Enabling system artifacts reuse through the semantic representation of engineering models: a case study of Simulink models

Roy Mendieta
The REUSE Company
Madrid, Spain
roy.mendieta@reusecompany.com

Eduardo Cibrián
Computer Science and Engineering Department
Carlos III University of Madrid
Madrid, Spain
ecibrian@inf.uc3m.es

Jose María Álvarez-Rodríguez
Computer Science and Engineering Department
Carlos III University of Madrid
Madrid, Spain
joalvare@inf.uc3m.es

Juan Llorens
Computer Science and Engineering Department
Carlos III University of Madrid
Madrid, Spain
llorens@inf.uc3m.es

Abstract—**Currently, digital twins are being designed to provide a virtual version of complex physical systems. Modelling and simulation techniques and tools are used to design these engineering products embedding domain knowledge in many system artifacts available under different protocols, formats and meta-models. The cost of development of these virtual artifacts is usually very high implying the need of saving time and costs by means of increasing the reusability factor of them. A first step to ease the reuse relies on the ability of looking up a system artifact according to some input query. To do so, it is necessary to design a knowledge management strategy unifying the structure and representation of these artifacts and provide a search service that can exploit the indexed information. In this work, authors propose a semantic model to represent system artifacts and demonstrate its application through a search service consuming simulation models (designed with the Matlab Simulink tool). Furthermore, an experiment has been conducted to show the precision and recall of this semantic search service.**

Index Terms—**information representation, physical system models, Simulink, model reuse, knowledge reuse**

I. Introduction

Last times have seen the emergence of Model-based Systems Engineering (MBSE) as a complete methodology to address the challenge of unifying the techniques, methods and tools to support the whole specification process of a system (conceptual design, system requirements, design, analysis, verification or validation, etc.) around the application of models. In the context of the well-known Vee lifecycle model, it means that there is "formalized application of modeling" to support the left-hand side of this system life-cycle implying that any process, task or activity will generate different system artifacts but all of them represented as models.

This approach is considered a cornerstone for the improvement of the current practice in the Systems Engineering discipline since it is expected to cover multiple modeling domains, to provide better results in terms of quality and productivity, lower risks and, in general, to support the concept of continuous and collaborative engineering, easing the interaction and communication between people (engineers, project managers, quality managers, etc.).

Although MBSE represents a shifting paradigm for the development of critical systems, the plethora of engineering methods supported by different tools implies the need of not only easing the communication between people but also considering its application to the universe of available tools. How could we do requirements management, simulation, diagramming, documenting, information retrieval or project management without the corresponding tools or IT systems? The more complex the problems are, the more complex computer tools must be delivered, and the main reason for that is, consequently, because those computer tools are demanded to be "smarter". Up to now, a computer tool is not human independent; it simply "acts" as smart according to its access to relevant data, information and knowledge. In order to enable a collaborative MBSE through IT systems, it is completely necessary to enable the possibility of communicating tools (interoperability) and reusing previous engineering designs saving costs and time.

In order to reuse the knowledge generated in Model-driven methodologies (MDB) such as MBSE, it is necessary to understand the underlying concepts and relationships that allows us to make a semantic interpretation of the models.For example in the automotive industry [1] modeling capabilities are applied to the whole engineering process, from the specification to the certification in a virtual twin environment. In the context of tool-chains for MDB, it is possible to find many suites such as Matlab Simulink [2] that can be applied to different engineering activities: architecting (descriptive modeling), simulation (analytical modeling) or testing of digital systems.

However, no one size fits all, and engineering environments are usually integrating many different tools. This situation generates a good number of system artifacts that are part of a specific product or service. Reuse capabilities are therefore constrained by the possibility of linking every system artifact (traceability) and, then, being able to represent, search and customize those relevant system artifacts. In this manner, when a system artifact is selected for being reuse (e.g. a component), it actually implies the necessity of bringing all connected system artifacts such as requirements, test cases, logical models, etc. The reusability factor will depend on the capability of creating an underlying knowledge graph that can serve us to deliver services that require an holistic view of the system such as change impact analysis, visualization or quality checking.

