

An Ontology Metamodel for Knowledge-Centric Systems Engineering in Practice^{*}

Jose Luis de la Vara^{1,*,†}, Pablo Mason^{1,†} and Clara Ayora^{1,†}

¹ Universidad de Castilla-La Mancha, Avda. España s/n, 02071 Albacete, Spain

Abstract

Knowledge-Centric Systems Engineering (KCSE) is an approach to systems and software engineering that advocates the development and use of knowledge bases that represent system domains. An example of industrial platform that implements it is SES Engineering Studio (SES), which uses ontologies in a distinct way. Although SES principles have been available since over 20 years ago, a detailed specification of the main concepts and relationships of an ontology in this context has not been developed yet. This can hinder the precise comprehension of how SES characterises and exploits ontologies. We present our work to provide such a specification. Based on different information sources and on our experience with SES, we have developed a novel ontology metamodel as a step towards a wider and clearer understanding of how ontologies are used for KCSE in practice. The metamodel also aids in determining how this usage could be adapted or extended.

Keywords

Knowledge-Centric Systems Engineering, ontology, metamodel, SES Engineering Studio

1. Introduction

The engineering of large and complex systems is usually a challenging activity. Various approaches exist with different principles on how to best address it. An example is Knowledge-Centric Systems Engineering (KCSE) [1], an industrial approach whose adoption is growing. It can be regarded as a specialization of model-based systems engineering that advocates that all lifecycle processes are affected by the existence of and can benefit from knowledge bases about a system. Such bases can be created by means of ontologies that represent system domains and consider different types of knowledge elements and of semantic aspects.

SES Engineering Studio (hereafter referred to as SES) [2] is an industrial platform developed by The REUSE Company (TRC) that implements KCSE. It can support and coordinate systems and software engineering processes and consists of several sub-tools for system specification, quality analysis, traceability, manual V&V, and ontology management. Knowledge Manager (KM) [3] is the main sub-tool for ontology management and considers five ontology parts (see Section 2): Vocabulary, Conceptual Model, Patterns, Formalization, and Reasoning. SES can interoperate with tens of other engineering tools (Capella, DOORS, Rhapsody, Simulink...) and is used mainly in critical domains and by large companies, e.g., by Airbus, Thales, and Toyota.


The ideas and principles on which SES is based have been developed for over 20 years, since an initial vision [4] to industrially accepted solutions. Despite this long period of development and use, a broad conceptualization of how ontologies are defined in SES is not available. There exist different information sources, including manuals, presentations and the sub-tools themselves, but not a dedicated specification of the concepts and relationships managed in SES ontologies. This can hinder

^{*}ER2025: Companion Proceedings of the 44th International Conference on Conceptual Modeling: Industrial Track, ER Forum, 8th SCME, Doctoral Consortium, Tutorials, Project Exhibitions, Posters and Demos, October 20-23, 2025, Poitiers, France

^{1*} Corresponding author.

[†] These authors contributed equally.

✉ joseluis.delavara@uclm.es (J.L. de la Vara); pablohoward.mason@alu.uclm.es (P. Mason); clara.ayora@uclm.es (C. Ayora)

 0000-0003-1813-398X (J. L. de la Vara); 0009-0002-8265-6531 (C. Ayora)



© 2025 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

the understanding of what is considered an ontology and of how ontologies are developed and exploited for KCSE in practice. An equivalent situation would be to have, e.g., UML (Unified Modeling Language) editors and information about them but not a complete metamodel that provides a deeper, more precise characterization of what could be modelled in UML diagrams and of its constraints.

As a step to fill this gap, we present our work on such a conceptualization. We have developed a novel ontology metamodel to provide a wider and clearer understanding of how ontologies are used for KCSE in practice in general and in SES in particular. By ontology metamodel, we refer to a structured specification in the form of a class diagram to which ontologies (i.e., metamodel instances) conform. We have studied different information sources and reflected on our experience with SES to determine the classes, attributes, and relationships necessary to develop the Vocabulary, Conceptual Model, Patterns, Formalization, and Reasoning parts of a SES ontology.

The metamodel is the first detailed characterization of the ontology elements used in SES that considers all its different ontology parts, in addition to how the elements and the parts relate. To the best of our knowledge, no prior (meta)model has addressed this. In addition, the metamodel shows how ontologies are used in practice for the engineering of large and complex systems. This is especially relevant nowadays because of the interest in exploiting AI techniques in this context, e.g., via knowledge bases. The metamodel can also aid in determining how SES ontology management could be adapted or extended. All in all, the paper helps practitioners and researchers to better understand KCSE in practice and to identify improvement opportunities.

The rest of the paper presents its background, the metamodel, and our main conclusions.

