Reuse of Physical System Models by means of Semantic Knowledge Representation: A Case Study applied to Modelica

Elena Gallego[1], Jose María Álvarez-Rodríguez[1] and Juan Llorens[1]

[1] Knowledge Reuse Group,
Department of Computer Science and Engineering,
University Carlos III of Madrid, Spain,
{elena.gallego,jmalvarez,llorens}@kr.inf.uc3m.es

Abstract

This paper presents the design and development of a solution to store and reuse physical system models by indexing and retrieving their content and metadata. To do so, a mapping between the representation modelling language and a semantic-based representation model (Relationship-RSHP) is defined. More specifically, electrical circuits designed in Modelica have been mapped to RSHP. A two-step process has been designed and implemented to parse Modelica artifacts and index the contents into a system knowledge repository. Afterwards, a case study has also been conducted to compare text vs. concept based information retrieval processes. A dataset of 25 electrical circuits and a set of 30 queries have been designed to extract precision and recall metrics assessing that the presented approach improves the retrieval of Modelica artifacts. As main conclusion, it is possible to state that a domain specific technology such as RSHP for knowledge representation can help the management of Modelica artifacts as knowledge assets.

Keywords: Information Representation, Physical System Models, Modelica Language, Model Reuse, Knowledge Reuse.

1 Introduction

Cyber-physical systems (CPS), a set of collaborative computational resources controlling physical entities, are considered "*the next computing revolution*" (Rajkumar et al. 2010) (K.-D. Kim and Kumar 2012). The design and deploy of these systems is currently based on the 5C architecture (connection, conversion, cyber, cognition, and configuration). Physical system models are designed at different levels of abstraction to analyze and study the mathematical equations that govern the CPS under different excitation configurations.

To do so, software tools (Fritzson 2015) supporting physical modelling languages are used to design and run the simulations that represent the physical system model behavior. During this stage of design and development a good number of logical artifacts are generated. In this context and with the aim of easing the development of the 5C architecture, software developing environments usually provide libraries of reusable components (M. Kim et al. 2010) through application patterns (Choi et al. 2013) and other techniques. These components are commonly represented in a particular modelling language and tagged with a predefined set of metadata properties that can only be accessed from the same development environment that produced them.

In order to reuse a component, the first step lies on the capability to search for them through a traditional interface, filtering the potential results depending on keywords or fixed values in the metadata fields.

Assuming that a physical system model in some modelling languages, such as Modelica, is a software artifact, it is possible then to apply the well-known techniques for information and software reuse (Jacobson, Griss, and Jonsson 1997) (Karlsson 1995). Reuse of information and software may have the potential of increasing productivity of engineers, improve quality and create a cost efficient development environment for cyber-physical systems.

However, the systematic support of reuse is affected by technical and non-technical issues (Smolárová and Návrat 1997):

1. Economical, organizational, educational or psychological issues and
2. Lack of standards to represent all software artifacts, lack of reusable component libraries or appropriate tools for boosting reuse among tools.

In the context of technical issues, those considered in this paper, the classical principles of (software) reuse: abstraction, selection, specialization and integration, can be found in a very good number of works (Jacobson, Griss, and Jonsson 1997) (Karlsson 1995) (Mcilroy 1969). In particular, *abstraction* (management of the intellectual complexity of an artifact) can be considered the essential feature for any reuse technique in order to specify when an artifact could be reused and how to reuse it. *Selection* refers to the discovery of artifacts covering from the representation and storage to the classification, location and comparison. *Specialization* consists on the set of parameters and transformations required to reuse an artifact, while *integration* refers to the capability of systems to communicate, collaborate and exchange data.

Thus, the reusability factor of artifacts will directly depend on how they are abstractly described, how they can be selected and specialized for reuse, and how they will integrate in the new complete system.

Currently, knowledge management has gained momentum in the software domain as a means to elevate the meaning of the implicit knowledge represented into software pieces. Software is becoming a commodity that is embedded in any work product or business process, being a new kind of intellectual asset that can be used to reduce costs and time to market by generating competitive advantage.

In this light, knowledge management techniques (Nonaka and Takeuchi 1995) can be applied to capture, structure, store and disseminate software-based artifacts to directly support the aforementioned software reuse principles of selection and integration. However, the selection of a proper knowledge management mechanism is still an open issue (Hull and King 1987) due to the fact that a suitable representation model can be reached in several ways.

In the context of cyber-physical systems development, physical system models seem to be a good candidate to take advantage of knowledge management and reuse techniques. Based on this concept, the Modelica modelling language (Fritzson and Engelson 1998) (Fritzson 2015) provides a comprehensible model data structure (Schamai, Fritzson, and Paredis 2013) in which it is possible to develop, design and run simulations.

However, there is much more at stake than the simple representation in a modelling language. Physical systems are represented by equation systems or by graphical models that represent their behavior. This valuable information must be organized and stored to be able to provide high-accurate information retrieval processes. One of the main challenges emerges from the complexity to transform physical systems into a logical structure that can be modeled and understood by knowledge management tools.

Semantic knowledge representation models appeared around year 2000 to cope with complex information representation problems. The most representative example of them can be Resource Description Framework (RDF) (Hayes 2004) and RSHP (pronounced "*arship*") (Llorens, Morato, and Genova 2004). RDF was created from the beginning to cope with web information management while RSHP's main goal was to represent information from all industrial work-products.

In order to overcome the existing limitations on reusing physical system models knowledge, a mapping between the Modelica modelling language and the RSHP information representation model is defined and implemented (Modelica2RSHP). Due to the intrinsic RSHP capabilities, it is possible to represent any kind of information such as textual descriptions, design models, code or even any piece of relation data under the same schema. A tool implementation for managing industrial work products has been developed by The Reuse Company (The Reuse Company Inc. 2014), named knowledgeMANAGER, enabling the possibility of applying knowledge management techniques to engineering domain.

As motivating example, Figure 1 shows a simple amplifier circuit comprising different electrical elements. This block could certainly be reused in different cyber-physical systems. However, in order to allow reuse the proper mechanisms must be provided to represent the elements and relationships within the circuit (metadata and contents), to store such elements in a repository, to define a retrieval algorithm that would allow the identification of physical models by content and to retrieve the block according to different queries. For instance, an engineer should be able to look up this circuit, see Figure 1, by expressing the next query: "*Give me all electrical circuits that contain a sine voltage source directly connected to an operational amplifier by a 20kΩ resistor*". In current Modelica environments, these tasks are hard to accomplish since they were not designed for these purposes. Advanced regular expressions could be a solution but an approach taking advantage of describing elements and relationships can really improve the retrieval of Modelica artifacts boosting the reusability factor of existing physical system models.

