
international symposium

Orlando, FL, USA
July 20 - 25, 2019

www.incose.org/symp2019