2. Background

The background of the paper presents SES ontologies and reviews related work.

2.1. SES Ontologies

As introduced above, SES supports different engineering activities via a specific implementation of KCSE. SES can be regarded as an expert, rule-based system for systems and software engineering that exploits semantic information about a domain and uses AI techniques, e.g., NLP and machine learning. It imports system data from other tools and integrates its management.

To the above ends, SES exploits ontologies with five parts (Figure 1):

- Vocabulary, which includes the terms used in a specific domain and their syntactic information; e.g., the word ‘car’ is a noun, and ‘decelerate’ is a verb.
- Conceptual Model, which focuses on semantics and relationships, including system architectures and relationship types; e.g., the semantics of ‘car’ can be ‘system’, it specializes ‘vehicle’, and it has an ‘engine’.
- Patterns, which are templates (aka boilerplates) used for structured system information specification and analysis; e.g., for textual requirements.
- Formalization, which involves the semantic representation of system data according to patterns and rules; e.g., for system data import and later analysis.
- Reasoning, with procedures to infer information; e.g., about the correctness of textual requirements or of model elements based on their content, structure, and semantics.

2.2. Related Work

SES uses ontologies in a distinct way and with a structure that departs from the usual practice presented in the literature. Therefore, existing ontology metamodels (and related ones) and prior KCSE-related solutions in other contexts do not reflect this usage.

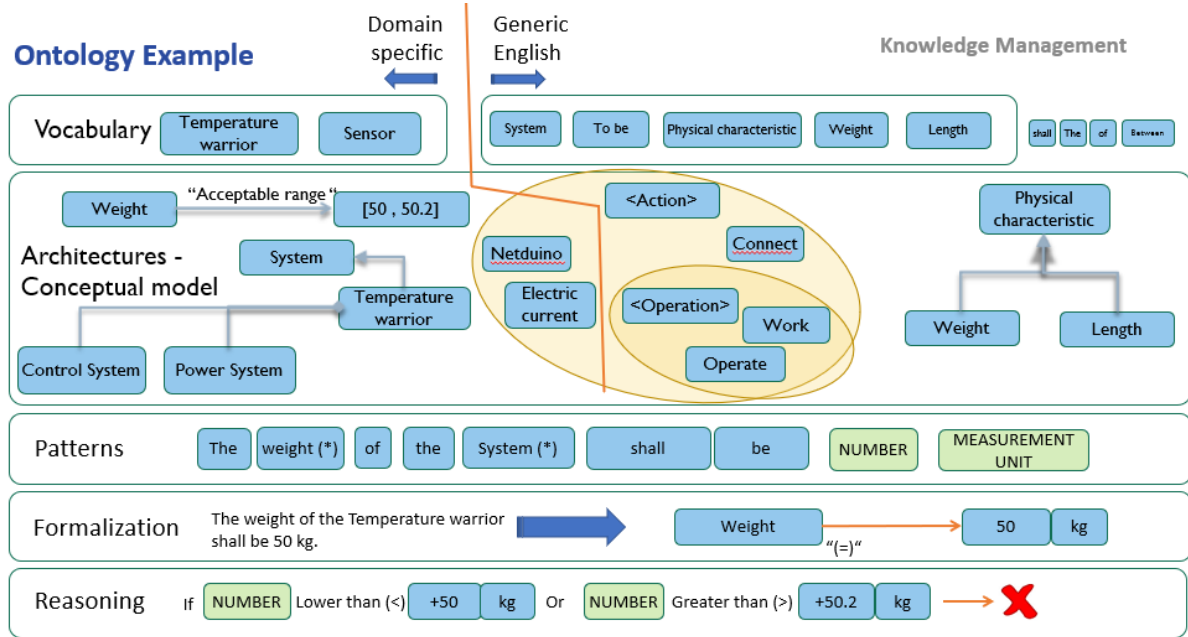


Figure 1: Example of SES ontology [3]

Prior publications have proposed different ontology metamodels. A key specification is OMG's Ontology Definition Metamodel [5], which includes metamodels for RDF (Resource Description Framework), OWL (Web Ontology Language), Common Logic, and Topic Map, considering aspects such as classes, properties, relationships, types, and terminology, among others. This relates to the Terminology and Conceptual Model parts of a SES ontology. SACM (Structure Assurance Case Metamodel) [6] is another OMG specification that includes metamodels for terminology (related to the Vocabulary part of a SES ontology) and for artefacts (related to the Formalization part). Although suitable for their purposes and intended uses, these metamodels do not correspond to how KCSE is applied in industry. The same applies to basic, foundational ontologies and their metamodels, such as BFO (Basic Foundational Ontology) [7] and ThingFO (Thing Foundational Ontology) [8]. There exists a gap between what these ontologies and their metamodels define and how they could be effectively applied in the engineering of, e.g., safety-critical systems.