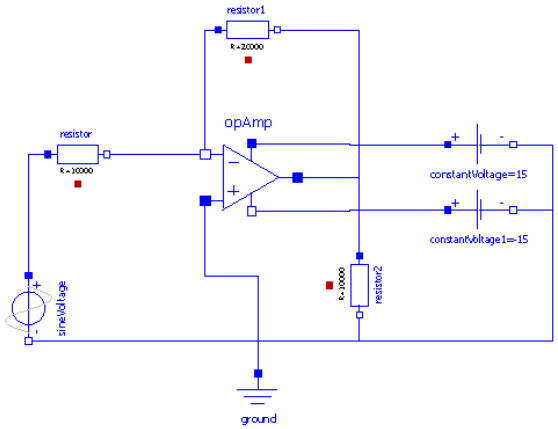

Figure 1. Simple Amplifier circuit which uses an operational amplifier (see example in Electrical-Analog circuits in OpenModelica).

2 Physical system models as software artifacts

Software reuse (Smolárová and Návrat 1997) as a discipline has been widely studied and surveyed from different perspectives. Reuse depending on software metrics and models (Frakes and Terry 1996), reuse of software libraries (A. Mili, Mili, and Mittermeir 1998), software repositories (Guo and others 2000), components in the industry (Land et al. 2009), success

DOI 10.3384/ecp15118747

factors (Basili and Rombach 1991) and reuse in software product lines (Thüm et al. 2014). In all of them, the different authors have explored and classified the mechanisms to store and retrieve software assets. One of the main conclusions in these studies is that successful reuse will come with sophisticated software components storage, representation and retrieval techniques. In this light, the authors in (Guo and others 2000) define a set of orthogonal attributes and six broad classes of methods for software reuse. They also establish criteria (technical, managerial and human factors) to assess and compare classes of methods for software reuse.

Other very relevant works have been focused on applying control engineering techniques (H. Mili 2002) for software reuse. Although some of good experiences have been reported (Tracz 1995), success and failure facts outlined in (Morisio, Ezran, and Tully 2002) and (Desouza, Awazu, and Tiwana 2006) are still open. This situation of software reuse is becoming critical in cyber-physical systems where the time to design, develop and deploy a system is more complicated due to the collaboration with other software and hardware components.

2.1 Physical system models sharing and reuse

When thinking about models reuse, engineers have to deal with the underlying information of a shared model and its relation with the design. Human experience is important to correctly understand, share or reuse models efficiently, while machines usually fail because of the tacit knowledge involved.

In (Winsberg 2001) the authors present a semantic driven design reuse for a 3D scene designed by computing the properties while modelling and enabling the system to recognize similar types by a vertex statics based algorithm.

As (Groza et al. 2009) outlines, over 20 billion CAD models exist with similar geometric aspects. Currently, indexers use alphanumeric numbers with different formats for each group. The developer could be able to design new models based on existing ones and reuse their similar components. More than 75% of new models design could be reused from previous models ensuring that the model fulfills the functionality for which it has been designed.

After this brief analysis, there are many technical problems (including data protection or copyrights) to create agreed knowledge-based representations such as ontologies that can ease the sharing and reuse of physical system models produced by different tools.

One of the most necessary elements, once a common knowledge representation is defined, is to have a good search engine supported by domain knowledge. This is the main goal for future works, to be able not only to store physical system models, but also to look for similar models and retrieve their information using concepts and relationships.

Functional Mock-up Interface (FMI) is described in (Otter, Blochwitz, and Arnold 2013) as a solution to model sharing and reuse. FMI allows to work with different simulation environments, as Modelica, Simulink and SIMPACK just in one interface to enhance model sharing avoiding incompatibilities.

Using current design tools it is possible to get both analytical and visual representation for every developed physical system model.

The analytical information describes the physical laws that model the system while the visual representation usually shows them graphically. Visual information represents a simplified view of the world that the system is modelling. When thinking about reuse of physical models, the approach should be to work with the analytical information, because of the knowledge contained. The analytic part of a model represents the different behaviors that could be in the real world for many configurations.

That is why; the choice made in this work is to index the analytical information of any physical system model, which can be complemented by graphical information when retrieving it easing the understanding of the underlying knowledge.

3 Physical System Models

The complex world where we live has the inherited characteristic to be governed by physic laws, which humans continuously try to control. Every physical system that engineers want to better understand has elements that behave according to a set of physical laws (Winsberg 2001).

Physical systems models represent the reality by means of relationships between physical and mathematical theories and their effect in the reality. There exist many ways to design physical system models but, almost all of them, are constructed under the same theories.

Therefore, if we are aware of the elements that define the system and the physical laws that govern it, we have the required information of the physical system model, in order to get the knowledge, with different abstraction levels, which can be used in other processes or projects.

Physical system models can be as complex as the reality they represent, thus, it is needed to clearly define the purpose of the model in order to get a reasonable result.

The goal, when modelling physical systems, is to get a mathematical representation of the system's behavior in terms of its variables. Depending on the nature of the system, electrical, mechanical or thermal, the system variables change. Despite of the differences, a common concept between the disciplines is energy, so it is possible to design the physical components of the system as energy manipulators (Wellstead)

Physical system models are built to represent the real world where the model is going to be used and its response to particular stimuli. The needs to create physical system models are described in (Valášek et al. 2003) as the real world system, the question to be answered by the simulation of the model, and the interpretation of the output is the solution.

3.1 Physical systems modelling environments

There are many models design environments that offer different capabilities depending on the domain.

Modelica-based modelling and simulation environments such as Dymola (Dempsey 2006), OpenModelica (Asgha and Tariq 2010) or JModelica (Åkesson et al. 2010), are examples of integrated development environments that make easier the visual development of models in domains such as: electric, mechanic or thermodynamic.

