Journal of Universal Computer Science, vol. 24, no. 11(2018), 1536-1560
submitted: 27/2/18, accepted: 7/9/18, appeared: 28/11/18 © J.UCS

Enabling System Artefact Exchange and Selection through a Linked Data Layer

Jose María Alvarez-Rodríguez
(Carlos III University of Madrid, Madrid, Spain
josemaria.alvarez@uc3m.es)

Roy Mendieta
(Carlos III University of Madrid, Madrid, Spain
roy.mendieta@kr.inf.uc3m.es)

Jose Luis de la Vara
(Carlos III University of Madrid, Madrid, Spain
jvara@inf.uc3m.es)

Anabel Fraga
(Carlos III University of Madrid, Madrid, Spain
afraga@inf.uc3m.es)

Juan Llorens
(Carlos III University of Madrid, Madrid, Spain
juan.llorens@uc3m.es)

Abstract: The use of different techniques and tools is a common practice to cover all stages in the systems development lifecycle, generating a very good number of system artefacts. Moreover, these artefacts are commonly encoded in different formats and can only be accessed, in most cases, through proprietary and non-standard protocols. This scenario can be considered a real nightmare for software or systems reuse. Possible solutions imply the creation of a real collaborative development environment where tools can exchange and share data, information and knowledge. In this context, the OSLC (Open Services for Lifecycle Collaboration) initiative pursues the creation of public specifications (data shapes) to exchange any artefact generated during the development lifecycle, by applying the principles of the Linked Data initiative. In this paper, the authors present a solution to provide a real multi-format system artefact reuse by means of an OSLC-based specification to share and exchange any artefact under the principles of the Linked Data initiative. Finally, two experiments are conducted to demonstrate the advantages of enabling an input/output interface based on an OSLC implementation on top of an existing commercial tool (the Knowledge Manager). Thus, it is possible to enhance the representation and retrieval capabilities of system artefacts by considering the whole underlying knowledge graph generated by the different system artefacts and their relationships. After performing 45 different queries over logical and physical models stored in Papyrus, IBM Rhapsody and Simulink, results of precision and recall are promising showing average values between 70-80%.

Keywords: Linked data, OSLC, data shapes, interoperability, retrieval systems
Categories: D.2.12, D.2.13, D.2.11

1 Introduction

Software and system artefact reuse [Mili et al., 1995] is commonly defined as a process to systematically specify, produce, classify, retrieve and adapt software-based artefacts for the purpose of using them in a development process. In general, software reuse [Krueger, 1992] may have the potential of increasing productivity of engineers, improve quality and create a cost efficient development environment. However, both technical and non-technical issues for a limited reuse can be found [Smolárová and Návrat, 1997]: 1) economical, organizational, educational or psychological issues and 2) lack of standards to represent artefacts, and lack of reusable component libraries or appropriate tools for boosting reuse and interoperability among tools.

In the context of technical issues, systems and software engineering techniques have been widely studied [Boehm, 1981] to support the classical principles of reuse [Krueger, 1992]: abstraction, selection, specialization and integration. More specifically, abstraction (i.e. management of the intellectual complexity of a software artefact) can be considered the essential feature for any reuse technique to specify when an artefact could be reused and how to reuse it. Selection refers to the discovery of software artefacts, covering from the representation and storage to the classification, location and comparison. Specialization consists on the set of parameters and transformations required to reuse a software artefact, while integration refers to the capability of software systems to communicate, collaborate and exchange data. Thus, the reusability factor of system artefacts [Fortune and Valerdi, 2008] will directly depend on how they are abstractly described, how they can be selected and specialized for reuse, and how they will be integrated in a target software-based system. Furthermore, a reuse approach implies that every artefact generated during the development lifecycle is not any more an isolated requirement specification, model, piece of source code or test case, but a knowledge item. However, after a long time, reuse promises [Jacobson et al., 1997] are still far from reaching the major objective of optimizing the system development lifecycle efforts.

In this context, last times have seen the emergence of Model-based Systems Engineering (MBSE) [INCOSE, 2004] as a complete methodology to address the challenge of unifying the techniques, methods and tools. This means a “formalized application of modelling” to support the left-hand side in the *Vee* lifecycle model implying that any process, task or activity will generate different system artefacts but all of them represented as a model. The MBSE approach is considered a cornerstone for the improvement of the current practice in the Systems Engineering discipline since it is expected to cover multiple 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. In the case of system artefact reuse, both disciplines are currently under study according to the works in [Shani and Broodney, 2015] [Smith, 2014] in which component models are applied to enable reuse. However, the MBSE approach considers that everything can be a model and this assumption is not always true in the development of a complex system. Requirements specifications, test cases or simulation data are just some examples of system artefacts which natural representation (as communication mechanism) is not a model.

Furthermore, abstraction and selection processes are not yet fully developed in a MBSE environment. Currently, interoperability initiatives (such as ISO 10303-STEP or OASIS OSLC-Open Services for Lifecycle Collaboration) are trying to boost reuse through data exchange. However, the first step to be able to exchange and reuse data lies on the provision of a proper environment for system artefacts selection. Existing platforms for the management of system engineering processes such as the Jazz Platform by IBM or Papyrus (Eclipse CDO), offer a kind of central repository in which engineers can upload their systems artefacts and perform tasks such as searching or traceability recovery. These centralized repositories represent artefacts as a set of metadata that are linked to a system artefact (content). Although in some cases, the use of metadata can be useful to look up artefacts by filtering certain properties, it seems too simple to really enable the proper reuse of the knowledge embedded in system artefacts.

