Automatically Generating Ontologies: A Case Study Examining the Lifecycle Modeling Language (LML)

Abstract

This study explores versioning of the Lifecycle Modeling Language (LML) ontology to address methods of constructing an unambiguous terminology. LML is a robust graphical modeling language that supports system development and design while adhering to systems engineering (SE) processes. Part of speech (POS) tagging is implemented with Python to extract ontological classes and object properties. Based on the research conclusions, innovative ontology techniques are established for the wider technical community to use and improve. These new methods will contribute to the modernization of manual approaches to creating ontologies. This paper/presentation begins by introducing key concepts based on a scholarly literature review. Research means will be described in detail with a subsequent discussion of limitations and results. After analysis, conclusions and recommendations will be provided. Future work is anticipated to contribute to ontology development.

Introduction

Systems engineering (SE) is responsible for delivering artifacts during product development and design that enable unambiguous communication among stakeholders. By constructing structural and behavioral models in a virtual environment, errors are caught earlier within the project lifecycle and can be mitigated with minimal effect of cost. A model that is viewed as an authoritative-source-of-truth (ASoT) enhances cross-discipline communication and encourages team collaboration.

There is a need to increase the speed in which formal ontologies are created. This research leverages the existing Lifecycle Modeling Language (LML) ontology to understand ways that modern technology enhances language development based on textual specifications.

Background

LML is an open-standard modeling language for SEs that supports the full product lifecycle and the integration with other project disciplines (Vaneman, et al., 2018). LML incorporates data-driven engineering into a standard that satisfies the following SE needs:

- easy to understand
- easy to extend
- supports functional and object-oriented approaches
- useful for stakeholders across the system life cycle
- supports all project life cycle stages from concept to disposal

Figure 1 highlights the four (4) core LML tenets.

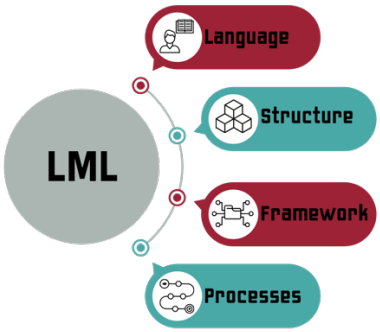

Figure 1 Essential LML tenets.

LML seeks to address discrepancies inherent to modeling languages that were created without a foundational ontology. The language represents functional and physical system aspects and introduces a hierarchy of classes that enable practitioners to capture essential design elements and the associated information (Vaneman, et al., 2018).

Modeling languages capture a formalized ontology (i.e., internal logic) to connect relationships and facilitate data sharing between tools (Vaneman, et al., 2018). Without clear agreement on basic terminology and relationships, a system becomes susceptible to failure (Madni & Sievers, 2018). A common characteristic of ontology engineering methods is the notion of a process model that guides activities to yield a consistent set of roles and policies (Simperl & Luczak-Rösch, 2014). Research performed by Yang et al., 2019, describes how ontologies benefit the SE discipline. The authors conclude that ontologies provide a uniform and consistent basis for knowledge representation.

Ontologies are viewed as the interface between the knowledge base and reality that guides information shareability, acquisition, and organization (Kang, et al., 2010). According to Noy & McGuinness, 2001, reasons for ontology development include:

- Creating a common understanding
- Enabling reuse of domain knowledge
- Explicitly defining domain assumptions
- Analysis of domain knowledge

A critical aspect of ontologies is the ability to execute data reasoners that produce semantic relationships, which is made possible due with formal logic languages that represent ontologies (Bravo, et al., 2019). Ontologies are the primary and most important component of the semantic web (Sattar, et al., 2020). The semantic web is a network of linked data (LD) and its exchange as defined by WC3. Mediator systems federate and integrate data from disparate sources to elicit information that cannot be provided by an individual source (Ludascher, et al., 2001). Semantic web concepts provide opportunities for an easier and more collaborative environment between teams and modeling tools (Jacobs, et al., 2014).

The LML taxonomy relies on the basic foundational ontology (BFO), a high-level ontology developed and designed to represent common categories of domain-specific languages (DSL) [Arp & Smith, 2011]. This specification is the basis for modern ontology languages. BFO gives structure to domain ontologies and is comprised of (1) continuants: entities that continue or persist through time, and (2) occurrents: the events in which continuants participate. Figure 2 shows the decomposition of the BFO continuant class.