More specifically, the Modelica language is an object-oriented programming language that allows physical systems modelling. Models can be expressed by differential, algebraic and discrete equations. Modelica allows reuse and share models by reducing the modelling effort (Martin-Villalba, Urquia, and Dormido 2008). Nevertheless, the knowledge management capabilities of these environments are restricted as it has been outlined in the introduction.

4 Knowledge representation of Physical System Models

In order to provide the proper knowledge management services for cyber-physical systems, it is necessary to select an adequate knowledge representation paradigm. Obviously, different types of knowledge require different types of representation (Davis, Shrobe, and Szolovits 1993) (Groza et al. 2009). In this light, expressions, rule-based systems, regular grammars, semantic networks, object-oriented representations, frames, intelligent agents or case-based models, to name just a few, are some of the main approaches to information and knowledge modelling.

More specifically, knowledge management also implies the standardization of data and information, that is, any block of information must be structured and stored for supporting other application services.

In this context, two main approaches can be highlighted: 1) the ISO 10303-STEP ("Standard for the Exchange of Product model data"), is an standard for the computer-interpretable representation and exchange of product manufacturing information and 2) the Open Services for Lifecycle Collaboration (Ryman, Hors, and Speicher 2013) (OSLC), an OASIS standard, that is seeking new methods to easily integrate System Engineering tools and build an ideal development and operations environment with special focus on interoperability.

Although both approaches represent very relevant actions to standardize and provide interoperable environments for developing complex systems, they do not directly define a knowledge model (Alvarez-Rodríguez et al. 2015) for representing metadata and contents of work products and artifacts. Besides, it has been demonstrated that the retrieval of information resources does not imply the need of any underlying logic formalism but a powerful framework for expressing concepts and relationships. Due to this fact and previous experiences (Alvarez-Rodríguez et al. 2015), the RSHP universal knowledge representation model has been selected as meta model to semantically describe the elements and relationships that can be found in a physical system model.

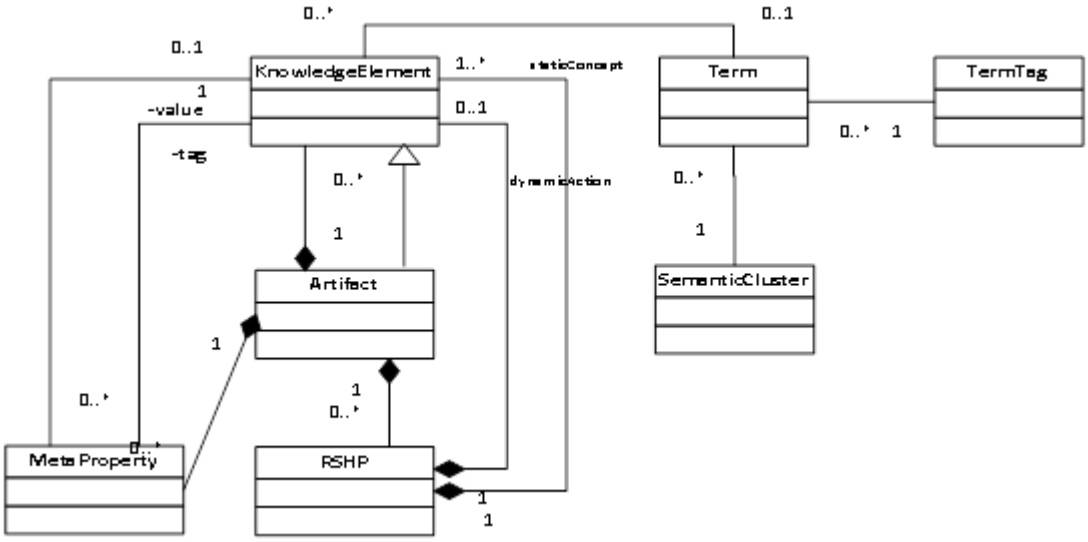

Figure 2. The RSHP representation model in UML.

DOI 10.3384/ecp15118747

4.1 RSHP in a nutshell

The RSHP universal knowledge representation model (Llorens, Morato, and Genova 2004), see Figure 2, is based on the ground idea that whatever information can be described as a group of relationships between concepts forming a conceptual graph. For example, Entity/Relationship data models (Chen 1976) are certainly represented as relationships between entities, processes can be represented as causal/sequential relationships between sub-processes, UML (Unified Modelling Language) or SysML meta models can also be modeled as a set of relationships between meta model elements, etc. Furthermore, free text information can certainly be represented as relationships between terms by means of the same structure. To represent human language text, a set of well-constructed sentences, including the *subject + verb + predicate* (SVP) should be used. The SVP structure can be then considered as a relationship typed V between the S and the predicated P. RSHP includes a repository model to store information and relationships with the aim of reusing all kind of knowledge chunks. The RSHP formal representation model, see Figure 2, is based on the following principles:

- The main description element is the relationship since it is the element in charge of linking knowledge elements.
- A Knowledge Element (KE) is an atomic knowledge brick that appears into an artifact and that is linked by one or more relationships with other KEs, to build information. It is defined by a concept, and it can also be an artifact (an information container found inside a wider artifact). A concept is represented by a normalized term (a keyword coming from a controlled vocabulary, or domain). Artifacts are knowledge containers of KEs and their relationships.

In RSHP, the simple representation model for describing the content of whatever artifact type (requirements, risks, models, tests, maps, text docs or source code) should be:

RSHP representation for artifact

$$\alpha = i_\alpha = \{(RSHP_1), (RSHP_2), \dots, (RSHP_n)\}$$

where every single RSHP is called RSHP-description and must be described using KE.

One important consequence of this representation model is that there is no restriction to represent a particular type of knowledge. Furthermore, RHSP has been used as underlying information model to build general-purpose indexing and retrieval systems, domain representation models (Díaz et al. 2005), quality assessment of requirements and knowledge management tools such as knowledgeMANAGER (The Reuse Company Inc. 2014) .

4.2 Mapping the Modelica language to RSHP

The use of Modelica as language for modelling complex physical systems is gaining momentum in the industry domain (Samlaus and Fritzson 2015). On the other hand, RSHP has been used for a long time in the Systems Engineering discipline for knowledge management. Given this situation, a strategy to map Modelica physical system models to RSHP must be defined. To do so a direct mapping is defined to perform simple transformations and to provide a basis for defining and comparing more complex transformations.

