Enhancing Model-Based Systems Engineering with the Lifecycle Modeling Language

Warren K. Vaneman, Ph.D.
Systems Engineering Department
Naval Postgraduate School
Monterey, CA

***Abstract*— As systems become more complex, the systems engineering community must find new and more efficient ways of dealing with complexity throughout the system's lifecycle. Model-Based Systems Engineering (MBSE) has proven to be effective at managing complexity through the development of systems in a virtual environment. Several languages have been developed in the spirit of MBSE; however, these languages often do not include the full spectrum of information needed for holistic system solutions. The Lifecycle Modeling Language (LML) has been developed to provide extensible language that contains both visualization models and an ontology. When LML is coupled with the Systems Modeling Language (SysML), the result is a modeling constructs that better supports systems engineering processes across the entire spectrum of lifecycle concerns. This coupling will not only be beneficial today, but could serve as a catalyst for a future MBSE environment.**

Keywords*— *Model-Based Systems Engineering; Systems Modeling Language (SysML); Lifecycle Modeling Language (LML)

I. INTRODUCTION

"It is common practice for systems engineers to use a wide range of modeling languages, tools and techniques on large systems projects." [1]

Today's systems are more complex, and change more rapidly than ever before. As sub-systems are added to systems, and systems are added to System of Systems (SoS), interfaces grow nonlinearly. As a result, interfaces and interactions are often difficult to comprehend, thus making requirements almost impossible to fully define, with a cascading effect leading to uncertain, and incomplete, systems tests.

The challenge is to deal with this increased complexity. However, modern systems engineering methods have not kept pace with the ever increasing complexities of systems. As a result, many system engineering activities have been relegated to the beginning of the systems engineering lifecycle. Development of complex systems requires: designs that are easy to understand by all stakeholders; engineering plans that are responsive to change; architectures that are easy to modify; methodologies that are appropriate to the problem at hand; and processes that support all phases of the system's lifecycle. Coupled with these technical planning and design challenges, are shrinking budgets and tightening schedules.

The Systems Engineering Community recognizes the need to evolve the traditional document-based approach to a model-based approach, where the models can be easily tailored to changing conditions and needs, re-used, and are executable to test the static architecture in a dynamic environment. Model-Based Systems Engineering (MBSE) was conceived to help address these system challenges.

Model-Based Systems Engineering is the formalized application of modeling to support systems design and analysis, throughout all phases of the system lifecycle, through the collection of related processes, methods (languages), and tools used to support the discipline of systems engineering in a "model-based" or "model-driven" context. Methods, languages, and frameworks such as the Unified Modeling Language (UML), and the Systems Modeling Language (SysML) have made tremendous progress, but do not address the full scope of the problem because they focus either on the logical construct (visual representation) or an ontology. An ontology is a collection of standardized, defined terms or concepts and the relationships among the terms and concepts [2]. Thus, individually, they are limited in the ability to express the wide range of entities, relationships, and attributes needed in today's systems engineering environment.

The Lifecycle Modeling Language (LML) was introduced as an attempt to provide a simpler language for planning, specifying, designing, analyzing, building, and maintaining modern complex systems. It combines logical constructs with a corresponding ontology, thus expressing a wide range of entities, relationships, and attributes to capture engineering and programmatic information. The goals of LML are [3]:

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 the entire system's lifecycle – cradle to grave;
6. To support both evolutionary and revolutionary system changes to system plans and designs over the lifespan of the system.

The remainder of this paper has three fundamental themes. First, it provides a brief overview of legacy modeling

languages (UML and SysML), and introduces LML. Second, it explores the current state of SysML and LML by comparing it against eight MBSE effectiveness measures. And third, this paper explores the potential of using LML as the ontology for SysML, while expanding while expanding SysML's logical constructs available for systems related disciplines with LML models.

