Lifecycle Modeling Language (LML)
SPECIFICATION

Document URL: http://www.lifecyclemodeling.org/spec/1.4
Current Document URL: http://www.lifecyclemodeling.org/spec/current

October 20, 2022

Table of Contents

1 Specification Information

As the Internet and Information Technology continue to evolve, systems have become more complex and change more rapidly than ever before. When coupled with ever tightening budgets and schedules the future becomes even more challenging for the modern system engineer Development of new systems requires:

- plans that are nimble and responsive to change,
- designs that are easy to understand for all stakeholders,
- architectures that are easy to modify,
- methodologies that are appropriate for the challenges at hand,
- processes that support all stages of the system lifecycle.

Systems engineering approaches, methods and tools have evolved, but they have not kept pace with the rate of change in modern systems. Model Based Systems Engineering, SysML, DODAF, MODAF, and UAF are tremendous steps forward but do not address the entire challenge. Large portions of the development lifecycle are ignored by these approaches. A new approach to analyzing, planning, specifying, designing, building and maintaining modern systems is needed. The Lifecycle Modeling Language (LML) is that approach.

LML was designed with 6 major goals.

1. To be easy to understand,
2. To be easy to extend,
3. To support both functional and object oriented approaches within the same design,
4. To be a language that can be understood by most system stakeholders, not just Systems Engineers,
5. To support systems from cradle to grave,
6. To support both evolutionary and revolutionary changes to system plans and designs over a system's lifecycle.

Most systems engineers recognize that MBSE's ability to evolve, reuse and execute models is a significant improvement over the "document-based" static view of a system. Good models can bridge the gap between written requirements and bending metal or writing code, the thing that is desired versus the thing that is delivered.

LML takes the principles of MBSE beyond development and production and into the conceptual, utilization, support and retirement stages. It provides a robust, easy to understand ontology that allows you to model complex interrelationships between system components and programmatic artifacts, as well as express system information using easy to understand diagrams. "LML was designed to integrate all lifecycle disciplines, such as program management, systems engineering, testing, deployment and maintenance, into a single framework. As a result, LML can be used throughout the lifecycle. LML uses common, everyday language to define its modeling elements such as entity, attribute, schedule, cost, and relationship Its primary modeling constructs are the box (which represents any part of the system that is necessary) and the directed arrow (which depicts a relationship between modeled elements such as "consists of," "derived from," or "costs"). This means that everyone from the least technical stakeholder to most highly skilled end users can model and understand systems LML becomes the Rosetta Stone that allows for easy communication between disparate disciplines across multiple industries.

1.1 Purpose of this Specification

The purpose of LML specifications is to provide a reference for users of the language to understand its goals, concepts and structure and to provide vendors a reference for implementing the language.

1.2 LML Steering Committee

The direction and evolution of this standard is overseen by the LML Steering Committee, which is part of the Lifecycle Modeling Organization (LMO). It consists of expert systems engineers and program managers from industry and academia. Their goal is to ensure LML evolves in such a way that it continues to meet the needs of its users. You may provide comments and suggestions on the LML website (www.lifecyclemodeling.org).

1.3 Documentation Conventions & Terminology

When referencing an entity, outside of a heading, the entity name is always in bold with the first letter capitalized, as in **Action**.

When referencing an attribute, outside of a heading, the attribute name is always in italics with the first letter lower case, as in *control*.

When clarification is needed to better identify an attribute, the entity name should be appended to the end of the attribute name as follows: "(**Entity Name**)"; where "**Entity Name**" would be the name of the entity. For example: *units*(**Characteristic**) and *units*(**Cost**), which clarifies the difference between the *units* attribute of the **Characteristic** entity and the *units* attribute of the **Cost** entity.

When referencing a relationship, outside of a heading, the relationship name is always in bold italics with the first letter lower case, as in ***traced to***.

When referencing an attribute on a relationship, outside of a heading, the attribute name is always underlined with the first letter lower case, as in trigger.

When referencing the Lifecycle Modeling Language (LML) it is referenced via the full name, Lifecycle Modeling Language, or with the abbreviation, LML.

2 Overview of Lifecycle Modeling Language (LML)

The basis for the LML formulation is the classic entity, relationship, and attribute (ERA) meta-meta model. This formulation modifies the classical approach by including attributes on relationships, to provide the adverb, as well as the noun (entity), relationship (verb), and attribute (adjective) language elements. Since LML was designed to translate into object languages, such as UML/SysML, these language elements correspond to classes (entity), relations (relationship), and properties (attribute).

Current Systems Engineering languages tend to add complexity to already complex problems, thus making it more difficult to communicate the underlying issues and develop effective solutions. LML was designed as a simpler language, both in its ontology and visual expressions. This feature makes it easy to understand by all stakeholders throughout the lifecycle. Such a simplified language may not include every "bin" of information needed for a particular domain, which is why LML is extensible. The process by which you can submit an extension is detailed at https://lifecyclemodeling.org/.

For example, UML and SysML use the term Actor to define a part of the system that performs actions. The DoDAF Metamodel 2 uses the word Performer for the same purpose. LML uses the term, **Asset**, but also allows the user to extend the language by defining Actors or Performers. However, we recommend modelers use the *type* attribute for **Assets** instead, to differentiate between these different names. New entities or child entities are recommended only when new attributes and/or relationships are needed. Thus, modelers using LML will know what entity to put something in immediately and can adjust the *type* as needed. We saw this problem in the DoDAF where modelers were often confused by the difference between an operational node and a systems node. When you have two "bins" for the same kind of information, people often get confused. Further examples of this appear in Appendices A and B.

This LML specification also defines common visualizations. For example, the *Risk* entity has a standard Risk Matrix as its basic diagram. Each entity has these kinds of common visualizations, and they need to be as simple as possible to reduce the complexity of the language and make it more understandable to stakeholders. Other visualizations are allowed and encouraged as they aid in expressing the information, which is the real goal of any language visualizations. These can and should be proposed as extensions to the language as well so that other practitioners can benefit from these visualizations.

Ontologies provide a set of defined terms and relationships between those terms to capture the physical, functional, performance, and programmatic aspects of the system. By system,[1] we mean the entire set of processes, people and things which operate for the benefit of people. Common ways of describing such ontologies is entity, relationship, and attribute (ERA). ERA is often used to define database schemas. LML uses the ERA approach but extends it by adding attributes to relationships. The extension reduces the number of relationships needed, just as attributes reduce the number of entities needed.

This section defines the ERAs for LML, thus providing the basic definitions of the data types used to collect information about the system. Furthermore, we describe how inheritance, extensions, limitations and instantiation can be used by tool developers to remain within the guidelines of this standard.

Entity, relationship and attribute have equivalent English language elements: noun, verb, and adjective. With the addition of attributes on the relationship, we also have the equivalent of the adverb. These equivalencies are provided to help explain the semantics of the language.

[1] INCOSE defines a system as a "combination of interacting elements organized to achieve one or more stated purposes."

2.1 Entity (noun)

An entity is something can exist by itself and is uniquely identifiable. In LML, we have defined 12 parent entities (**Action**, **Artifact**, **Asset**, **Characteristic**, **Connection**, **Cost**, **Decision**, **Input/Output**, **Location**, **Risk**, **Statement** and **Time**). The rationale for this set is presented in Section 3. Several child entities have also been defined as they have specific utility in capturing the information needed by the system lifecycle stakeholders. The child entities are **Conduit** (child of **Connection**), **Logical** (child of **Connection**), **Measure** (child of **Characteristic**), **Orbital** (child of **Location**), **Physical** (child of **Location**), **Requirement** (child of **Statement**), **Resource** (child of **Asset**), and **Virtual** (child of **Location**). These child classes have the attributes and relationships of the parents plus additional attributes and relationships that make them unique. More on this "inheritance" of attributes and relationships is discussed in section 2.6. These twenty (20) entities shall be part of any LML-compliant language extension.

Every entity shall have a *name* or *number* or *description* attribute (or combination of the three) to identify it uniquely. The *name* is a word or small collection of words providing an overview of information about the entity. The *number* provides a numerical way to identify the entity. The *description* provides more detail about that entity.

In terms of the English language, an entity is like a noun.

2.2 Attribute (adjective)

An attribute is an inherent characteristic or quality of an entity. It further describes the entity, enhancing its uniqueness. Every attribute shall have a *name* to identify it uniquely within an entity. The *name* is a word or small collection of words providing an overview of information about the attribute. The attribute data type (see 2.5 below) specifies the data associated with the attribute.

Attribute names shall be unique within an entity, but may be used in other entities, such as the example provided earlier: *units*(**Characteristic**) and *units*(**Cost**), to avoid confusion within an entity specification.

In terms of the English language, an attribute is like an adjective.

2.3 Relationship (verb)

A relationship connects entities to each other. In LML, all relations shall be defined in both directions and shall have unique names with the same verb. For example, the standard parent child relationship (used by all LML entities) is ***decomposed by*** and its inverse is ***decomposes***. Each relationship and its inverse shall have unique names. Relationship names shall also be unique across the whole schema. The relationships enable an English reading of the way entities connect. For example, when connecting an **Action** to a **Statement,** LML uses ***traced from*** as the relationship: an **Action** is ***traced from*** a **Statement**. The inverse relation of ***traced from*** is ***traced to***, and thus would be read as: a **Statement** is ***traced to*** an **Action**. Figure 2-1 shows this and the Entity-Relationship Diagram (ERD) for this relationship.

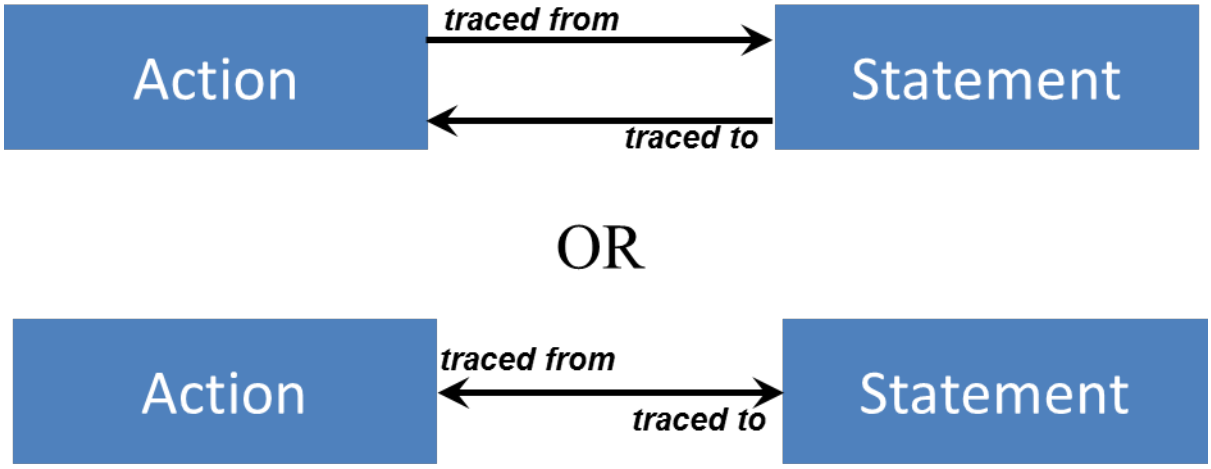

Figure 2-1. Entity Relationship Diagram for the Relationship Between an Action and a Statement.

For relationships within the same entity, such as ***decomposed by*** and ***decomposes*** can be visualized as an ERD as well (see Figure 2-2).

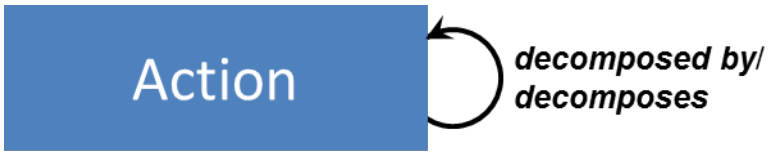

Figure 2-2. Entity Relationship Diagram for the Relationship Between an Action and Itself.

In terms of the English language, a relationship is like a verb.

2.4 Attributes on Relationships (adverb)

The classic ERA modeling does not include attributes on relationships. However, this addition is useful for LML.

Figure 2-3 shows an example of how the attribute on a relationship is depicted in an extended version of the ERD. The attribute on a relationship is shown with a dashed line to the relationship. The attribute on a relationship shall have a unique name for that relationship, but can be used in other relationships, if necessary to enhance communication.

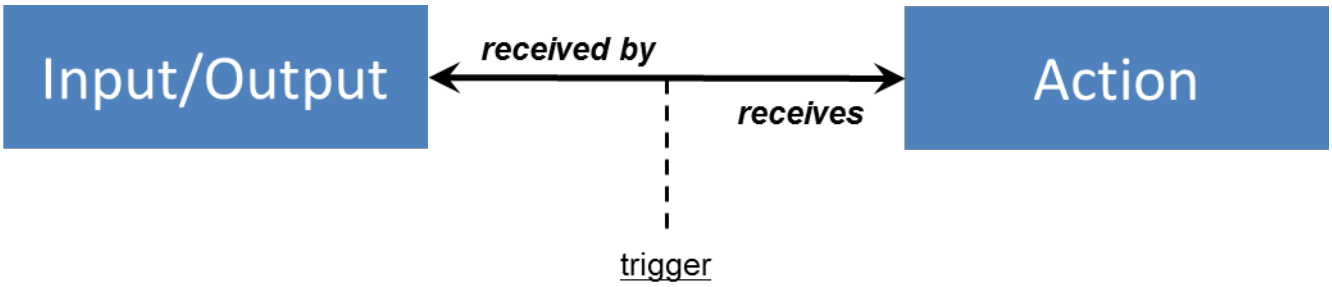

Figure 2-3. Extended Entity Relationship Diagram for the Attribute on the Relationship Between an Action and an Input/Output.

In terms of the English language, an attribute on a relationship is like an adverb.

2.5 Attribute Data Types

For a complete specification, defining which data types are appropriate for the attributes is indispensable. This is because they can vary significantly, and because specification interoperability with other schemas (i.e., translations) would be exceptionally difficult without this. The following sub-sections present the current acceptable set of attribute data types for LML. Extensions (see Section 2.7) may also extend this list of data types as well.

2.5.1 Text

A Text data type represents: a single character, single word, or multiple words.

2.5.2 Number

A Number data type represents any individual real number.

2.5.3 Boolean

A Boolean data type represents the two possible logical values “true” and “false”.

2.5.4 Percent

A Percent data type represents a special case of Number where the value is restricted to values between zero and one hundred.

2.5.5 DateTime

A DateTime data type represents a specific method of storing a given date and time value. Commonly stored as “YYYY-MM-dd hh:mm:ss”. Where “YYYY” is the four digit year, “MM” is the two digit month, “dd” is the two digit day, “hh” is the two digit hour (in twenty-four hours), “mm” is the two digit minute, and “ss” is the two digit second.

2.5.6 URI

An URI date type represents a special case of Text where the value must be a Uniform Resource Identifier. Examples of URI: “C:/Program Files”, “http://www.google.com”, and “test@test.com”.

2.5.7 Enumeration

An Enumeration data type represents a set of defined choices, where the selection of only one choice is permitted. An enumeration must contain at minimum two choices. All choices within the set must be of the same data type (having a set containing both Text data type and Number data type is not permitted). Also, within a set, a repeated choice is not permitted (A set cannot have repeated “Apple” as in: “Apple”, “Orange”, “Apple”, and “Plum”).

2.5.8 GeoPoint

A GeoPoint data type represents a longitude and latitude pair on the surface of a body.

2.6 Inheritance

With LML it is possible to create child entities. The child entity inherits all the attributes and relationships from the parent entity with the EXCEPTION of the attribute *type*; this attribute is overridden by the child entity.

An example of a child entity is **Resource** with parent entity **Asset**.

A child entity will also inherit relationships from the parent. Additional relationships may also be defined for the child.

2.7 Extensions

Extensions for domain-specific needs are allowed and encouraged when necessary. However, it is strongly recommended that you avoid simply aliasing the current ontology, but instead that you identify aliases as another “*type*” of the entity, which is a common attribute for every entity. Tool developers may want to display the entity *type*, rather than the entity *name*, to enhance the communication with these other languages.

The schema user can also apply a **Characteristic** entity in many cases to provide “attributes” of an **Asset** or **Action**, thus making many other extensions unnecessary. We recommend that you explore this option first, before creating an extension.

All extensions must be submitted to the LML Steering Committee for adjudication before they will be recognized as official extensions to LML.

2.8 Instantiation

Actual instantiation of the LML specification will be up to tool vendors. If they find portions of the specification difficult to implement, they can apply to the LML Steering Committee for guidance.

3 LML Ontology

Systems engineers, enterprise architects and program managers have overlapping needs for information and LML provides a basic but comprehensive set of design elements that satisfy all of them. For example, the systems engineer is concerned with optimizing cost, schedule and performance. Performance includes form, fit, and function. Enterprise architects often use the "5WH" model (What, Why, When, Where, Who, and How) to capture their information. The Program Manager is primarily concerned with cost, schedule, tasks, resources, and risks. Table 3-1 shows these various information needs and how LML satisfies them. Each of these entities has unique and common attributes and relationships.

Table 3-1. Comparison Between LML and the Information Needs of Key System Lifecycle Stakeholders

Systems Engineering	Architecture	Program Management	Lifecycle Modeling Language
Cost	(How Much)	Cost	Cost
Schedule	When	Schedule	Time/Action
Performance			
Form	Who	Organization	Asset
	What	Resource	Resource
	Where	Location	Location
	Why	Goal, Objective & Decision	Decision & Statement/Requirement
Function	How	Task	Action
Metric (Fit)		Metric	Characteristic/Measure
Interface			Connection (Conduit) & Input/Output
Risk		Risk	Risk
		Artifact	Artifact

This section provides an overview of LML and then subsequently delves into the detailed attributes, relationships, and attributes on relationships for each entity. This information is fundamental for tool developers planning to instantiate LML.