In order to design this direct mapping, both models are represented using the commonly defined abstract data types set and list. The definitions follow a type-as-specification approach (Schamai, Fritzson, and Paredis 2013); thus models are based on dependent types that can also include cardinality. More specifically, Table 1 and Table 2 show both specifications as a kind of regular tree grammars that can be used to specify a rule-based transformation between two grammars (denotational semantics). Thus, a transformation between a partial set of production rules of the Modelica language and RHSP can be defined as a function, Modelica2RSHP, that takes the Modelica grammar (v3.2), $G_{Modelica}$, a valid Modelica model, $Modelica_k$, the RSHP grammar G_{RSHP} and a set of direct mapping rules, $M_{Modelica2rshp}$ (see Table 3 where sub-indexes refer to attributes and relationships of the elements), to generate a valid $RSHP_{graph}$.

$$Modelica2RSHP: G_{Modelica} \times Modelica_k \times G_{RSHP} \times M_{Modelica2rshp} \rightarrow RSHP_{graph}$$

Table 1. Selected Production rules of the Regular Tree Grammar of Modelica: $\boldsymbol{G_{Modelica}}$

```
(1) class_definition   ::=      class_prefixes
    class_specifier
(2) class_prefixes ::= (model)
(3) class_specifier::=  long_class_specifier |
(4) short_class_specifier
(5) long_class_specifier                    ::=
     IDENT string_comment composition  end IDENT
    | extends IDENT [class_modification]
(6)  string_comment composition  end  IDENT
(7) short_class_specifier
    ::= IDENT "=" base_prefix name [array_subs
    cripts]    [class_modification]    comment
             | IDENT "=" enumeration "(" ( [enu
    m_list] | ":" ) ")" comment
(8) component_clause    ::=        type_prefix
    type_specifier          [array_subscripts]
    component_list
(9) type_specifier ::=  name
(10) name ::= ["."] IDENT {"." IDENT }
(11) component_list ::= component_declaration {
    "," component_declaration }
(12) component_declaration       ::=declaration
    [condition_attribute] comment
(13) declaration  ::=  IDENT  [array_subscripts]
    [modification] ->KE | Term
```

```
(14) connect_clause    ::=        connect    "("
     component_reference "," component_reference
     ")
(15) component_reference    ::=    ["."]    IDENT
     [array_subscripts]        {"."        IDENT
     [array_subscripts] }
```

Table 2. Regular Tree Grammar of RSHP: G_{RSHP}

```
 (1) Artifact ::= (Set(RHSP), MetaProperty{0,*})
(2)  RSHP ::= (Subject, Verb, Object, Semantics)
(3)  Subject ::= KE {0,1}
(4)  Verb    ::= KE {0,1}
(5)  Object  ::= KE {0,1}
(6)  KE :: = (Term {0,1}) | Artifact
(7)  Term    ::=    (lexicalForm,    languageTag,
     TermTag)
(8)  TermTag    ::= lexicalForm
(9)  MetaProperty ::= (Tag, Value)
(10) Tag   ::= {KE, lexicalForm}
(11) Value ::= {KE {0,1}, lexicalForm {0,1}}
(12) SemanticCluster ::= (Term)
```

Table 3. Set of mapping rules $M_{Modelica2rshp}$ to transform Modelica physical system models into RSHP

```
(1)  class_definition ::=  Artifact
(2)  class_prefixes       ::=       MetaProperty
     (Tag="type", Value="model")
(3)  class_specifier::=  long_class_specifier |
     short_class_specifier
(4)  long_class_specifier                        ::=
     Artifact(physical_name=IDENT)
(5)  short_class_specifier                       ::=
     Artifact(physical_name=IDENT)
(6)  component_clause    ::=        type_prefix
     type_specifier    [    array_subscripts    ]
     component_list
(7)  type_specifier ::=  name
(8)  name ::= SemanticCluster (Term=IDENT)
(9)  component_list ::= component_declaration {
     "," component_declaration }
(10) component_declaration  ::=  declaration  [
     condition_attribute ] comment
(11) declaration ::= KE (Term = IDENT)
(12) connect_clause ::=RSHP(KE, KE, KE, KE)
(13) component_reference ::= KE (Term = IDENT)
```

Although, the presented mapping does not cover all production rules in $G_{Modelica}$, it is correct since only valid Modelica and RSHP models will be accepted and generated.

4.3 Implementation details

In order to implement the mapping rules presented in Table 3, a stepwise process has been carried out. Taking into account that RSHP and its underlying technology (the CAKE API[1]) are implemented in the .NET platform and considering the diversity of Modelica parsers, we selected the option of building the JModelica sources (Java) for Modelica version 3.2.

More specifically, the last JModelica sources[2] were checked out (January 2015) and built using Apache Ant for Java. Afterwards, a JAR (Java Archive) analyzer tool[3] was used to extract the dependencies between the different Java libraries and to generate a script that transformed the required Java libraries to .NET DLLs (Dynamic-link library).

This approach was enough to demonstrate the possibility of integrating a Modelica parser in the .NET platform. Thus, it is possible now to offer a universal information representation model to index and retrieve physical system models metadata and contents.

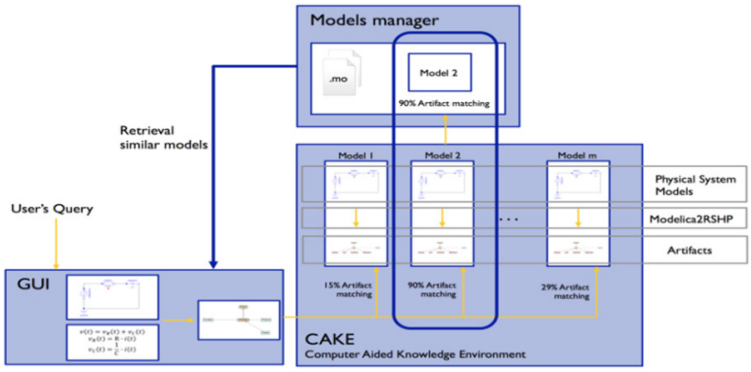

Figure 3. Process to index, search and retrieve a physical system model.

These DLLs are then interpreted in .NET through the IKVM[4] (a Java interpreter for this platform) providing a port and implementation of the Modelica parser. Finally, this set of .NET libraries are used to implement the set of mapping rules in Table 3 and to connect to the CAKE API as Figure 3 shows. Moreover, the knowledgeMANAGER tool can be used to manage all the generated artifacts, see Figure 4.

