What Is Systems Engineering?

SPEC INNOVATIONS

INCOSE’s Definition of Systems Engineering

- “Traditional Definition”
  - “Systems Engineering (SE) is an interdisciplinary approach and means to enable the realization of successful systems. It focuses on holistically and concurrently understanding stakeholder needs; exploring opportunities; documenting requirements; and synthesizing, verifying, validating, and evolving solutions while considering the complete problem, from system concept exploration through system disposal. (INCOSE 2012, modified.)”
- INCOSE Fellows’ Definition
  - “A transdisciplinary and integrative approach to enable the successful realization, use, and retirement of engineered systems, using systems principles and concepts, and scientific, technological, and management methods.”

Source: SE Body of Knowledge (https://www.sebokwiki.org/wiki/Systems_Engineering_Overview) Accessed 10/10/2022

INCOSE's Primary Focus of Systems Engineering Today

- INCOSE's Definition of MBSE
  - "Model-based systems engineering (MBSE) is the formalized application of modeling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases."
- The original idea was to transform systems engineering from "document-based" to "model-based"
- By this definition, we have been doing MBSE every since someone used a template to draw a picture
- SysML has been proposed as "The MBSE Language" by many

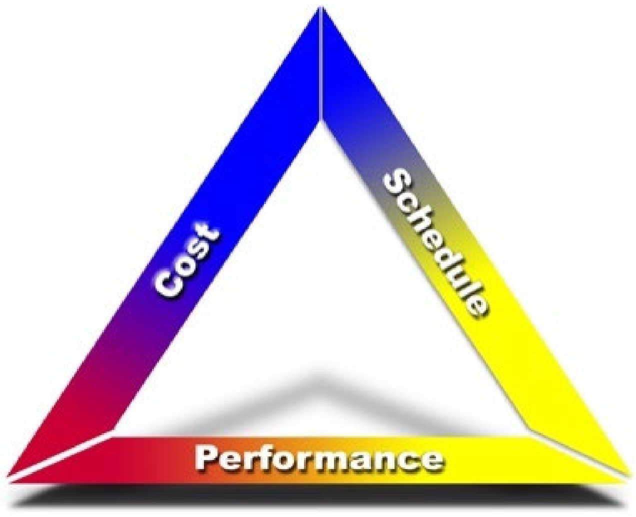
What Do Systems Engineers Do?

- To determine if SysML is sufficient, we first need to identify what systems engineers do and how SysML supports those activities
- The essence of Systems Engineering is optimization
  - We optimize cost, schedule, and performance, while mitigating risk in each area
  - We optimize the design across all the engineering disciplines

We solve problems, but we need to do so in a way people understand and have confidence in the solution

Cost, Schedule, and Performance Optimization

- Cost estimating is an engineering job
  - Requires engineering judgement and experience
  - Conducts trade-off studies for the system, which includes cost
  - Often creates the technical portion of the work breakdown structure (WBS)
- Schedule estimating requires engineering input
  - Also requires engineering judgement and experience
  - Knowledge of the detailed development processes
- Performance requires modeling and simulation
  - Used to predict performance over entire lifecycle
  - Traded off with other factors