II. Overview Of Systems Engineering Modeling Languages

The Systems Engineering Community recognizes MBSE's ability to evolve, reuse and execute models is a significant improvement over the classic "document-based" approach's static view of a system. These capabilities help ensure a comprehensive and high quality design. 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 [3].

The MBSE environment consists of modeling languages and tools. Languages serve as the basis of tools, and tools enable the development of system models. While the languages serve as the foundation for MBSE tool development, tool vendors often interpret the languages to enable the best implementation for their tools. Thus while a common language is used, the MBSE tools can be very different. As an example, MBSE tools that support SysML support a common set of visual models, but typically have unique data schemas due to SysML not having a defined data schema.

This section provides a brief introduction to the UML, SysML, and LML. While UML is not a language widely used in system engineering, it does serve as the foundation for SysML, therefore a brief introduction is warranted for completeness. To better describe and evaluate these models, MBSE effectiveness measures are discussed first.

A. MBSE Effectiveness Measures

Friedenthal and Burkhart [4] identified eight MBSE effectiveness measures, which can be used to assess modeling languages and tools. Those effectiveness measures are [4]:

- Expressiveness – The ability to express system concepts;
- Precise – System representation is unambiguous and concise;
- Presentation/Communication – Ability to effectively communicate with diverse stakeholders;
- Model Construction – Ability to efficiently and intuitively construct models;
- Interoperable – Ability to exchange and transform data with other models and structured data;
- Manageable – The ability to efficiently manage and change models;
- Usable – The ability for stakeholders to efficiently and intuitively create, maintain, and use the model;
- Adaptable/Customizable – The ability to extend models to support domain-specific concepts and terminology.

While the theme of this paper is to discuss how LML can enhance SysML – primarily a modeling language discussion - a casual consideration of how tools will be affected by the languages needs to occur. The eight effectiveness measures will be used in the following sub-sections to provide a cursory evaluation of the current state of SysML and LML, and where applicable how the language applies to MBSE tools.

B. Unified Modeling Language (UML)

Developed by the Object Management Group (OMG) to serve the software engineering community, UML is a standardized general-purpose, object-oriented, modeling language, divided into fourteen structural and behavior models that represent the interactions within software systems. The seven structure diagrams emphasize those elements that must be present to make the system complete. The structure diagrams are used extensively in documenting the software architecture of the system. The seven behavior models emphasize what must happen within the system being modeled. The behavior models are used extensively to describe the functionality of software systems. Fig. 1 shows the UML model hierarchy [5]. The shaded boxes are models that were adopted for SysML.

C. Systems Modeling Language (SysML)

System Modeling Language (SysML) is a general-purpose graphical modeling language for specifying, designing, analyzing, and verifying systems that may include hardware, software, people, facilities, information, and procedures [4].

Given the success of UML, OMG developed SysML for the Systems Engineering Community. Hence, SysML is a profile of UML in that it extends UML, as shown in the Venn Diagram in Fig. 2 [6]. SysML uses seven of the fourteen models from UML, plus two new models based on the needs of the systems engineering discipline. These models support requirements specification, analysis, design, validation and verification, for systems that include hardware, software, information, process, and people [1].

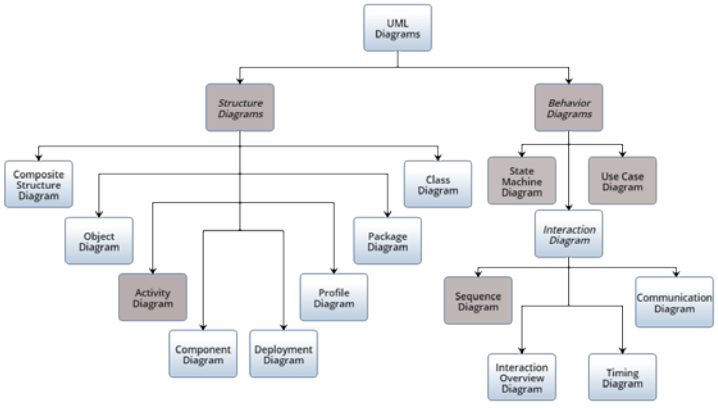

Fig. 1. UML Diagram Taxonomy [5].