Ontology metamodels have also been proposed and used for specific ontology-based systems and software engineering solutions [9], i.e., for specific KCSE or KCSE-related tasks. For instance, Chen et al. [10] used OWL for requirements verification. Alanen et al. [11] and, more recently, Maunero et al. [12] developed dedicated metamodels for risk assessment. Ernadote [13] uses a metamodel close to foundational ontologies. While interesting and potentially useful, these metamodels and usages are different to how SES exploits ontologies. Their scope is also narrower.

Prior work has dealt with SES-focused data models. However, they have only considered some part of its ontologies. The original, initial data model is referred to as RSHP [4], to reflect the fact that it focuses on relationships between different elements. It also includes concepts such as artefact, property, term, and semantics. RSHP is the foundation of the Vocabulary, Conceptual Model, and Formalization parts of a SES ontology, but it does not reflect how they have evolved and their current needs and usage. More recently, SRL (System Representation Language) [14] has been proposed as an evolution of RSHP focused on the system artefact data that SES manages. Although SRL is currently the main basis of the Formalization part of a SES ontology, it lacks details of the relationship with other parts. Other artefact metamodels can be found in the literature (e.g., in [15]), but they have not been developed for KCSE.

Prior work has also studied the correspondence between parts of SES ontologies and specific metamodels, e.g., for system assurance and certification [16] and for system modelling [17]. This mostly relates to the Formalization part of a SES ontology and is what SES connectors implement when a metamodel explicitly exists for external tools (e.g., Capella and Papyrus).

In summary, ontology metamodels and related ones can be found in the literature, including metamodels on or connected to SES-specific aspects. There are metamodels that deal with SES ontology parts to some extent, but no prior metamodel addresses the individual parts completely or all the parts how they are specified nowadays. The scope of these prior metamodels is different to the scope of the metamodel in this paper. Therefore, a wide characterization of SES ontologies is missing. This is necessary to better understand how ontologies are used in KCSE in practice.

3. Ontology Metamodel

The development of the metamodel has been based on (1) our work on and with SES since 2015 for different systems and software engineering tasks (see, e.g., [18, 19, 20, 21]), (2) a detailed review of SES and KM manuals (630 pages), and (3) a detailed analysis of the default KM relational database in English (173 tables and 12,412 rows). Especial attention has been paid to thoroughly identify classes and relationships of the metamodel from the manuals and the database, as inconsistencies exist. For instance, some database content corresponds to implementation decisions (e.g., dedicated tables for faster information retrieval that conceptually match others) and to legacy aspects that do not represent accurately how ontologies are used in SES nowadays (e.g., shift from a platform focused on requirements quality analysis to one that can assist in any systems and software engineering task).

The next sub-sections present the classes and relationships of the metamodel and discuss the metamodel.

3.1. Classes and Relationships

Figures 2-6 show the ontology metamodel. For space reasons, we do not explain the elements that regard as easy to understand (e.g., ID and name attributes of the classes), association names are shown only when two classes relate in more than one way, and it is indicated in Figure 5 when a class is defined in Figure 3, omitting its details in the former.

The base classes of the metamodel (Figure 2) are *Ontology* and *Ontology Element*. An ontology consists of ontology elements of the different parts (Vocabulary, Patterns, etc.). All the other concrete classes of the metamodel specialize *Ontology Element*.

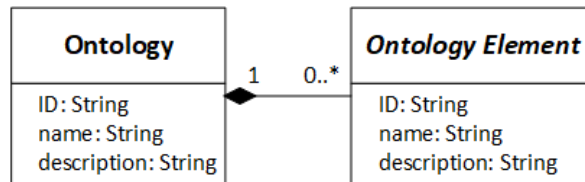


Figure 2: Base classes

Figure 3 shows the classes and associations of the Vocabulary, Conceptual Model, and Patterns parts of an ontology. The **Vocabulary** classes are as follows:

- *Term*, which represents words, expressions, abbreviations, and acronyms. A term has a number and a gender, which might change. For instance, in Spanish some terms vary for male and female genders, such as “usuario” and “usuaria” for user.
- *Term Tag*, which corresponds to syntactical categories of the terms, such as noun, verb, adjective, and adverb. A term tag can have a parent and child term tags, e.g., modal verbs are a special type of verb.
- *Simple Term* represents a term that consists of a single word, e.g., ‘sensor’ in Figure 1.
- *Compound Term* represents a term that consists of several simple terms, e.g., ‘physical characteristic’ in Figure 1.
- *Term Slot* represents the parts of a compound term. A term slot has a position and refers to a simple term, e.g., the position of ‘physical’ is 1 in ‘physical characteristic’.