[1] The CAKE (Computer Aided Knowledge Environment) API (Application Programming Interface).

[2] http://trac.jmodelica.org/browser/trunk/Compiler/ModelicaCompiler

[3] https://code.google.com/p/jar2ikvmc/

[4] http://www.ikvm.net/

DOI 10.3384/ecp15118747

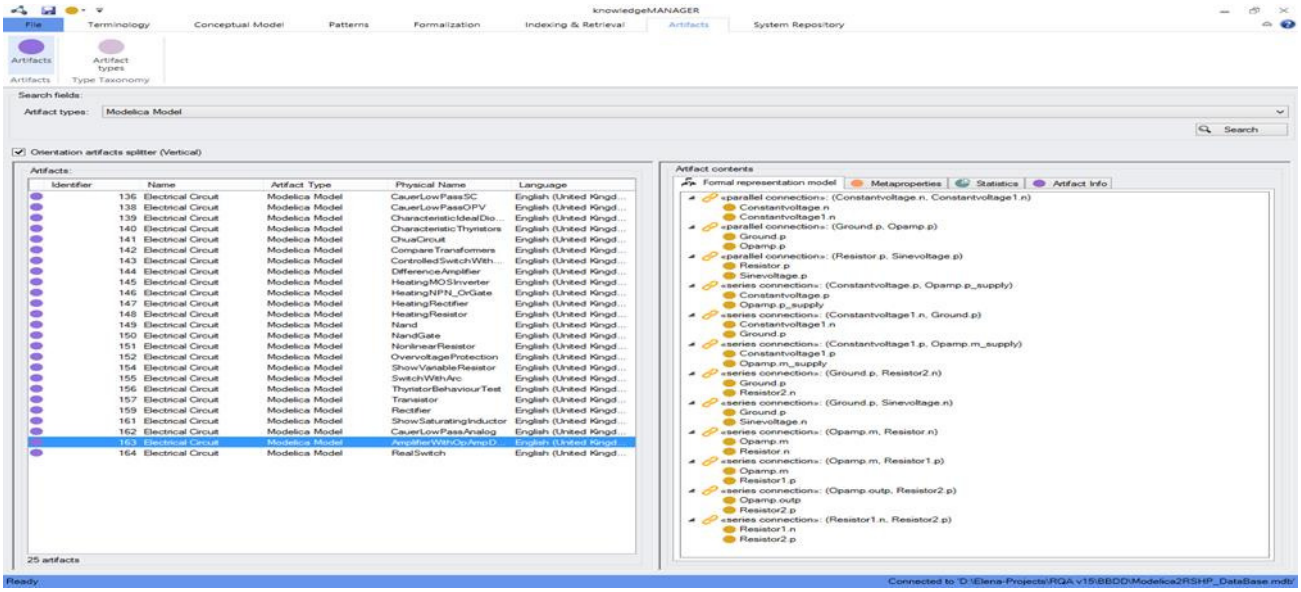

Figure 4. Representation of the physical system models in knowledgeMANAGER

5 Case Study: Indexing and retrieval of Modelica physical system models

To illustrate the approach for reusing physical models, a case study based on the comparison of precision and recall measures of the two approaches to retrieve physical system models (OpenModelica vs knowledgeMANAGER) is presented below.

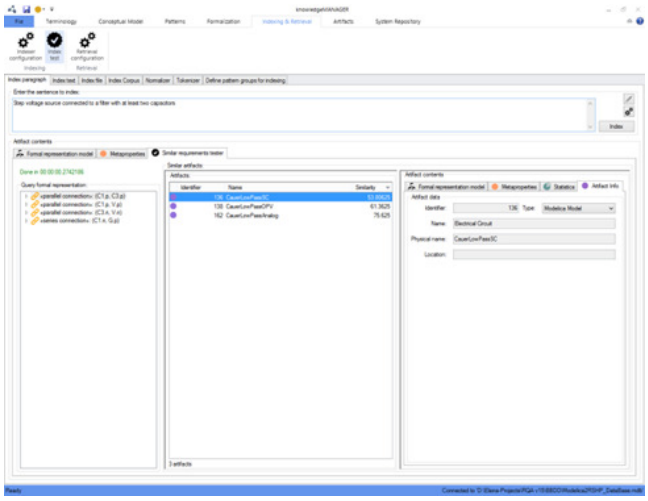

Figure 5. Example of physical system model retrieval in knowledgeMANAGER.

5.1 Research design

One of the main stages in a reuse process consists on looking up the proper artifacts according to a set of preferences or query. This can be interpreted as a search system in which given a query (text-based or even a target model) and a set of resources (a set of physical models), it is necessary to stablish which are the best models that match the input query. To do so, the following steps will be carried out:

3. Design a domain-based vocabulary, O, to represent the concepts and relationships that will be used to represent physical models. In this case, the built in domain ontology in the knowledgeMANAGER has been used. It is actually a taxonomy comprising three main entities: System, Subsystem and Component and hierarchy relationships (*part-of, is-a, broader/narrower*).
4. Define a test dataset of physical models specifications $D = \{d_1, d_2, \dots, d_k, \dots, d_n\}$. To do so, the public dataset of electrical circuits available in OpenModelica has been selected. This dataset comprises 25 physical system models for electrical circuits that have been also indexed in knowledgeMANAGER, see Figure 4.
5. Define a set of queries and expected results, Q where each query q_k will return a set of physical models D_k. To do so, a random walk process on top of the dataset D has been implemented to automatically generate search queries based on the combination of the different elements that can be found in a circuit (between 1-5). Afterwards, a panel of three experts has validated the expected circuits for every query, see Table 4.

$$Q = \{ (q_1, D_1), (q_2, D_2), \dots (q_k, D_k) \dots, (q_n, D_n)\}.$$

6. Run the indexing and retrieval processes implemented on top of the knowledgeMANAGER APIs and the OpenModelica editor. See an example in Figure 5.
7. Extract measures of precision (P), recall (R) and the $F1$ score (the harmonic mean of precision and recall) making a comparison of the expected and generated results. Being $P = \frac{tp}{tp+fp}$, $R = \frac{tp}{tp+fn}$ and , $F1 = \frac{2\,P*R}{P+R}$ where given a target dataset of physical

models, D, and a query q_k which expected results is the set D_k:

- tp (true positive) is *the number of physical models in D_k that have been retrieved and are in D*,
- fp (false positive) is *the number of physical models in D_k that have been retrieved and are not in D*,
- tn (true negative) is *the number of physical models in D_k that have not been retrieved and are not in D* and
- fn (false negative) is *the number of physical models in D_k that have not been retrieved and are in D*.