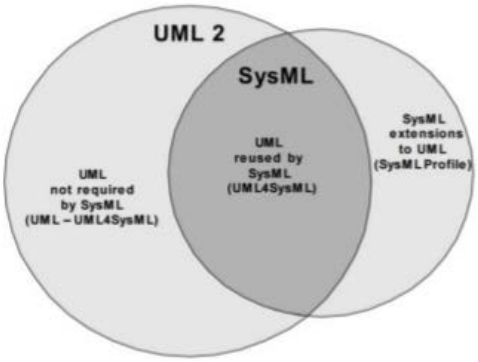

Fig. 2. UML and SysML Venn Diagram [7].

Like UML, SysML has included the behavior and structural models, plus a requirements diagram and parametric models. The diagrams contained in the behavior diagrams include [1]:

- State Machine Diagram – Describes the states and the state transitions of the system;
- Activity Diagram – Describes operational flows of components within the systems;
- Sequence Diagram – Depicts how system elements (objects) communicate with each other in terms of a sequence of messages and events;
- Use Case Diagram - Depicts the system's functionality in terms of actors, dependencies between use cases, and use case goals.

The diagrams contained the structural diagram are [1]:

- Block Definition Diagram – Depicts the principal parts of a system as a series of blocks, with interconnections to represent the relationships;
- Package Diagram – Describes how a system is divided into logical groupings by showing the dependencies among the groupings;
- Internal Block Diagram – Describes the internal structure of a system in terms of component properties, and the relationship of the constituent parts.

The requirements diagram is used to describe system requirements through visual models. The parametric diagram is the newest diagram to SysML, and represents engineering analysis within the design models [1]. The nine models are shown the SysML Diagram Taxonomy in Fig. 3 [1].

When evaluated against the eight MBSE effectiveness measures, SysML is limited to the nine models that are currently included in the language. The lack of an ontology hinders SysML from representing precise complex system relationships and interactions, and forces tool developers to define data schemas for tool execution. The complete evaluation is in Table 1.

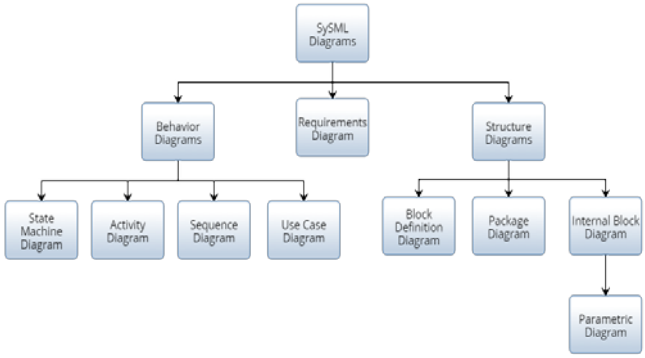

Fig. 3. SysML Diagram Taxonomy [1].

D. Lifecycle Modeling Language

The Lifecycle Modeling Language (LML) was developed as an approach for incorporating a logical construct and ontology within the same framework. LML defines a logical construct using visualization models in four areas: functional models; physical models; documentation entities; and parametric and program entities (Fig. 3). Many of the LML models are equivalent to the familiar models that have been developed over time by UML, SysML, Business Process Modeling Notation (BPMN), and other engineering disciplines such as electrical engineering. The LML Specification 1.1 contains a subset of the possible visualizations [3].