3.1 LML Entities

Table 3-2 summarizes the LML entities, their parent entity, description and examples (where appropriate) of how they might be used. Note that these examples can be part of the *type* attributes used as aliases for the entity itself.

Table 3-2. Summary of the LML Entities

Entity Name	Parent Entity	Description	Examples
Action	None	An **Action** entity specifies the mechanism by which inputs are transformed into outputs.	Activity, Capability, Event, Function, Process, Task
Artifact	None	An **Artifact** entity specifies a document or other source of information that is referenced by or generated in the knowledgebase.	Document, E-mail, Procedure, Specification
Asset	None	An **Asset** entity specifies an object, person, or organization that performs Actions, such as a system, subsystem, component, or element.	Component, Entity, Service, Sub-system, System
Characteristic	None	A **Characteristic** entity specifies properties of an entity.	Attribute, Category, Power, Role, Size, Weight
Conduit	**Connection**	A **Conduit** entity specifies the means for physically transporting **Input/Output** entities between **Asset** entities. It has limitations (attributes) of capability and latency.	Data Bus, Interface, Pipe
Connection	None	A **Connection** entity specifies the means for relating **Asset** instances to each other.	Abstract entity
Cost	None	A **Cost** entity specifies the outlay or expenditure (as of effort or sacrifice) made to achieve an objective associated with another entity.	Earned Value, Work Breakdown Structure, Actual Cost, Planned Cost
Decision	None	A **Decision** entity specifies a challenge and its resolution.	Major Decision, Challenge, Issue, Problem
Input/Output	None	An **Input/Output** entity specifies the information, data, or object input to, trigger, or output from an **Action**.	Item, Trigger, Information, Data, Energy
Location	None	A **Location** entity specifies where an entity resides.	Abstract entity
Logical	**Connection**	A **Logical** entity represents the abstraction of the relationship between two entities (e.g., **Asset** entities with the type “Entity”)	Has, “is a”, “relates to”
Measure	**Characteristic**	A **Measure** entity specifies properties of measurements and measuring methodologies, including metrics.	Key Performance Parameter (KPP), Measure of Effectiveness (MOE), Measure of Performance (MOP), Metric

Entity Name	Parent Entity	Description	Examples
Orbital	**Location**	An **Orbital** entity specifies a location along an orbit around a celestial body.	Orbit
Physical	**Location**	A **Physical** entity specifies a location on, above, or below the surface.	Map Coordinates
Requirement	**Statement**	A **Requirement** entity identifies a capability, characteristic, or quality factor of a system that must exist for the system to have value and utility to the user.	Functional Requirement, Performance Requirement, Safety Requirement
Resource	**Asset**	A **Resource** entity specifies a consumable or producible **Asset**.	Fuel, Bullets, Missiles, People
Risk	None	A **Risk** entity specifies the combined probability and consequence in achieving objectives.	Cost Risk, Schedule Risk, Technical Risk
Statement	None	A **Statement** entity specifies text referenced by the knowledgebase and usually contained in an **Artifact**.	Need, Goal, Objective, Assumption
Time	None	A **Time** entity specifies a point or period when something occurs or during which an action, asset, process, or condition exists or continues.	Milestone, Phase
Virtual	**Location**	A **Virtual** entity specifies a location within a digital network.	URL

3.2 LML Relationships

The relationships between the entities are provided in the Table 3-3. Note that the same verb is used for the inverse relationships.

* - Implies the inverse relation is present, just not shown.

Note: This matrix does not include the extensions found in the Appendices.

Table 3-3. Summary Table of LML Relationships

	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 (satisfied 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*

3.3 Traceability

Because LML was designed to simplify tracing requirements to their implementation mechanisms (Asset class entities), the primary path for traceability is in Figure 3-1. This diagram does not reflect all the relationships shown in Table 3-3.

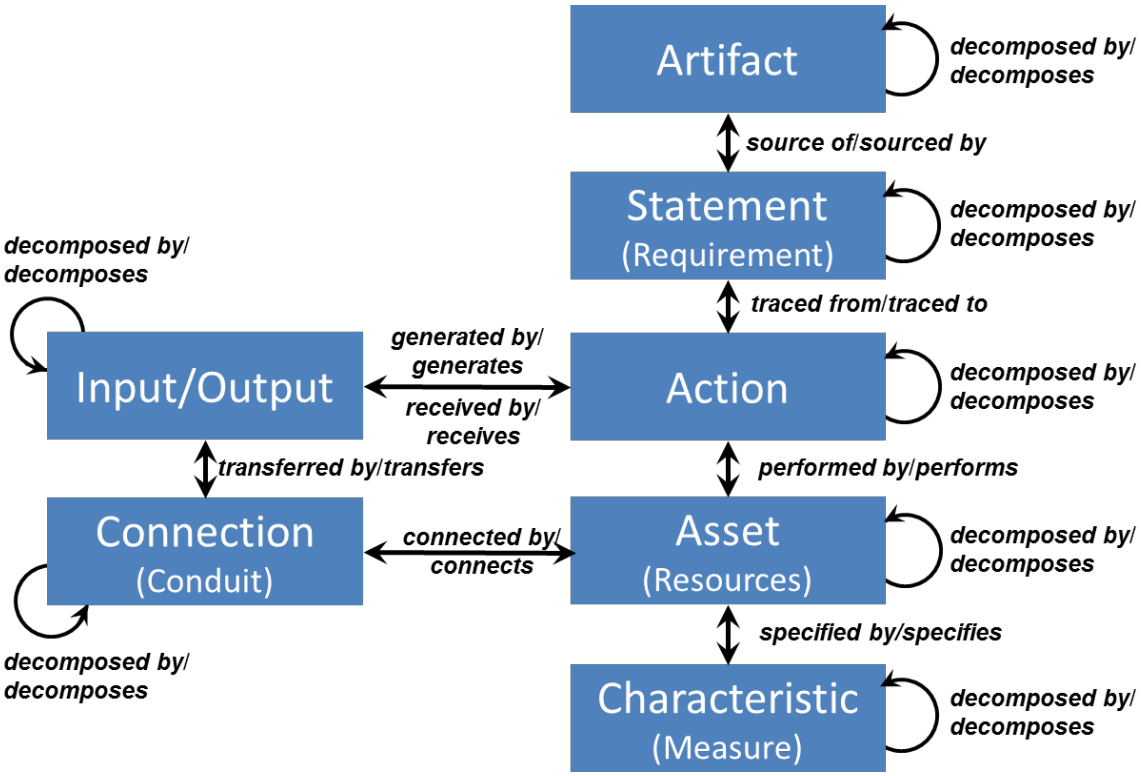

Figure 3-1. Principal Entities and Relationships for Design LML Traceability

3.4 Entity Specifications

The following tables and subsections provide the detailed entity, relationship and attribute specifications for the LML ontology. These specifications shall be used by language users to provide the basis for all LML-related extensions. The **Inverse** provides the name of the relationship from the perspective of the target entity. **Target entities** provide their names on the other side of the relationship.

These attributes are common to all entities.

Attribute	Data Type	Description
name	Text	*name* designates the particular instance of an entity called element.
number	Text	*number* provides the element's place in a hierarchy.
description	Text	*description* captures the text needed to describe this element.

3.4.1 Action

Entity	Parent Entity	Child Entities	Description
Action	None	None	An **Action** entity generates effects and may have pre-conditions before it can be executed. This **Action** can include transforming inputs into outputs. Examples: Process, Discover, Calculate.

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

Relationship	Inverse	Target Entity	Description
causes	***caused by***	**Risk**	***causes*** identifies the **Risk** resulting from this entity.
consumes	***consumed by***	**Resource**	***consumes*** identifies the **Resource** that this **Action** uses. After this **Action** is completed, the amount consumed is not returned to the **Resource**.
	Attribute	Data Type	Description
	amount	Number	amount represents how much of the resource is ***consumed by*** the **Action**. Units are relative to the units selected for the **Resource**.
decomposed by	***decomposes***	***Action***	***decomposed by*** identifies the children of this entity.
decomposes	***decomposed by***	***Action***	***decomposes*** identifies the parent of this entity.
enables	***enabled by***	***Decision***	***enables*** identifies the *Decision* that is empowered by this entity.
generates	***generated by***	***Input/Output***	***generates*** identifies the **Input/Output** that this **Action** transforms.
incurs	***incurred by***	***Cost***	***incurs*** identifies the **Cost** value for this entity.
located at	***locates***	***Location***	***located at*** identifies the **Location** where this entity exists.

Relationship	Inverse	Target Entity	Description
mitigates	***mitigated by***	***Risk***	***mitigates*** identifies the **Risk** that this entity plan alleviates.
occurs	***occurred by***	***Time***	***occurs*** identifies the **Time** (or timespan) in which this entity happens.
performed by	***performs***	**Asset**	***performed by*** identifies the **Asset** that executes this **Action**.
produces	***produced by***	**Resource**	***produces*** identifies the **Resource** that is created by this **Action**. **Resources** are produced when the execution of the **Action** completes.
	Attribute	Data Type	Description
	amount	Number	amount represents how much of the **Resource** is ***produced by*** the **Action**. Units are relative to the units selected for the **Resource**.
receives	***received by***	**Input/Output**	***receives*** identifies the **Input/Output** that is taken in by this **Action**.
	Attribute	Data Type	Description
	trigger	Boolean	trigger represents this relation as an enabling requirement for the **Action**. An **Action** begins execution when it has received control enablement, all of its triggers have arrived, and its necessary resources are available.
references	***referenced by***	**Artifact**	***references*** identifies the **Artifact** that specifies and/or enhances the definition of this entity.
related to	***relates***	**Action**	***related to*** identifies the entity that ties in a peer-to-peer way with this entity.
	Attribute	Type	Description
	context	Text	context represents a description of this relation.
relates	***related to***	**Action**	***relates*** identifies the entity that ties in a peer-to-peer way with this entity.
	Attribute	Type	Description
	context	Text	context represents a description of this relation.
resolves	***resolved by***	**Risk**	***resolves*** identifies the **Risk** that is closed by this entity.
results in	***result of***	**Decision**	***results in*** identifies the **Decision** that is caused by this entity.

Relationship	Inverse	Target Entity	Description
seizes	***seized by***	**Resource**	***seizes*** identifies the **Resource** that this **Action** uses. After this **Action** has completed the **Resource** is released for use by other **Actions**.
	Attribute	Type	Description
	amount	Number	amount represents how much of the resource is ***captured by*** the **Action**. Units are relative to the units selected for the **Resource**.
specified by	***specifies***	**Characteristic**	***specified by*** identifies a **Characteristic** that provides further information about this entity.
traced from	***traced to***	**Statement**	***traced from*** identifies a **Statement** that is accredited to this entity.

3.4.2 Artifact

Entity	Parent Class	Child Entities	Description
Artifact	None	None	An **Artifact** entity specifies a document or other source of information that is referenced by or generated in the knowledgebase. Example: Requirements Report.

Attribute	Type	Description
date published	DateTime	*date published* represents the date when this **Artifact** was accessed (webpage), published, or uploaded to the knowledgebase.
file	URI	The *file* that represents this **Artifact**.
type	Text	*type* provides aliases for the entities. For **Artifact** these can include: Briefing, Change Notice, Change Request, Concept of Operations, Directive, Doctrine, Document, E-Mail, Guidance, Instruction, Interface Control Document, Interface Requirements Specification, Manual, Matrix, Meeting Minutes, Memorandum, Mitigation Plan, Model, Operational Concept, Policy, Procedure, Protocol, Proposal, Regulation, Requirements Document, Request for Proposals, Software Requirements Specification, Standard, System Requirements Document, System/Segment Design Document, System/Segment Specification, Test & Evaluation Plan, Test & Evaluation Report, Text Message, Trade Study, White Paper

Relationship	Inverse	Target Class(es)	Description
causes	***caused by***	**Risk**	***causes*** identifies the *Risk* resulting from this entity.
decomposed by	***decomposes***	**Artifact**	***decomposed by*** identifies the children of this entity.
decomposes	***decomposed by***	**Artifact**	***decomposes*** identifies the parent of this entity.
defines protocol for	***defined protocol by***	**Conduit**	***defines protocol for*** identifies the **Conduit** that uses the standard identified in this **Artifact**.
enables	***enabled by***	**Decision**	***enables*** identifies the **Decision** that is empowered by this entity.
incurs	***incurred by***	**Cost**	***incurs*** identifies the **Cost** value for this entity.
located at	***locates***	**Location**	***located at*** identifies the **Location** where this entity exists.
mitigates	***mitigated by***	**Risk**	***mitigates*** identifies the **Risk** that this entity plan alleviates.

occurs	***occurred by***	**Time**	***occurs*** identifies the **Time** (or timespan) this entity happens.
referenced by	***references***	**Action, Artifact, Asset, Characteristic, Connection, Cost, Decision, Input/Output, Risk, Statement**	***referenced by*** identifies the entity that specified and/or enhanced its definition by this **Artifact**.
related to	***relates***	**Artifact**	***related to*** identifies the entity that ties in a peer-to-peer way with this entity.
	Attribute	Type	Description
	context	Text	context represents a description of this relation.
relates	***related to***	**Artifact**	*relates* identifies the entity that ties in a peer-to-peer way with this entity.
	Attribute	Type	Description
	context	Text	context represents a description of this relation.
resolves	***resolved by***	**Risk**	***resolves*** identifies the **Risk** that is closed by this entity.
results in	***result of***	**Decision**	***results in*** identifies the **Decision** that is caused by this entity.
source of	***sourced by***	**Statement**	***source of*** identifies the **Statement** that is contained in this **Artifact**.
specified by	***specifies***	**Characteristic**	***specified by*** identifies a **Characteristic** that provides further information about this entity.
traced from	***traced to***	**Statement**	***traced from*** identifies a **Statement** that is accredited to this entity.

3.4.3 Asset

Entity	Parent Entity	Child Entities	Description
Asset	None	**Resource**	An *Asset* entity specifies an object, person, or organization that used to create value and perform **Actions**, such as a system, subsystem, component, or element. Example: Infrared Sensor, Accounting Department, Internal Revenue Service

Attribute	Type	Description
type	Text	*type* provides aliases for the entities. For **Asset** these can include: Architecture, Assembly, Component, Context, CSC, CSCI, CSU, Element, Environment, External System, Facility, Hardware, Human, HW Element, HWCI, Infrastructure, LRU, Materiale, Operational Element, Organization, Part, Performer, Personnel, Segment, Service, Software, Subassembly, Subsystem, System, System Instantiation, Test Equipment, Test Software, Unit

Relationship	Inverse	Target Entity(es)	Description
causes	***caused by***	**Risk**	***causes*** identifies the **Risk** resulting from this entity.
connected by	***connects to***	**Connection**	***connected by*** identifies the **Connection** that adjoins this **Asset**.
	Attribute	Type	Description
	origin	Boolean	origin represents if the **Asset** is the origin of the **Connection**. This attribute enables unidirectional (one or the other **Asset**'s relationship origin is false), in additional to the default bi-directional flow (both origins are set to true as a default).
	multiplicity	Text	multiplicity, also called cardinality, represents the number of relationships (one to many, many to one, etc.) that can occur.
decomposed by	***decomposes***	**Asset**	***decomposed by*** identifies the children of this entity.
decomposes	***decomposed by***	**Asset**	***decomposes*** identifies the parent of this entity.
enables	***enabled by***	**Decision**	***enables*** identifies the **Decision** that is empowered by this entity.
incurs	***incurred by***	**Cost**	***incurs*** identifies the **Cost** value for this entity.
located at	***locates***	**Location**	***located at*** identifies the **Location** where this entity exists.

made	***made by***	**Decision**	***made*** identifies the **Decision** that this **Asset** decided.
mitigates	***mitigated by***	**Risk**	***mitigates*** identifies the **Risk** that this entity plan alleviates.
occurs	***occurred by***	**Time**	***occurs*** identifies the **Time** (or timespan) in which this entity happens.
orbited by	***orbits***	**Asset**	***orbited by*** identifies **Asset** entity that acts as a satellite to this **Asset**.
orbits	***orbited by***	**Asset**	***orbits*** identifies the **Asset** that this **Asset** moves around.
performs	***performed by***	**Action**	***performs*** identifies the **Action** that is executed by this *Action*.
references	***referenced by***	**Artifact**	***references*** identifies the **Artifact** that specifies and/or enhances the definition of this entity.
related to	***relates***	**Asset**	***related to*** identifies the entity that ties in a peer-to-peer way with this entity.
	Attribute	Type	Description
	context	Text	context represents a description of this relation.
relates	***related to***	**Asset**	***relates*** identifies the entity that ties in a peer-to-peer way with this entity.
	Attribute	Type	Description
	context	Text	context represents a description of this relation.
resolves	***resolved by***	**Risk**	***resolves*** identifies the **Risk** that is closed by this entity.
responds to	***responded by***	**Decision**	***responds to*** identifies the **Decision** that is acted on by this **Asset** (usually a person or organization).
	Attribute	Type	Description
	responsibility	Enumeration [**Primary**, Secondary]	responsibility represents the **Asset** that has the responsibility for resolving the **Decision**.
results in	***result of***	**Decision**	***results in*** identifies the **Decision** that is caused by this entity.
specified by	***specifies***	**Characteristic**	***specified by*** identifies a **Characteristic** that provides further information about this entity.
traced from	***traced to***	**Statement**	***traced from*** identifies a **Statement** that is accredited to this entity.