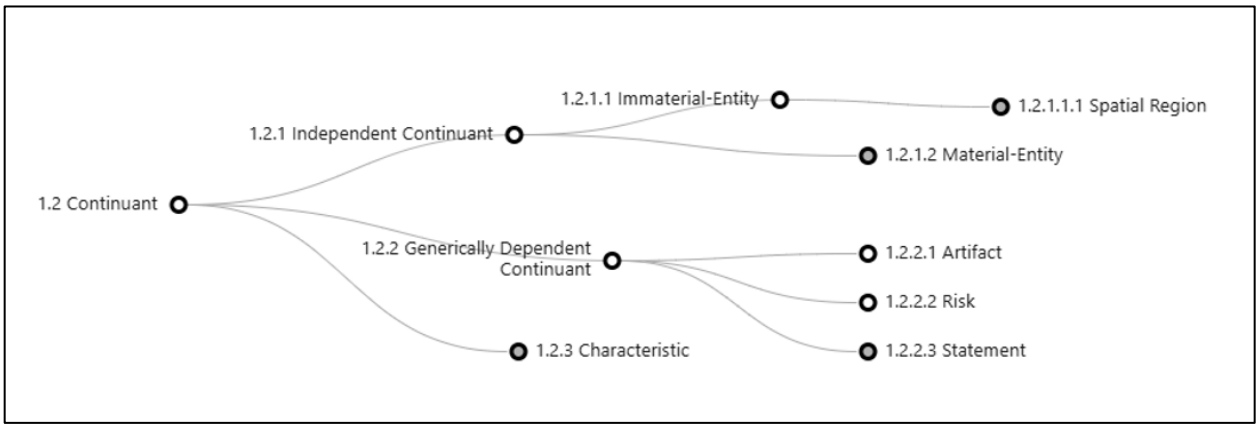

Figure 2 BFO continuants.

The Web Ontology Language (OWL) is a syntax independent language that has several common representations (Drummond, 2006). OWL ontologies map to resource description framework (RDF) graphs and include annotations of classes and properties. OWL can enhance both precision and accessibility, but it requires mediation to enhance model semantics and resolve conflicts (Kulvatunyou, at al., 2014). RDF provides a common approach for expressing information to prevent data exchange between software applications from losing meaning (W3C, 2004). Each entity is given a unique internationalized resource identifier (IRI) to ensure statements are machine-readable. RDF defines all data as triples composed of a subject, predicate, and object (SPO) [Ernadote, 2015]. Each RDF triple represents a statement (i.e., fact) of a relationship between the subject and the object (Angles & Gutierrez, 2008). The SPO approach aligns with a graph-based data model profile that uses nodes and edges to convey domain structure.

Terse RDF Triple Language (Turtle) allows an RDF graph to be written in a compact, natural text form (OMG, 2014) and facilitates the automatic transformation of data by direct mapping. Python has the ability to create Turtle files that are interpretable by ontology editors and include classes, object properties, and data properties (Permatasari & Jayadianti, 2022). Given the similarities between ontologies and relational databases (RDB) [Zhang & Li, 2011], there is an expectation that this information can be transformed directly into axioms by directly mapping the corresponding fields. An *axiom* defines concepts that are always true (Blázquez, et al., 1998). *Classes* represent domain concepts while *object properties* describe the relationships between classes (Noy & McGuinness, 2001).

A natural language processing (NLP) token is a group of characters (i.e., string) that represents a single meaning. Part-of-speech (POS) tagging is a grammatical classification used in NLP to identify individual words as a noun, adjective, verb, etc (Chiche & Yitagesu, 2022). The Natural Language Toolkit (NLTK) within Python is a platform that works with human-interpretable languages libraries that support tokenization, parsing, and semantic reasoning (Bird, et al., 2009).