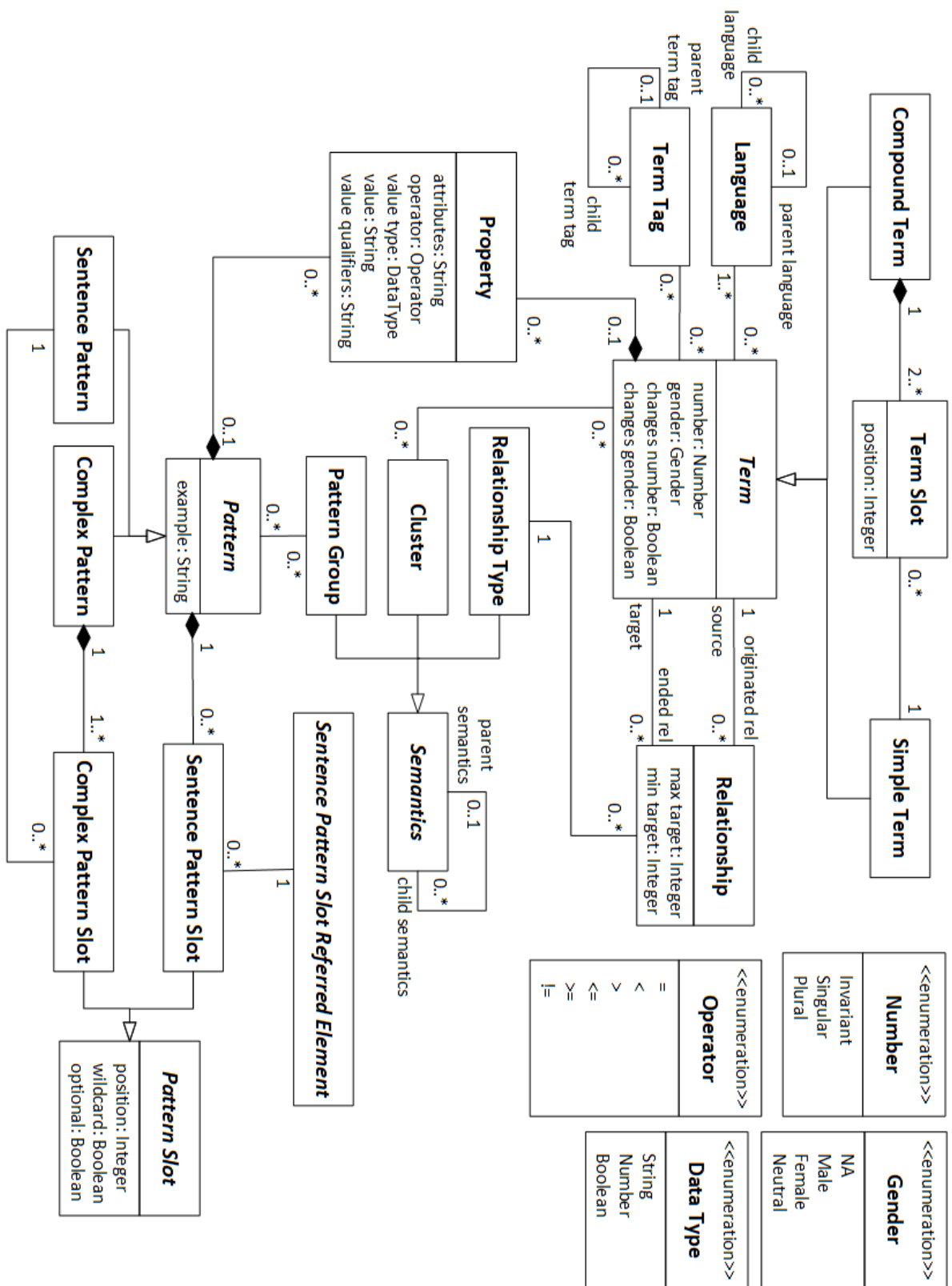


Figure 3: Metamodel fragment for Vocabulary, Conceptual Model and Patterns

- *Language*, such as English and Spanish. A language can have a parent and child languages, e.g., American English and British English.
- *Property* is used to specify characteristics (equal to, greater than...) of the object(s) that a term represents, e.g., the weight of a system in the scope of Figure 1. This includes the units of a property (value qualifiers; e.g., kilogram). Properties can also be specified for patterns and artefacts in the Patterns and Formalization parts.