3.4.4 Characteristic

Entity	Parent Entity	Child Entities	Description
Characteristic	None	**Measure**	A **Characteristic** entity specifies or captures properties of an entity. Examples: Blue, no heavier than 2 oz, accurate to within 1%

Attribute	Type	Description
units	Text	*units* represents this **Characteristic**'s units used to measure the value, such as pounds or miles per hour.
value	Text	*value* represents this **Characteristic**'s current value.
type	Text	*type* provides aliases for the entities. For **Characteristic** these can include: Condition, Data Element, Environmental, Functional, Performance, Physical, Scenario, Security, Verification Category

Relationship	Inverse	Target Entity(es)	Description
causes	***caused by***	**Risk**	***causes*** identifies the **Risk** resulting from this entity.
decomposed by	***decomposes***	**Characteristic**	***decomposed by*** identifies the children of this entity.
decomposes	***decomposed by***	**Characteristic**	***decomposes*** identifies the parent of this entity.
enables	***enabled by***	**Decision**	***enables*** identifies the **Decision** that is empowered by this entity.
incurs	***incurred by***	**Cost**	***incurs*** identifies the **Cost** value for this entity.
located at	***locates***	**Location**	***located at*** identifies the **Location** where this entity exists.
mitigates	***mitigated by***	**Risk**	***mitigates*** identifies the **Risk** that this entity plan alleviates.
occurs	***occurred by***	**Time**	***occurs*** identifies the **Time** (or timespan) in which this entity happens.
references	***referenced by***	**Artifact**	***references*** identifies the **Artifact** that specifies and/or enhances the definition of this entity.
related to	***relates***	**Characteristic**	***related to*** identifies the entity that ties in a peer-to-peer way with this entity.
	Attribute	Type	Description
	context	Text	context represents a description of this relation.

relates	***related to***	**Characteristic**	***relates*** identifies the entity that ties in a peer-to-peer way with this entity.
	Attribute	Type	Description
	context	Text	*context* represents a description of this relation.
resolves	***resolved by***	**Risk**	***resolves*** identifies the **Risk** that is closed by this entity.
results in	***result of***	**Decision**	***results in*** identifies the **Decision** that is caused by this entity.
specified by	***specifies***	**Characteristic**	**specified by** identifies a **Characteristic** that provides further information about this entity.
specifies	***specified by***	**Action, Artifact, Asset, Characteristic, Connection, Cost, Decision, Input/Output, Location, Risk, Statement, Time**	***specifies*** identifies an entity that this **Characteristic** provides further information about.
traced from	***traced to***	**Statement**	***traced from*** identifies a **Statement** that is accredited to this entity.

3.4.5 Conduit (Connection)

Entity	Parent Entity	Child Entities	Description
Conduit	**Connection**	None	A **Conduit** entity specifies the connection between Assets and has capacity and latency, which carries an Input/Output. Examples: SpaceWire, Bluetooth, railway

Attribute	Type	Description
capacity	Number	*capacity* represents the maximum rate supported by the **Conduit**. (Used in simulation as to add time delay by dividing the **Input/Output** size with the *capacity* value.)
latency	Number	*latency* represents the time required to transmit the information or **Input/Output** entity over this **Conduit**. This value does not factor in any delays due to capacity limitations.
units	Text	*units* represents this **Conduit**'s units used to measure the capacity, such as bits per second or gallons per minute.
type	Text	*type* provides aliases for the entities. For **Conduit** these can include: Cable, Downlink, Human-in-the-Loop, Human Machine Interface, Interface, Landline, Link, Needline, Network, Pipe, RF - Satcom, RF - Terrestrial, Roadway, Service Interface, Uplink, Wireless.

Relationship	Inverse	Target Entity(es)	Description
decomposed by	***decomposes***	**Conduit**	***decomposed by*** identifies the children of this entity.
decomposes	***decomposed by***	**Conduit**	***decomposes*** identifies the parent of this entity.
defined protocol by	***defines protocol for***	**Artifact**	***defined protocol by*** identifies the **Artifact** that contains the standard used by this **Conduit**.
transfers	***transferred by***	**Input/Output**	***transfers*** identifies the **Input/Output** that is passed by this **Conduit**.

All attributes and relationships from **Connection** are inherited by the **Conduit** entities.

3.4.6 Connection

Entity	Parent Entity	Child Entities	Description
Connection	None	**Conduit**, **Logical**	A **Connection** entity specifies the mechanism relating Assets. This is an abstract class.

Relationship	Inverse	Target Entity(es)	Description
causes	***caused by***	**Risk**	***causes*** identifies the **Risk** resulting from this entity.
connects to	***connected by***	**Asset**	***connects to*** identifies the **Asset** that this *Connection* adjoins.
	Attribute	Type	Description
	origin	Boolean	origin represents if the **Asset** is the origin of the *Connection*. This attribute enables unidirectional (one or the other **Asset***'s* relationship origin is False), in additional to the default bi-directional flow (both origins are set to True by default).
	multiplicity	Text	multiplicity, also called cardinality, represents the number of relationships (one to many, many to one, etc.) that can occur.
enables	***enabled by***	**Decision**	***enables*** identifies the **Decision** that is empowered by this entity.
incurs	***incurred by***	**Cost**	***incurs*** identifies the **Cost** value for this entity.
joined by	***joins***	**Connection**	***joined by*** identifies the **Connection** that connects to this **Connection**. ***joined by*** implies the end of the relationship.
	Attribute	Type	Description
	bidirectional	Boolean	bidirectional represents if the connection between two **Connections** is bidirectional. If the **Connection** is not bidirectional, the *Connection* that has the relation ***joins*** is the start and the Connection with ***joined by*** is the end.

joins	***joined by***	**Connection**	***joins*** identifies the **Connection** that is connected to this **Connection**. *Joins* implies the start of this relationship.
	Attribute	Type	Description
	bidirectional	Boolean	bidirectional represents if the connection between two **Connection**s is bidirectional. If the **Connection** is not bidirectional, the **Connection** that has the relation ***joins*** is the start and the Connection with ***joined by*** is the end.
located at	***locates***	**Location**	***located at*** identifies the **Location** where this entity exists.
mitigates	***mitigated by***	**Risk**	***mitigates*** identifies the **Risk** that this entity plan alleviates.
occurs	***occurred by***	**Time**	***occurs*** identifies the **Time** (or timespan) this entity happens.
references	***referenced by***	**Artifact**	***references*** identifies the **Artifact** that specifies and/or enhances the definition of this entity.
related to	***relates***	**Connection**	***related to*** identifies the entity that ties in a peer-to-peer way with this entity.
	Attribute	Type	Description
	context	Text	context represents a description of this relation.
relates	***related to***	**Connection**	***relates*** identifies the entity that ties in a peer-to-peer way with this entity.
	Attribute	Type	Description
	context	Text	context represents a description of this relation.
resolves	***resolved by***	**Risk**	***resolves*** identifies the **Risk** that is closed by this entity.
results in	***result of***	**Decision**	***results in*** identifies the **Decision** that is caused by this entity.
specified by	***specifies***	**Characteristic**	***specified by*** identifies a **Characteristic** that provides further information about this entity.
traced from	***traced to***	**Statement**	***traced from*** identifies a **Statement** that is accredited to this entity.

3.4.7 Cost

Entity	Parent Entity	Child Entities	Description
Cost	None	None	A **Cost** entity specifies the outlay or expenditure (as of effort or sacrifice) made to achieve an objective associated with another entity. Examples: $100, 6 man-hours

Attribute	Type	Description
amount	Number	*amount* represents this **Cost**'s value.
category	Enumeration [Procurement, MILCON, MILPERS, O&M, RDT&E, SCN, **N/A**]	*category* represents the part of the lifecycle for which the money is used (commonly called Color of Money). This matches the Federal Government (DoD) cost phases.
contract type	Enumeration [CPFF, CPAF, CPIF, FFP-Completion, FFP-LOE, T&M, **N/A**]	*contract type* represents this **Cost** way to identify the reimbursement structure (i.e., CPFF vs T&M).
rate	Enumeration [**Fixed**, Per Hour]	*rate* represents this **Cost'**s billing rate.
units	Text	*units* represents this **Cost'***s* units used to measure the amount, such as $ or Euro.
type	Text	*type* provides aliases for the entities. For **Cost** these can include: Actual, Earned Value, Estimated, Overrun, Plan

Relationship	Inverse	Target Entity(es)	Description
causes	***caused by***	**Risk**	***causes*** identifies the **Risk** resulting from this entity.
decomposed by	***decomposes***	**Cost**	***decomposed by*** identifies the children of this entity.
decomposes	***decomposed by***	**Cost**	***decomposes*** identifies the parent of this entity.
enables	***enabled by***	**Decision**	***enables*** identifies the **Decision** that is empowered by this entity.
incurred by	***incurs***	**Action, Artifact, Asset, Characteristic, Connection, Decision, input/Output, Risk, Statement**	***incurred by*** identifies the entity that has this **Cost** value.

located at	***locates***	**Location**	***located at*** identifies the **Location** where this entity exists.
mitigates	***mitigated by***	**Risk**	***mitigates*** identifies the **Risk** that this entity plan alleviates.
occurs	***occurred by***	**Time**	***occurs*** identifies the **Time** (or timespan) in which this entity happens.
references	***referenced by***	**Artifact**	***references*** identifies the **Artifact** that specifies and/or enhances the definition of this entity.
related to	***relates***	**Cost**	***related to*** identifies the entity that ties in a peer-to-peer way with this entity.
	Attribute	Type	Description
	context	Text	context represents a description of this relation.
relates	***related to***	**Cost**	***relates*** identifies the entity that ties in a peer-to-peer way with this entity.
	Attribute	Type	Description
	context	Text	context represents a description of this relation.
resolves	***resolved by***	**Risk**	***resolves*** identifies the **Risk** that is closed by this entity.
results in	***result of***	**Decision**	***results in*** identifies the **Decision** that is caused by this entity.
specified by	***specifies***	**Characteristic**	***specified by*** identifies a **Characteristic** that provides further information about this entity.
traced from	***traced to***	**Statement**	***traced from*** identifies a **Statement** that is accredited to this entity.

3.4.8 Decision

Entity	Parent Entity	Child Entities	Description
Decision	None	None	A **Decision** entity specifies an opportunity to make a choice. Examples: slip schedule by two months, accept risk, hire ten new employees

Attribute	Type	Description
assumptions	Text	*assumptions* represents facts that are used as a basis for this **Decision**. These assumptions can be taken for granted by the decision maker.
justification	Text	*justification* represents the reason and context for making this **Decision**. For additional justification, the user may want to apply the ***enabled by*** relationship to link it to other **Statements**.
status	Enumeration [**Open**, Closed]	*status* represents the state of the **Decision** (open or closed).
type	Text	*type* provides aliases for the entities. For **Decision** these can include: Challenge, Issue, Problem.

Relationship	Inverse	Target Entity(es)	Description
alternative	***alternative of***	**Statement**	***alternative*** identifies the **Statement** that is a potential choice for this **Decision**.
	Attribute	Type	Description
	choice	Boolean	choice represents if this alternative was the choice selected. If the **Decision** is still open, none of these alternatives would be True.
causes	***caused by***	**Risk**	***causes*** identifies the **Risk** resulting from this entity.
date resolved by	***date resolves***	**Time**	***date resolved by*** identifies the **Time** when this **Decision** was closed. For open issues, this attribute can be left blank.
decomposed by	***decomposes***	**Decision**	***decomposed by*** identifies the children of this entity.
decomposes	***decomposed by***	**Decision**	***decomposes*** identifies the parent of this entity.
decision due	***decided by***	**Time**	***decision due*** identifies the **Time** when this **Decision** is scheduled to be closed.

enabled by	***enables***	**Action, Artifact, Asset, Characteristic, Connection, Cost, Input/Output, Risk, Statement**	***enabled** by* identifies the entity that empowers this **Decision** to be made.
incurs	***incurred by***	**Cost**	***incurs*** identifies the **Cost** value for this entity.
located at	***locates***	**Location**	***located at*** identifies the **Location** where this entity exists.
made by	***made***	**Asset**	***made by*** identifies the **Asset** that made this *Decision*.
mitigates	***mitigated by***	**Risk**	***mitigates*** identifies the **Risk** that this entity plan alleviates.
occurs	***occurred by***	**Time**	***occurs*** identifies the **Time** (or timespan) in which this entity happens.
references	***referenced by***	**Artifact**	***references*** identifies the **Artifact** that specifies and/or enhances the definition of this entity.
related to	***relates***	**Decision**	***related to*** identifies the entity that ties in a peer-to-peer way with this entity.
	Attribute	Type	Description
	context	Text	context represents a description of this relation.
relates	***related to***	**Decision**	***relates*** identifies the entity that ties in a peer-to-peer way with this entity.
	Attribute	Type	Description
	context	Text	context represents a description of this relation.
resolves	***resolved by***	**Risk**	***resolves*** identifies the *Risk* that is closed by this entity.
responded by	***responds to***	**Asset**	***responded by*** identifies the **Asset** (usually a person or organization) that acts on this **Decision**.
	Attribute	Type	Description
	responsibility	Enumeration [**Primary**, Secondary]	*responsibility* represents the **Asset** that has the responsibility for resolving the **Decision**.

result of	***results in***	**Action,** **Artifact,** **Asset,** **Characteristic,** **Connection,** **Cost,** **Decision,** **Input/Output,** **Risk,** **Statement**	***result of*** identifies the entity that caused this **Decision**.
results in	***result of***	**Decision**	***results in*** identifies the **Decision** that is caused by this entity.
specified by	***specifies***	**Characteristic**	***specified by*** identifies a **Characteristic** that provides further information about this entity.
traced from	***traced to***	**Statement**	***traced from*** identifies a **Statement** that this **Decision** comes from.

3.4.9 Input/Output

Entity	Parent Entity	Child Entities	Description
Input/Output	None	None	An **Input/Output** entity specifies the matter, energy, and/or information input to, triggers (controls), or output from an Action. Examples: Nickel/gumball, gasoline/horsepower, investment/return

Attribute	Type	Description
size	Number	*size* represents the amount or proportion of this **Input/Output**, such as 100 as 100 Gigabytes or number of entities (e.g., 10 as in 10 tokens).
units	Text	*units* represents this **Input/Output**'*s* units used to measure the size such as Gigabytes or tokens.
type	Text	*type* provides aliases for the entities. For **Input/Output** these can include: Analog, Data, Digital, Event, Information, Item, Mixed, Physical, Product, Verbal.

Relationship	Inverse	Target Entity(es)	Description
causes	***caused by***	**Risk**	***causes*** identifies the **Risk** resulting from this entity.
decomposed by	***decomposes***	**Input/Output**	***decomposed by*** identifies the children of this entity.
decomposes	***decomposed by***	**Input/Output**	***decomposes*** identifies the parent of this entity.
enables	***enabled by***	**Decision**	***enables*** identifies the **Decision** that was empowered by this entity.
generated by	***generates***	**Action**	***generated by*** identifies the **Action** that transformed this **Input/Output**.
incurs	***incurred by***	**Cost**	***incurs*** identifies the **Cost** value for this entity.
located at	***locates***	**Location**	***located at*** identifies the **Location** where this entity exists.
mitigates	***mitigated by***	**Risk**	***mitigates*** identifies the **Risk** that this entity plan alleviates.
occurs	***occurred by***	**Time**	***occurs*** identifies the **Time** (or timespan) during which this entity happens.
received by	***receives***	**Action**	***received by*** identifies the **Action** that takes in

			this *Input/Output*.
	Attribute	Type	Description
	trigger	Boolean	trigger represents this relation as an enabling requirement for the **Action**. An **Action** begins execution when it has received control enablement, all of its triggers have arrived, and its necessary resources are available.
references	***referenced by***	**Artifact**	***references*** identifies the **Artifact** that specifies and/or enhances the definition of this entity.
related to	***relates***	**Input/Output**	***related to*** identifies the entity that ties in a peer-to-peer way with this entity.
	Attribute	Type	Description
	context	Text	context represents a description of this relation.
relates	***related to***	**Input/Output**	***relates*** identifies the entity that ties in a peer-to-peer way with this entity.
	Attribute	Type	Description
	context	Text	context represents a description of this relation.
resolves	***resolved by***	**Risk**	***resolves*** identifies the **Risk** that is closed by this entity.
results in	***result of***	**Decision**	***results in*** identifies the **Decision** that is caused by this entity.
specified by	***specifies***	**Characteristic**	***specified by*** identifies a **Characteristic** that provides further information about this entity.
traced from	***traced to***	**Statement**	***traced from*** identifies a **Statement** that is accredited to this entity.
transferred by	***transfers***	**Connection**	***transferred by*** identifies the **Connection** that passes this **Input/Output**.

3.4.10 Location

Entity	Parent Entity	Child Entities	Description
Location	None	**Orbital, Physical, Virtual**	A **Location** entity specifies where an entity resides. Abstract Class.

Relationship	Inverse	Target Entity(es)	Description
decomposed by	***decomposes***	**Location**	***decomposed by*** identifies the children of this entity.
decomposes	***decomposed by***	**Location**	***decomposes*** identifies the parent of this entity.
locates	***located at***	**Action, Artifact, Asset, Characteristic, Connection, Cost, Decision, Input/Output, Statement**	***locates*** identifies an entity that exists at this **Location**.
mitigates	***mitigated by***	**Risk**	***mitigates*** identifies the **Risk** that this entity plan alleviates.
occurs	***occurred by***	**Time**	***occurs*** identifies the **Time** (or timespan) during which this entity happens.
related to	***relates***	**Location**	***related to*** identifies the entity that ties in a peer-to-peer way with this entity.
	Attribute	Type	Description
	context	Text	context represents a description of this relation.
relates	***related to***	**Location**	***relates*** identifies the entity that ties in a peer-to-peer way with this entity.
	Attribute	Type	Description
	context	Text	context represents a description of this relation.
specified by	***specifies***	**Characteristic**	***specified by*** identifies a **Characteristic** that provides further information about this entity.
traced from	***traced to***	**Statement**	***traced from*** identifies a **Statement** that is accredited to this entity.

3.4.11 Logical (Connection)

Entity	Parent Entity	Child Entities	Description
Logical	**Connection**	None	A **Logical** entity specifies relationship between Assets. It is primarily used in database schema development and entity-relationship diagrams.