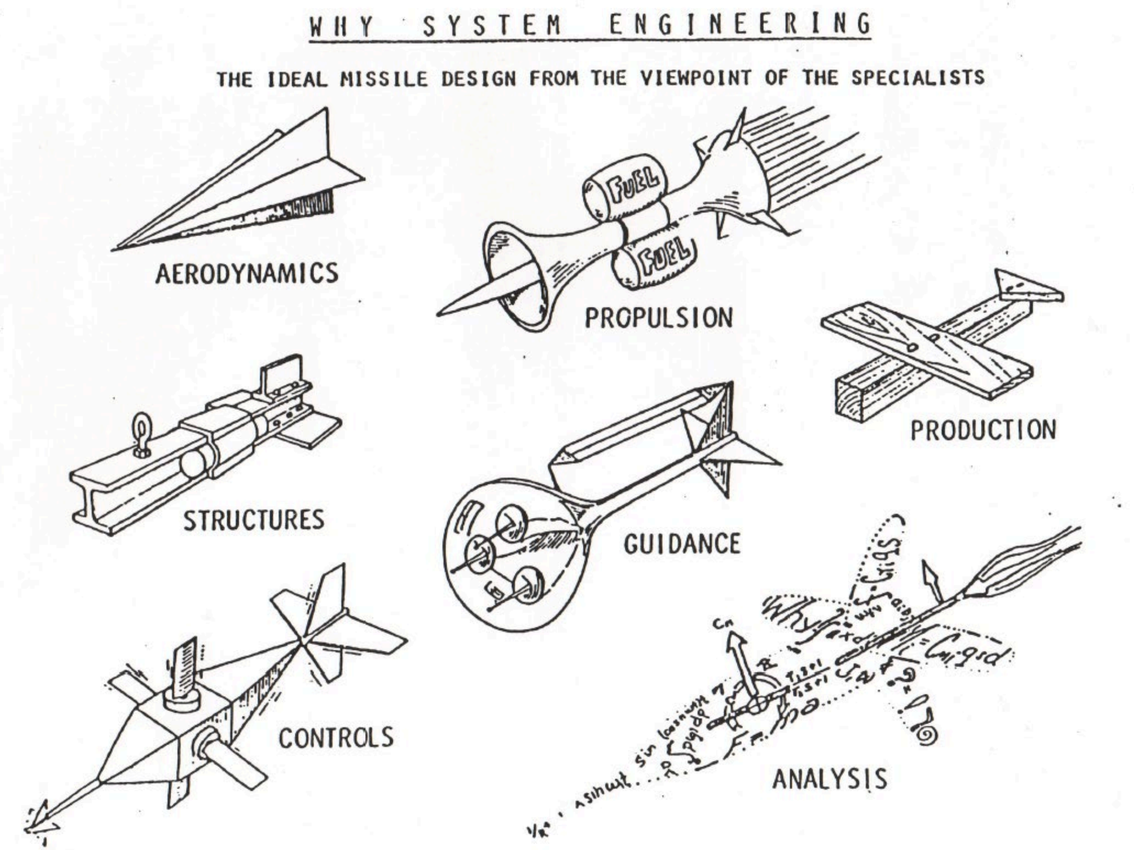


A shared responsibility with program management

Too often cost and schedule get fixed early in the program and so performance degrades

Optimizing the Design

- Classic picture seems funny, but a lot of truth in it
- Balancing all the different perspectives requires significant judgement and temperament
- Requires being a translator between all these different languages
- Must be seen as an "honest broker"



Software is just another detailed design discipline – SysML is based on a software "language"

How Can SE Be “Faster, Better, and Cheaper”?

- Faster comes from executing many processes and tasks in parallel
  - Tools can help us keep the information coherent, while many activities occur
  - Enables large number of technical stakeholders to collaborate on same design at the same time
  - Enables communication of progress to non-technical stakeholders
- Better comes from greater accuracy
  - Simulators provide means to test processes before they are implemented
  - Rule checkers verify drawings are created uniformly
  - Advanced techniques can even identify quality factors in requirements
- Cheaper comes from taking less time to do the job
  - Tools must be easy enough to use that it takes less time to perform a task

An MBSE Methodology must be used to enable these benefits

MBSE Methodology

- Methodology is defined by:
  - Technique
  - Processes
  - Tools
- In systems engineering the technique is the language
  - IDEF
  - SysML
  - LML
- Language consist of nouns, verbs, adjectives, and adverbs
  - A diagramming framework does not provide these explicitly
  - Language requires an ontology