The **Conceptual Model** part is based on:

- *Semantics*, which represents the meaning of different ontology elements. There can be parent and child semantics, such as system and automotive system, or UML element and use case actor.
- *Cluster* for the semantics of terms, e.g., system or component. In the scope of a use case diagram, there can be terms that correspond to actors or use cases (i.e., their names).
- *Relationship Type* for the semantics of relationships between terms, e.g., composition (is part of), specialisation (is a), synonymy (same or almost the same meaning), and association (between actors and use cases in use case diagrams).
- *Pattern Group* for the semantics of patterns (Patterns part), e.g., patterns for reliability, safety, or security requirements.
- *Relationship*, as the specific occurrence of a relationship type between two terms, e.g., composition between 'car' and 'wheel', and the association between a specific actor and a specific use case. In SES, cardinality can be specified for the target of a relationship, e.g., a 'car' has four 'wheels'.

The **Patterns** part is defined through:

- *Pattern*, i.e., templates for structured system information specification and analysis, as shown in Figure 1 to specify the weight of a system. An example of pattern instantiation can be stored. A pattern can have properties to indicate object characteristics that are considered in its usage, such as the specific weight of a system in the scope of Figure 1.
- *Sentence Pattern* corresponds to simple, atomic patterns, e.g., the pattern in Figure 1.
- *Complex Pattern* corresponds to patterns that consist of others. For instance, considering Figure 1, there could be a sentence pattern for 'the system shall' as a core basic part, and the pattern shown could include it.
- *Pattern Slot* represents the parts of a pattern. A pattern slot has a position, can be optional, and can correspond to any term (wildcard).
- *Sentence Pattern Slot* represents individual, atomic parts of a pattern, such as the different boxes of the pattern shown in Figure 1.
- *Sentence Pattern Slot Referred Element* (Figure 4) generalizes ontology elements that can be used for a given slot, e.g., 'be' (verb) and 'NUMBER' (term tag) in the pattern in Figure 1.
- *Complex Pattern Slot* represents the inclusion of a sentence pattern in a complex pattern.

Figure 5 shows the classes and associations of the Formalization and Reasoning parts of an ontology. In essence, the classes of the Formalization part deal with how data is stored in SES (ontologies) for later analysis in the Reasoning part. Overall, data storage is based on artefacts, artefact properties, and relationships of the artefacts, as other languages for artefacts, entities, or classes. The main difference is a more detailed consideration of semantic aspects.

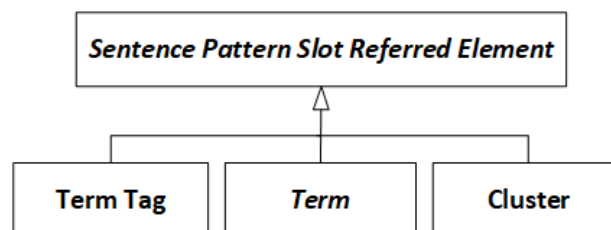


Figure 4: Sentence pattern slot referred elements

The classes of the **Formalization** part are:

- *Formalization* corresponds to the representation, in SES format, of data from external sources such as requirements from DOORS or diagrams from Rhapsody. Figure 1 includes a basic example for the requirement about the weight of a system (50 kg). A formalization could be created for, e.g., a use case diagram.
- *Artefact* represents the main way to structure the data stored in SES. Its semantics can be specified with clusters. As for terms and patterns, properties can be specified for artefacts. When representing a use case diagram, there could be a root artefact for the diagram as a whole and different sub-artefacts for the actors and use cases.
- *Metaproperty* is used to store data about artefact management, not about the artefacts themselves and their characteristics. The latter is done via properties. A metaproperty could be used, e.g., to store when a requirement was last modified and its author.
- *Artefact Content* supports the specification of what an artefact consists of and what its components refer to (other artefacts, relationships, or terms). The corresponding elements can have a given position. In the example in Figure 1, 'Weight', the arrow, '50', and 'kg' are referred to in different positions of the overall artefact, i.e., they represent content of the overall artefact.
- *Artefact Content Element* generalizes elements that can be used as artefact content. In Figure 1, 'Weight', '50', and 'kg' are terms, and the arrow depicts a relationship.
- *Formalization Rule* defines how data is imported into SES. This can be specified programmatically (source code) or by mapping source elements to formalization abstract elements (i.e., to the types of elements that can be used in a formalization).
- *Formalization Source Element* corresponds to what a formalization can be created from, e.g., the pattern in Figure 1 (i.e., requirements that match the pattern).
- *Resource Element* corresponds to elements other than patterns and pattern slots that a formalization can be created from, such as a given use case diagram in XML format. In principle, any structured or partially structured data source could be considered.
- *Formalization Abstract Element* represents a target element and its type for a formalization. These elements can correspond to a semantics or a term tag.
- *Formalization Abstract Artefact* represents artefact-specific formalization rule aspects, e.g., to indicate that actors and use cases will be mapped to artefacts.
- *Formalization Abstract Relationship* represents relationship-specific formalization rule aspects, e.g., to map associations of actors and use cases to relationships.