As a motivating example, in traditional information retrieval systems (text-based) if someone is looking for documents (text), she will express queries as text (or keywords) and the search engine will match documents according to the input query. In all of them, the representation of information, queries and results are working under the same primitive: text. The same kind of behaviour can be found in the Google Image search service where it is possible to look up images by entering an image. Moreover, and considering the plethora of tools, system artefacts and formats, a retrieval system for a MBSE environment shall be able to represent, store and retrieve any kind of artefact by using as input query any kind of system artefact: a requirement, an architectural, a physical model or event just a text. Thus, information retrieval techniques will equip engineers with a method to discover existing system artefacts based on contents not just metadata.

In this frame, the application of knowledge management techniques has gained momentum to elevate the meaning of the implicit knowledge coded into system artefacts and allow engineers to reuse existing data and knowledge. Software-based artefacts are a new kind of intellectual asset that can be used to reduce costs and save time to market generating competitive advantage in the construction and operation of complex systems. That is why, knowledge management techniques [Nonaka and Takeuchi, 1995] are being applied to capture, structure, store and disseminate system artefacts and support the aforementioned reuse principles of selection and integration. However, one of the cornerstones in knowledge management lies in the selection of an adequate knowledge representation paradigm. After a long time [Hull and King, 1987], this problem still persists since a suitable representation format (and syntax) can already be reached in several ways [Davis et al., 1993]. Any bit of information must be structured and stored for supporting other application services such as business analytics or knowledge discovery. This situation also creates an impedance mismatch between the system and the outside world.

Therefore, one of the current trends to boost systems engineering processes lies in improving interoperability and collaboration through the exchanging of system artefacts under common data models, formats and protocols. In this context, OSLC is creating a collaborative engineering ecosystem through the definition of data shapes that serve us as a contract to get access to information resources applying the Linked Data principles. The Representational State Transfer (REST) software architecture style is used to manage information resources that are publicly represented and exchanged in RDF. However, RDF has been also demonstrated [Powers, 2003] to contain some restrictions to represent certain knowledge features such as N-ary relationships [Noy and Rector, 2006], practical issues dealing with reification [Nguyen et al., 2014] and blank nodes [Mallea et al., 2011]. Moreover, some common services such as indexing or retrieval of any kind of information resource are restricted to the internal storage and the query capabilities offered by each tool (usually a SPARQL interface).

That is why, in this paper the authors present an industry-oriented approach based on existing standards, OSLC and RDF, to support the principles of abstraction and selection of system artefacts. In particular, an OSLC Resource Shape for any knowledge item is refined [Alvarez-Rodríguez et al., 2015] and implemented. Afterwards, an experiment is conducted to integrate, exchange and retrieve different tools and types of system artefacts and to share them under the designed data shape. Finally, some discussion and open issues are outlined with the aim of evaluating the capabilities of this approach to enable software practitioners to develop a software reuse strategy.

2 Related work

In the early days of the Semantic Web, formal ontologies [Benjamins et al., 1998] designed in RDFS (Resource Description Framework Schema) or OWL (Ontology Web Language) were the key technologies to model and share knowledge. From upper ontologies to specific vocabularies, the process to share knowledge consisted in designing a formal ontology for a specific domain and populate data (instances) for that domain. Although the reuse of existing ontologies was expected, the reality demonstrated that every party willing to share knowledge and data would create its own ontologies. Thus, the main idea behind web ontologies was partially broken since just a few concepts were really reused.

Once the Linked Data initiative emerged to unleash the power of existing databases, a huge part of the Semantic Web community realized that a formal ontology was not completely necessary to exchange data. Taking into account that ontologies were still present, these efforts were based on validating data consistency [Baclawski et al., 2002] through the execution of procedures such as: 1) reasoning processes and 2) rules (e.g. SPARQL [Hogan et al., 2012]). Depending on the size and complexity of the ontologies, these procedures are not recommended because of performance issues. As a new evolution, the community realized that ontology-based reasoning was not the most appropriate method for data querying and validation when exchanging RDF resources. Thus, it is possible to find works that focused on exploiting Linked Data from an information technology perspective [Colomo-Palacios et al., 2012] in different domains such as the financial domain [Sánchez-Cervantes et al., 2018] or in the field of sensor data management [Sánchez-Cervantes et al., 2016].

That is why, the RDF community has seen an emerging interest to manage and validate RDF datasets according to different shapes and schemes. New specifications and methods for data validation are being designed to turn reasoning-based validation into a kind of grammar-based validation. These methods take inspiration from existing approaches in other contexts such as DTD (Document Type Definition), XML-Schema or Relax NG (REgular LAnguage for XML Next Generation) for XML, or DDL (Data Definition Language) for SQL (Structured Query Language).