A key goal of any technique should be a means to capture the information needed

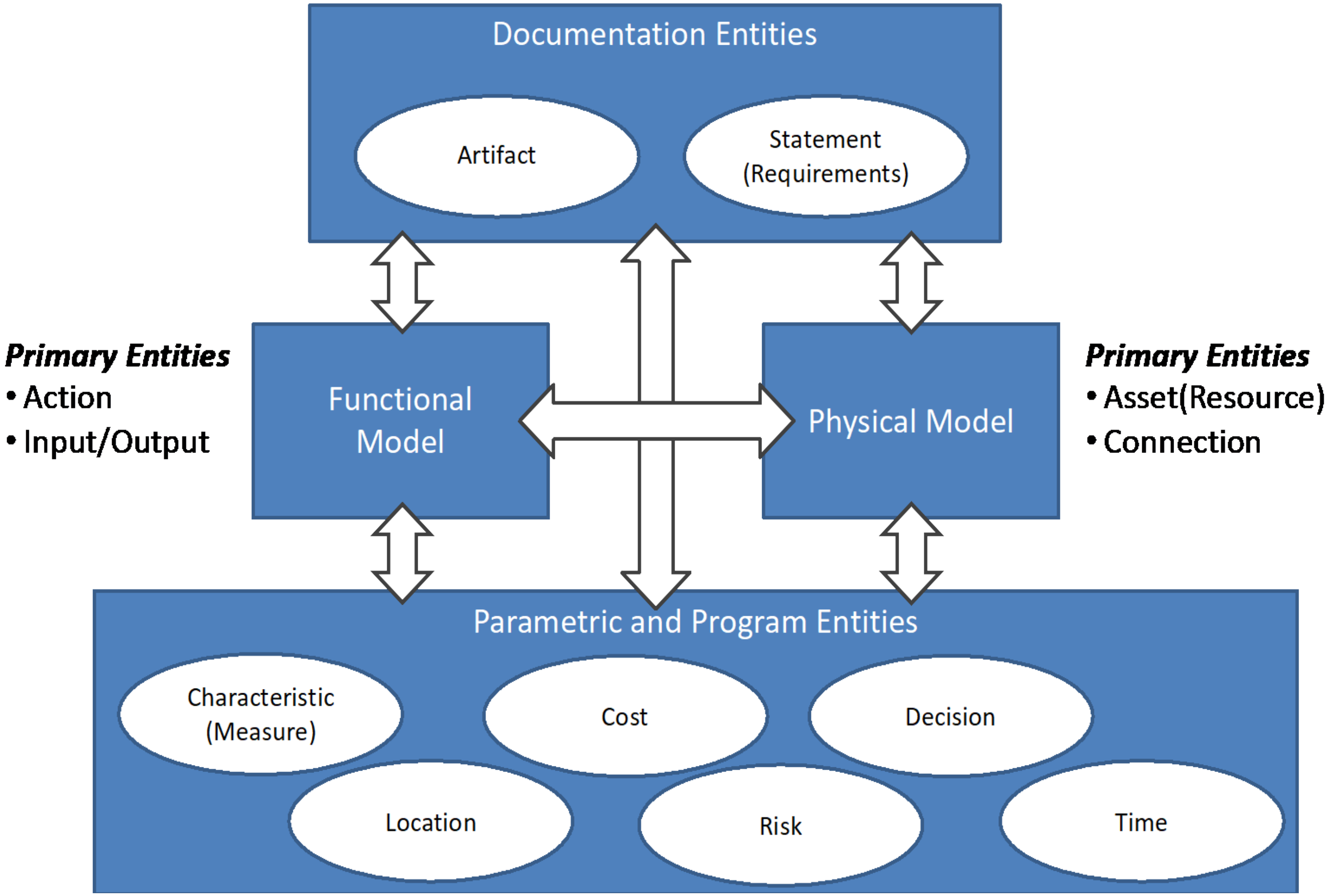
LML – A Real MBSE Technique

- LML consists of an ontology and a small set of “mandatory” diagrams
- Diagrams are used to visualize information
- LML contains 12 primary and 8 child classes as a basis
- The number of diagram types that would be required to fully express all this information visually is roughly 20! or 10^{18}
- LML was designed as a language for systems engineering and program management, by systems engineers and program managers

LML Classes have proven to capture at least 80% of the information needed

LML Entity Classes and Models

- All four models are built from the entities in the diagram
- Types of each class can be used to distinguish entities within each class (e.g., Action, Activity, Function, Task, etc.)
- Typing can be implemented as an enumerated attribute or label
- Attributes on the entities represent the adjectives for the language



This simple set of classes have many relationships between them

LML Relationships

- Same verb forms in each direction (e.g., performed by/performs)
- All hierarchical relationships are the same terms (decomposed by/decomposes)
- Attributes on relationships also provide the needed adverbs for the language