The **Reasoning** part of a SES ontology deals with the information and knowledge that can be derived from the content of other parts. This has historically been quantitative and qualitative quality analyses, first for requirements and later for any system artefact that can be formalized (i.e., for which formalization rules have been defined). These are the classes that characterize this part:

- *Inference Rule* represents procedures to derive information or knowledge, e.g., in Figure 1, to indicate that a weight lower than 50 kg or greater than 50.2 kg is not adequate. The procedures can be developed programmatically (source code) or can be based on the number of times that certain elements are used (e.g., number of words of a requirement) or that certain conditions hold (e.g., terms in a requirement that do not relate to a specific cluster or to any). For a use case diagram, actors could be the main elements of inference rules. The rules could be based on the total number of actors or of actors not associated with any use case (related element).
- *Inference Source Element* (Figure 6) generalizes ontology elements that an inference rule can consider, e.g., use case actors (cluster).
- *Quality Range* establishes what is regarded as low, medium, or high quality depending on the number of times that certain elements are used or that certain conditions hold. For instance, for requirement words, low quality could be 0-4 or 31- ∞ words, medium quality could be 21-30, and high quality could be 5-20. For actors, having some actor that is not associated with some use case could be low quality. These kinds of ranges can be adjusted to project- or

company-specific needs and practices and to general guidelines, e.g., about requirements specification quality. An example is the INCOSE Guide to Writing Requirements [22].

- *Inference Result* corresponds to the outcome of the procedure of an inference rule, such as 25 requirements words (medium quality) or no actor not associated with some use case (high quality). Inference results might be specified manually by a user from no specific formalization or no specific inference rule, e.g., as a checklist item on which a reviewer indicates that the layout of a use case diagram is regarded as adequate.

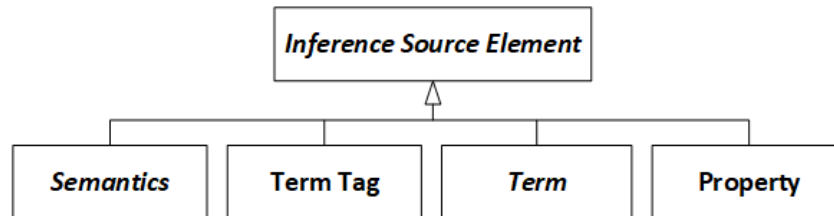


Figure 6: Inference source elements

3.2. Discussion

After presenting the metamodel, this section presents and discusses how it has been validated and some practical considerations.

For **validation**, we formulated the following research question: “Can the metamodel be a feasible means to represent ontologies for KCSE in practice?”. To answer it, we developed a prototype with the Eclipse Modelling Tools package [23] to create instances of the metamodel (Figure 7). An ecore version of the metamodel was created and user interfaces based on KM ones (i.e., trying to reproduce them) were designed with EMF Forms. The instances created replicated examples from SES and KM manuals and ontology development with SES that we had conducted in the past in different projects and contexts (e.g., for [19, 21, 24]), such as the examples mentioned in the description of the classes and relationships of the metamodel. This includes ontology development for real, industrial systems. We considered the following 13 user tasks:

- Specify terminology
- Specify semantic categories
- Relate semantic categories
- Specify semantic information of terminology
- Relate terms
- Specify system specification patterns
- Reuse system specification patterns
- Specify semantic information of patterns
- Specify formalization rules
- Specify formalization results
- Specify inference rules
- Specify inference results
- Maintain ontologies

All the classes and relationships of the metamodel were used in the set of tasks.

We regard the outcome of the validation as successful and thus the answer to our research question as positive. We were able to effectively execute several scenarios for each user task considered. This also allowed us to identify errors and missing details in the metamodel, e.g., errors in some multiplicities and the fact that a formalization abstract element can relate to a term tag. All in all, we consider that the metamodel can contribute to effective ontology management in the scope of KCSE in practice.