Methodology

The NLTK library was leveraged to organize the ontology data related to POS. For this research, modern AI applications (e.g., ChatGPT, Claude.ai) were prompted to code Python scripts that will assist in ontology construction. The PyCharm integrated development environment (IDE) was used with Python to extract nouns (i.e., classes) and verbs (i.e., object properties). The LML v1.4 specification was loaded into the virtual environment for NLP parsing. Plurals of nouns were removed from the findings and

variations of verbs were removed (e.g., “working” instead of “work”). Python proved able to produce a Turtle (.ttl) file with classes, object properties, and inverse object properties with proper prompt engineering.
The initial ontology (http://www.semanticweb.org/stevendam/ontologies/2017/4/lifecyclemodelinglanguage.owl) is based on LML v1 (LML Steering Committee, 2015) and was last updated in 2017 according to the documentation.

Figure 3 visually represents the LML v1 taxonomy.

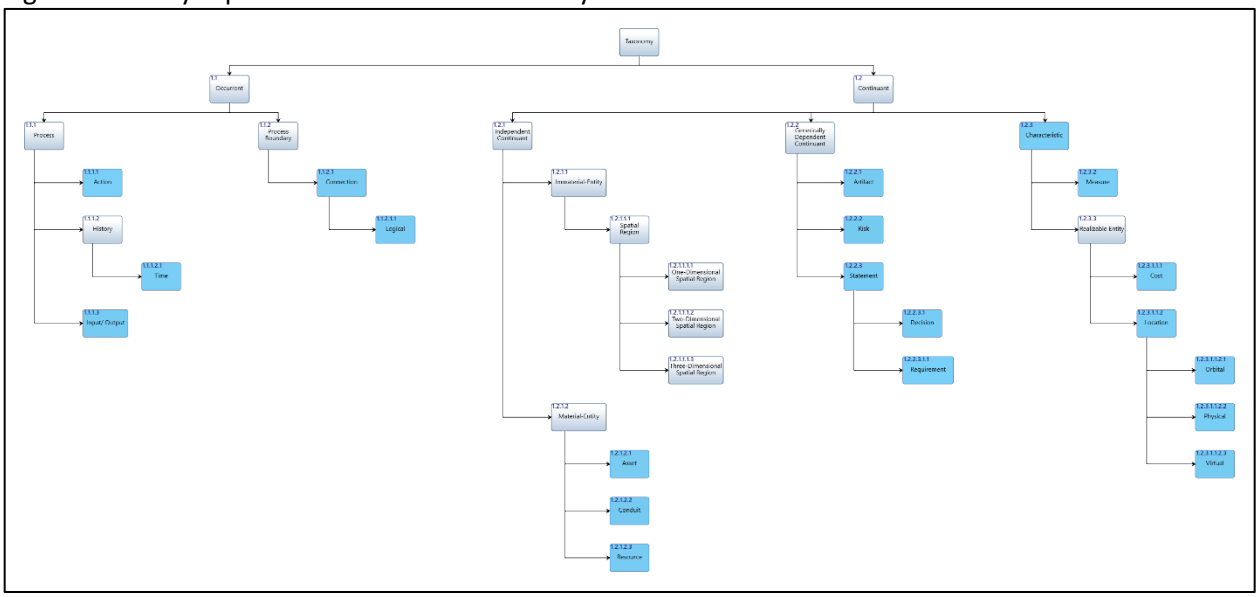

Figure 3 LML v1 taxonomy

Table 1 shows basic metrics of the LML v1.1 ontology.

Table 1 LML v1 Ontology Metrics

METRIC	AMOUNT
AXIOM	1011
CLASSES	20
OBJECT PROPERTIES	87

Figure 4 shows the twenty (20) core LML classes as *nodes* within the Protégé ontology editor using the Ontograf plug-in.