More specifically, in the context of model reuse, it is necessary to define a knowledge management strategy for reusing system artifacts. The use of semantics may help to improve the reusability factor of a system artifact by identifying similar artifacts through a comparison under a common and representation model. Model reuse still remains challenging due to the diversity of domains and information embedded in the models. Furthermore, engineering tools have not been designed to look up similar artifacts. The cost of reuse will mainly depend on the complexity of the entity to be reused [3] and, this implies, that an enriched representation may help to improve the fist step to reuse: discoverability.

In order to build an underlying knowledge graph, there are works, such as OSLC Resource Shapes [4] [5] [6] or ISO Step meta-models, focusing on the description of artifact meta-data [7]. However, the representation of both artifact meta-data and contents is not fully addressed by a common representation model.

In this work, authors aims to effectively reuse the knowledge embedded in Simulink models. The solution called Simulink2RSHP makes use of an ontology-based approach for indexing and retrieving information following a meta-model, RSHP [8]. Under this schema, both meta-data and contents are represented using a common domain vocabulary and taxonomy creating a property graph that can be exploited for system artifact discovery. To do so, a mapping between the Matlab Simulink meta-model and the RSHP meta-model is defined to represent and serialize analytical models in a repository. Then, a retrieval process is implemented on top of this repository to allow users to perform text-based queries and look up similar artifacts. To validate the proposed solution, 38 Simulink models have been used and 20 real user queries have been designed to study the effectiveness, in terms or precision and recall, of the proposed solution [9] against the Matlab Simulink searching capabilities.

The paper is organized as follows: The related work is presented in Section II. Section III describes the background and defines the proposed solution for Simulink model reuse. Section IV describes the validation, while Section V summarizes the main conclusions and outlines some future research directions.

II. Related Work

Semantic representation of analytical models for retrieval purposes is the cornerstone of this work. In the case of models reuse, [10] presents a work to represent and retrieve CAD (Computer-Aided Design) by implementing a mapping function between the features of different CAD models. In [11], an ontology-based retrieval technique is introduced to perform a semantic similarity process between UML class diagrams.

In the case of Simulink models, [12] describes a solution focused on design patterns to develop reusable model structures without considering semantic features. In [13], tool for automatically identifying, classifying and formalizing submodel patterns in Simulink models is presented. This tool implements a retrieval process based on text-comparison.

Regarding RSHP applications for system artifact representation, some prior works can be found to reuse electric circuits designed in the Modelica language [14]. In [15], authors also use a similar approach for SysML models where a mapping between the SysML meta-model and the RSHP meta-model is presented. Based on previous experiences, the RSHP meta-model fits to represent both meta-data and contents of different types of models. In the context of this work, Simulink models have been selected to test the feasibility of reusing analytical models applying the same principles of knowledge representation.

Unlike previous approaches, where reuse is based on specific features of the domain knowledge or where reuse is basically focused on text comparison. The proposed solution aims to improve the reuse of the embedded information in the Simulink models by providing: 1) A semantic representation of Simulink models using an existing meta-model like RSHP and 2) A retrieval process based on comparing the underlying graphs of a query against a repository of Simulink models.

III. Simulink2RSHP: Implementation of a tool for reuse Simulik models

A. Background

Since the first step to provide a reuse mechanism for Simulink models relies on the representation of information using the RHSP metamodel, the main building blocks of this framework are here outlined.

1) The RSHP metamodel.: RSHP [8] is an information representation model based on defining concepts (artifacts) and relationships among them under a specific semantics. It has been used to different types of information such as textual, design models or source code using the same representation schema. The meta-model, as a class diagram, is presented in Figure 1 and it comprises the following elements:

- Artifact. An artifact is a knowledge container can be represented through only Knowledge Elements or through other Artifacts.