Table 4. Set of queries to search for physical system models.

Q	**Human-based query**
q_1	Step voltage source with an RLC filter
q_2	LC filter with any kind of voltage source
q_3	Step voltage source connected to a filter with at least two capacitors
q_4	Step voltage source and operational amplifier
q_5	Comparator operational amplifier
q_6	Diode connected to a sine voltage source
q_7	Ideal Operational amplifier integrator
q_8	Rectifiers with ideal diodes
q_9	Sine voltage source connected to a load by a diode
q_{10}	Sine voltage source connected to a load by two ideal thyristors
q_{11}	Sine voltage source connected to a load by one ideal thyristor
q_{12}	Circuits with thermal resistor and LC filter
q_{13}	Sine voltage source connected to a potentiometer (variable resistor) before a RC filter
q_{14}	Sine voltage source connected to a potentiometer (variable resistor)
q_{15}	Rectifiers with inductances to any load
q_{16}	Inductance filter to a sine voltage source
q_{17}	Sine voltage source connected to a potentiometer to supply a resistive load
q_{18}	Circuits with sine voltage source and a variable resistor
q_{19}	Sine voltage source connected to a resistor
q_{20}	Constant voltage source connected to a LR filter by a switch
q_{21}	Constant voltage source connected to a load by a switch
q_{22}	Sine voltage source and operational amplifier
q_{23}	Simplified transformer connected to a resistive load by resistors
q_{24}	Simplified transformer connected to a resistive load by inductors
q_{25}	Ideal transformer connected to a sine voltage source
q_{26}	switch controlled by a sine voltage source
q_{27}	Sine voltage source with an RLC filter
q_{28}	Sine voltage source connected to a transistor
q_{29}	Circuit whit thermal conductor and heat capacitor
q_{30}	Sine voltage source connected to a capacitive load

8. Check the robustness of the comparison by performing statistical hypothesis testing.

5.2 Results and Discussion

Table 5 shows the metrics of precision, recall and the F1 measure of the different executions. The first column corresponds to the query identifier; the next three columns contain the metric values when the OpenModelica search capabilities are used to look up circuits. After that, the second experiment shows the metric values when the presented approach implemented on top of knowledgeMANAGER is executed. According to the results, it seems clear that the presented approach is better than the results provided by OpenModelica, as Figure 6 depicts. The main reason of this behavior is due to the fact that the presented approach can take advantage of exploiting semantic relationships (knowlegeMANAGER) while the text-based approach (OpenModelica) can only perform string comparisons.

Nevertheless, the precision values can be improved and higher-values would be expected in both approaches. In the case of knowledgeMANAGER, this is because of the detail of the query, when it has more components to compare, the precision is higher. The tool prefers not to return false positives keeping precision higher.

On the other hand, a statistical hypothesis testing has been carried out to demonstrate if results will vary depending on the type of method or tool used to search physical models. To do so, a comparison of the precision values of both tools and approaches has been formulated through the next hypotheses:

$\boldsymbol{H_o}$: There is no change in the calculation of precision when searching using OpenModelica or knowledgeMANAGER.

$\boldsymbol{H_1}$: There is change in the calculation of precision when searching using OpenModelica or knowledgeMANAGER.

In order to run the statistical hypothesis testing, the F-Test with alpha 0.05 has been carried out to ensure that variances are unequal (there is statistical significance). After that, the t-Test of two-sample assuming unequal variances has been performed with alpha 0.05 to assert whether H_o is rejected or not. According to Table 6, H_o can be rejected, since the t Stat is less than "-t Critical (two tail)". In conclusion, the knowledgeMANAGER tool method exploiting semantic relationships can improve in terms of precision the problem of retrieving the proper physical system models.

DOI 10.3384/ecp15118747

Table 5. Precision and recall metrics for a retrieval process in OpenModelica and knowledgeMANAGER.

	OpenModelica		knowledgeMANAGER			
Q	**P**	**R**	**F1**	**P**	**R**	**F1**
q_1	0.017	0.500	0.032	1.000	0.080	0.148
q_2	0.017	1.000	0.033	1.000	0.040	0.077
q_3	0.034	1.000	0.066	1.000	0.120	0.214
q_4	0.200	1.000	0.333	1.000	0.080	0.148
q_5	0.300	1.000	0.462	1.000	0.083	0.154
q_6	0.018	1.000	0.035	1.000	0.000	0.000
q_7	0.300	1.000	0.462	1.000	0.083	0.154
q_8	0.036	1.000	0.069	1.000	0.042	0.080
q_9	0.006	0.500	0.012	1.000	0.042	0.080
q_{10}	0.000	0.000	N/A	1.000	0.000	0.000
q_{11}	0.000	0.000	N/A	1.000	0.000	0.000
q_{12}	0.000	1.000	0.000	1.000	0.000	0.000
q_{13}	0.002	1.000	0.004	1.000	0.040	0.077
q_{14}	0.004	1.000	0.008	0.333	0.045	0.080
q_{15}	0.006	0.500	0.012	0.500	0.043	0.080
q_{16}	0.000	0.000	N/A	0.200	0.056	0.087
q_{17}	0.002	1.000	0.004	1.000	0.040	0.077
q_{18}	0.002	1.000	0.004	1.000	0.000	0.000
q_{19}	0.100	1.000	0.182	0.500	0.048	0.087
q_{20}	0.077	1.000	0.143	0.500	0.042	0.077
q_{21}	0.077	0.500	0.133	0.500	0.043	0.080
q_{22}	0.100	1.000	0.182	0.333	0.043	0.077
q_{23}	0.007	1.000	0.015	1.000	0.040	0.077
q_{24}	0.007	1.000	0.015	1.000	0.040	0.077
q_{25}	0.007	1.000	0.015	1.000	0.040	0.077
q_{26}	0.011	1.000	0.022	1.000	0.040	0.077
q_{27}	0.000	1.000	0.000	1.000	0.000	0.000
q_{28}	0.011	0.500	0.022	1.000	0.042	0.080
q_{29}	0.006	1.000	0.011	0.667	0.083	0.148
q_{30}	0.000	0.000	N/A	1.000	0.000	0.000

Table 6. The t-Test of two-sample assuming unequal variances to compare OpenModelica vs knowledgeMANAGER for physical models retrieval.