LML uses a simplified ontology of 12 primary entities and eight secondary entities to capture the system characteristics, relationships, and interactions. LML's primary and secondary entities are [3]:

- Asset – An Asset entity specifies an object, person, or organization that performs Actions, such as a system, sub-system, component, person, or element. Asset is the parent to the Resource child entity (e.g. component, service, sub-system, system);
- Resource – A Resource is a child entity of Asset that specifies a consumable or producible Asset (e.g. fuel, ammunition, people);
- Characteristic – A Characteristic entity specifies properties of an entity. Characteristic is the parent for the Measure child entity (e.g. category, attribute, power, size, weight);
- Measure – A Measure is a child entity to Characteristic that specifies properties of measurement and measuring methodologies, including metrics (e.g. Measures of Effectiveness (MOE), Key Performance Parameters (KPP), Measures of Performance (MOPS));
- Connection – A Connection entity specifies the means for relating Asset instances to each other. Connection is the parent to the Conduit and Logical child entities (e.g. abstract entity);

Table 1. SysML Current State Evaluated Against the MBSE Effectiveness Measures.

Effectiveness Measure	SysML Current State
Expressiveness	Nine models allow for many system concepts to be expressed. However, a robust data model needs to be defined to more effectively capture and expand the system concepts.
Precise	Models provides semantics (meaning) and notation (representation of meaning), which could lead to interpretation. No ontology currently exists, therefore it lacks the expression of entities, relationships, and attributes needed to fully describe the system.
Presentation/ Communication	Views and viewpoints are limited to the nine current models. Better visualizations and flexibility to convey the message to wider stakeholder set it needed.
Model Construction	Tool dependent. Current language does not inherently promote efficient model construction.
Interoperable	Tool dependent. Current language does not contain a standard API, therefore limits the inherent ability to be interoperable.
Manageable	The ability to track changes is a function of individual tools, and is not a characteristic of SysML *per se*. However, implementation in a cloud environment will assist in managing the model when multiple, and possible geographically disperse users are engaged on a single model.
Usable	SysML has proven to be arduous, therefore has primarily been limited to engineers.
Adaptable/ Customizable	The nine models are predefined, and do not easily allow for extendibility into other domains.

- Conduit – A Conduit is a child entity to Connection that provides a means for physically transporting Input/Output entities between Asset entities. Conduits are constrained by the attributes of capability and latency (e.g. data bus, communications interface, and pipe);
- Logical – A Logical entity is a child entity to Connection that represents the abstraction of the relationship between any two entities (e.g. "Has", "is a", "relates to");
- Cost – A Cost entity specifies the outlay of expenditure made to achieve an objective associated with another entity (e.g. Earned Value, Work Breakdown Structure, actual cost, planned cost);

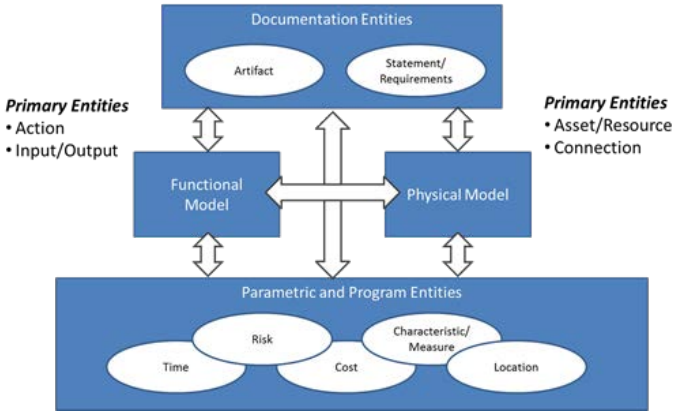

Fig. 3. Categories of LML Models [3].