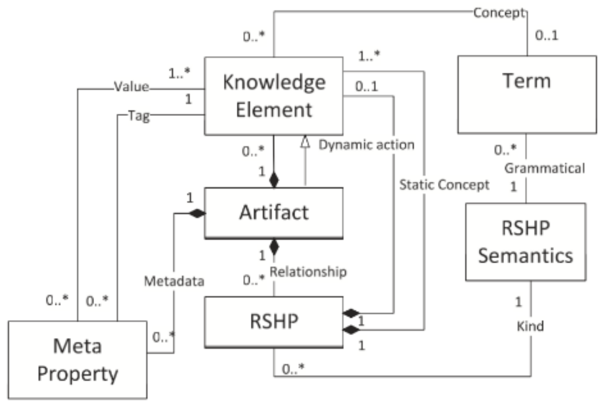

Fig. 1. The RSHP representation metamodel.

- Knowledge Element. A Knowlege Element represents the occurrence of a Term. It is the smallest unit of knowledge.
- Term. A term represents an element of the domain vocabulary (with some specific semantics, if defined).
- RSHP. An RSHP represents an n-ary relationship between Artifacts.
- RSHP Semantics. It is the relationship type assigned to a relationship between two Artifacts.
- Meta-property. It is used to add meta-data to the Artifacts.

The RSHP meta-model has also been exposed as an OSLC Resource Shape. It can be serialized as RDF using the interface known as OSLC-KM (Open Services for Lifecycle Collaboration-Knowledge Manager) [4], which is a kind of flavor of OSLC as a result of the CRYSTAL European project.

B. The RSHP reusability framework

Reuse semantic information needs to deal with a lot of factors that have to be consider in reuse techniques. Most major frameworks are focused on specific types of information such us software [16] restricting the knowledge manager capabilities.

One of the main objectives is to identify, classify, organize and represent Simulink models using semantics. To do so, the proposed solution applies a domain ontology to model such information and build a retrieval information process based not only on calculating the similarity of the underlying graphs between two artifacts.

The implementation of this approach makes use of a framework that supports semantic information indexing and retrieval, the CAKE API. CAKE is an ontology-based framework that allows us to provide a technical solution exploiting a domain ontology to shift the representation of Simulink models from text-based (names of blocks, etc.) relationships to concept-based relationships.

In this manner, it is possible to enrich the domain language within the Simulink models to make a better interpretation of the embedded information, this mainly requires the mapping between the model and a the domain ontology. CAKE uses the concept of ontology as a way to restrict concepts, which can be

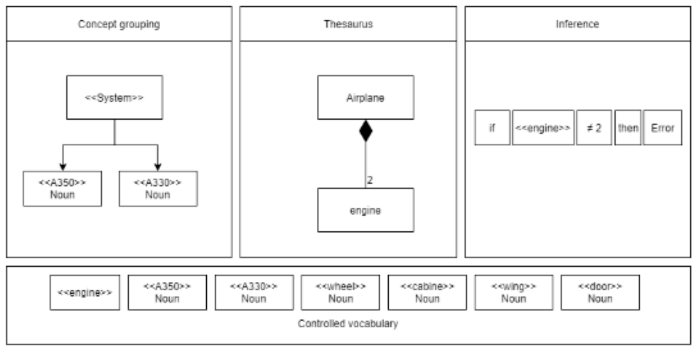

Fig. 2. Example of the knowledge layers representation in RSHP.

used to represent knowledge, as well as endow this vocabulary with syntactic, semantic and pragmatic information.

Figure 2 also shows an example of how knowledge is represented across the different layers within the CAKE-RSHP framework:

1) The controlled vocabulary layer refers to all terminology in a specific domain and it is the basis of the other layers.
2) Grouping terms by concepts allows us to add more semantics to the terminology and will be always restricted by the controlled vocabulary.
3) Thesaurus allows us to represent structure, for example a break down structure, just as in previous stages thesaurus is restricted by the controlled vocabulary.
4) Inference layer allows to execute logic using terminology. This logic can be used to infer new knowledge, or to execute validation rules based on the domain knowledge. These rules can be also executed against any source of knowledge that is represented in RSHP using CAKE, for example logical models, physical models, even textual information.