Relationship	Inverse	Target Entity(es)	Description
Note for specified by (see description)	***specifies***	**Characteristic**	***specified by*** identifies a **Characteristic** that provides further information about this entity. This relationship is used to specify an attribute on a relationship in an extended Entity-Relationship-Attribute (ERA) schema.

All attributes and relationships from **Connection** are inherited by the **Logical** entities.

3.4.12 Measure (Characteristic)

Entity	Parent Entity	Child Entities	Description
Measure	**Characteristic**	None	A **Measure** entity specifies the set of measurements used to provide managers, system developers, and systems engineers with insight into the system definition, and the analysis of technical solutions with respect to performance, cost, and risk. Examples: 19 inches, 22 grams, 1.21 gigawatts

Attribute	Type	Description
improvement direction	Enumeration [N/A, **Positive**, Negative]	*improvement direction* represents the direction in which metric improvement occurs. It is the direction from the *threshold value* to the *objective value*.
objective value	Text	*objective value* represents the goal for this **Measure**.
projected value	Text	*projected value* represents this **Measure**'s expected value to be achieved with existing resources.
threshold value	Text	*threshold value* represents the minimum acceptable value for this **Measure**.
tolerance	Text	*tolerance* represents the percentage of the value that forms the positive and negative tolerance bands. *tolerance* is used when *value* represents the planned measure value at a given time.
type	Text	*type* provides aliases for the entities. For **Measure** these can include: Critical Operational Issue (COI), Key Performance Parameter (KPP), Mean Time Between Failures (MTBF), Measure of Effectiveness (MOE), Measure of Performance (MOP), Metric, Technical Performance Measure (TPM).

All attributes and relationships from **Characteristic** are inherited by the **Measure** entities.

3.4.13 Orbital (Location)

Entity	Parent Entity	Child Entities	Description
Orbital	**Location**	None	An **Orbital** entity specifies a location (ephemeris) along an orbit around a celestial body. Note that this includes transfer orbits as well. Examples: Mars orbit, transfer to lunar orbit

Attribute	Type	Description
Inclination	Number	*inclination* represents the angle between the orbital plane and a reference plane, such as the equatorial plane for Earth. The *Inclination* must be specified with the *longitude of ascending node* to characterize the orbital plane.
Semimajor Axis	Number	*semimajor axis* represents the one half of the length of the longest diameter of the orbital ellipse. The *semimajor axis* must be specified with the *eccentricity* to characterize the orbital ellipse shape.
Longitude of Ascending Node	Number	*longitude of ascending node* represents the angle from the *origin of longitude* to the ascending node, measured in the *reference plane*. For the Earth, the *origin of longitude* is typically the Prime Meridian. The *longitude of ascending node* must be specified with the *Inclination* to characterize the orbital plane.
Reference Plane	Text	*reference plane* represents the reference plane from where the Inclination angle will be calculated.
Argument of Periapsis	Number	*argument of periapsis* represents the angle between the Periapsis and the ascending node as measured in the orbital plane in the direction of motion.
Origin of Longitude	Text	*origin of longitude* represents the reference meridian from where the *longitude of ascending node* will be calculated.
Mean Anomaly	Number	*mean anomaly* represents the proportion of orbital area swept since the last *periapsis* at the specified time. It is used to define the position of the orbiting **Asset** along the orbital ellipse.
Apoapsis	Number	*apoapsis* represents the point of greatest distance from the celestial body being orbited. For Earth, the term is apogee. For the Sun the term commonly used is aphelion. The *apoapsis* must be specified with the *periapsis* to characterize the orbital ellipse shape.
Periapsis	Number	*periapsis* represents the point of closest distance from the celestial body being orbited. For Earth, the term is perigee. For the Sun the term commonly used is perihelion. The *periapsis* must be specified with the *apoapsis* to characterize the orbital ellipse shape.
Eccentricity	Number	*eccentricity* represents the amount the orbit deviates from a perfect circle (0 being perfectly circular and 1 is a parabola - no longer a closed orbit). The *eccentricity* must be specified with the *semimajor axis* to characterize the orbital ellipse shape.

All attributes and relationships from **Location** are inherited by the **Orbital** entities.

3.4.14 Physical (Location)

Entity	Parent Entity	Child Entities	Description
Physical	**Location**	None	A **Physical** entity specifies a location on, below, or above the surface of a celestial body. Examples: North Pole, Camp Lejeune

Attribute	Type	Description
type	Text	*type* provides aliases for the entities. For **Physical** these can include: Geospatial Location, Map Coordinates

Attribute	Data Type	Description
address	Text	*address* represents this **Location's** complete address.
altitude/depth	Number	*altitude/depth* represents the distance above (positive values) or below (negative values) the surface.
coordinates	GeoPoint	*coordinates* represents the coordinate points for this **Location** (GPS or other system).
units	Text	*units* represents the units used to measure the *altitude/depth* of the **Physical** location.

Note that this entity definition uses Cartesian Coordinates (x, y, z). It may be desirable to establish other coordinate systems (e.g., Cylindrical, Spherical, etc.) for other implementations.

All attributes and relationships from **Location** are inherited by the **Physical** entities.

3.4.15 Requirement

Entity	Parent Entity	Child Entities	Description
Requirement	**Statement**	None	A **Requirement** entity identifies a capability, characteristic, or quality factor of a system that must exist for the system to have value and utility to the user. Example: pump shall weigh no more than 1.2 kilograms.

Attribute	Type	Description
rationale	Text	*rationale* provides a place to capture the reason(s) behind this **Requirement**.
type	Text	*type* provides aliases for the entities. For **Requirement** these can include: Functional Requirement, Safety Requirement, Support Requirement, Verification Requirement

Note the quality attributes below are optional. Other sets of quality attributes may be provided by the tool developer, or these may be user-definable. However, some form of quality attributes is recommended.

Attribute	Type	Description
clear	Boolean	*clear* represents if this **Requirement** is unambiguous and not confusing.
complete	Boolean	*complete* represents if this **Requirement** expresses a whole idea.
consistent	Boolean	*consistent* represents if this **Requirement** is not in conflict with other requirements.
correct	Boolean	*correct* represents if this **Requirement** describes the user's true intent and is legally possible.
design	Boolean	*design* represents if this **Requirement** does not impose a specific solution on design; says "what", not "how".
feasible	Boolean	*feasible* represents if this **Requirement** can be implemented with existing technology, within cost and schedule.
modular	Boolean	*modular* represents if this **Requirement** can be changed without excessive impact on other requirements.
traceable	Boolean	*traceable* represents if this **Requirement** is uniquely identified, and able to be tracked to predecessor and successor lifecycle items/objects.
verifiable	Boolean	*verifiable* represents if this **Requirement** is provable (within realistic cost and schedule) that the system meets the requirement.

All attributes and relationships from **Statement** are inherited by the **Requirement** entities.

3.4.16 Resource

Entity	Parent Entity	Child Entities	Description
Resource	**Asset**	None	A **Resource** entity specifies a consumable or producible **Asset**. Example: $5, 2 kilowatts, natural gas

Attribute	Type	Description
initial amount	Number	*initial amount* represents this **Resource**'s starting amount.
maximum amount	Number	*maximum amount* represents this **Resource**'s maximum amount allowed.
minimum amount	Number	*minimum amount* represents this **Resource**'s minimum amount allowed.
units	Text	*units* represents this **Resource**'s units used to measure the amounts, such as each or tons.
type	Text	*type* provides aliases for the entities. For **Resource** these can include: Computer Resource, Human Resource, Fuel

Relationship	Inverse	Target Entity(es)	Description
consumed by	***consumes***	**Action**	***consumed by*** identifies the **Action** that uses this **Resource**. After the **Action** is completed, the amount consumed is not returned to the **Resource**.
	Attribute	Type	Description
	amount	Number	amount represents how much of the resource is ***consumed by*** the ***Action***. Units are relative to the units selected for the **Resource**.
produced by	***produces***	**Action**	***produced by*** identifies the **Action** that creates this **Resource**. **Resources** are produced when the execution of the action completes.
	Attribute	Type	Description
	amount	Number	*amount* represents how much of the *Resource* is *produced by* the *Action*. Units are relative to the units selected for the *Resource*.
seized by	***seizes***	**Action**	***seized by*** identifies the **Action** that uses this

			Resource. After the **Action** has completed, this **Resource** is released for use by other **Actions**.
	Attribute	Type	Description
	amount	Number	amount represents how much of the resource is ***captured by*** the **Action**. Units are relative to the units selected for the **Resource**.

All attributes and relationships from **Asset** are inherited by the **Resource** entities.

3.4.17 Risk

Entity	Parent Entity	Child Entities	Description
Risk	None	None	A **Risk** entity specifies the combined probability and consequence in achieving objectives. Example: the risk of a large meteorite hitting the earth in the next 100 years is low but it could cause the extinction of life as we know it.

Attribute	Type	Description
type	Text	*type* provides aliases for the entities. For **Risk** these can include: Cost Risk, Schedule Risk, Technical Risk

Attribute	Type	Description
consequence	Percent	*consequence* represents the level of effect from this **Risk** occurring.
consequence description	Text	*consequence description* represents the result of this **Risk** occurring.
mitigation status	Enumeration [**Accept**, Avoid, Mitigate, Transfer]	*mitigation status* represents the status of the mitigation technique for this **Risk**.
probability	Percent	*probability* or *likelihood* represents the chance that this **Risk** will occur.
status	Enumeration [**Open**, Duplicate, Declined, Resolved]	*status* represents the current state of this **Risk**.
trend	Enumeration [Decreasing, Increasing, New, **Unchanged**]	*trend* indicates the change in the **Risk** over time as to whether it is increasing, decreasing, or staying the same.

Relationship	Inverse	Target Entity(es)	Description
caused by	***causes***	**Action, Artifact, Asset, Characteristic, Connection, Cost, Decision, Input/Output, Risk, Statement**	***caused by*** identifies the entity that this **Risk** results.

causes	***caused by***	**Risk**	***causes*** identifies the **Risk** resulting from this entity.
decomposed by	***decomposes***	**Risk**	***decomposed by*** identifies the children of this entity.
decomposes	***decomposed by***	**Risk**	***decomposes*** identifies the parent of this entity.
enables	***enabled by***	**Decision**	***enables*** identifies the **Decision** that is empowered by this entity.
incurs	***incurred by***	**Cost**	***incurs*** identifies the **Cost** value for this entity.
located at	***locates***	**Location**	***located at*** identifies the **Location** where this entity exists.
mitigated by	***mitigates***	**Action,** **Artifact,** **Asset,** **Characteristic,** **Connection,** **Cost,** **Decision,** **Location,** **Statement,** **Time**	***Mitigated by*** identifies the entity that contains the plan to alleviate this **Risk**.
occurs	***occurred by***	**Time**	***occurs*** identifies the **Time** (or timespan) during which this entity happens.
references	***referenced by***	**Artifact**	***references*** identifies the **Artifact** that specifies and/or enhances the definition of this entity.
related to	***relates***	**Risk**	***related to*** identifies the entity that ties in a peer-to-peer way with this entity.
	Attribute	Type	Description
	context	Text	context represents a description of this relation.
relates	***related to***	**Risk**	***relates*** identifies the entity that ties in a peer-to-peer way with this entity.
	Attribute	Type	Description
	context	Text	context represents a description of this relation.
resolved by	***resolves***	**Action,** **Artifact,** **Asset,** **Characteristic,** **Connection,** **Cost,** **Decision,** **Input/Output,** **Risk,** **Statement**	***resolved by*** identifies the entity that closes this **Risk**.

resolves	***resolved by***	**Risk**	***resolves*** identifies the **Risk** that is closed by this entity.
results in	***result of***	**Decision**	***results in*** identifies the **Decision** that is caused by this entity.
specified by	***specifies***	**Characteristic**	***specified by*** identifies a **Characteristic** that provides further information about this entity.

3.4.18 Statement

Entity	Parent Entity	Child Entities	Description
Statement	None	**Requirement**	A **Statement** entity specifies text referenced by the knowledgebase and usually contained in an Artifact. Example: Elvis is king!, Our goal is to be the first on Mars

Attribute	Type	Description
type	Text	*type* provides aliases for the entities. For **Statement** these can include: Acronym, Assumption, Constraint, Definition, Directive, Doctrine, Goal, Need, Objective, Plan, Policy, Question, Rule, Scope, Standard, Vision

Relationship	Inverse	Target Entity(es)	Description
alternative of	***alternative***	**Decision**	*alternative of* identifies the *Decision* that has this *Statement* as a potential choice.
	Attribute	Type	Description
	choice	Boolean	choice represents if this alternative was the choice selected.
causes	***caused by***	**Risk**	***causes*** identifies the **Risk** resulting from this entity.
decomposed by	***decomposes***	**Statement**	***decomposed by*** identifies the children of this entity.
decomposes	***decomposed by***	**Statement**	***decomposes*** identifies the parent of this entity.
enables	***enabled by***	**Decision**	***enables*** identifies the **Decision** that is empowered by this entity.
incurs	***incurred by***	**Cost**	***incurs*** identifies the **Cost** value for this entity.
located at	***locates***	**Location**	***located at*** identifies the **Location** where this entity exists.
mitigates	***mitigated by***	**Risk**	***mitigates*** identifies the **Risk** that this entity plan alleviates.
occurs	***occurred by***	**Time**	***occurs*** identifies the **Time** (or timespan) during which this entity happens.
references	***referenced by***	**Artifact**	***references*** identifies the **Artifact** that specifies and/or enhances the definition of this entity.
related to	***relates***	**Statement**	***related to*** identifies the entity that ties in a peer-to-peer way with this entity.
	Attribute	Type	Description
	context	Text	context represents a description of this relation.

relates	***related to***	**Statement**	***relates*** identifies the entity that ties in a peer-to-peer way with this entity.
	Attribute	Type	Description
	context	Text	context represents a description of this relation.
resolves	***resolved by***	**Risk**	***resolves*** identifies the **Risk** that is closed by this entity.
results in	***result of***	**Decision**	***results in*** identifies the **Decision** that is caused by this entity.
sourced by	***source of***	**Artifact**	***sourced by*** identifies the **Artifact** that contains this *Statement*.
specified by	***specifies***	**Characteristic**	***specified by*** identifies a **Characteristic** that provides further information about this entity.
traced to	***traced from***	**Action, Artifact, Asset, Characteristic, Connection, Cost, Input/Output, Location**	***traced to*** identifies an entity that is accredited to this **Statement**.

3.4.19 Time

Entity	Parent Entity	Child Entities	Description
Time	None	None	A **Time** entity specifies a point or period when an action, asset, process, or condition exists, finishes, starts, or continues. Example: Milestone A, Phase 2

Attribute	Type	Description
duration	Number	*duration* represents the period of time this **Time** occurs. A zero (0) duration indicates a milestone.
start	DateTime	*start* represents the time when this **Time** begins.
end	DateTime	*end* represents the time when this **Time** finishes. It can be computed from the *start* and *duration* attributes.
type	Text	*type* provides aliases for the entities. For **Time** these can include: Duration, Milestone, Point In Time, Time Frame

Relationship	Inverse	Target Entity(es)	Description
date resolves	***date resolved by***	**Decision**	***date resolves*** identifies the **Decision** that is closed at this **Time**.
decided by	***decision due***	**Decision**	***decided by*** identifies the **Decision** scheduled for closure at this **Time**.
decomposed by	***decomposes***	**Time**	***decomposed by*** identifies the children of this entity.
decomposes	***decomposed by***	**Time**	***decomposes*** identifies the parent of this entity.
mitigates	***mitigated by***	**Risk**	***mitigates*** identifies the *Risk* that this entity plan alleviates.
occurred by	***occurs***	**Action, Artifact, Asset, Characteristic, Conduit, Cost, Decision, Input/Output, Location, Risk, Statement**	***occurred by*** identifies the entity that happens at this **Time**.
related to	***relates***	**Time**	***related to*** identifies the entity that ties in a peer-to-peer way with this entity.
	Attribute	Type	Description
	context	Text	context represents a description of this relation.

relates	***related to***	**Time**	***relates*** identifies the entity that ties in a peer-to-peer way with this entity.
	Attribute	Type	Description
	context	Text	***context*** represents a description of this relation.
specified by	***specifies***	**Characteristic**	***specified by*** identifies a **Characteristic** that provides further information about this entity.

3.4.20 Virtual (Location)

Entity	Parent Entity	Child Entities	Description
Virtual	**Location**	None	A **Virtual** entity specifies a location within a digital network. Example: http://www.google.com

Attribute	Type	Description
address	URI	*address* represents the identification address using the Uniform Resource Identifier (URI) protocols.

All attributes and relationships from **Location** are inherited by the **Virtual** entities.

4 Visualizations

The following diagrams represent the common forms of visualizing information. They do not attempt to encompass every possible visualization. Only one is unique to LML: the Action Diagram. Many similar models have been developed over the years to express functional sequencing, such as the Flow Charts, Activity Diagram in UML/SysML, Business Process Modeling Notation (BPMN), and others. Although these various notations are accurate, they use many different symbols, which often confuse the non-expert viewers/recipients of the visualization. As seen below, the visualization of this functional sequencing in LML is much simpler, but it appears to have all the necessary information for language execution. The usual constructs are replaced by special cases of Actions to denote decision points.

The other visualizations should be considered standard diagrams, used over many years by different techniques. We also provide suggested diagrams that LML users may want to consider as well. A few do not appear to require visualization beyond a hierarchy diagram. We denote which diagrams are appropriate for which classes.