- Decision – A Decision entity specifies a challenge and its resolution (e.g. major decision, challenge, issue, problem);
- Input/Output – An Input/Output entity specifies the information, data, or object input to, trigger, or output from an Action (e.g. item, trigger, information, data, matter, energy);
- Location – A Location entity specifies where an entity resides. Location is the parent entity for Orbital, Physical, and Virtual child entities (e.g. abstract entity);
- Orbital – An Orbital entity is a child entity of Location that specifies a location along an orbit around a celestial body (e.g. orbit, ephemeris);
- Physical – A Physical entity is a child entity of Location that specifies the location on, above, or below the surface (e.g. geospatial coordinates, altitude, depth);
- Virtual - A Virtual entity is a child entity of Location that specifies a location within cyber space or a physical network (e.g. Uniform Resource Locator);
- Risk – A Risk entity specifies the combined probability and consequence in achieving objectives (e.g. cost risk, schedule risk, technical risk);
- Statement – A Statement entity specifies text referenced by the knowledgebase and is usually contained in an Artifact. Statement is the parent entity to the Requirement child entity (e.g. need, goal, objective, assumption) ;
- Requirement – A Requirement is a child entity to Statement that identifies a capability, characteristic, or quality factor of a system that must exist for the system to have value and utility to the system or user (e.g. functional requirement, performance requirement, safety requirement);
- Time – A Time entity specifies a point or period when something occurs or during which an action, asset, process, or condition exists or continues (e.g. milestone, phase).

Fig. 4 shows LML's principal entities and relationships. LML was designed to support the systems engineering lifecycle (i.e. the traceability of requirements to their implementation to system elements (Assets)) [3]. The systems engineering process begins with the communication of a need, or capability, typically in the form of an authoritative document (Artifact). Statements of need (Statements or Requirements) are contained in the documents, and serve as the basis for the beginning of the systems architecting and engineering process. Functional models define operational activities (Actions) that the system must perform based on the system's inputs, controls, and outputs (Input/Output). Operational activities are performed by systems, sub-systems, and components (Asset or Resources) and are connected through a series of physical interfaces (Connection or Conduit). Systems have physical properties (e.g. size, weight, power required) that are described by Characteristic or Measure entities.

LML Specification 1.1 introduced the addition of one class (Equation) and one subclass – Port – as a subclass of Asset, to visualize the complete set of SysML models. The definitions of these proposed LML entities are [3].

- Equation - An Equation entity specifies a mathematical or logical equation that can be used to describe a portion of the model;
- Port - A Port entity is a child entity of Asset that represents an interaction point of a block, specifying the input and output flow.

A fundamental tenet of LML is that each entity has at least one corresponding visualization. The language contains three mandatory diagrams: Action diagrams for functional modeling; Asset diagrams for physical modeling; and, Spider diagrams for traceability. All other visualizations are optional, but recommended, to represent the full set of LML entities. Additional visualizations are possible due to the flexibility and extensionability of LML. Table 2 provides of a list of possible visualization models that can be used to represent LML entities [3].

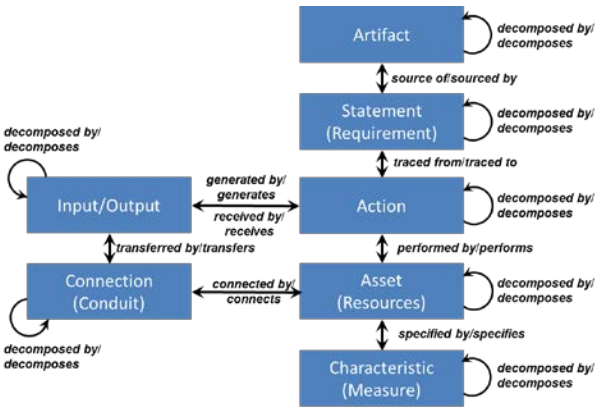

Fig. 4. LML Principal Entities and Relationships [3].

Evaluating LML against the eight effectiveness measures (Table 3) depicts a language that is more versatile than other MBSE languages since it contains both visualization models and ontology. This allows the system concepts to more precisely and efficiently defined. However, the LML Standard [3] is silent with respect to application programming interfaces (API) – instead leaving the API for tool developers to implement.