C. Technological implementation

The proposed solution consists of an application developed in Visual Studio .Net 2019 with framework 4.8, which allows us to parse Simulink files using a Simulink software library for Java [17] and the IKVM (to run Java code within the .NET framework), and to create a semantic representation of the Simulink models using the CAKE-RSHP model. As a consequence of using this framework, it is possible to use the built-in mechanisms already available for indexing and retrieving information [18]. Figure 3 shows the architecture of the proposed solution, which consists of three main elements:

- Simulink2RSHP. This component groups the other two, basically it allows us to semi-automatically apply the mappings between the Simulink elements and the RSHP meta-model creating an underlying semantic graph based on the domain ontology, see Table I.
- Simulink Library. This component groups the other two, basically it allows mapping the objects that are obtained from invoking the reading processes of the Simulink library. Once the information is obtained from the files, it is represented using the Cake API.

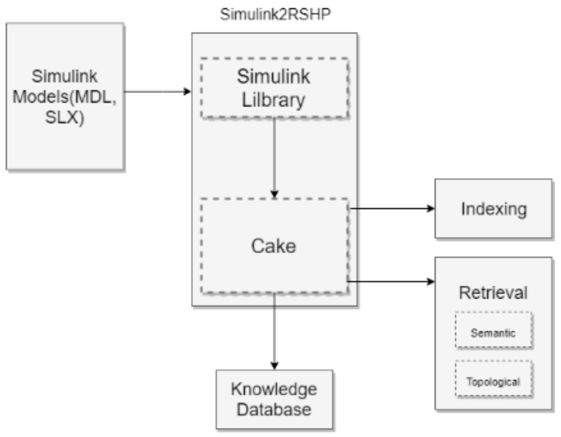

Fig. 3. Conceptual architecture of the proposed solution (RSHP).

TABLE I
EQUIVALENCE BETWEEN SIMULINK AND RSHP.

Simulink Element	**RSHP Element**
Model	Artifact
Block	Artifact
Block Type	Artifact Type
Block Name	Artifact Name, noun term
Block Properties	Metaproperties
Line	RSHP

- CAKE. Once the information is represented in the RSHP language, it is possible to use the built-in capabilities for information retrieval and indexing. The CAKE API internally implements a pattern matching algorithm between graphs that returns a value of similarity.

Following a detailed explanation of the mapping between Simulink elements and the RSHP metamodel is provided, see Table I.

- The global Simulink model is represented as a RSHP Artifact of type Simulink Model, this artifact contains meta-data such as model image, creation date, modification date and any other additional description.
- Each block is represented as an Artifact and the properties of the blocks become RSHP Metaproperties of the artifact. It is important to mention that RSHP and Simulink are very compatible since Simulink blocks contain typology that is represented as the Artifact Type of each block artifact. There are cases where the blocks have names or descriptions, these are represented as the name and description attributes of the artifact.
- Simulink models, unlike SysML, have a single type of relation which is the line, this element is represented in RSPH as a relationship of type ”Line”.
- In cases where Simulink blocks have names, these names are represented as terms with a syntactic tag of type ”Noun”, this also adds more semantic information to the components.

Finally, CAKE also gives us the possibility of grouping the terminology in semantic clusters, which allow adding context to the representation language. In the case of experimentation (see Section IV), no groupings of terms were made.

IV. CASE STUDY: INDEXING AND RETRIEVING SIMULINK MODELS

To illustrate the approach for reusing Simulink models, a case study of indexing and retrieving Simulink models has been conducted.