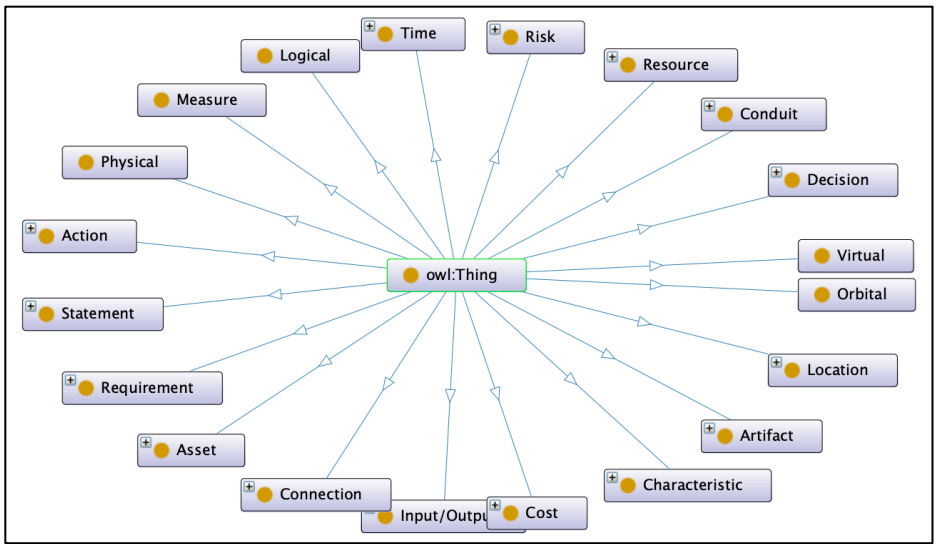

Figure 4 LML core entities represented as nodes.

Figure 5 shows a subset of LML core entities represented as *nodes* and the relationships between them as *arcs*. The graphic exemplifies the complexity of *predicates* between *subjects* and *objects.*

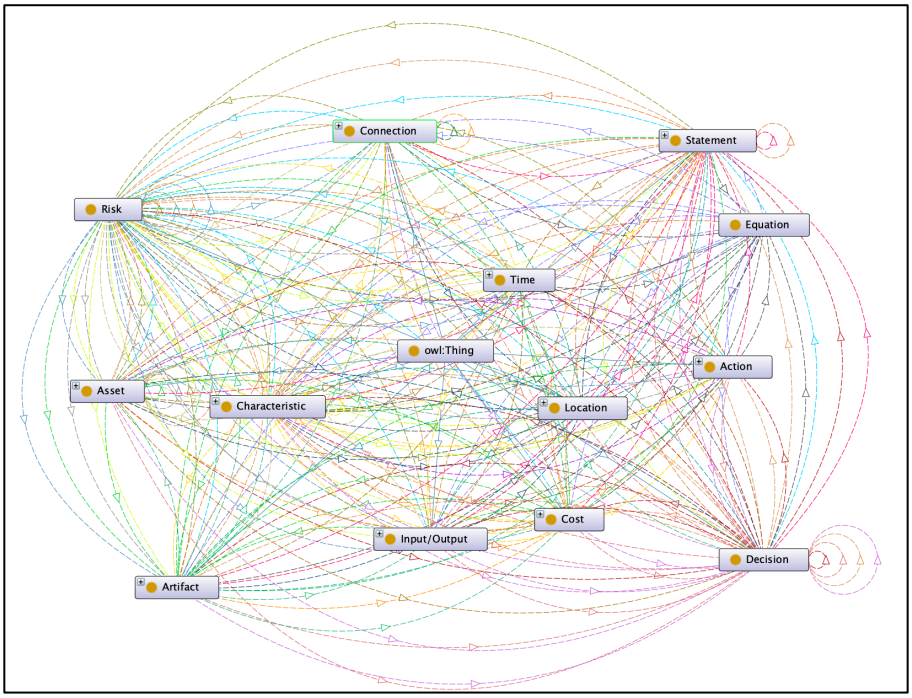

Figure 5 Subset of LML entities with relationships.

Relationships (i.e., predicates) are shown as *arcs* connecting the *nodes* (i.e., subjects and objects).

Results

After brief cleaning of the .csv files output by the Python code, the entire LML v1.4 specification document resulted in 426 classes and 278 predicates (and therefore 278 inverse predicates) for a total of 556 object properties. Applying the maximum multiplier (~21) and the minimum multiplier (~7) to the

number of axioms resulted in an estimated 7,077 and 21,231, respectively. Applying the ~14 average to the initial number of axioms, approximately 14,000 are present in the specification itself as shown in Table 2.

Table 2 Corrected LML v1 ontology metrics.