III.MAPPING SysML to LML

A mapping between the SysML diagrams and the LML entities (Table 4) will provide SysML with the long-sought ontology to more fully represent system types, properties, and interrelationships. This process begins with exploring commonality among the SysML and LML visualization models, and creating a mapping between those models. Once mapped, the LML visualization models can be associated to the corresponding LML entity, and by extension provides an ontology for SysML. Providing this ontology will prove important to practitioners as they will be able to better represent the complexities of a system.

Coupling SysML with LML could immediately serve as the foundation for MBSE tool vendors to have a common SysML data schema, instead of the individually defined solutions that currently exist. This coupling will allow SysML visualizations to more precisely represent system concepts, especially where complexity is involved, and allow for better presentation of

Table 2. Suggested Visualization Models to Represent LML Entities.

LML Entity	LML Model
Action	Action Diagram
Artifact	Photo, Diagram, etc.
Asset	Asset Diagram
Resource (Asset)	Asset Diagram
Port (Asset)	Asset Diagram
Characteristic	State Machine, Entity-Relationship, and Class Diagrams
Measure (Characteristic)	Hierarchy, Spider, and Radar Charts
Connection	Asset Diagram
Conduit (Connection)	Asset Diagram
Logical (Connection)	Entity-Relationship Diagram
Cost	Pie/Bar/Line Charts
Decision	
Input/Output	State Machine Diagram
Location	Map
Physical (Location)	Geographic Maps
Orbital (Location)	Orbital Charts
Virtual (Location)	Network Maps
Risk	Risk Matrix
Statement	Hierarchy and Spider Charts
Requirement (Statement)	Hierarchy and Spider Charts
Time	Gantt Chart, Timeline Diagram
Equation	*Equation*

Table 3. LML Current State Evaluated Against the MBSE Effectiveness Measures.

Effectiveness Measure	LML Current State
Expressiveness	Provides both visualizations and an ontology to express a wide variety of system and project concepts.
Precise	Visualizations coupled with the ontology allows for precise system representations.
Presentation/ Communication	Multiple visualization models allow for communication with diverse stakeholders.
Model Construction	Tool dependent. The well-defined ontology and visualization will allow models to be constructed efficiently.
Interoperable	Tool dependent. Current language does not contain a standard API, therefore limits the inherent ability to be interoperable.
Manageable	Tool dependent. The ability to track changes is a function of individual tools, and is not a characteristic of LML per se. However, implementation in a cloud environment will assist in managing the model when multiple, and possible geographically disperse users are engaged on a single model.
Usable	Whether developing models visually or populating a data base, LML has been shown to be easily usable due to the simplified language of 12 primary entities and eight child entities.
Adaptable/ Customizable	LML is a flexible language that allows for the extension of ontology or visualization models to support multiple domains.

data. Table 5(a-b) presents the potential immediate benefits for MBSE using SysML and LML together, and considers how this coupling could be beneficial towards progressing MBSE to an immediate next step and future MBSE state.

Two organizations The International Council on Systems Engineering (INCOSE) Systems Engineering Vision 2025 [7, p. 38] describes the future MBSE state as:

> "Formal systems modeling is a standard practice for specifying, analyzing, designing, and verifying systems, and is fully integrated with other engineering models. System models are adapted to the application domain, and include a broad spectrum of models for representing all aspects of systems. The use of internet-driven knowledge representation and immersive technologies enable highly efficient and shared human understanding of systems in a virtual environment that span the full lifecycle from concept through development, manufacturing, operations, and support."

Table 4. Mapping of SysML Diagrams to LML Diagrams and Entities [2].

SysML Models	LML Models	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

The OMG Systems Engineering Domain Special Interest Group (SEDSIG) has chartered the System Modeling Assessment and Roadmap Working Group to determine how SysML is supporting MBSE, and define a roadmap for the future state of MBSE. To date, 11 high-level requirements have been defined, that describe future state for both MBSE tolls and languages [4]. While conducting the assessment for the potential immediate benefits of coupling SysML and LML in Table 5(a-b), these future requirements were considered.