	OpenModelica Precision	*knowledgeMANAGER Precision*
Mean	0.044886824	0.851111111
Variance	0.006732369	0.068102171
Observations	30	30
Hypothesized	0	
Df	35	
t Stat	-16.14230163	
P(T<=t) one-tail	4.32626E-18	
t Critical (one tail)	1.689572458	
P(T<=t) two tail	8.65252E-18	
t Critical (two tail)	2.030107928	

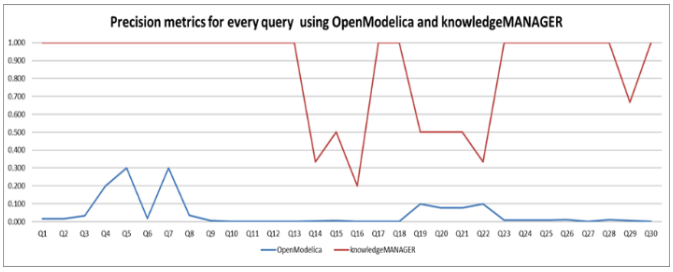

Figure 6 Precision and recall for every query and approach.

5.3 Research Limitations

Some key limitations of the presented work must be outlined. The first one relies on the sample size; our research study has been conducted in a closed world. More specifically, the physical models have been taken from a public repository and the set of queries has been automatically generated through a random walk process. That is why results in a broad or real scope could change, in terms of precision, since more complex relationships in circuits and queries could be designed. Nevertheless, the research methodology, the design of experiments and the creation of a kind of benchmark for testing retrieval processes have been demonstrated to be representative and creditable.

Regarding the generation of queries, the process creates queries similar to the way a domain expert would do. In this case, we have focused on a random combination of circuit elements due to the fact that the handmade creation of queries requires a more in-depth analysis of every circuit. This situation also implies a high probability of losing robustness due to the fact that the same domain can be interpreted according to different experts and domain discourses. However, we consider that the precision and recall metrics are helpful to make a first estimation of the advantages of using a domain ontology and knowledge representation mechanisms to retrieve physical models.

Besides, it has not been possible to fully compare both OpenModelica Connection Editor with knowledgeMANAGER because of the structure of the queries. In the text-based browser of OpenModelica it is complicated to look for several components at the same time and no advanced query mechanisms such as regular expressions are available. That is why, the precision is lower but the recall is most of times very high.

Building on the previous comments, we cannot either figure out the internal budget, methodologies, domain vocabularies, experience and background of specific domain-experts to create and query physical models. We merely observe and re-use existing public and on-line knowledge sources to provide an accurate information reuse process for physical model artifacts.

6 Conclusions and Future Work

Physical system models are not anymore isolated pieces of code to design a physical system. Current trends to develop and deploy cyber-physical systems imply the need of applying knowledge management techniques to save time and to develop safer and more secure systems. In this context, the reuse of existing and well-tested knowledge embedded into physical system models is a challenging task that can be carried out by using the proper mechanism for knowledge management. The RSHP representation model offers a flexible technique to represent any kind of knowledge through concepts and relationships. It also includes technology support through the knowledgeMANAGER tool. It seems clear that the shifting of the underlying information in physical system models to a more adequate representation improves the capabilities to discover and reuse existing knowledge.

As future work, we plan to extend the approach to any kind of physical system model (full support to the Modelica language) providing semantic engines for indexing and retrieving information. Furthermore, we will extend the experiments to make comparisons in a broad scope (tools, models and queries) releasing also the information under the principles of the OpenScience initiative.

Acknowledgements

The research leading to these results has received funding from the ARTEMIS Joint Undertaking under grant agreement Nº 332830-CRYSTAL (CRitical sYSTem engineering AcceLeration project) and from specific national programs and/or funding authorities. This work has been supported by the Spanish Ministry of Industry.

References

Åkesson, J., K. E. Årzén, M. Gäfvert, T. Bergdahl, and H. Tummescheit 2010 Modeling and Optimization with Optimica and JModelica.org-Languages and Tools for Solving Large-Scale Dynamic Optimization Problems. Computers and Chemical Engineering 34(11): 1737–1749.

Alvarez-Rodríguez, Jose Maria, Juan Llorens, Manuela Alejandres, and Jose Fuentes 2015 OSLC-KM: A Knowledge Management Specification for OSLC-Based Resources. In Proceedings of the 25th Annual INCOSE International Symposium (Accepted).

Asgha, Syed Adeel, and Sonia Tariq 2010 Design and Implementation of a User Friendly OpenModelica Graphical Connection Editor.

Basili, V. R., and H. D. Rombach 1991 Support for Comprehensive Reuse. Softw. Eng. J. 6(5): 303–316.

Chen, Peter Pin-Shan 1976 The Entity-Relationship Model—toward a Unified View of Data. ACM Transactions on Database Systems (TODS) 1(1): 9–36.

Choi, Jong-Seok, Tim McCarthy, Maneesh Yadav, et al. 2013 Application Patterns for Cyber-Physical Systems. In Cyber-Physical Systems, Networks, and Applications (CPSNA), 2013 IEEE 1st International Conference on Pp. 52–59. IEEE.

Davis, Randall, Howard Shrobe, and Peter Szolovits 1993 What Is a Knowledge Representation? AI Magazine 14(1): 17.

Dempsey, Mike 2006 Dymola for Multi-Engineering Modelling and Simulation. 2006 IEEE Vehicle Power and Propulsion Conference, VPPC 2006.

Desouza, Kevin C., Yukika Awazu, and Amrit Tiwana 2006 Four Dynamics for Bringing Use Back into Software Reuse. Commun. ACM 49(1): 96–100.

Díaz, Irene, Juan Llorens, Gonzalo Genova, and José Miguel Fuentes 2005 Generating Domain Representations Using a Relationship Model. Information Systems 30(1): 1–19.