METRIC	INITIAL	FINAL	MULTIPLIER
AXIOM	1,011	14,000	13.8
CLASSES	20	426	21.3
OBJECT PROPERTIES	87	556	6.4

The vast difference between the LML v1 specification and the official OWL file is not unique, yet highlights universal difficulties encountered when constructing modeling languages that meet stakeholder needs while being machine and human interpretable to enhance collaboration and communication.

Python code is currently being developed to extract RDF triples directly from a .txt file and each inverse statement. The first viable iteration of this implementation resulted in almost 115,000 triples (and therefore an equivalent amount of inverse triples), which suggests an exponential increase in triples based on classes and object properties within an ontology shown in Table 3.

Table 3 Calculated LML v1 ontology metrics.

METRIC	INITIAL	FINAL	MULTIPLIER
AXIOM	1,011	230,000	227.5
CLASS	20	426	21.3
OBJECT PROPERTIES	87	556	6.4

Figure 6 further demonstrates the exponential pattern involved in the axiom relationship with the iterations of varying classes and object properties.

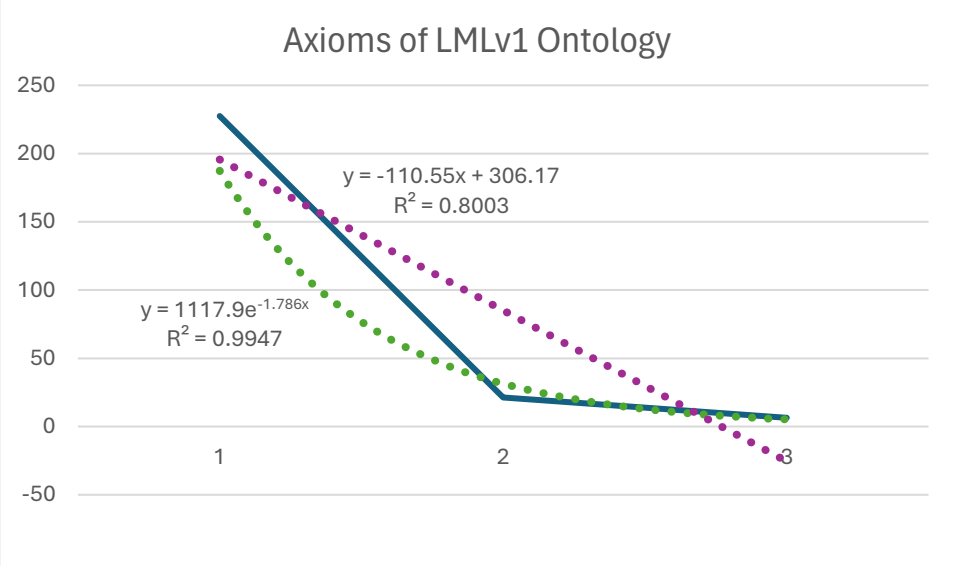

Figure 6 Trendlines of LML v1 ontological axiom data.

Refinement to this approach is ongoing to reduce noise and clarify explicit triples within the LML specification. Future research will apply this methodology to assess additional modeling languages specific to the SE domain.

References

Angels, R., & Gutierrez, C. (2008). Survey of graph database models. *ACM Computing Surveys*, *40*(1), 1–39. https://doi.org/10.1145/1322432.1322433

Arp, R., & Smith, B. (2011). *Realizable Entities in Basic Formal Ontology*.

Bird, S., Loper, E. & Klein, E. (2009), *Natural Language Processing with Python*. O'Reilly Media Inc.

Blázquez, M., Fernández, M., García-Pinar, J. M., & Gómez-Pérez, A. (1998). *Building Ontologies at the Knowledge Level using the Ontology Design Environment*.

Bravo, M., Hoyos Reyes, L. F., Reyes Ortiz, J. A., Bravo, M., Hoyos Reyes, L. F., & Reyes Ortiz, J. A. (2019). Methodology for ontology design and construction. *Contaduría y Administración*, *64*(4). https://doi.org/10.22201/fca.24488410e.2020.2368

Chiche, A., & Yitagesu, B. (2022). Part of speech tagging: A systematic review of deep learning and machine learning approaches. *Journal of Big Data*, *9*(1), 10. https://doi.org/10.1186/s40537-022-00561-y

Dam, S. (2022). *Lifecycle Modeling Language (LML) SPECIFICATION*.