IV. CONCLUSION

The Lifecycle Modeling Language defines a new approach to MBSE that simplifies the ontologies and logical constructs found previous MBSE methods and languages. Coupling SysML and LML provides an environment with an ontology that allows system concepts to be better represented by denoting underlying properties, relationships, and interrelationships. LML provides a means to improve how we model system functionality to ensure functions are embedded in the design at the proper points and captured as part of the functional and physical requirements needed for design and test.

Table 5a. SysML and LML Combined State Evaluated Against the MBSE Effectiveness Measures.

Effectiveness Measure	SysML & LML Combined State
Expressiveness	Provides an ontology to accompany the visualization models to represent system concepts. The LML schema provides the framework to extend SysML representations to include certain program management concepts. Needs to further be expanded to represent a wider array of engineering concepts.
Precise	Existing SysML visualizations are better defined by linking it with the LML ontology, and allows for design and requirements changes to be propagated throughout the model. However, this language linkage alone does not achieve a state where models can be validated to be logically consistent, nor does it assess the impact of design and requirements changes on the system.
Presentation/ Communication	LML's ability to support a larger stakeholder set could allow SysML to be expanded to better communicate. However, the visualizations will needs to be expanded to include advanced visualization such as representing a dynamic environment that is orchestrated through simulation.
Model Construction	The ease of model construction will continue to be tool dependent. However, adding an ontology will allow more compressive system relationships to be represented, visualization models to be generated from a database, or a database to be developed from visualizations.
Interoperable	Neither SysML nor LML contain a standard APIs, so interoperability will depend on tool developers adhering to an open architecture standard. Future modeling environments will require an API be present within the language.

Table 5b. SysML and LML Combined State Evaluated Against the MBSE Effectiveness Measures.

Effectiveness Measure	SysML & LML Combined State
Manageable	While management of the models will continue to be tool dependent, coupling the visualization models with the ontology will allow relationships to me made that were not heretofore possible with SysML alone. LML has been implanted in a cloud environment allowing for more efficient collaboration and change management when multiple, geographically disperse users to engage on a single model. Future instantiations of the language will also require single models to be developed by multiple tools.
Usable	Using SysML and LML together will enhance usability due to the visualizations being linked with an ontology. LML is a simplified language that is usable by technical and nontechnical disciplines alike. However, significant usability improvements are tool developer dependent.
Adaptable/ Customizable	Combining the SysML visualizations with LML allows the engineering models to be developed, visualized, and analyzed for the systems engineering and program management domains. The ontology and visualizations will need to be expanded to represent other domains.

REFERENCES

[1] Object Management Group, "OMG Systems Modeling Language (OMG SysML), Version 13," June 2012, www.omg.org/spec.SysML /20120401'SysML.xm1.

[2] S.H. Dam, "What is a model? Understanding the first word in MBSE", 16th Annual Systems Engineering Conference, National Defense Industrial Association, 2015.

[3] Lifecycle Modeling Language Steering Committee, "Lifecycle Modeling Language (LML) Specification, Version 1.1," December 1, 2015, www.lifecyclemodeling.org.

[4] S. Friedenthal and R. Burkhart, "Evolving SysML and the Systen Modeling Environment to support MBSE"., Insight v.18, no. 2, pp. 39-41, International Council on Systems Engineering, August 2015.

[5] Object Management Group, "OMG Unified Modeling Language (OMG UML) Superstructure Version 2.4.1," 9 April 2014. http://www.omg.org/spec/UML/2.4.1/Superstructure/PDF

[6] S.H. Dam, "LML to SysML and back: A Look into the Lifecycle Modeling Language and System Modeling language", SPEC Innovations webinar, 2016.

[7] Internation Council on Systems Engineering, "INCOSE Systems Engineering Vision 2025," International Council on Systems Engineering, p.38, 2014.