The W3C has recently launched the W3C Recommendation “Shapes Constraint Language (SHACL)“ to support the notion RDF Data Shapes. The ShEX (Shape Expressions) language [Boneva et al., 2014] is an alternative option to SHACL addressing the same objectives. Both are formal languages for expressing constraints on RDF graphs including cardinality constraints as well as logical connectives for disjunction and polymorphism. As other examples of data exchange and validation, OSLC Resource Shapes [Ryman et al., 2013], Dublin Core Description Set Profiles [Coyle and Baker, 2013], and RDF Unit [Kontokostas et al., 2014] are also constraint languages for domain specific RDF resources.

This focus on RDF data validation for easing data exchange and avoiding complex processes such as semantic reasoning also represents an opportunity to bring the principles of Linked Data to the Systems Engineering discipline. In this context, the OSLC initiative [Ryman et al., 2013] is a joint effort between academia and industry to boost data sharing and interoperability among applications by applying the Linked Data principles. Led by the OASIS OSLC working group, OSLC is based on a set of specifications that take advantage of web-based standards to share system artefact data under a common data model (RDF) and protocol (HTTP). Every OSLC specification defines a *shape* for a type of resource. For instance, requirements, changes, test cases or estimation and measurement metrics, to name a few, have already a defined shape (also called OSLC Resource Shape).

In the knowledge management area, the *Assets Management* and the *Tracked Resource Set* are the most convenient specifications for managing artefacts. However, there are many artefacts generated during the development lifecycle which may not fit to existing shapes or standard vocabularies. Simulation models, logical models, business rules or physical circuits are examples of potential artefacts whose an OSLC resource shape is not yet defined. Furthermore, some common and useful services such as indexing, naming, retrieval, quality assessment, visualization or traceability must be provided by all tool vendors, creating a tangled environment of query languages, interfaces, formats and protocols. However, some specific works can be also found in this area of semantically representing and retrieving system artefacts such as system models (e.g. Modelica RC circuits [Gallego et al., 2015] or SysML models [Mendieta et al., 2017]).

Like OSLC, Agosense Symphony offers an integration platform for application and product lifecycle management with a huge implantation in the industry due to the possibility of connecting existing tools. WSO2 is another middleware platform for service-oriented computing based on standards for business process modelling and management. However, none of them offer a standard input/output interfaces based on lightweight data models and software architectures such as RDF and REST. Other industry platforms such as PTC Integrity, Siemens Team Center, IBM Jazz Platform or HP PLM are now offering OSLC interfaces for different types of artefacts.

As it has been introduced, software and system artefact reuse [Mili et al., 1995] [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 [Mili et al., 1998], reuse in software repositories [Guo and others, 2000], reuse of components in the industry [Land et al., 2009], reuse success 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. Requirements specifications and knowledge management techniques are presented in [Bolanle, 2014] to address the challenge of software reuse using formal ontologies and reasoning methods. Other very relevant works have been focused on applying control engineering techniques [Mili, 2002]. Although some of good experiences have been reported [Tracz, 1995], success and failure facts outlined in [Morisio et al., 2002] and [Desouza et al., 2006] are still open.

In the specific case of software and systems engineering and reuse, the application of semantics-based technologies has also been focused in the creation of OWL ontologies [Castañeda et al., 2010] to support requirements elicitation, and to model development processes [Kossmann et al., 2008] or information systems [Beydoun et al., 2014] or Model Driven Architecture [Gaševic et al., 2006], to name just a few. These works only leverage ontologies to formally design a meta-model.

In conclusion, software and system artefact reuse is an active research area that evolves according to the current trends in development lifecycles. It may have the potential of leveraging new technologies such as the web environment, service-oriented computing, semantics and Linked Data. However, data exchange [García-Rodríguez et al., 2012] does not necessarily imply reuse. From service providers to data items, a knowledge strategy is required to really represent, store and search system artefacts metadata and contents. In this light, the OSLC initiative is following this approach, having impact on the main players of software and systems industry. Nevertheless, it only covers a restricted set of artefacts and some cross-cutting and basic services for reuse, such as selection (discovery), must be provided by all third-parties. Lastly, a system and software repository for systematic reuse shall fulfil the following three requirements:

1) A language for representing any artefact’s metadata and contents;

2) A system for indexing and retrieval; and

3) A standard input/output interface (data shape+REST+RDF) to share and exchange artefact metadata and contents.

3 Definition of a Linked Data layer for system artefact exchange

As it has been introduced, the OSLC initiative is making a strong commitment to apply the principles of Linked Data, RDF and REST to boost interoperability and tool integration. Some specifications or, more precisely, data shapes, have already been defined to model metadata and contents of requirements, assets, test cases, changes and estimation and measurement metrics. However, there are still some artefacts for which there is no shape. Moreover, some cross-cutting services are delegated in third-party tools, preventing the implementation of one of the cornerstones for system artefact reuse: selection.