Frakes, William, and Carol Terry 1996 Software Reuse: Metrics and Models. ACM Computing Surveys (CSUR) 28(2): 415–435.

Fritzson, Peter 2015 Principles of Object-Oriented Modeling and Simulation with Modelica 3.3: A Cyber-Physical Approach. 2. ed. New York: John Wiley & Sons Inc.

Fritzson, Peter, and Vadim Engelson 1998 Modelica - A Unified Object-Oriented Language for System Modelling and Simulation. In ECOOP'98 - Object-Oriented Programming, 12th European Conference, Brussels, Belgium, July 20-24, 1998, Proceedings Pp. 67–90. http://dx.doi.org/10.1007/BFb0054087.

Groza, Tudor, Siegfried Handschuh, Tim Clark, S Buckingham Shum, and Anita de Waard 2009a A Short Survey of Discourse Representation Models.

Groza, Tudor, Siegfried Handschuh, Tim Clark, S Buckingham Shum, and Anita de Waard 2009b A Short Survey of Discourse Representation Models.

Guo, Jiang, and others 2000 A Survey of Software Reuse Repositories. In Engineering of Computer-Based Systems, IEEE International Conference on the Pp. 92–92. IEEE Computer Society.

Hayes, Patrick 2004 RDF Semantics. World Wide Web Consortium. http://www.w3.org/TR/rdf-mt/.

Hull, Richard, and Roger King 1987 Semantic Database Modeling: Survey, Applications, and Research Issues. ACM Computing Surveys (CSUR) 19(3): 201–260.

Jacobson, Ivar, Martin Griss, and Patrik Jonsson 1997 Software Reuse: Architecture, Process and Organization for Business Success. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co.

Karlsson, Even-André, ed. 1995 Software Reuse: A Holistic Approach. New York, NY, USA: John Wiley & Sons, Inc.

Kim, Kyoung-Dae, and Panganamala R Kumar 2012 Cyber–physical Systems: A Perspective at the Centennial. Proceedings of the IEEE 100(Special Centennial Issue): 1287–1308.

Kim, Minyoung, M-O Stehr, Jinwoo Kim, and Soonhoi Ha 2010 An Application Framework for Loosely Coupled Networked Cyber-Physical Systems. In Embedded and

Ubiquitous Computing (EUC), 2010 IEEE/IFIP 8th International Conference on Pp. 144–153. IEEE.

Land, Rikard, Daniel Sundmark, Frank Lüders, Iva Krasteva, and Adnan Causevic 2009 Reuse with Software Components-a Survey of Industrial State of Practice. In Formal Foundations of Reuse and Domain Engineering Pp. 150–159. Springer.

Llorens, Juan, Jorge Morato, and Gonzalo Genova 2004 RSHP: An Information Representation Model Based on Relationships. In Soft Computing in Software Engineering. Ernesto Damiani, Mauro Madravio, and LakhmiC. Jain, eds. Pp. 221–253. Studies in Fuzziness and Soft Computing. Springer Berlin Heidelberg. http://dx.doi.org/10.1007/978-3-540-44405-3_8.

Martin-Villalba, Carla, Alfonso Urquia, and Sebastian Dormido 2008 An Approach to Virtual-Lab Implementation Using Modelica. Mathematical and Computer Modelling of Dynamical Systems 14(4): 341–360.

Mcilroy, Doug 1969 Mass-Produced Software Components. In Proceedings of Software Engineering Concepts and Techniques. J. M. Buxton, P. Naur, and B. Randell, eds. Pp. 138–155. Garmisch, Germany: NATO Science Committee. http://homepages.cs.ncl.ac.uk/brian.randell/NATO/nato1968.PDF.

Mili, Ali, Rym Mili, and Roland T Mittermeir 1998 A Survey of Software Reuse Libraries. Annals of Software Engineering 5: 349–414.

Mili, Hafedh 2002 Reuse Based Software Engineering: Techniques, Organization and Measurement. New York: Wiley.

Morisio, M., M. Ezran, and C. Tully 2002 Success and Failure Factors in Software Reuse. IEEE Transactions on Software Engineering 28(4): 340–357.

Nonaka, Ikujiro, and Hirotaka Takeuchi 1995 The Knowledge-Creating Company: How Japanese Companies Create the Dynamics of Innovation. New York: Oxford University Press.

Otter, Martin, Torsten Blochwitz, and Martin Arnold 2013 Functional Mock-up Interface for Model Exchange and Co-Simulation: 1–120.

Rajkumar, Ragunathan Raj, Insup Lee, Lui Sha, and John Stankovic 2010 Cyber-Physical Systems: The next Computing Revolution. In Proceedings of the 47th Design Automation Conference Pp. 731–736. ACM.

Ryman, Arthur G., Arnaud Le Hors, and Steve Speicher 2013 OSLC Resource Shape: A Language for Defining Constraints on Linked Data. In LDOW.

Samlaus, Roland, and Peter Fritzson 2015 Semantic Validation of Physical Models Using Role Models. Simulation 91(4): 383–399.

Schamai, Wladimir, Peter Fritzson, and Christiaan J. J. Paredis 2013 Translation of UML State Machines to Modelica: Handling Semantic Issues. Simulation 89(4): 498–512.

Smolárová, Mária, and Pavol Návrat 1997 Software Reuse: Principles, Patterns, Prospects. CIT. Journal of Computing and Information Technology 5(1): 33–49.

The Reuse Company Inc. 2014 knowlegeMANAGER (KM). Industry website. knowledgeMANAGER. http://www.reusecompany.com/knowledgemanager, accessed October 15, 2014.

Thüm, Thomas, Sven Apel, Christian Kästner, Ina Schaefer, and Gunter Saake 2014 A Classification and Survey of Analysis Strategies for Software Product Lines. ACM Computing Surveys 47(1): 1–45.

Tracz, Will 1995 Confessions of a Used Program Salesman: Institutionalizing Software Reuse. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

Valášek, M, P Steinbauer, J Kolář, and J Dvořák 2003 Concurrent Design of Railway Vehicles by Simulation Model Reuse 43(6): 9–15.

Wellstead, Peter E 1979 Introduction to Physical System Modelling. London: Academic Press.

Winsberg, Eric 2001 Simulations, Models, and Theories: Complex Physical Systems and Their Representations. Philosophy of Science 68(S1): S442.