4.1 Action Diagram (Mandatory for *Action* entities with children)

The Action Diagram (see Figure 4-1) represents the functional sequencing of **Actions** along with the data flow provided by the **Input/Output** entities. This combination of **Actions** with **Input/Outputs** is similar to the SREM "Behavior Diagram" or UML Activity Diagram. Without the **Input/Output** entities, the Action Diagram would be the equivalent of the classic Function Flow Block Diagram (FFBD) or IDEF 3. The main difference between LML and these other diagrams is the use of special kinds of **Actions** to replace the constructs used in these other languages. The construct set is shown below.

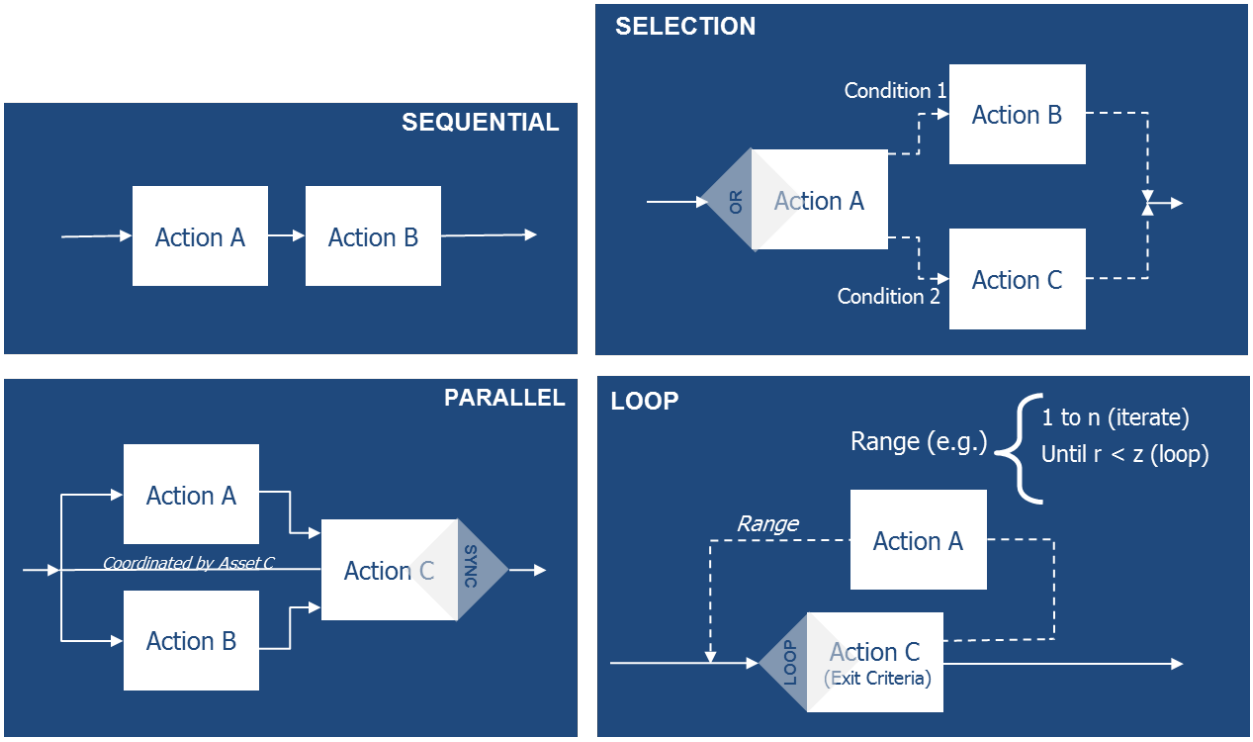

Figure 4-1. Special Actions for the Action Diagram to Capture Decision Points.

The special cases of **Actions**, denoted by the rectangle with a diamond embedded in it (showing ½ the diamond as a point on the rectangle), represent decision points. For example, in a loop the key decision is the exit criteria for the loop. This criterion can be as simple as the number of iterations of the loop or more complex logic that determines when the loop must stop.

The "OR" decision point represents an exclusive selection of one path or the other. The decision point in the case can be a probability or a specific criterion for path selection.

The final decision point type is the "SYNC." The SYNC provides the functional rationale for ending parallel branches. Note that in the physical view, two separate entities can exist without any synchronization between them. However, in the functional view between two physical entities that are interacting, it is necessary decide how to terminate that interaction. We see this in software that when two parallel processes are spawned, these threads must be synchronized to complete the program.

The Action Diagram can include **Input/Output** entities as well. An example of this inclusion is shown in Figure 4-2.

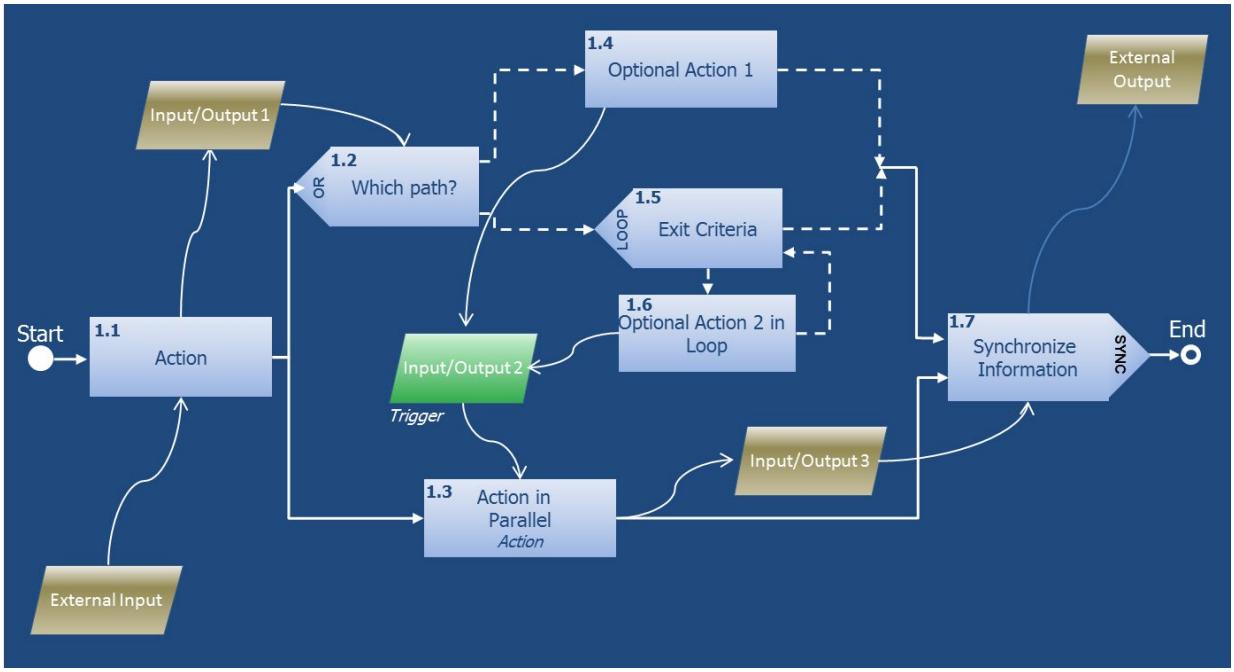

Figure 4-2. An Example Action Diagram with Inputs/Outputs.

The **Input/Output** entities are shown as parallelograms, reminiscent of the classical flow charting symbols used in the 1950s and 1960s. Two colors (or some other mechanism) are used to distinguish triggering **Input/Outputs** from optional **Input/Outputs**. It is recommended that a different type of line be used to show the **Input/Output** flow, thus making it easy to distinguish between the data flow and functional sequencing lines. The diagram should also contain a way to show the **Start** and **End** of the functional sequencing.

No other constructs have been determined to be necessary. Other languages include a “Replicate,” however the research done by the LML developers indicated that it was a way to identify a physical instantiation of the functional entity in more than one physical entity at a time. This representation appears more appropriately in the physical diagram (see Asset Diagram in next section).

While this specification does not directly specify or standardize the Action Diagram, the constructs above are to be used as guidance for the overall look of an Action Diagram. The key takeaway is the inclusion of logic flow into the Action entities themselves, instead of relying on separate logic flow diagram elements.

4.2 Asset Diagram (Mandatory for *Asset* entities with children)

LML must have a physical representation of design elements as well as the functional one provided by the Action Diagram. Figure 4-3 shows one way of providing this information. The **Assets** are shown as rectangles, with **Conduits** displayed as lines connecting the **Assets**. Since **Resource** is a subclass of **Asset**, they could also be displayed by this diagram. Other information, such as the capacity and latency attributes of the **Conduit**, may be overlaid on this diagram as well. Also, since the **Asset** *type* may interest most users, it should be desirable to include that attribute on this diagram as well.

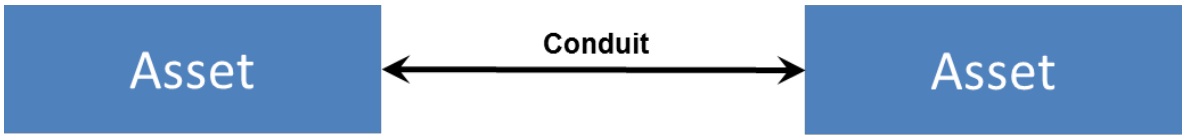

Figure 4-3. A Simple Asset Diagram.

Nodes and connections could be replaced with pictures as shown in Figure 4-4.

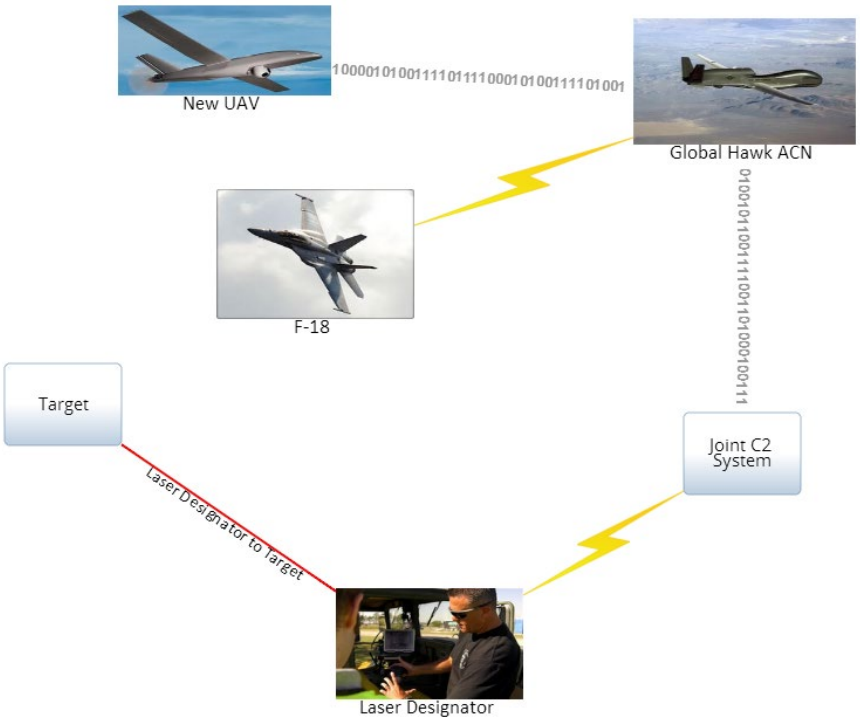

Figure 4-4. Asset Diagram with Pictures and Special Lines for Conduits.

The Asset Diagram is mandatory only for those Assets with children.

4.3 Spider Diagram (Mandatory for Traceability)

The spider diagram shows how entities are related to one another. This diagram is similar to the ERD shown above, but reflects LML's schema, not an abstract schema. This diagram (see Figure 4-5) shows traceability of LML entities with their relationships.

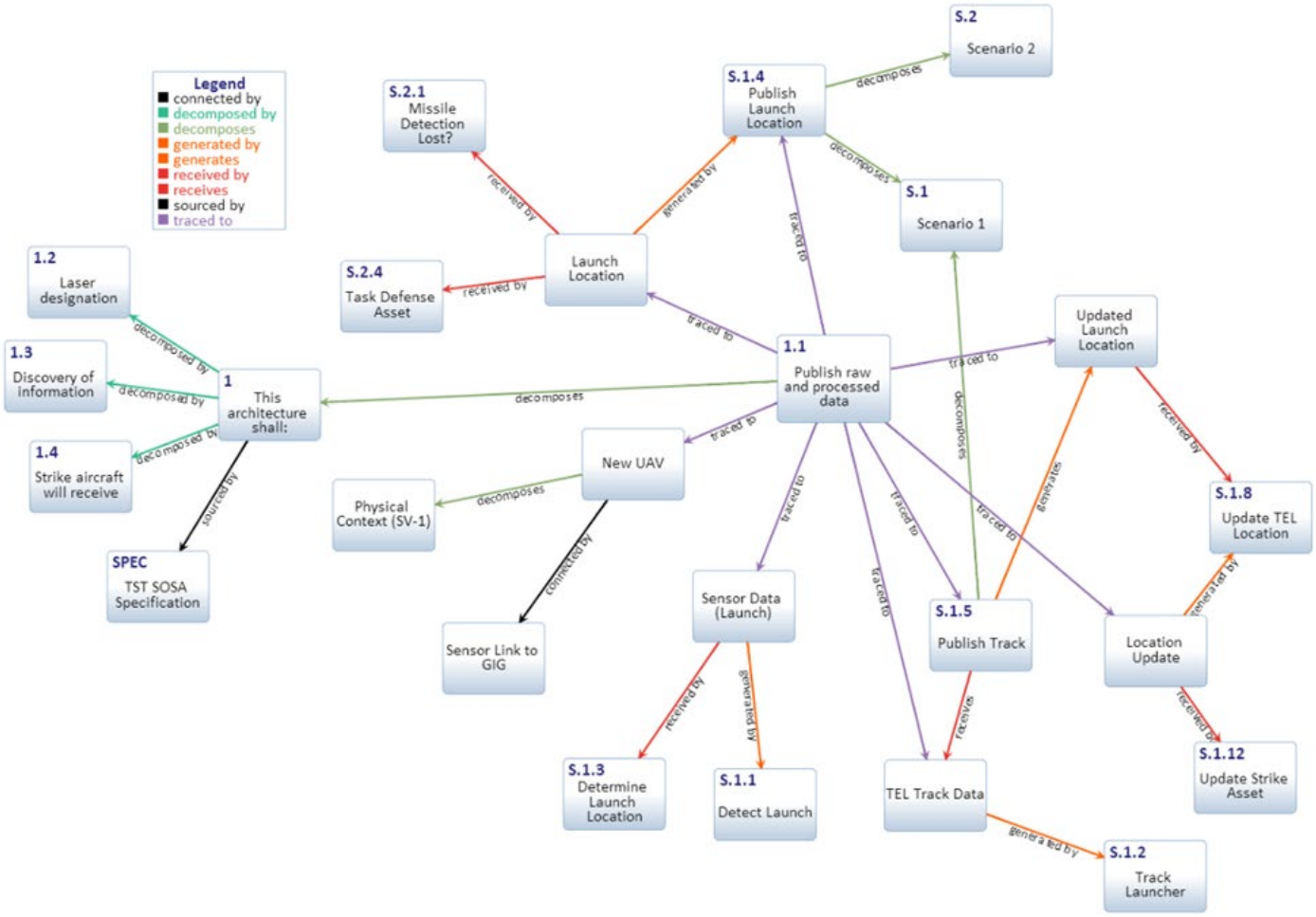

Figure 4-5. Example Spider Diagram to Show Traceability between Entities Using the LML Relationships.

4.4 Example Views for Other LML Entities

LML was designed to support the full lifecycle. All of its entities can support the common visualizations that architects, systems engineers, software engineers, test engineers, operators, logisticians, and program managers use. The examples below are suggestions of how to implement these common visualizations using LML.

4.4.1 Class Diagram

The diagram below shows how to implement the UML Class Diagram using LML classes. Since the Class Diagram has become a standard for software development, it seems a good candidate for inclusion in LML's approach. In this diagram, the LML entities that match the diagram elements include: **Asset** with the *type* "object"; attributes are captured as **Characteristics**; methods are **Actions**; and relationships are **Logical** connections.

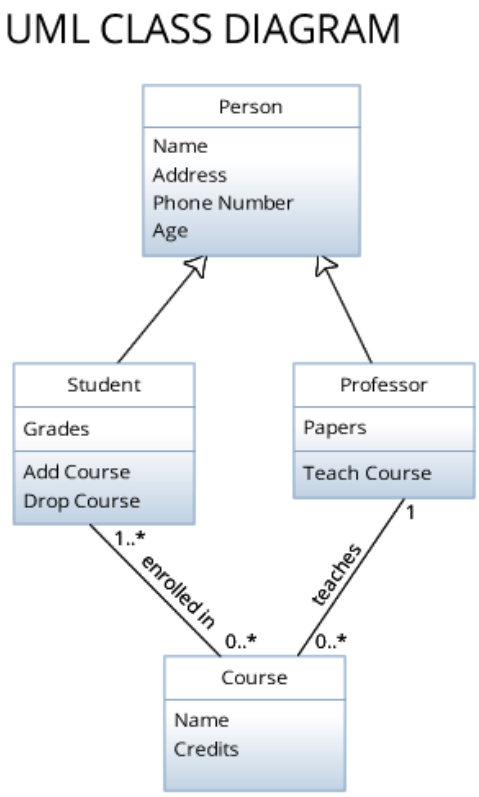

Figure 4-6. LML Encourages Use of the Class Diagram and Provides the Schema Entities to Support Its Creation.

4.4.2 Entity-Relationship Diagram

Another way to model information has been the classic entity-relationship diagram (ERD). An example of this is shown below for the LML schema. The relationships are expressed using the **Logical Connection**. Entities are represented by **Assets**. You can also capture attributes as **Characteristics**.