Figure 7: Prototype for ontology specification

Regarding **practical considerations** of the metamodel:

- The Vocabulary and Conceptual Model elements are, in general, similar to what terminology and ontology metamodels (e.g., [5, 6]) commonly consider, i.e., terms (or concepts), attributes, relationships, and categories.
- The Patterns part is more specific to SES and is mainly a result of its historical focus on structured requirements specification and on quality analysis. Nonetheless, how patterns are defined in SES is flexible and enables general specification and analysis of structured text and of other structured representations such as system, software, and hardware diagrams.
- Formalization in SES ontologies strongly depends on how SES stores system data from external sources, i.e., on its internal system data representation schema. The applicability of this part in other contexts is not direct but would require adaptation.
- The Inference elements are largely based and depend on what has been and is mostly reasoning in SES, i.e., on quality analysis. More general, non-quality-focused reasoning is not supported in a structured way explicitly but is implemented through source code, e.g., to determine how similar two system artefacts are.
- We conjecture that a holistic and more generic metamodel for KCSE in practice could be developed. It could support, in a structured way, how SES manages ontologies and further needs, such as wider reasoning possibilities from ontology elements. The metamodel could be based on OMG standards (e.g., [5]).
- Metamodels of basic, foundational ontologies (e.g., [7, 8]) can be regarded as more widely applicable because they are more generic. However, they lack details that reflect the specific needs of KCSE in practice. This also happens with the original RSHP data model [4]. Their high level of abstraction can be an advantage or a disadvantage depending on the purpose of their use.

4. Conclusion

KCSE is an approach used in practice for systems and software engineering that exploits knowledge bases, e.g., ontologies. Although KCSE has been developed for over two decades and companies from different domains have adopted it for their projects, a dedicated and complete characterization of ontology concepts and their relationships in this scope had not been provided yet.

As a solution, we have presented a novel ontology metamodel for KCSE with SES, an industrial tool used for the engineering of large and complex systems such as airplanes, cars, and trains. From

the analysis of the Vocabulary, Conceptual Model, Patterns, Formalization, and Reasoning parts of SES ontologies and of their usage, we have identified 36 classes, 39 attributes, 63 relationships, and six enumerations. Some elements are similar to those in related metamodels (e.g., the Vocabulary and Conceptual Model elements), whereas others are more SES-specific (e.g., the Formalization and Reasoning elements). We have used all the classes and relationships to replicate, with an Eclipse-based prototype, examples in SES manuals and past ontology development with SES in different projects and contexts. This has allowed us to conclude that the metamodel can be a feasible means to represent ontologies for KCSE in practice.

The metamodel is the first detailed, wide and implementation-independent characterization of the ontology elements that SES uses. This usage is different to how ontologies are commonly used in other contexts and represents how the systems and software engineering industry is working on and with ontologies to make their processes more effective and more efficient. The metamodel also facilitates the analysis of how SES ontology management could be adapted or extended, e.g., considering further ontology management needs or other metamodels. This can help practitioners and researchers to better understand KCSE in practice and its application, as well to identify improvement opportunities (e.g., for alignment with related standards and recommendations). We conjecture that a holistic and more generic metamodel could be developed.

As future work, we plan to conduct a more detailed comparison of the metamodel presented with related ones, including the study of how, e.g., existing OMG metamodels could enable and extend ontology management in SES. We would also like to discuss the results of the comparison with TRC staff and customers. They could provide feedback on the extent to which they find a possible alignment with OMG metamodels and an extension useful and necessary.

Acknowledgements

The work leading to this paper has received funding from the REBECCA (HORIZON-KDT ref. 101097224; MCIN/AEI ref. PCI2022-135043-2; NextGen.EU/PRTR), AETERNAL (MCIN/AEI ref. PID2023-149753OB-C21; ERDF), FDT4S (JCCM ref. SBPLY/24/180225/000020; ERDF), “Una propuesta integral para el desarrollo independiente de dominio de gemelos digitales” (UCLM ref. 2025-GRIN-38441; ERDF), “Preparing for Society 5.0” (Cátedra Ciudad de Albacete ref. 13585/2025), and “Paradigmas de interacción para la nueva era de resiliencia digital” (UCLM ref. 2022-GRIN-34436; ERDF) projects. We are also grateful to TRC and its employees for granting access to SES, for sharing information about SES and its sub-tools, and for assisting us when requested.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

- [1] J. Llorens, Knowledge Centric Systems Engineering, in: South European Systems Engineering (SESE 2014)
- [2] The REUSE Company, SES Engineering Studio, 2025. URL: <https://www.reusecompany.com/ses-engineering-studio> (accessed Jun 2, 2025)
- [3] The REUSE Company, KM - Knowledge Manager, 2025. URL: <https://www.reusecompany.com/km-knowledge-manager>
- [4] J. Llorens, J. Morato, G. Genova, RSHP: an information representation model based on relationships, in: Soft computing in software engineering, Springer, 2004, pp. 221–253. doi: 10.1007/978-3-540-44405-3_8
- [5] OMG, Ontology Definition Metamodel, version 1.1, 2014. URL: <https://www.omg.org/spec/ODM/>
- [6] OMG, Structured Assurance Case Metamodel, version 2.3, 2023. URL: <https://www.omg.org/spec/SACM>