In this section, a data shape, coined as OSLC KM (Knowledge Management) [Alvarez-Rodríguez et al., 2015], for any system artefact is introduced. This data shape gives a response to the basic requirements that have been previously identified for a modern system artefact knowledge repository, and fits to common processes in a knowledge management strategy, see Table 1:

Process	Support
Capture/Acquire	Access OSLC repositories in the context of Systems Engineering for all existing specifications and other RDF-based services or SPARQL endpoints.
Organize/Store	RDF as a public exchange data model and syntax. The data shape is also a kind of general-purpose representation model to build a system artefact repository.
Access/Search/Disseminate	RDF query language (e.g. SPARQL), natural language [Paredes-Valverde et al., 2016] or a native query language (if any). A set of entities and relationships creating an underlying graph.
Use/Discover/Trace/Exploit	Entity reconciliation based on graph comparison.
Visualization	A generic graph-based visualization framework that can show not only entities and relationships, but also interpret them as type of diagram. E.g. Class diagram.
Exploit	Index, search, trace or assess quality based on the internal representation model.
Share/Learn	An OSLC interface on top of the system artefact repository that offers both data and services.
Create	Third-party tool that exports artefacts using an OSLC-based interface.

Table 1: Mapping of the OSLC KM approach to the knowledge management processes presented in Figure 1.

3.1 A data shape for representing system artefacts

The previous section has outlined the different approaches for data validation and definition of data shapes. In this case and taking into account the guidelines and definitions of the OSLC Core specification, the data shape for a system artefact shall conform the next basic OSLC definitions [Ryman et al., 2013]:

1. "*An OSLC Domain is one ALM (Application Lifecycle Management) or PLM (Product Lifecycle Management) topic area*". Each domain defines a specification. In this case, a new domain is being defined: Knowledge Management (KM).
2. "*An OSLC Specification is comprised of a fixed set of OSLC Defined Resources*". The following UML class diagram, see Figure 1, represents a simple metamodel that will be used as the underlying shape for representing system artefacts. The key concepts here are the classes Artefact and Relationship. An artefact is a container of relationships between terms under a specific semantics and

syntax (e.g. “Car has 4 Wheel”) that can have metadata (authoring, versioning, visualization features and, in general, provenance information) and data (e.g. attribute-expressions of a system artefact, such as “Temperature=50”). This shape is inspired by previous works on representing engineering knowledge.

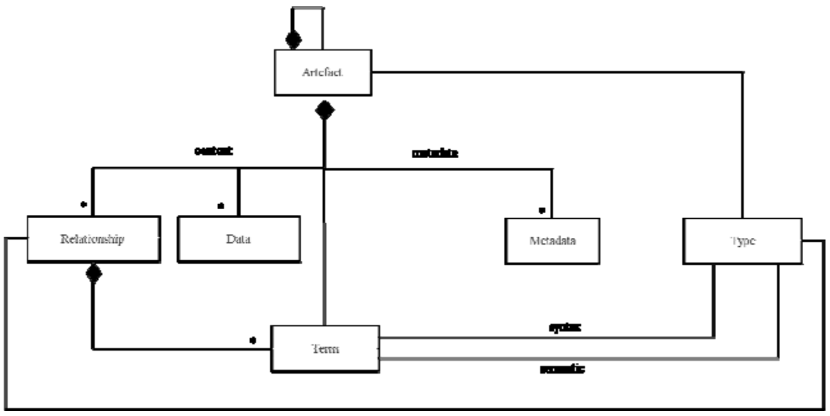

Figure 1: UML Class Diagram of the OSLC Resource Shape for representing system artefacts based on **[Llorens et al., 2004]**.

3. *“An OSLC Defined Resource is an entity that is translated into an RDF class with a type”*. Every resource consists of a set of defined properties whose values may be set when the resource is created or updated. In this case and following the previous design, a shape for every class has been defined. Table 2 presents a brief description of the resource shape and provides a link to the official definition (*prefix:name*, e.g. *Ios_km*[1]*:Artefact*) and a brief description of the resource.

Taking into account that the Linked Data Initiative has also seen the creation of methodologies, guidelines or recipes [Hyland and Terrazas, 2011] to publish RDF-encoded data, we have paid special attention to follow a similar approach by reusing existing RDF-based vocabularies. More specifically, the following rules have been applied to create the OSLC resource shapes:

4. If there is an RDF-based vocabulary that is already a *W3C Recommendation* or it is being promoted by other standards body, it must be used as it is, by creating an OSCL Resource Shape.

5. If there is an RDF-based vocabulary but it is just a *de-facto* standard, it should be used as it is, by including minor changes in the creation of an OSCL Resource Shape.

[1] The prefix *Ios_km* refers to the URI: https://www.eca-ios.orgmediawiki/index.php/ that was part of the “Interoperability Specification” developed within the CRYSTAL project. See web link: http://www.crystal-artemis.eu/fileadmin/user_upload/Deliverables/CRYSTAL_D_601_023_v3.0.pdf

6. If there is not an RDF-based vocabulary, try to take advantage (reusing properties and classes) of existing RDF-based vocabularies to create the OSLC Resource Shape.

Class in Figure 1	**OSLC Resource Shape Item**	**Description**
Artefact	`Ios_km:Artefact`	A container of relationships between concepts and metaproperties to semantically describe any piece of information. It is the basis for the creation of an underlying semantic network.
Relationship	`Ios_km:Relationship`	A relationship represents a link between any set of resources. It is possible to add semantics and it can contain any number of elements representing binary, ternary or even n-ary relationships.
Data	`Ios_km:Data`	An attribute-value expression that represents a property of the artefact under description.
MetaData	`Ios_km:MetaData`	A tag-value attribute representing typical metadata properties. Dublin Core is used here to represent such information. Both can be any type of resource or, more specifically, concepts.
Term	`Ios_km:Concept`	This concept follows the semantics and shape of a *skos:Concept* [Baker et al., 2013]. More specifically: "*the notion of a SKOS concept is useful when describing the conceptual or intellectual structure of a knowledge organization system, and when referring to specific ideas or meanings established within a KOS (Knowledge Organization System)*".
Type	`Ios_km:Concept`	Everything has a type and a type is a kind of concept coming from a classification. E.g. The types of UML metamodel, such as Class, Use Case, etc.