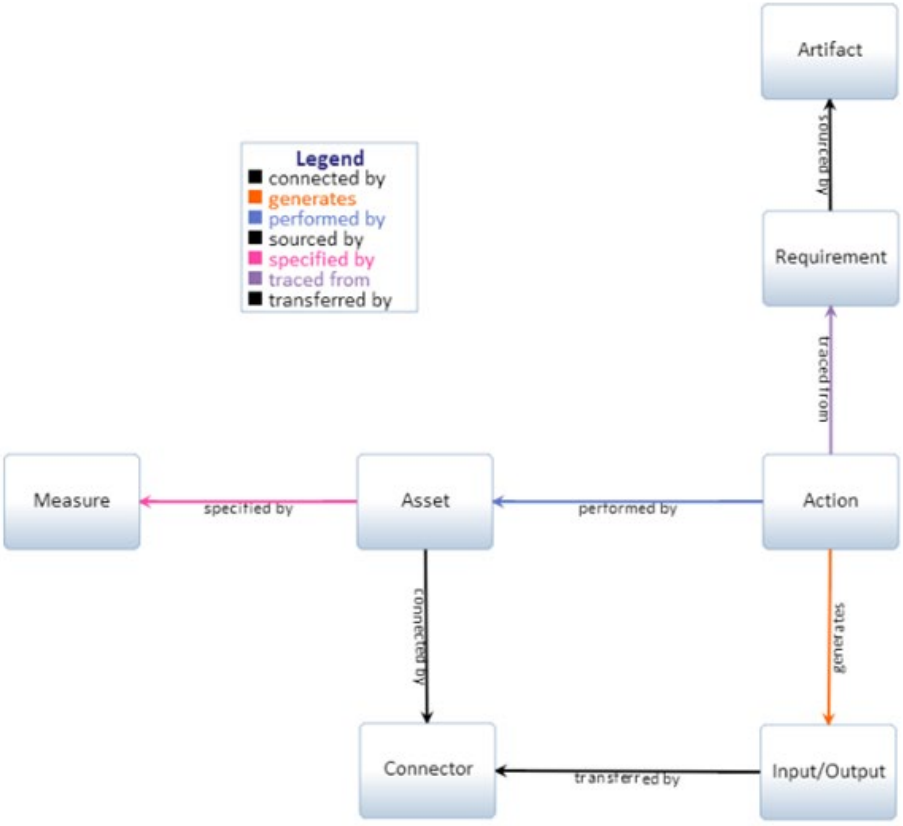

Figure 4-7. Entity-Relationship Diagram Developed Using the Asset Diagram.

4.4.3 Timelines

The **Actions** and **Times** can be displayed using a classic Gantt Chart shows how functional elements execute over time. An example of this is shown below.

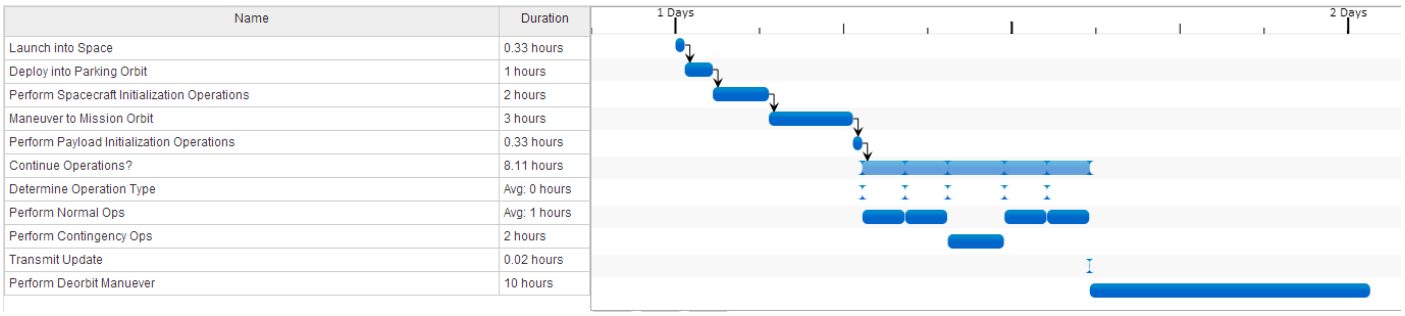

Figure 4-8. Use of Gantt Chart for Displaying Actions and Durations.

Time with **Actions** or **Assets** can also be visualized in many other ways. One of the most useful is shown below.

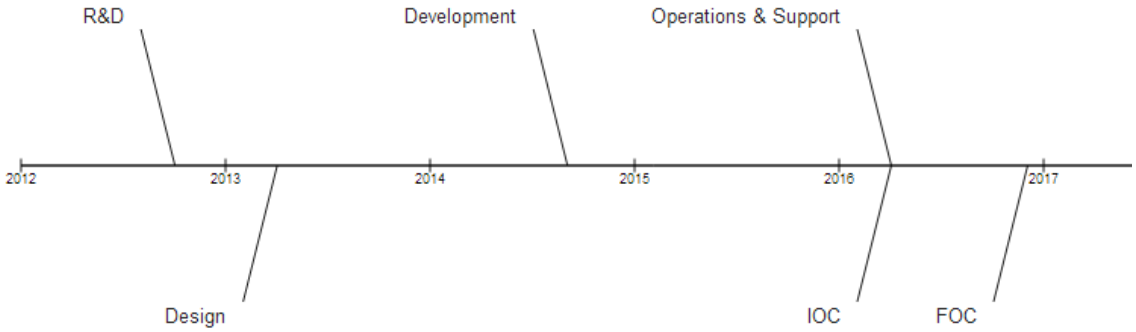

Figure 4-9. Timeline Diagram Showing Actions at Specific Times.

4.4.4 Hierarchy Diagram

A hierarchy chart is used in LML to show decomposition of elements. The figure below shows an example of requirements decomposition. This chart uses the ***decomposed by*** relationship.

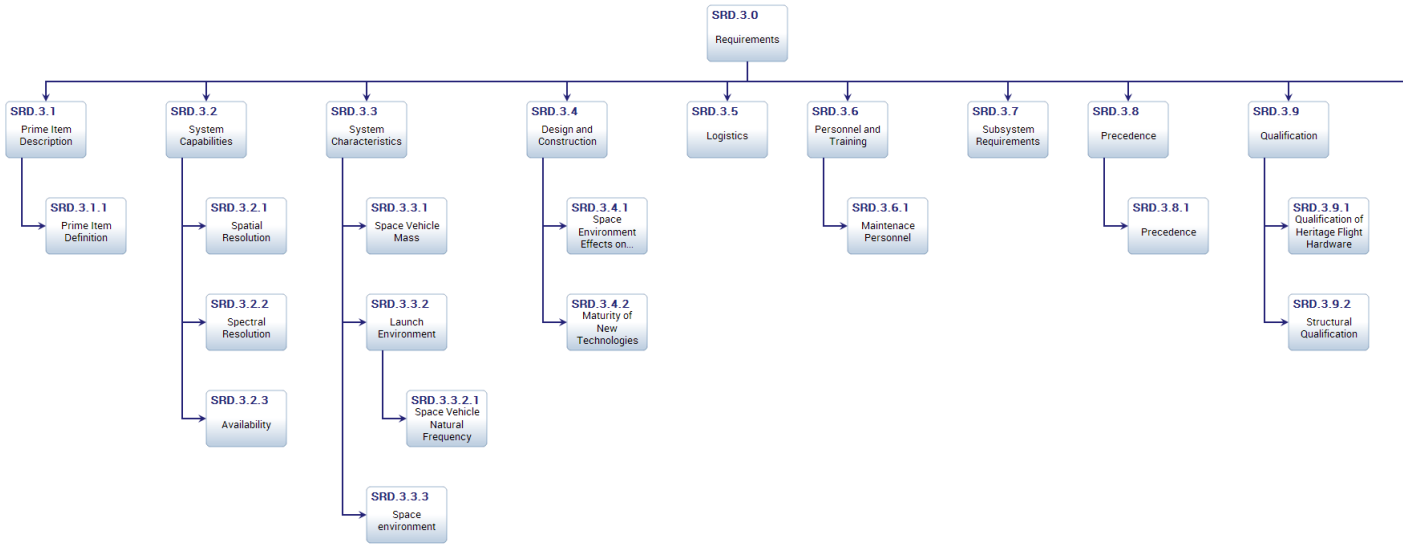

Figure 4-10. A Hierarchy Diagram Is a Good Way to Show Decomposition.

4.4.5 Risk Matrix

A standard DoD risk matrix (shown below) or other form can be used to display **Risk** entity information. Another type of risk analysis uses probabilities to create a fault-tree. A fault tree is often shown as a hierarchy diagram with the probabilities shown for each branch.

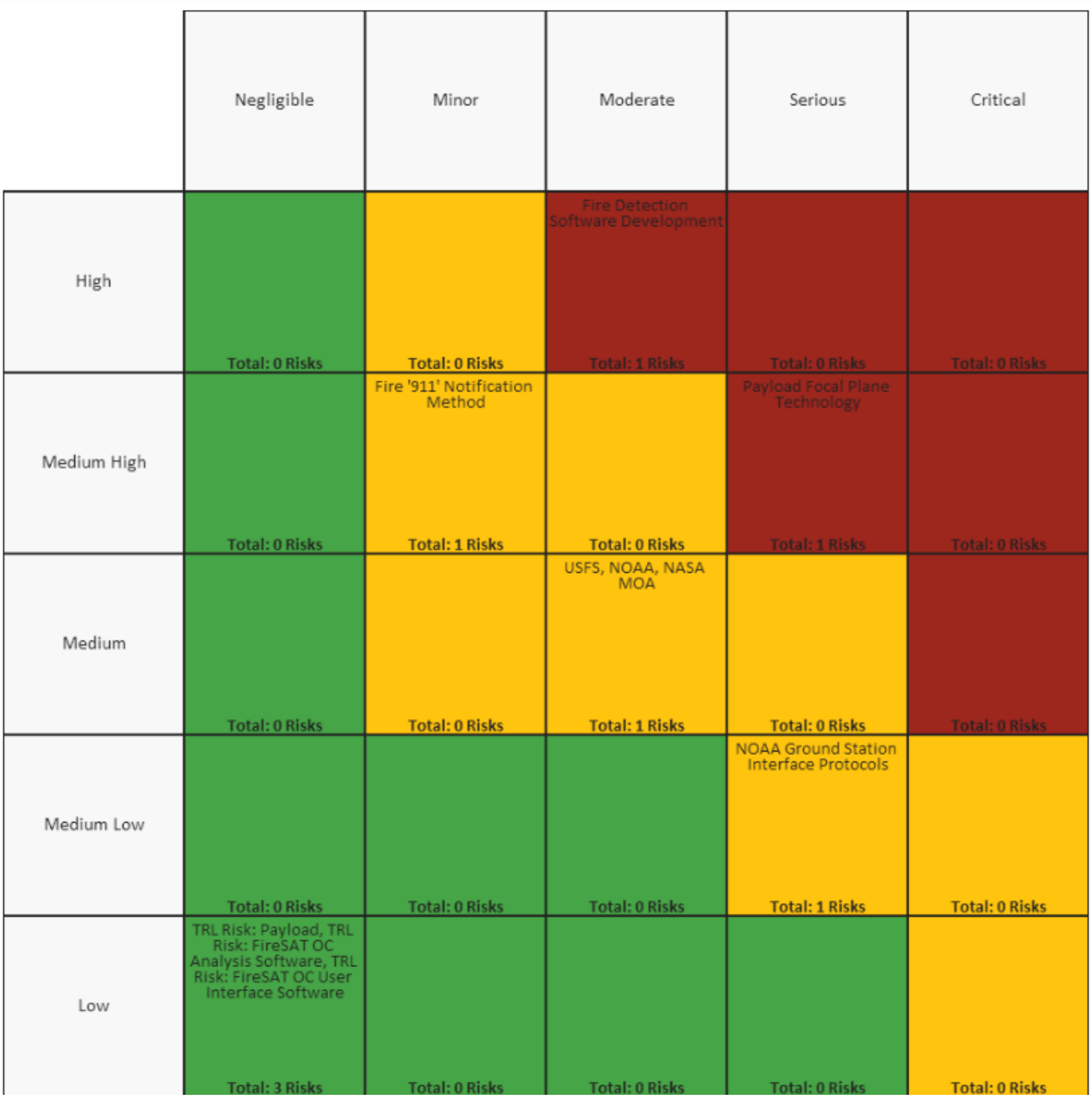

Figure 4-11. Typical Risk Matrix.

4.4.6 State Machine Diagram

The state machine diagram expresses how an **Asset** transitions from one state to another. In the diagram below, the state (or **Characteristic**) transition occurs when the **Action** entity event causes the transition to the other state.

Name	Class	Description
State	**Characteristic**	Means that it's a state of the system
Initial State	**Characteristic**	Means that it's the initial state
Final State	**Characteristic**	Means that it's the final state.

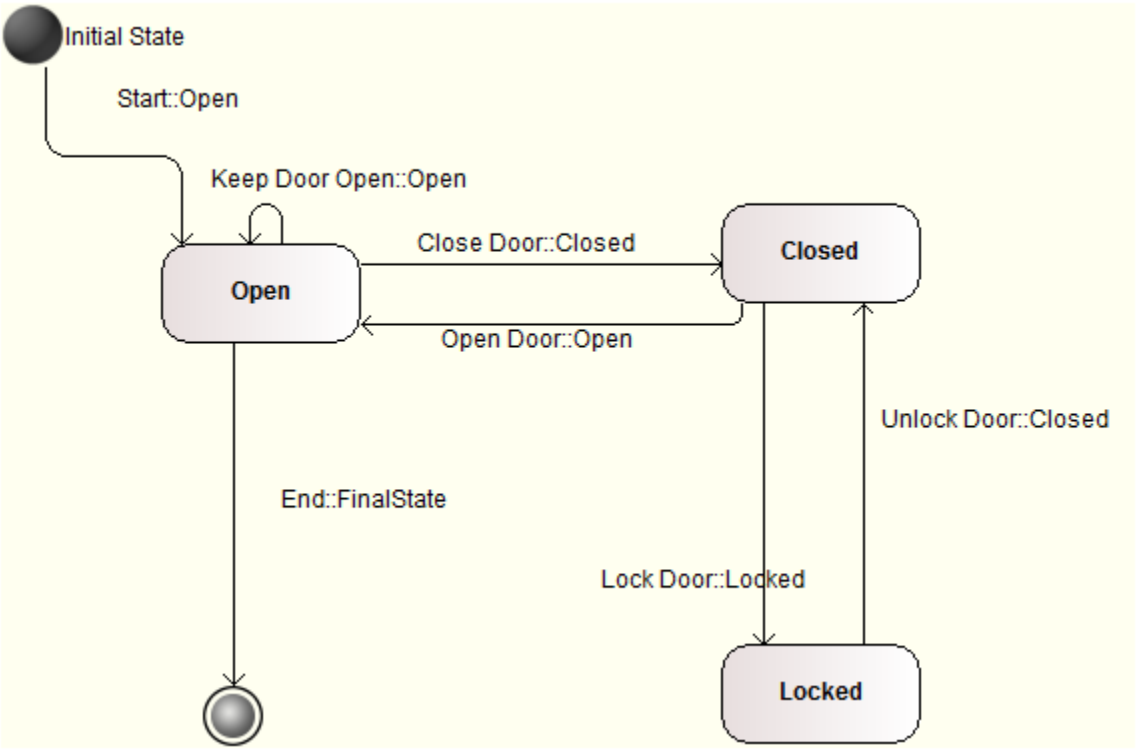

Figure 4-12. The State Machine Diagram Has Proven Useful and LML Supports It.

Appendix A. SysML Mapping to LML

SysML focuses mainly on diagrams, with an underlying ontology embedded in the XMI that each diagram represents. Currently (October 2013) a complete ontology is under development. The table below shows the various SysML diagrams and the LML equivalent diagram and associated classes.

Table A-1. SysML Diagram Mapping to LML Diagrams and Ontology

SysML Diagram	LML Diagram	LML Entities
Activity	Action Diagram	**Action**, **Input/Output**
Sequence	Sequence	**Action**, **Asset**
State Machine	State Machine	**Characteristic** (State), **Action** (Event)
Use Case	Asset Diagram	**Asset**, **Connection**
Block Definition	Class Diagram, Hierarchy Chart	**Input/Output** (Data Class), **Action** (Method), **Characteristic** (Property)
Internal Block	Asset Diagram	**Asset**, **Connection**
Package	Asset Diagram	**Asset**, **Connection**
Parametric	Hierarchy, Spider, Radar	**Characteristic**
Requirement	Hierarchy, Spider	**Requirement** and related entities

Although the Systems Modeling Language (SysML) does not have an official ontology many tool vendors have created these models from database schemas. The purpose of this appendix is to identify the entities, relationships and attributes necessary to completely visualize the SysML models in LML. The SysML 1.4 standard is available at http://www.omg.org/spec/SysML/1.4/ (as of 1 December 2015). This appendix will only address the changes to LML required to produce these models.

Note that significant changes between the Action Diagram and the Activity Diagram must occur if developers want to adhere to all the SysML requirements, since Action Diagrams do not contain the large number of constructs used in an Activity Diagram. This specification does not recommend such a large number of constructs as they impede understanding of the diagram. The same content is provided by the ontology.

SPEC Innovations' Innoslate® tool was used as the basis for this extension. Innoslate added one class (Equation) and one subclass to Asset (Port) to visualize the complete set of SysML models. Numerous relationships were added to accommodate the SysML visualizations. The changes to Innoslate's LML schema for SysML are shown in the tables below.

New Classes

Class	Parent	Description
Equation		An Equation entity specifies an equation (mathematical or logical) that can be used to describe a part of the model.

Class	Parent	Description
Port	Asset	An interaction point of a block, specifying the input and output flow.

Equation Class

An Equation entity specifies an equation (mathematical or logical) that can be used to describe a part of the model.

Equation Properties

Name	Type	Description
Value	Text	*Value* represents this Equation's text.