	Action	Artifact	Asset (Resource)	Characteristic (Measure)	Connection (Conduit, Logical)	Cost	Decision	Input/Output	Location (Orbital, Physical, Virtual)	Risk	Statement (Requirement)	Time
Action	decomposed by* related to*	references	(consumes) performed by (produces) (seizes)	specified by	-	incurs	enables results in	generates receives	located at	causes mitigates resolves	(satisfies) traced from (verifies)	occurs
Artifact	referenced by	decomposed by* related to*	referenced by	referenced by specified by	defines protocol for referenced by	incurs referenced by	enables referenced by results in	referenced by	located at	causes mitigates referenced by resolves	referenced by (satisfies) source of traced from (verifies)	occurs
Asset (Resource)	(consumed by) performs (produced by) (seized by)	references	decomposed by* orbited by* related to*	specified by	connected by	incurs	enables made responds to results in	-	located at	causes mitigates resolves	(satisfies) traced from (verifies)	occurs
Characteristic (Measure)	specifies	references specifies	specifies	decomposed by* related to* specified by*	specifies	incurs specifies	enables results in specifies	specifies	located at specifies	causes mitigates resolves specifies	(satisfies) spacifies traced from (verifies)	occurs specifies
Connection (Conduit, Logical)	-	defined protocol by references	connects to	specified by	decomposed by* joined by* related to*	incurs	enables results in	transfers	located at	causes mitigates resolves	(satisfies) traced from (verifies)	occurs
Cost	incurred by	incurred by references	incurred by	incurred by specified by	incurred by	decomposed by* related to*	enables incurred by results in	incurred by	located at	causes incurred by mitigates resolves	incurred by (satisfies) traced from (verifies)	occurs
Decision	enabled by result of	enabled by references result of	enabled by made by responded by result of	enabled by result of specified by	enabled by result of	enabled by incurs result of	decomposed by* related to*	enabled by result of	located at	causes enabled by mitigated by result of resolves	alternative enabled by traced from result of	date resolved by decision due occurs
Input/Output	generated by received by	references	-	specified by	transferred by	incurs	enables results in	decomposed by* related to*	located at	causes mitigates resolves	(satisfies) traced from (verifies)	occurs
Location (Orbital, Physical, Logical)	locates	locates	locates	locates specified by	locates	locates	locates	locates	decomposed by* related to*	locates mitigates	locates (satisfies) traced from (verifies)	occurs
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 specified by	caused by mitigated by resolved by	caused by incurs mitigated by resolved by	caused by enables mitigated by results in resolved by	caused by mitigated by resolved by	located at mitigated by	caused by* decomposed by* related to* resolved by*	caused by mitigated by resolved by	occurs mitigated by
Statement (Requirement)	(satisfied by) traced to (verified by)	references (satisified by) sourced by traced to (verified by)	(satisified by) traced to (verified by)	(satisified by) specified by traced to (verified by)	(satisified by) traced to (verified by)	incurs (satisified by) traced to (verified by)	alternative of enables traced to results in	(satisified by) traced to (verified by)	located at (satisified by) traced to (verified by)	causes mitigates resolves	decomposed by* traced to* related to*	occurs (satisified by) (verified by)
Time	occurred by	occurred by	occurred by	occurred by specified by	occurred by	occurred by	date resolves decided by occurred by	occurred by	occurred by	occurred by mitigates	occurred by (satisfies) (verifies)	decomposed by* related to*

The relationships enable complex interactions and visualizations between different entity classes

LML Class Attributes

- Each class has a set of common attributes and attributes specific to a specific class
- The common ones are: name, number, description. Every entity class has these attributes.
- The LML specification provides the specific attributes for each class
- Add other attributes as needed
- Can also use Characteristics class to provide common or floating attributes

Example of Action Class Specific Attributes

Attribute	Data Type	Description
duration	Number	duration represents the period of time this Action occurs.
percent complete	Percent	percent complete represents the percentage this Action is complete.
start	DateTime	start represents the time when this Action begins.
type	Text	type provides aliases for the entities. For Action these can include: Activity, Capability, Event, Function, Mission, Operational Activity, Program, Service Orchestration, Simulation Workflow, Subprocess, System Function, Task, Training, Use Case, Work Process, Workflow

Class attributes should be added only when they will be used by most instances of that class

LML Relationship Attributes

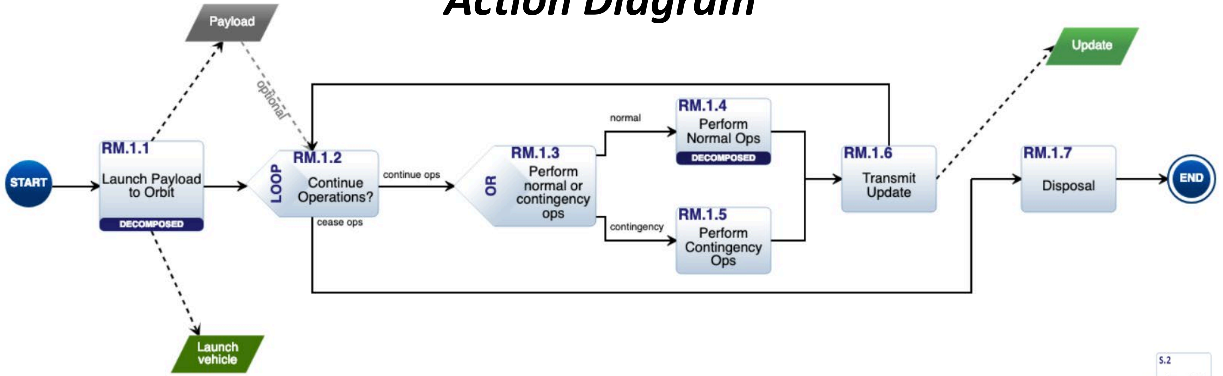
- Attributes on relationships provide a means to modify the relationship information
- The value of the attribute can be different depending on the direction of the relationship (e.g. *related to* vs. *relates*)
- Examples:
  - *context* is used on the *related to/relates* relationship to better describe how the entities relate to one another
  - *amount* is used on the *consumes*, *produces*, and *seizes* relationships to enable resource modeling
  - *trigger* is used on the *receives* relationship to determine if the Action needs to wait for a specific Input/Output entity before executing. This relationship is critical for sequencing Actions