A. Research design

The experiment to evaluate the advantages of a semantic representation of Simulink models has been designed as follows:

1) Define a dataset of Simulink models from the public website repository of MathWorks [1]. General, automotive and aerospace models have been downloaded to test different domains. This dataset comprises 38 physical models (21 general models, 9 automotive and 8 aerospace) that have been indexed.
2) Define a dataset of queries to evaluate the retrieval capabilities of the proposed solution. Each query has been designed with different common components of models to return a set of Simulink schemes. These queries have also been indexed see Table II.
3) Execute the experiment. For each query defined in the previous step, analyze the models retrieved by Simulink2RSHP taking into account all the semantic information represented into the dataset.
4) Analyze the results and validate them using the schema proposed in [19]. Extract measures of: 1) precision (fraction of retrieved information that is relevant); 2) recall (fraction of relevant information that is retrieved), and 3) a combination between the last two measures, the F1 score.

B. Analysis of results

The analysis of results is based on the levels of ”goodness” (see Table III) established in [20]. Table IV shows the metrics of precision, recall an F1 for each input query. It was found that 10% of queries are at an acceptable level of ”goodness” for precision and a 5% for recall. A 10% of queries obtained a good level for precision metric and a 5% for recall. In the same manner, a 70% of the queries obtained an excellent level for both precision and recall metrics. Finally, just a 10% of the queries obtained a value of precision below acceptable and, in the case of recall, just a 20% of the queries.

The global average of the metrics was excellent for precision and good for recall, since they are above 60% and 70% respectively, as Figure 4 depicts.

This is likely due to the fact that the degree of similarity between the input queries and the dataset of models, was calculated using semantic and topological algorithms, since

[1]https://es.mathworks.com/help/simulink/examples.html

TABLE II
LIST OF QUERIES EXECUTED TO RETRIEVE SIMILAR MODELS.

Q	Query Description
Q_1	Signal connected to a memory and sum block
Q_2	Clock connected with logical operator
Q_3	Vertical channel
Q_4	Sensor
Q_5	Clock connected to an output
Q_6	Logical operator connected to other logical opperator
Q_7	Integrator connected to a Gain connected to a sum
Q_8	Integrator connected to a gravity component
Q_9	Gain connected to a produc connected to another product
Q_{10}	Integraor connected to a signum block
Q_{11}	Clock connected to a relational operator connected to a constant
Q_{12}	Step
Q_{13}	Input connected to a Mux
Q_{14}	Inport connected to a Function
Q_{15}	Costant connected to a switch
Q_{16}	Signum block connected to a transfer function
Q_{17}	Scope connected to a integrator connected to a Gain
Q_{18}	Signum connected to a product
Q_{19}	Ram
Q_{20}	Relay

TABLE III
GOODNESS LEVELS FOR PRECISION AND RECALL METRICS [20].

Level of "goodness"	Precision	Recall
Acceptable	$\geq$20%	$\geq$60%
Good	$\geq$30%	$\geq$70%
Excellent	$\geq$50%	$\geq$80%

TABLE IV
PRECISION, RECALL AND F1 METRICS FOR EACH QUERY.

	Precision	Recall	F1
Q_1	0.2857	0.6667	0.4000
Q_2	0.3333	1.0000	0.5000
Q_3	1.0000	1.0000	1.0000
Q_4	0.2000	0.3333	0.2500
Q_5	0.6250	0.8333	0.7143
Q_6	0.5000	0.5000	0.5000
Q_7	1.0000	1.0000	1.0000
Q_8	0.7500	0.8571	0.8000
Q_9	1.0000	0.9000	0.9474
Q_{10}	1.0000	0.9000	0.9474
Q_{11}	0.8000	1.0000	0.8888
Q_{12}	0.0000	0.0000	0.0000
Q_{13}	0.9412	0.9412	0.9412
Q_{14}	1.0000	1.0000	1.0000
Q_{15}	0.5000	1.0000	0.6667
Q_{16}	0.5000	1.0000	0.6667
Q_{17}	0.6667	1.0000	0.8000
Q_{18}	0.3333	0.7500	0.4615
Q_{19}	1.0000	1.0000	1.0000
Q_{20}	0.0000	0.0000	0.0000
Avg	**0.6218**	**0.7841**	**0.6742**

the more information available in the Simulink blocks such as names and descriptions, is, the more accurate the results were.