Table 2: OSLC Resource Shapes description within the Knowledge Management domain.

In the case of knowledge management, we have selected the Simple Knowledge Organization System (SKOS), a W3C recommendation, to define concepts, since it has been designed for promoting controlled vocabularies, thesauri, taxonomies or even simple ontologies to the Linked Data initiative. That is why, in the proposed data shape, most of the entities can be considered as a *skos:Concept* and we have created the shape of this standard definition within the resource *Ios_km:Concept*.

7. "*An OSLC Defined Property is an entity that is translated into an RDF property*". It may define useful information such as the type of the property, datatypes and values, domain, range, min. and max. cardinality, representation (inline or reference) and readability.

The detailed and preliminary description of all properties for every defined resource can be found in the public deliverable "Interoperability Specification – V3" of the CRYSTAL project and [Alvarez-Rodríguez et al., 2015].

8. An OSLC Service Provider is a tool that offers data implementing an OSLC specification in a REST-fashion.

Figure *2* shows a functional architecture for an OSLC Knowledge Management provider. It shall be able to process any kind of OSLC-based resource or even any piece of RDF by applying the mappings described in [Alvarez-Rodríguez et al., 2015]. Once the data is in the OSLC KM processor, a reasoning process can be launched to infer new RDF triples (if required). Afterwards, data is validated and indexed into the system and software knowledge repository. On top of this repository, services such as semantic search, naming, traceability, quality checking or visualization may be provided, generating new OSLC KM Resources.

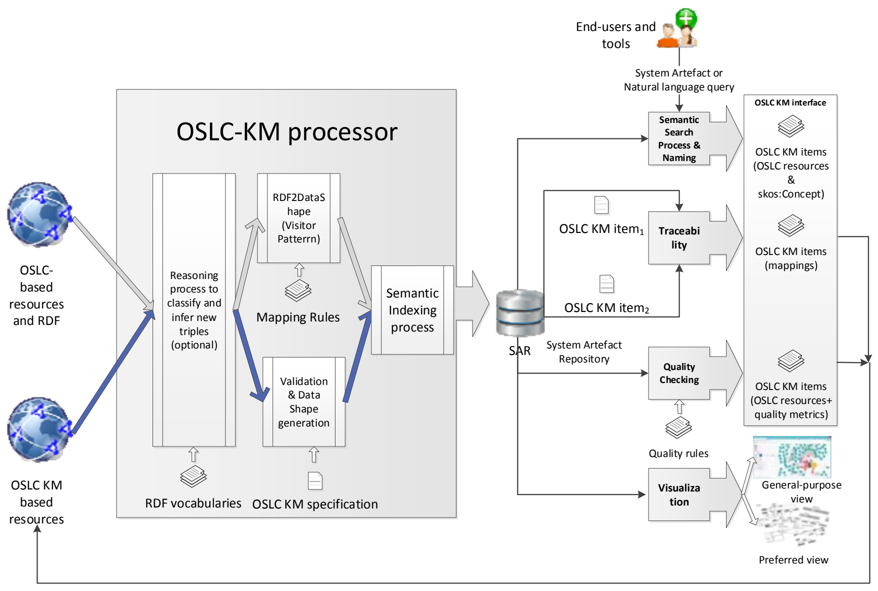
End-users and tools
System Artefact or Natural language query
OSLC-KM processor
OSLC-based resources and RDF
OSLC KM based resources
Reasoning process to classify and infer new triples (optional)
RDF2DataShape (Visitor Patterrn)
Mapping Rules
Validation & Data Shape generation
Semantic Indexing process
RDF vocabularies
OSLC KM specification
SAR
System Artefact Repository
Semantic Search Process & Naming
OSLC KM item1
OSLC KM item2
Traceability
Quality Checking
Quality rules
Visualization
OSLC KM interface
OSLC KM items (OSLC resources & skos:Concept)
OSLC KM items (mappings)
OSLC KM items (OSLC resources+ quality metrics)
General-purpose view
Preferred view

Figure 2: Functional Architecture and core services for knowledge management based on the OSLC KM specification.