Equation Relations

Name	Classes	Description
decomposed by	Equation	*Decomposed by* identifies the children of this entity.
decomposes	Equation	*Decomposes* identifies the parent of this entity.
equation for	Cost	*Equation for* identifies the entity that this Equation represents.
equation for	Time	*Equation for* identifies the entity that this Equation represents.
equation for	Characteristic	*Equation for* identifies the entity that this Equation represents.
equation for	Statement	*Equation for* identifies the entity that this Equation represents.
equation for	Decision	*Equation for* identifies the entity that this Equation represents.
equation for	Action	*Equation for* identifies the entity that this Equation represents.
equation for	Risk	*Equation for* identifies the entity that this Equation represents.
equation for	Location	*Equation for* identifies the entity that this Equation represents.
equation for	Asset	*Equation for* identifies the entity that this Equation represents.
equation for	Artifact	*Equation for* identifies the entity that this Equation represents.
equation for	Connection	*Equation for* identifies the entity that this Equation represents.
equation for	Input/Output	*Equation for* identifies the entity that this Equation represents.
has variable	Characteristic	*Has variable* identifies the Characteristic that is represented in this Equation.
related to	Equation	*Related to* identifies the entity that ties in a peer-to-peer way with this entity.
relates	Equation	*Relates* identifies the peer-to-peer entity that is tied to this

Name	Classes	Description
		entity.

Port Class

An interaction point of a block, specifying the input and output flow.

Port Properties

Name	Type	Description
Direction	Enumeration	*Direction* represents the flow of data on this port.

Action Relations

Name	Classes	Description
depends on	Action	*Depends on* identifies the Action that this Action has a dependency on.
equation of	Equation	*Equation of* identifies the Equation that represents this entity.
extend	Action	*Extend* identifies the Action (use case) that is added to this Action (use case).
extended by	Action	*Extended by* identifies the Action (use case) that is added from this Action (use case).
fetches	Characteristic	*Fetches* identifies the State Characteristic that this Action receives.
has dependent	Action	*Has Dependent* identifies the Action that depends on this Action.
include	Action	*Include* identifies the Action (use case) that is added to this Action (use case).
included by	Action	*Included by* identifies the Action (use case) that is added from this Action (use case).
pushed by	Characteristic	*Pushed by* identifies the State Characteristic that this Action is generated from.
satisfies	Requirement	*Satisfies* identifies the Requirement that is fulfilled by this entity.
verifies	Requirement	*Verifies* identifies the Requirement that is supported by this entity.

Artifact Relations

Name	Classes	Description
equation of	Equation	*Equation of* identifies the Equation that represents this entity.
satisfies	Requirement	*Satisfies* identifies the Requirement that is fulfilled by this entity.
verifies	Requirement	*Verifies* identifies the Requirement that is supported by this entity.

Asset Relations

Name	Classes	Description
equation of	Equation	*Equation of* identifies the Equation that represents this entity.
extend	Asset	*Extend* identifies the Action (use case) that is added to this Action (use case).
extended by	Asset	*Extended by* identifies the Action (use case) that is added from this Action (use case).
instantiated by	Asset	*Instantiated by* identifies another entity that is an instance of this entity.
instantiates	Asset	*Instantiates* identifies another entity that has this entity as an instance.
represented in	Asset	*Represented in* identifies the Asset whose diagrams include this Asset.
represents	Asset	*Represents* identities the Asset that is included in this Asset's diagrams.
satisfies	Requirement	*Satisfies* identifies the Requirement that is fulfilled by this entity.
verifies	Requirement	*Verifies* identifies the Requirement that is supported by this entity.

Characteristic Relations

Name	Classes	Description
equation of	Equation	*Equation of* identifies the Equation that represents this entity.
fetched by	Action	*Fetched by* identifies the Action that this State Characteristic is received by.
generates	Input/Output	*Generates* identifies the Input/Output or Action that this Action or Characteristic transforms.
generates	Action	*Generates* identifies the Input/Output or Action that this Action or Characteristic transforms.
instantiated by	Characteristic	*Instantiated by* identifies another entity that is an instance of this entity.
instantiates	Characteristic	*Instantiates* identifies another entity that has this entity as an instance.
pushes	Action	*Pushes* identifies the Action that this State Characteristic generates.
satisfies	Requirement	*Satisfies* identifies the Requirement that is fulfilled by this entity.
variable of	Equation	*Variable of* identifies the Equation that this Characteristic is represented in.
verifies	Requirement	*Verifies* identifies the Requirement that is supported by this entity.

Conduit Relations

Name	Classes	Description
instantiated by	Conduit	*Instantiated by* identifies another entity that is an instance of

Name	Classes	Description
		this entity.
instantiates	Conduit	*Instantiates* identifies another entity that has this entity as an instance.

Connection Relations

Name	Classes	Description
equation of	Equation	*Equation of* identifies the Equation that represents this entity.
satisfies	Requirement	*Satisfies* identifies the Requirement that is fulfilled by this entity.
verifies	Requirement	*Verifies* identifies the Requirement that is supported by this entity.

Cost Relations

Name	Classes	Description
equation of	Equation	*Equation of* identifies the Equation that represents this entity.
satisfies	Requirement	*Satisfies* identifies the Requirement that is fulfilled by this entity.
verifies	Requirement	*Verifies* identifies the Requirement that is supported by this entity.

Decision Relations

Name	Classes	Description
equation of	Equation	*Equation of* identifies the Equation that represents this entity.

Input/Output Relations

Name	Classes	Description
equation of	Equation	*Equation of* identifies the Equation that represents this entity.
instantiated by	Input/Output	*Instantiated by* identifies another entity that is an instance of this entity.
instantiates	Input/Output	*Instantiates* identifies another entity that has this entity as an instance.
satisfies	Requirement	*Satisfies* identifies the Requirement that is fulfilled by this entity.
verifies	Requirement	*Verifies* identifies the Requirement that is supported by this entity.

Location Relations

Name	Classes	Description
satisfies	Requirement	*Satisfies* identifies the Requirement that is fulfilled by this entity.
verifies	Requirement	*Verifies* identifies the Requirement that is supported by this entity.

Requirement Relations

Name	Classes	Description
copied by	Requirement	*Copied by* identifies the Requirement that is a clone of this Requirement.
copies	Requirement	*Copies* identifies the Requirement that this Requirement clones.
derived by	Requirement	*Derived by* identifies the Requirement that is developed from this Requirement.
derives	Requirement	*Derives* identifies the Requirement that this Requirement develop from.
refined by	Requirement	*Refined by* identifies the Requirement that clarifies this Requirement.
refines	Requirement	*Refines* identifies the Requirement that this Requirement clarifies.
satisfied by	Cost	*Satisfied by* identifies the entity that fulfills this Requirement.
satisfied by	Time	*Satisfied by* identifies the entity that fulfills this Requirement.
satisfied by	Action	*Satisfied by* identifies the entity that fulfills this Requirement.
satisfied by	Artifact	*Satisfied by* identifies the entity that fulfills this Requirement.
satisfied by	Asset	*Satisfied by* identifies the entity that fulfills this Requirement.
satisfied by	Location	*Satisfied by* identifies the entity that fulfills this Requirement.
satisfied by	Input/Output	*Satisfied by* identifies the entity that fulfills this Requirement.
satisfied by	Characteristic	*Satisfied by* identifies the entity that fulfills this Requirement.
satisfied by	Connection	*Satisfied by* identifies the entity that fulfills this Requirement.
verified by	Input/Output	*Verified by* identifies the entity that supports this Requirement.
verified by	Characteristic	*Verified by* identifies the entity that supports this Requirement.
verified by	Asset	*Verified by* identifies the entity that supports this Requirement.
verified by	Artifact	*Verified by* identifies the entity that supports this Requirement.
verified by	Time	*Verified by* identifies the entity that supports this Requirement.
verified by	Action	*Verified by* identifies the entity that supports this Requirement.
verified by	Requirement	*Verified by* identifies the entity that supports this Requirement.
verified by	Cost	*Verified by* identifies the entity that supports this Requirement.
verified by	Connection	*Verified by* identifies the entity that supports this Requirement.
verified by	Location	*Verified by* identifies the entity that supports this Requirement.
verifies	Requirement	*Verifies* identifies the Requirement that is supported by this entity.

Risk Relations

Name	Classes	Description
equation of	Equation	*Equation of* identifies the Equation that represents this entity.

Statement Relations

Name	Classes	Description
equation of	Equation	*Equation of* identifies the Equation that represents this entity.

Time Relations

Name	Classes	Description
equation of	Equation	*Equation of* identifies the Equation that represents this entity.
satisfies	Requirement	*Satisfies* identifies the Requirement that is fulfilled by this entity.
verifies	Requirement	*Verifies* identifies the Requirement that is supported by this entity.

Appendix B. DoDAF MetaModel 2.0 (DM2) Mapping to LML

The Department of Defense has developed a schema called the DoDAF (DoD Architecture Framework) MetaModel 2.0 (DM2). The Conceptual Data Model is shown below. The physical data model contains over 500 entries. As shown below, this is a specialized model that focuses mainly on the DoD nomenclature and may be less useful in other domains. For example, since the DoD has developed their acquisition process around the concept of "Capability," that becomes a critical item in the top level of this schema, thus driving additional relationships. In LML we identified that capability could be a type of **Action**, **Asset**, **Characteristic** or even a **Statement**. However, LML does not preclude the user or tool vendor from adding "Capability" as an entity class. Schema extensions are actually encouraged for different domains. As such extensions are made and standardized; the LML Steering Committee will consider adding them as extensions to LML.

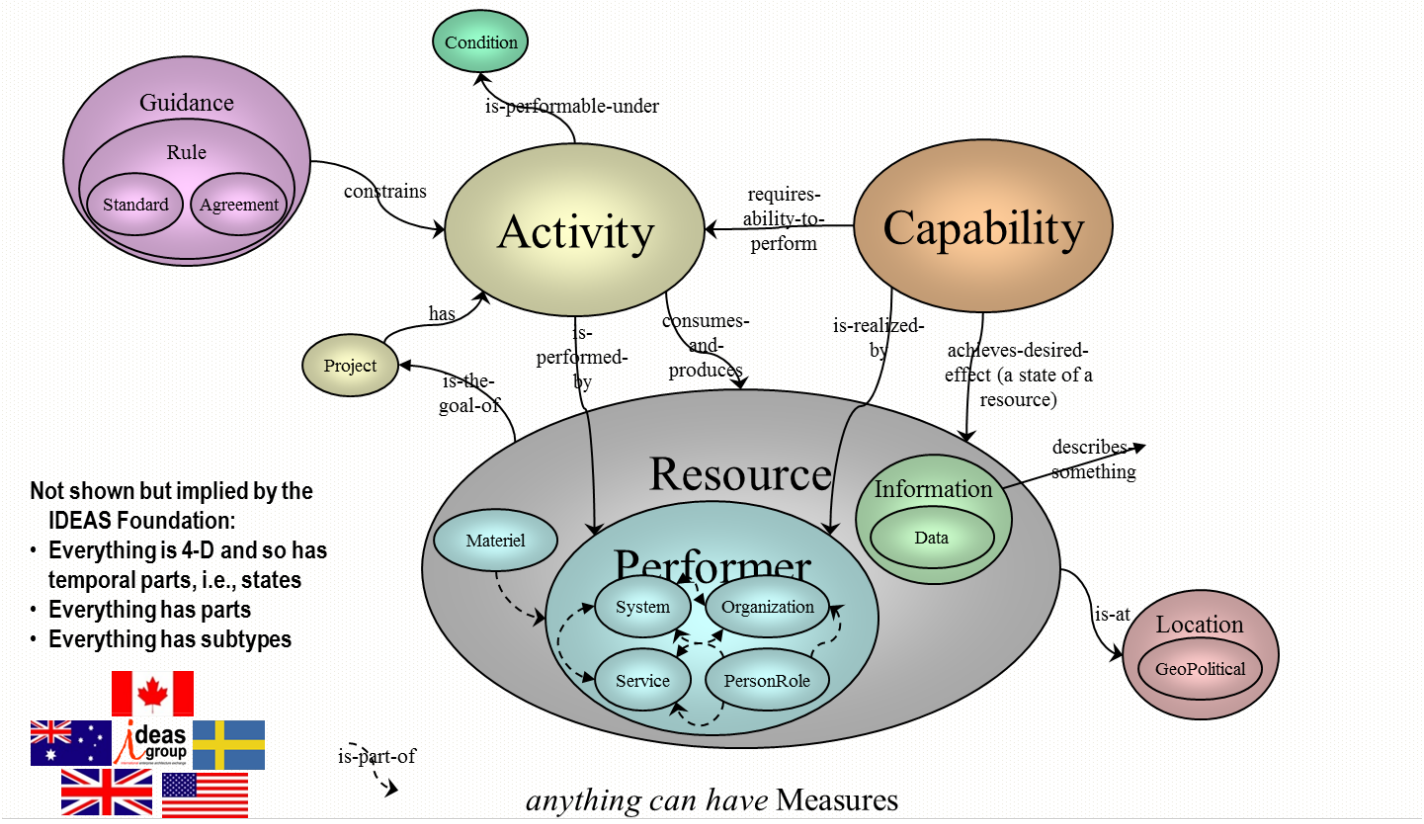

Figure B-1. DM2 Conceptual Data Model.

For a quick guide from the DM2 schema to LML, please see the table below.

Table B-1. DM2 Conceptual Data Model Mapping to LML

DM2 Schema Element (Conceptual)	LML Equivalent
Activity	Action
Capability	Action with “Capability” label
Condition	Characteristic with “Condition” label
Information/Data	Input/Output
Desired Effect	Statement with “Desired Effect” label
Guidance	Statement with “Guidance” label
Measure	Measure
Measure Type	Measure labels
Location	Location
Project	Action with “Project” label
Resource	Asset with labels for “Materiel,” “Organization,” etc.
Skill	Characteristic with label “Skill”
Vision	Statement with label “Vision”

Appendix C. Application to Verification and Validation

The purpose of this appendix is to document an approach to include verification and validation (V&V) techniques into LML. This approach requires the addition of three new subclasses: Test Suite, Test Case, and Verification Requirement. We have used the word "Test" to simplify the name of the class, but it applies to all types of V&V methods, including analysis, demonstration, inspection, modeling & simulation, and test. The classes along with the attributes and relationships are provided below.

New Classes

Class	Parent	Description
Test Case (Verification Event)	Action	A Test Case entity specifies a verification/validation task, as well as its expected and actual results.
Test Suite	Artifact	A Test Suite entity specifies a container for Test Cases.
Verification Requirement	Statement	A Verification Requirement entity specifies what is required to confirm that a requirement is satisfied

Test Case (subclass of Action)

A Test Case entity specifies a verification/validation task. This subclass provides a way to capture expected and actual results of the V&V activity.

Test Case Properties (additional to those inherited)

Name	Type	Description
Actual Result	Text	Actual Result summarizes the recorded results of the V&V task.
Expected Result	Text	Expected Result summarizes the predicted results of the V&V task.
Status	Enumeration	Status provides the state of the V&V task. Suggested values are: "Not Run," "In Progress," "Blocked," "Failed," and "Passed"
Setup	Text	Setup details steps required prior to conducting the V&V task. Setup details might require specific HWIL, SWIL, or surrogate input streams.
Event Conditions	Text	Specifies conditions and setup details to be used during the test. Conditions might specify parameters that simulate scenarios or physical analogs of processed materials.
Event Constraints	Text	Captures information that impose constraints on verification events, such as the need for independent verification, witness points, etc. Constraints might also indicate sampling requirements.

Test Case Relations (additional to those inherited)

Name	Classes	Description
references	Test Suite	References the Test Suite that provides the container for the top level Test Cases.
traced from	Verification Requirement	A verification event is traced from a verification requirement
evaluates	Asset	A verification event evaluates an Asset included in the event

Note: we have found from experimenting that a “File” attribute for the actual results is often added to accommodate the detailed actual results.

Test Suite (subclass of Artifact)

A Test Suite entity specifies a container for Test Cases. This subclass provides an organizational construct for various sets of Test Cases, such as for a particular unit test or a full operational test and evaluation suite.

Test Suite Properties (additional to those inherited)

Name	Type	Description
None	N/A	N/A

Test Suite Relations (additional to those inherited)

Name	Classes	Description
referenced by	Test Case	Referenced by Test Cases provides the top level link between the Test Cases and the Test Suite.

Note: The lack of specified properties and unique relationships means that this subclass can be implemented as a type of Artifact, with the extension of the “referenced by” relationship including the new Test Case subclass as a target.

Verification Requirement (subclass of a Statement)

A Verification Requirement entity specifies a verification/validation event. Multiple verification requirements can be linked to a single event, which aids in build verification plans, such as a Test and Evaluation Plan, or a Design Verification Plan. It is specific to either a asset in design (in the case of design verification), or a specific set of physical assets (in the case of product verification/acceptance).

Verification Requirement Properties (additional to those inherited)

Name	Type	Description
Verification Method	Enumeration	Minimum values: “analysis”, “inspection”, “demonstration”, “test”. Other possibilities include “modeling and simulation”, “design review”, “supplier guarantee”. The user needs to be able to add to this list.
Acceptance Criteria	Text	Acceptance Criteria state what results must be achieved to be considered satisfactory. These criteria should be apparent from the requirement statement and appropriate for the method of verification selected
Evidence	Text	Evidence specifies the type of documented or otherwise validated information and materials required to be obtained and delivered in order to confirm that a requirement was satisfied. For example, “Material test report with sample coupons”, “Inspection record recording actual geo-location measurements”, “Weld Radiographs”, “Analysis report based on results from a qualified model”, “written supplier warrantee”
Rationale	Text	Rationale captures the reasoning for the verification requirement properties

Verification Requirement Relations (additional to those inherited)

Name	Classes	Description
verifies	Requirement	Verifies a Requirement is satisfied by an Asset
traced to	Verification Event	Traced to a Verification Event to support planning activities
verifies	Asset	Verifies an Asset satisfies a requirement

Appendix D. Structuring Artifacts

The purpose of this appendix is to document an approach to structuring Artifacts, such as documents. This approach requires new metadata for the Artifact class, the addition of one new subclass (i.e. Heading) and one new relationship (originated by). This approach provides multiple advantages over the previous model, including:

- Eliminates numbering conflicts that arise when a requirement exists in multiple documents
- Enabling building content from any Entity class rather than being limited to Statement Class Entities
- Eliminates convoluted traceability resulting when Sections decompose to multiple levels of depth

The Artifact metadata, Heading class along with its attributes and relationships, and the “originated by” relationship are defined below. An entity relationship diagram illustrating the use of these items is also provided.