Drummond, N. (2006). *A Practical Introduction to Protégé OWL*.

Ernadote, D. (2015). *An ontology mindset for system engineering* (p. 460). https://doi.org/10.1109/SysEng.2015.7302797

Feuerriegel, S., Hartmann, J., Janiesch, C., & Zschech, P. (2024). Generative AI. *Business & Information Systems Engineering*, *66*(1), 111–126. https://doi.org/10.1007/s12599-023-00834-7

Jacobs, S., Wengrowicz, N., & Dori, D. (2014). Defining Object-Process Methodology in Web Ontology Language for Semantic Mediation. *2014 IEEE International Conference on Software Science, Technology and Engineering*, 87–95. https://doi.org/10.1109/SWSTE.2014.14

Kang, D., Lee, J., Choi, S., & Kim, K. (2010). An ontology-based Enterprise Architecture. *Expert Systems with Applications*, *37*(2), 1456–1464. https://doi.org/10.1016/j.eswa.2009.06.073

Kulvatunyou, B., Ivezic, N., Lee, Y., & Shin, J. (2014). An analysis of OWL-based semantic mediation approaches to enhance manufacturing service capability models. *International Journal of Computer Integrated Manufacturing*, *27*(9), 803–823. https://doi.org/10.1080/0951192X.2013.834477

Ludascher, B., Gupta, A., & Martone, M. E. (2001). Model-based mediation with domain maps. *Proceedings 17th International Conference on Data Engineering*, 81–90. https://doi.org/10.1109/ICDE.2001.914816

Madni, A. M., & Sievers, M. (2018). Model-Based Systems Engineering: Motivation, Current Status, and Needed Advances. In A. M. Madni, B. Boehm, R. G. Ghanem, D. Erwin, & M. J. Wheaton (Eds.), *Disciplinary Convergence in Systems Engineering Research* (pp. 311–325). Springer International Publishing. https://doi.org/10.1007/978-3-319-62217-0_22

Nargesian, F., Samulowitz, H., Khurana, U., Khalil, E. B., & Turaga, D. (2017). Learning Feature Engineering for Classification. *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, 2529–2535. https://doi.org/10.24963/ijcai.2017/352

Noy, N. F., & McGuinness, D. L. (2001). *Ontology Development 101: A Guide to Creating Your First Ontology*.

Sattar, A., Salwana, E., Surin, M., Ahmad, M., Malaysia, U., Bangi, Kfueit, Khan, R., Ahmad, M., & Mahmood, A. K. (2020). Comparative Analysis of Methodologies for Domain Ontology

Development: A Systematic Review. *International Journal of Advanced Computer Science and Applications*, *11*. https://doi.org/10.14569/IJACSA.2020.0110515

Simperl, E., & Luczak-Rösch, M. (2014). Collaborative ontology engineering: A survey. *The Knowledge Engineering Review*, *29*(1), 101–131. https://doi.org/10.1017/S0269888913000192

Vaneman, W. K. (2016). Enhancing model-based systems engineering with the Lifecycle Modeling Language. *2016 Annual IEEE Systems Conference (SysCon)*, 1–7. https://doi.org/10.1109/SYSCON.2016.7490581

Vaneman, W. K. (2018). Evolving Model-Based Systems Engineering Ontologies and Structures. *INCOSE International Symposium*, *28*(1), 1027–1036. https://doi.org/10.1002/j.2334-5837.2018.00531.x

Vaneman, W.K., Sellers, J.J. & Dam, S.H. (2018). *Essential LML: Lifecycle Modeling Language (LML): a Thinking Tool for Capturing, Connecting and Communicating Complex Systems.* SPEC Innovations.

Yang, L., Cormican, K. & Yu, M. (2019). Ontology-based systems engineering: A state-of-the-art review. *Computers in Industry*, *111*, 148–171. https://doi.org/10.1016/j.compind.2019.05.003

Yang, L., Cormican, K. & Yu, M. (2021). Ontology Learning for Systems Engineering Body of Knowledge. *IEEE Transactions on Industrial Informatics*, *17*(2), 1039–1047. https://doi.org/10.1109/TII.2020.2990953