3.2 Technological environment

Figure 3 depicts the elements of the functional architecture. Here, it is important to remark that a service-oriented architecture is proposed to implement the approach. More specifically, the next building blocks and technologies have been used to develop this architecture:

- Tool_k. It is the target tool from which artefacts are expected to be exposed following the OSLC Resource Shape defined for Knowledge Management, see Figure 1.
- OSLC KM adapter. It is a wrapper on top of a target tool, $tool_k$, that must implement the transformation rules from the internal representation format to the resource shape in Figure 1. Currently, there are some available implementations based on .Net, Java and XSLT.
- OSLC KM Provider. It is an OSLC service provider that offers a Linked Data API (Application Programming Interface) to access the artefacts available in $tool_k$. Currently, there are two implementations for .Net and Java.
- OSLC KM Client & Provider. It is an OSLC client and Provider for OSLC KM resources. There are again two available implementations in .Net and Java.
- CAKE (*Computer-Aided Knowledge Environment).* It is an API on top of the Knowledge Manager (KM) tool and a repository that offers natural language processing techniques and ontology management capabilities to promote any kind of resource to a semantic-based representation creating an underlying knowledge graph. The CAKE v18 has been used to implement this functional block.
- KM. It is the acronym of the Knowledge Manager[2] v18, a commercial tool developed by The Reuse Company, that offers capabilities to design ontologies and a semantic-based retrieval engine based on graph-matching techniques.
- Common services. Once any piece of data and information is stored in the repository as a graph, it is possible to reuse some of the operations available in the KM tool such as naming, traceability recovery, quality checking or semantic retrieval.

One relevant implication of this architecture is that the reuse of a new type of system artefact only requires the implementation of an OSLC KM adapter.

[2] https://www.reusecompany.com/knowledge-manager

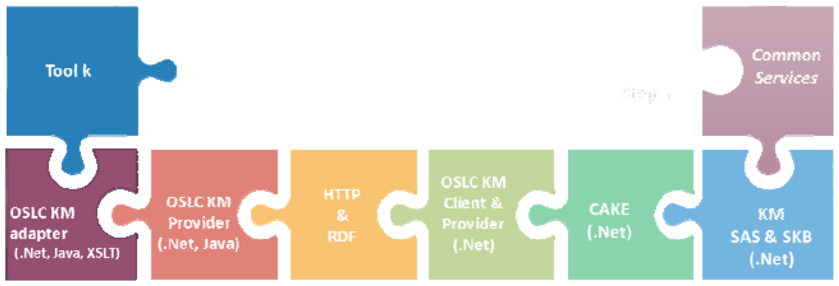
Tool k
Common Services
OSLC KM adapter (.Net, Java, XSLT)
OSLC KM Provider (.Net, Java)
HTTP & RDF
OSLC KM Client & Provider (.Net)
CAKE (.Net)
KM SAS & SKB (.Net)

Figure 3: Building blocks of the functional architecture and technology.

4 Experiment: Exchange and selection of logical and physical models

4.1 Motivation

An organization [Berglund, 2013] developing a cyber-physical system, a rugged computer, is looking for a solution to integrate all tools involved in the development lifecycle. Instead of using a complete ALM or PLM suite, they follow a decentralized and federated approach where different tool providers can be found. They use a requirements management tool (RMT) for gathering and storing stakeholder and system requirements. They also have tools for designing logical models (e.g. Papyrus and IBM Rhapsody) and physical models (Simulink). The organization is looking for the best way to integrate and reuse all the system artefacts that are being continuously generated. Sometimes, requirements contain entities that do not appear in the models, preventing the possibility of recovering traceability links. Thus, the cost of reusing any existing artefact is becoming higher, since it is not possible to completely trace a component from its inception to the final release. In conclusion, this organization is facing some issues:

1. Lack of a product breakdown structure (or a metamodel) to drive the development lifecycle.
2. Name mismatches.
3. Point-to-point and ad-hoc integrations between a client and a tool provider.
4. A plethora of heterogeneous protocols and data models (most of them non-standard ones).
5. Impossibility of reusing artefacts since system artefacts cannot be easily discovered.
6. Lack of a system artefact repository to store and search for system artefacts (metadata and contents).
7. Standalone applications not ready for a collaborative web environment.
8. Vendor lock-in.

Due to all these reasons, they are interested in a holistic and standard approach that can tackle these issues, easing the development lifecycle and boosting the reuse of existing and future artefacts. Since natural language is the most common mean

[Valencia-García and García-Sánchez, 2013] of expression, even in the Linked Data world [Paredes-Valverde et al., 2016], string queries will be used to retrieve artefacts.

4.2 Research design

The validation of the presented approach is conducted using the schema proposed by [Juristo and Moreno, 2001], where experimentation is divided into four main activities: definition of the objectives of the experimentation, design of the experiment, execution of the experiment, and analysis of the results. In this case, two different experiments have been conducted to test the retrieving capabilities of:

1. Exchange and selection of logical models in SysML. Papyrus and IBM Rhapsody have been selected as target tools that can manage logical models and are expected to provide reuse capabilities.
2. Exchange and selection of physical models. In this case, Simulink has been selected as a tool to create and manage physical system models.

In both cases, the reuse capabilities of these tools are compared to the proposed approach.

4.2.1 Main research objective

The following objective was defined for the validation: "*To study the effectiveness of the proposed solution in finding reusable SysML and Simulink models (i.e. suitable for reuse) from a model repository*".