New Class

Class	Parent	Description
Heading	None	A Heading entity provides structure to an Artifact, representing Sections of a document

Heading

A Heading entity structures an Artifact and contains the content of the Section it represents.

Heading Relations (in addition to those that are standard for all entities)

Name	Classes	Description
sourced by	Artifact	A Heading is sourced by an Artifact when the Heading functions as a Level 1 Section
contains	Any	A Heading contains any other class. When viewed as a document, the Heading displays whatever attributes have been selected for the contained class

New Relationship (originated by)

Class	Name	Class	Description
Artifact	originates/ originated by	Requirement	Used to establish a direct link between a Requirement and the Artifact that functions as the originating authoritative source

The entity relationship diagram below (Figure D-1) shows how an Artifact might be structured. The Heading for Section 2.0 is directly linked to the Artifact using the “source of”/ “sourced by” relationship since it represents a Level 1 section. That section is also decomposed to capture deeper document structure, including a child section (2.1) and a grandchild section (2.1.1). Use of the Heading object allows a section to contain multiple

objects of any class, as can be seen in this example for Section 2.1 and 2.1.1. This enables constructing more complex Artifacts rather than being constrained to only the Statement class and it's subclasses. This supports common cases such as when a section contains both statement and requirement class objects, or such as when an artifact, representing an operating procedure built directly from an Action diagram, has sections that contain Action class objects. In a document view, the Headings would show the structure with its section numbering, name and description attributes. The paragraph contents displayed underneath the Heading would include at least the description attribute of each contained object. The name and number of the "contained" object could optionally be displayed.

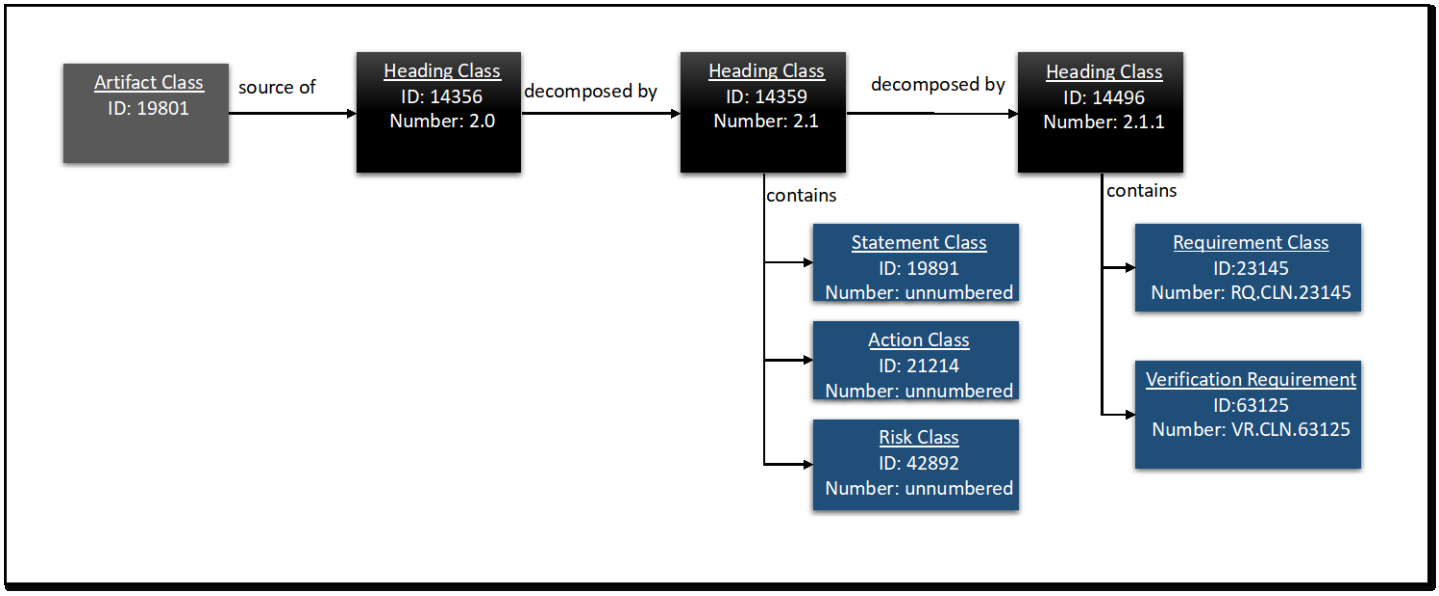

Figure D-1. Proposed New Classes for Structured Artifacts

Appendix E. Program Management Extensions

The purpose of this appendix is to document an approach extending the Program Management capabilities of LML. The basis for this extension comes from research performed for the US Missile Defense Agency (MDA) and the US Navy. This research was converted to the Innoslate® commercial tool as a testbed of implementation. Appendix E documents these extensions so that other tools may benefit from this research and foster interoperability between tools.

LML version 1.0 provided a number of program management-related classes, including: Decision, Risk, Cost, Time (for scheduling and milestones), and Actions (with the Task type and attributes of Start, Duration and Percent Complete). We assume that Start is the actual start day/time. A Planned Start has been added to capture that information as well. The Duration is the estimated duration. For this extension in LML version 1.3, two new subclasses (Task and Dependency) are proposed to be added to the LML schema. These subclasses are defined below:

Class	Parent	Description
Dependency	Connection	A Dependency entity specifies a connection between two Tasks in a Gantt Chart. It defines the relationship a Task depends on another Task in order to Start or Finish.
Task	Action	A Task entity specifies an Action that must be completed for a particular project. It serves as a "To-Do" for the Project.

The Dependency subclass has one additional relationship: "*has dependent*" links a Dependency entity to the new subclass, Task. The table below shows this relationship.

Dependency Relations (in addition to those that are standard for Connection entities)

Name	Target		Description
has dependent	Task		Has dependent identifies the Task that depends on this Task.
	Attribute	**Type**	**Description**
	Origin	Boolean	Origin identifies which Task is depending on another Task for completion.
	Dependency	Enumeration	Dependency identifies if the Origin is at the start or finish of each Task. This capability enables the distinguishing between start-to-finish, finish-to-start, start-to-start, and finish-to-finish.

The Task subclass has several new relationships, as shown below:

Task Relations (in addition to those that are standard for Action entities)

Name	Target	Description
depends on	Dependency	Inverse relationship of has dependent.
scheduled by	Time	Scheduled by identifies a Kanban Column (Time entity) that a Task is planned to take place within.
tracks	Time	Tracks identifies the Time entities that are Kanban Columns in a Kanban Board (Task entity).

The Task subclass also has several attributes as shown below:

Task Properties (additional to those inherited from Action class)

Name	Type	Description
Status	Enumeration	Open, In Progress, in Review, Closed.
Finish	DateTime	Date and time the Task is actually finished.
Estimated Date of Completion	DateTime	Date and time the Task is expected to be completed.
Date Due	DateTime	Date and Time the Task is required to be completed.
Planned Start	DateTime	Date and Time the Task is (or was) planned to begin.
Priority	Enumeration	Priority type identifies the level of urgency for Task completion. Values of: Low Priority, Medium Priority, High Priority.
Type	Enumeration	Work Package, Step, Milestone, Schedule Activity, Baseline, Forecast, Kanban Board, Gantt Chart

Note: In LML 2.0, it is recommended when including the Task subclass in the base ontology that the Start and Percent Complete attributes from the Action class be moved to the Task subclass. This move will reduce potential confusion when using an Action only as a systems engineering functional entity and not also as a programmatic entity.

Appendix F. Application Programming Interfaces

Application Programming Interfaces (APIs) allow multiple computer programs to talk to each other. They act like user interfaces, but instead of connecting computer programs to people, APIs connect computer programs to other computer programs. In this case, Applications refer to any software program with a distinct function, and Interfaces are like contracts of service between two of these Applications. Also, it is important to appreciate the distinction between client programs and server programs when talking about APIs.

The most popular kind of APIs are Representational State Transfer (REST) APIs. These work with a set of functions (e.g., GET, PUT, DELETE, etc.) that clients can use to access data on the server. The server responds to client requests in plain data, without graphical rendering. The server also does not preserve session information from the client (i.e., statelessness). The REST APIs in Innoslate 4.x for entities and schemas are listed below:

Entity

- Fetch entities from the search query
  - Method: GET
  - URL: api/v4/o/:organizationSlug/entities
  - Parameters:
    - query
    - project Id
  - Optional Parameters:
    - limit
    - offset
  - Response: Array of Entity Objects
  - Example: api/v4/o/demo/entities?query="class:Action"&projectId=64&limit=30&offset=0
- Update Entities
  - Method: POST, PUT
  - URL: api/v4/o/:organizationSlug/entities
  - Payload: Array of Entity Objects
  - Payload Example:
    [{"number":"1","labelIds":[],"sortNumber":"000001.00000.0000.000.000.000.000.15639","classId":3,"createdIn":0,"modifiedIn":0,"linkedLabelId":0,"rels":[],"attrs":{},"controlStep":null,"controlType":"SERIAL","branches":[],"diagrams":{},"isArchived":false,"isLocked":false,"followers":["john.doe"],"projectId":64,"globalId":"I_6168Y8QCZCK6S_AKAJ8HC1TC3ES","isRedacted":false,"id":15639,"name":"Asset","description":"","created":1541771921633,"modified":1541771973213,"createdBy":"john.doe","modifiedBy":"john.doe","version":2}]
  - Response: Array of Entity Objects
- Fetch entities by Ids
  - Method: GET
  - URL: api/v4/o/:organizationSlug/entities/:ids

    - Optional Parameters:
        - includeChildren
        - includeArchived
        - includeRelations
        - levels
    - Response: Array of Entity Objects
- Update entities by Ids
    - Method: POST, PUT
    - URL: api/v4/o/:organizationSlug/entities/ids
    - Payload: Array of Entity Object
    - Payload Example: [234, 233]
    - Response: Array of Entity Objects
- Delete entities by Ids
    - Method: DELETE
    - URL: api/v4/o/:organizationSlug/entities/ids
    - Payload: Array of Entity Object
    - Payload Example: [234, 233]
    - Response: Array of Entity Objects
- Transform entities into another class
    - Method: PUT
    - URL: api/v4/o/:organizationSlug/entities/transform/:classId/:ids
    - Response: Array of new transformed Entity Objects
    - Example: api/v4/o/demo/entities/transform/3/9390,9450
- Revert entities to previous version
    - Method: PUT
    - URL: api/v4/o/:organizationSlug/entities/revert/:ids/:versionNumbers
    - Response: Array of reverted Entity Objects
    - Example: api/v4/o/demo/entities/transform/3/9390,9450
- Restore deleted entities
    - Method: PUT
    - URL: api/v4/o/:organizationSlug/entities/restore/:ids
    - Response: Array of reverted Entity Objects
    - Example:
- Auto number entities
    - Method: PUT
    - URL: api/v4/o/:organizationSlug/entities/autonumber/:id
    - Optional Parameters:
        - startNumber

        - singleLevel
        - useControlStep
    - Response: Array of auto numbered Entity Objects
    - Example:

Schema

- Fetch organization schema
    - Method: GET
    - URL: api/v4/o/:organizationSlug/schema
    - Response: Schema Object
- Update organization schema
    - Method: POST, PUT
    - URL: api/v4/o/:organizationSlug/schema
    - Payload: Schema Object
    - Payload Example:
    - Response: Schema Object
- Delete organization schema properties or labels
    - Method: DELETE
    - URL: api/v4/o/:organizationSlug/schema/:type/:ids
    - Response: Schema Object
- Fetch project schema
    - Method: GET
    - URL: api/v4/o/:organizationSlug/p/:projectId/schema
    - Response: Schema Object
- Update project schema
    - Method: POST, PUT
    - URL: api/v4/o/:organizationSlug/p/:projectId/schema
    - Payload: Schema Object
    - Payload Example:
    - Response: Schema Object
- Delete project schema properties or labels
    - Method: DELETE
    - URL: api/v4/o/:organizationSlug/p/:projectId/schema/:type/:ids
- Response: Schema Object

Appendix G. Interface Diagrams

A number of interface diagram types have been suggested over the years. Most of these diagrams come from drawing approach that show different aspects of the interface. LML's Asset Diagram provides part of an interface definition by showing how Assets are connected by Conduits. But more information is needed to completely define an interface, including the *capacity* and *latency*, as well as the Input/Output entities that are transferred across the Conduit. Conduits can also be decomposed, so parent-child relationships need a means of being visualized too. Of course other information, such as protocols used, will more completely define the interface, but at some point the diagrams get to be too cluttered to be useful in helping the stakeholders understanding of the interface. Figure G-1 shows a potential Interface Diagram that displays most of the information needed to define an Interface.

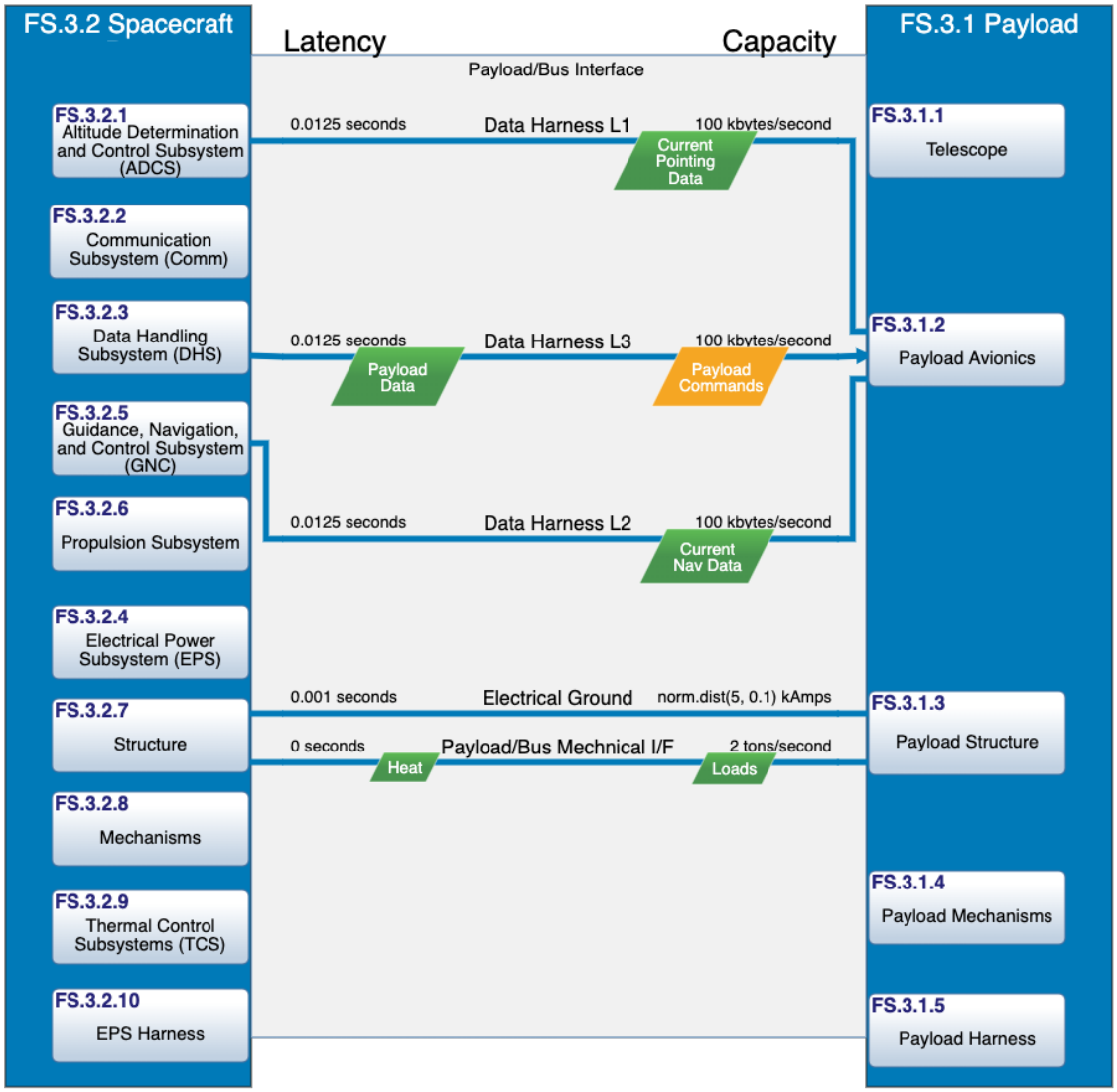

Figure G-1. Interface Control Diagram

This figure shows the two Assets (Spacecraft and Payload) with the Conduit (Payload/Bus Interface) between them. The next level of decomposition for both the Assets and the Conduits are also shown. The child Conduits are attached to the child Assets. Note that an arrow can be used to depict the directionality of the child Conduit. The *latency* and *capacity* values of each child Conduit are shown as well. The green and gold Conduits in this diagram depict the Input/Output entities transferred by the child Conduits. The difference between a green and gold Input/Output in this case indicate a potential error in the *size* units of the Input/Output entity. While error detection is not required for this diagram, such capabilities are desirable in all diagrams, where appropriate.