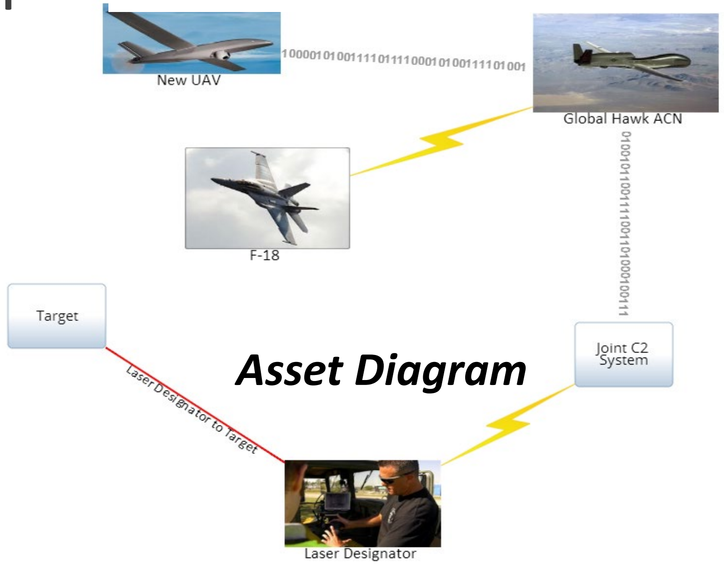
Relationship attributes should be added only when they will be used by most instances of that relationship

LML Diagrams

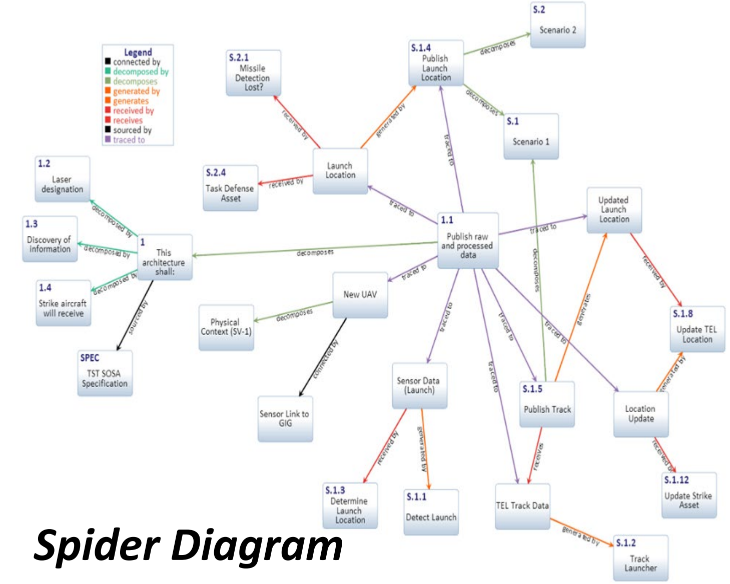
- 3 "Mandatory" diagrams
  - Behavioral (Action)
  - Physical (Asset)
  - Traceability (Spider)
- Additional diagrams types are recommended using common forms, e.g.
  - Timeline
  - Risk Matrix
  - …
- All 9 SysML diagrams can be generated from LML

Action Diagram





Asset Diagram



Spider Diagram

You should be able to visualize the information anyway you need

LML Enables Us to Move from MBSE to DDSE

- We define DDSE as:
  - *the transformation of user needs to requirements for design engineering and the transformation of design engineering data into verified and validated system-level information for decision makers to make better decisions throughout the lifecycle*
- This definition refocuses us on the underlying basis for systems engineering and explicitly identifies the benefit to all stakeholders
  - Design engineers get clear, easy to understand requirements
  - Decision makers get the information they need to make good decisions
- By using a language driven approach, we can focus on the data and less on the form (”model”)
- This data-driven approach gives us a new way to think about systems engineering, thus enabling us to focus on our job

For More on LML

- Go to https://www.lifecyclemodeling.org/
- Download the latest version of the specification at https://www.lifecyclemodeling.org/specification