4.2.2 Detailed Design

To evaluate the reuse capabilities of the selected tools in comparison to the presented approach, the following metrics [Croft et al., 2010] have been selected:

- Precision: fraction of retrieved models that are relevant to the query.

$$(P)recision = \frac{|\{relevant\ models\} \cap \{retrieved\ models\}|}{|\{retrieved\ models\}|}$$

- Recall: fraction of relevant models that are retrieved.

$$(R)ecall = \frac{|\{relevant\ models\} \cap \{retrieved\ models\}|}{|\{relevant\ models\}|}$$

- F1: a combination of precision and recall that measures the accuracy of the test.

$$F1 = 2 * \frac{P * R}{P + R}$$

Furthermore, some levels of „*goodness*" for them are defined based on [Hayes et al., 2005]:

- 1) Precision: above 20% it is acceptable, good above 30%, and excellent above 50%.

- 2) Recall: above 60% it is acceptable, good above 70%, and excellent above 80%.

In the case of the first experiment, a set of 44 SysML models based on existing requirements diagrams from two sources [Friedenthal et al., 2014] [OMG, 2016] have been loaded in Papyrus and IBM Rhapsody. Morevover, a set of 25 queries, see Table 3, has been designed to compare and validate the retrieving capabilities of the proposed approach. These queries were designed by experts considering functional aspects common to several of the models, the components of the models, and the terminology of the models. After creating the queries, the relevant items to be retrieved were defined allowing us to calculate the performance metrics.

Id	Query string
Q1	System availability
Q2	Maximum rate of failure
Q3	Manage Traffic flow
Q4	System for purify water
Q5	System using remote control component
Q6	System use cameras
Q7	System with an statistical data component
Q8	System Performance Requirements
Q9	Requirements of System Usability
Q10	System with Simulation Component
Q11	Group Creation
Q12	System Restrictions Requirements
Q13	System that use Sensors
Q14	Gather and Interpret Information Module
Q15	Adaptive Control
Q16	Consistency in transaction
Q17	Manual Control
Q18	intruders detection
Q19	Time Validation
Q20	computer response time
Q21	System validation cards
Q22	tasks and scenarios
Q23	traffic management based in the region
Q24	semaphores automatic operation
Q25	Control standard

Table 3: Queries for retrieving logical models.

In the second case, a set of Simulink models available as libraries have been selected. 20 queries have been again designed to compare both approaches.

Id	**Query string**
Q1	A flow between a constant, product, block sum and a outport block.
Q2	A flow between an inport, product, an a block sum.
Q3	A flow between an inport, block sum and integrator.
Q4	A flow between a subsystem and outport block.
Q5	A flow between a subsystem and to Workspace block.
Q6	A flow between a Transport Delay and Subsystem block.
Q7	A flow between a Integrator block, Transport Delay and Subsystem block.
Q8	A flow between a Inport and constant blocks with a product block.
Q9	A flow between a Inport and constant blocks with a product block and the product block with outport block
Q10	A flow between a Integrator and Subsystem, Add block and subsystem and Subsystema with Subsystem
Q11	A flow between a Integrator and Subsystem, Add block and subsystem and Subsystema with Subsystem1 and subsystem2
Q12	A flow between a Integrator and Subsystem, Add block and subsystem and Subsystema with Subsystem1 and subsystem2 with to Workspace block
Q13	Model with no flows only inport block, outport block and product block
Q14	Two submodels of A flow between an inport, product, an a block sum and outport.
Q15	Two submodels of A flow between an inport, product, an a block sum and outport with two constants
Q16	A flow between inport and add block, and two inports nodes without flow
Q17	A flow between add bloc and constant with divide block.
Q18	A flow between divide block tro integrator nodes and tree outports block
Q19	A flow between integrator block and aoutport block and two outports block and one add block with no flows
Q20	A flow between 4 transfer delay with two subsystems.

Table 4: Queries for retrieving physical models.

4.3 Results and analysis

Table 5 and Figure 4 present the results of the first experiment in which it is possible to find the next metrics (only good, excellent and perfect results are commented):

Query	**OSLC KM**			**Papyrus**			**IBM Rhapsody**		
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
Q1	0.45	0.94	0.61	1.00	0.44	0.61	1.00	0.44	0.61
Q2	1.00	0.83	0.91	1.00	0.67	0.80	1.00	0.67	0.80
Q3	0.67	1.00	0.80	0.75	0.38	0.50	0.80	0.50	0.62
Q4	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Q5	1.00	0.75	0.86	0.00	0.00	0.00	0.00	0.00	0.00
Q6	0.36	1.00	0.53	1.00	1.00	1.00	1.00	1.00	1.00
Q7	1.00	1.00	1.00	1.00	0.56	0.71	1.00	0.56	0.71
Q8	0.33	1.00	0.50	0.50	1.00	0.67	0.00	0.00	0.00
Q9	0.38	1.00	0.55	0.75	1.00	0.86	0.75	1.00	0.86
Q10	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Q11	0.50	1.00	0.67	0.33	1.00	0.50	0.33	1.00	0.50
Q12	0.50	1.00	0.67	0.12	1.00	0.22	0.12	1.00	0.22
Q13	0.36	1.00	0.53	0.12	1.00	0.22	0.14	1.00	0.24
Q14	1.00	0.73	0.84	0.73	1.00	0.85	0.71	0.91	0.80
Q15	0.75	1.00	0.86	1.00	0.67	0.80	1.00	0.67	0.80
Q16	1.00	1.00	1.00	1.00	0.75	0.86	1.00	1.00	1.00
Q17	1.00	1.00	1.00	1.00	0.75	0.86	1.00	0.75	0.86
Q18	1.00	1.00	1.00	0.50	1.00	0.67	0.50	1.00	0.67
Q19	0.70	1.00	0.82	0.88	1.00	0.93	0.88	1.00	0.93
Q20	1.00	1.00	1.00	0.18	1.00	0.31	0.24	1.00	0.38
Q21	1.00	1.00	1.00	0.12	1.00	0.22	1.00	1.00	1.00
Q22	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Q23	0.78	1.00	0.88	0.16	1.00	0.27	0.88	1.00	0.93
Q24	1.00	1.00	1.00	0.50	1.00	0.67	0.50	1.00	0.67
Q25	0.43	0.75	0.55	1.00	1.00	1.00	1.00	1.00	1.00
Average	**0.77**	**0.96**	**0.82**	**0.67**	**0.85**	**0.66**	**0.71**	**0.82**	**0.70**