However, it was also determined the need to consider within the similarity algorithm more specific aspects within the Simulink blocks. For example, in cases such as logical operator blocks, the algorithm assumes a similarity between these blocks regardless of the type of operator. In other words, for the algorithm there was a similarity between the

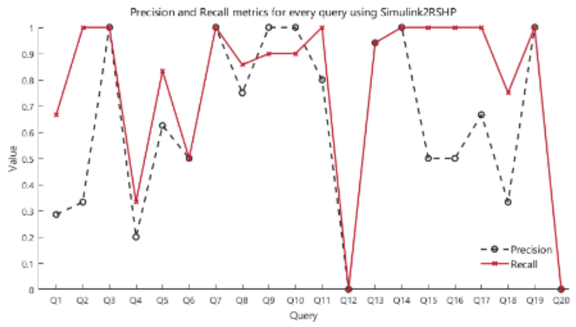

Fig. 4. Precision and Recall metrics results obtained for the proposed solution.

logical operators AND and OR, regardless of whether they are semantically different.

This could be improved using semantic clusters and a controlled vocabulary, to differentiate this type of aspects, and consider more information when determining the similarity of the components. These capabilities are also available in the CAKE API, but in order to refine the algorithm it is necessary to spend more time populating the domain ontoloty. In general, it implies the creation of more specific terminology, thesaurus and semantic clusters for specific Simulink components.

C. Research limitations

One of the main limitations in the research lies in the size of the repository where the queries were made. To carry out more accurate test cases, it would be necessary to have a larger set of Simulink models. Furthermore, a more specific domain ontology for physical models would be necessary to take advantage of other CAKE API capabilities, adjusting the semantic representation to the matching algorithm.

Additionally, the creation of the queries was carried out by randomly selecting components presented in the sample models. A more robust experiment would require the study of each model and the behavior of users to create a more realistic dataset of queries.

V. CONCLUSIONS AND FUTURE WORK

Despite the importance of the reuse of physical models and, the existing alternatives for reusing components and models, the existing MBD tools lack of advanced retrieval mechanisms. Although, these tools have not been designed for this purpose, the reuse mechanisms are a bit naive and, in most of the cases, a mere search query based on some keywords seems too simple to really exploit the information embedded in the models.

In this work, the Simulink2RHSP approach seems to be a promising alternative, considering that unlike many of the retrieval tools that perform text searches, it determines the similarity using a combination of semantic and topological algorithms. The results obtained in the experimentation demonstrate the feasibility of the approach. It is possible to build indexing and retrieval engines for physical models using a semantic representation.

As future work, improvements in the representation of system artifacts are planned, including terminology, thesaurus and semantic clusters. Other types of models will be also included in the experimentation such as those supported in the Modelica language. In terms of experimentation, this small setting is representative to demonstrate the feasibility of the approach. However, larger settings including real user needs are completely required to provide a more significant and realistic validation.

Acknowledgment

The research leading to these results has received funding from the project H2020-ECSEL Arrowhead Tools under grant agreement nº 826452 and from specific national programs and/or funding authorities.

References

[1] B. Schätz, S. Voss, and S. Zverlov, "Automating design-space exploration: Optimal deployment of automotive sw-components in an iso26262 context," in *Proceedings of the 52nd Annual Design Automation Conference*, ser. DAC '15. New York, NY, USA: Association for Computing Machinery, 2015. [Online]. Available: https://doi.org/10.1145/2744769.2747912

[2] M. I. Simulink, "Homepage," [Last accessed 3 May 2020].