- [7] J.N., Otte, J. Beverley, A. Ruttenberg. "BFO: Basic formal ontology." *Applied ontology* 17(1) (2022): 17-43. doi: 10.3233/AO-220262
- [8] L. Olsina, Thing Foundational Ontology: ThingFO v1.3's Terms, Properties, Relationships and Axioms, arXiv:2107.09129 (2022). doi: 10.48550/arXiv.2107.09129
- [9] L. Yang, K. Cormican, M. Yu. "Ontology-based systems engineering: A state-of-the-art review." *Computers in Industry* 111 (2019): 148-171. doi: 10.1016/j.compind.2019.05.003
- [10] R. Chen, C.H. Chen, Y. Liu, X. Ye, X. "Ontology-based requirement verification for complex systems." *Advanced Engineering Informatics* 46 (2020): 101148. doi: 10.1016/j.aei.2020.101148
- [11] J. Alanen, J. Linnosmaa, T. Malm, N. Papakonstantinou, T. Ahonen, E. Heikkilä, R. Tiusanen. "Hybrid ontology for safety, security, and dependability risk assessments and Security Threat Analysis (STA) method for industrial control systems". *Reliability Engineering & System Safety* 220 (2022): 108270. doi: 10.1016/j.ress.2021.108270
- [12] N. Maunero, F. de Rosa, P. Prinetto, Towards Cybersecurity Risk Assessment Automation: an Ontological Approach, in: 21st IEEE International Conference on Dependable, Autonomic & Secure Computing (DASC 2023). doi: 10.1109/DASC/PiCom/CBDCCom/Cy59711.2023.10361456
- [13] D. Ernadote, Ontology-based pattern for system engineering, in: 20th International Conference on Model Driven Engineering Languages and Systems (MODELS 2017). doi: 10.1109/MODELS.2017.4
- [14] J.M. Álvarez-Rodríguez, R. Mendieta, J. Llorens, Elevating the meaning of data and operations within the development lifecycle through an interoperable toolchain, in: INCOSE International Symposium 2019. doi: 10.1002/j.2334-5837.2019.00652.x
- [15] J.L. de la Vara, A. García, J. Valero, C. Ayora. "Model-Based Assurance Evidence Management for Safety-Critical Systems". *Software and Systems Modeling* 21(6) (2022): 2329-2365. doi: 10.1007/s10270-021-00957-z
- [16] J.L. de la Vara, E. Parra, L. Alonso, B. López, J.M. Álvarez-Rodríguez, Integration of Tool Support for Assurance and Certification and for Knowledge-Centric Systems Engineering, in: 9th IEEE International Workshop on Software Certification (WoSoCer 2019). doi: 10.1109/ISSREW.2019.00092
- [17] R. Mendieta, J.L. de la Vara, J. Llorens, J.M. Álvarez-Rodríguez, Towards Effective SysML Model Reuse, in: 5th International Conference on Model-Driven Engineering and Software Development (MODELSWARD 2017). doi: 10.5220/0006267605360541
- [18] J.L. de la Vara, A. Gómez, E. Gallego, G. Génova, A. Fraga, Representation of Safety Standards with Semantic Technologies Used in Industrial Environments, in: 6th International Workshop on Next Generation of System Assurance Approaches for Safety-Critical Systems (SASSUR 2017). doi: 10.1007/978-3-319-66284-8_22
- [19] J.L. de la Vara, J.M. Morote, C. Ayora, G. Giachetti, L. Alonso, R. Mendieta, D. Muñoz, R. Ruiz-Nolasco, A. González, Early V&V in Knowledge-Centric Systems Engineering: Advances and Benefits in Practice, in: 18th IEEE International Conference on Software Testing, Verification and Validation (ICST 2025). doi: 10.1109/ICST62969.2025.10988975
- [20] B. Lopez, J.M. Alvarez-Rodriguez, E. Parra, J.L de la Vara, Ontology Configuration Management for Knowledge-Centric Systems Engineering in Industry, in: 50th IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2020). doi: 10.1109/DSN-S50200.2020.00022
- [21] E. Parra, L. Alonso, R. Mendieta, J.L. de la Vara, Advances in Artefact Quality Analysis for Safety-Critical Systems, in: 30th International Symposium on Software Reliability Engineering (ISSRE 2019). doi: 10.1109/ISSREW.2019.00047
- [22] INCOSE, Guide to Writing Requirements, version 4, 2023.
- [23] Eclipse, Eclipse Modeling Tools, 2025. URL: <https://www.eclipse.org/downloads/packages/>
- [24] J.L. de la Vara, H. Bahamonde, C. Ayora. "Assessment of the Quality of the Text of Safety Standards with Industrial Semantic Technologies". *Computer Standards & Interfaces* 88 (2024): 103803. doi: 10.1016/j.csi.2023.103803