[3] G. Beydoun, A. Hoffmann, R. V. Garcia, J. Shen, and A. Gill, "Towards an assessment framework of reuse: a knowledge-level analysis approach," *Complex & Intelligent Systems*, vol. 6, no. 1, pp. 87–95, Apr. 2020. [Online]. Available: http://link.springer.com/10.1007/s40747-019-0116-1

[4] J. M. Á. Rodríguez, R. Mendieta, J. L. de la Vara, A. Fraga, and J. L. Morillo, "Enabling system artefact exchange and selection through a linked data layer," *J. UCS*, vol. 24, pp. 1536–1560, 2018.

[5] J. M. Alvarez-Rodríguez, R. M. Zuñiga, and J. Llorens, "Elevating the meaning of data and operations within the development lifecycle through an interoperable toolchain," in *INCOSE International Symposium*, vol. 29, no. 1. Wiley Online Library, 2019, pp. 1053–1071.

[6] J. M. Alvarez-Rodríguez1, J. Llorens1, M. Alejandres1, and J. M. Fuentes2, "Oslc-km: A knowledge management specification for oslc-based resources," in *INCOSE International Symposium*, vol. 25, no. 1. Wiley Online Library, 2015, pp. 16–34.

[7] P. Atzeni, L. Bellomarini, P. Papotti, and R. Torlone, "Meta-mappings for schema mapping reuse," *Proc. VLDB Endow.*, vol. 12, no. 5, p. 557–569, Jan. 2019. [Online]. Available: https://doi.org/10.14778/3303753.3303761

[8] J. Llorens, J. Morato, and G. Genova, *RSHP: an information representation model based on relationships*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 221–253.

[9] W. B. Croft, D. Metzler, and T. Strohman, *Search engines: Information retrieval in practice*. Addison-Wesley Reading, 2010, vol. 520.

[10] B. Huang, "An effective retrieval approach of 3D CAD models for macro process reuse," *Int J Adv Manuf Technol*, p. 23, 2019.

[11] K. Robles, A. Fraga, J. Morato, and J. Llorens, "Towards an ontology-based retrieval of uml class diagrams," *Information and Software Technology*, vol. 54, no. 1, pp. 72–86, 2012.

[12] M. W. Whalen, A. Murugesan, S. Rayadurgam, and M. P. E. Heimdahl, "Structuring simulink models for verification and reuse," in *Proceedings of the 6th International Workshop on Modeling in Software Engineering - MiSE 2014*. Hyderabad, India: ACM Press, 2014, pp. 19–24. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2593770.2593776

[13] J. R. Cordy, "Submodel pattern extraction for simulink models," in *Proceedings of the 17th International Software Product Line Conference*, ser. SPLC '13. New York, NY, USA: Association for Computing Machinery, 2013, p. 7–10. [Online]. Available: https://doi.org/10.1145/2491627.2492153

[14] E. Gallego, J. M. Alvarez Rodríguez, and J. Llorens, "Reuse of physical system models by means of semantic knowledge representation: A case study applied to modelica," Sep. 2015, pp. 747–757.

[15] R. Mendieta, J. L. de la Vara, J. L. Morillo, and J. M. Álvarez-Rodríguez, "Towards effective sysml model reuse." in *MODELSWARD*, 2017, pp. 536–541.

[16] T. Xin and L. Yang, "A framework of software reusing engineering management," in *2017 IEEE 15th International Conference on Software Engineering Research, Management and Applications (SERA)*, 2017, pp. 277–282.

[17] CQSE, "Simulink Library." [Online]. Available: https://www.cqse.eu/en/products/simulink-library-for-java/overview/

[18] A. Rodrigues, "Tools Exhibits. In UML Modeling Languages and Applications," pp. 281–291, 2005.

[19] N. Juristo and A. M. Moreno, *Basics of software engineering experimentation*. Springer Science & Business Media, 2013.

[20] J. H. Hayes, A. Dekhtyar, and S. K. Sundaram, "Improving after-the-fact tracing and mapping: Supporting software quality predictions," *IEEE software*, vol. 22, no. 6, pp. 30–37, 2005.