Table 5: Results of retrieving logical models with OSLC KM, Papyrus and IBM Rhapsody.

- The precision and recall are at least good for all the queries, and excellent on average.
- The precision is good for 8 queries (32%), and excellent for 17 (68%) being perfect (1,00) for 13 queries (52%).
- The recall is good for 3 queries (12%) and excellent for 22 (88%) being perfect (1,00) for 20 queries (80%).

- F1 is perfect (1,00) for 10 queries (40%).

On the other hand, Table 6 and *Figure 5* present the results of the second experiment comparing the Simulink capabilities for retrieving models and the presented approach. An analysis of the results shows that:

- The precision and recall are at least good for all the queries, and excellent on average.
- The precision is good for 3 queries (12%), and excellent for 14 (56%) being perfect (1,00) for 6 queries (25%).
- The recall is good for 18 queries (72%) and excellent for 16 (64%) being perfect (1,00) for 11 queries (44%).
- F1 is perfect (1,00) for 3 queries (40%).

Query	**OSLC KM**			**Simulink**		
	P	**R**	**F1**	**P**	**R**	**F1**
Q1	0.44	0.41	0.42	0.64	0.94	0.76
Q2	0.58	0.65	0.61	0.64	0.94	0.76
Q3	0.53	1.00	0.69	1.00	0.90	0.95
Q4	1.00	1.00	1.00	0.00	0.00	0.00
Q5	1.00	1.00	0.00	0.00	0.00	0.00
Q6	1.00	1.00	1.00	0.00	0.00	0.00
Q7	0.10	1.00	0.18	0.00	0.00	0.00
Q8	1.00	0.44	0.61	1.00	0.69	0.81
Q9	0.13	1.00	0.22	0.00	0.00	0.00
Q10	0.67	1.00	0.80	0.24	1.00	0.39
Q11	0.40	1.00	0.57	0.16	1.00	0.28
Q12	0.33	0.75	0.46	0.16	1.00	0.28
Q13	1.00	0.26	0.42	0.00	0.00	0.00
Q14	0.90	0.56	0.69	0.80	0.25	0.38
Q15	0.90	0.56	0.69	0.67	1.00	0.80
Q16	0.17	0.25	0.20	0.32	1.00	0.48
Q17	0.58	0.85	0.69	0.31	0.31	0.31
Q18	0.90	1.00	0.95	0.25	0.56	0.34
Q19	0.90	1.00	0.95	0.28	0.56	0.37
Q20	1.00	1.00	1.00	0.00	0.00	0.00
Average	**0.68**	**0.79**	**0.61**	**0.32**	**0.51**	**0.35**

Table 6: Results of retrieving physical models with OSLC KM and Simulink.

In both experiments, the presented approach of transforming existing system artefacts into an OSLC data shape to be indexed in the system artefact repository is a valuable solution to make logical and physical models reuse effective. The level of precision and recall has been excellent for most of the queries. Results also suggest that a user would be able to find almost all the models that could be reused for a given purpose. Here, it is necessary to remark, that is more important to retrieve all potential reusable models than retrieving models that are not reusable. The F1 score is also promising, suggesting that the presented approach can be used to effectively implement a reuse strategy.

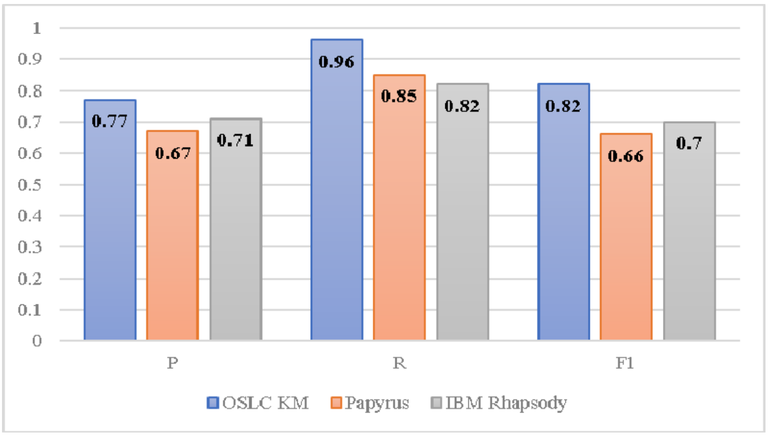

Figure 4: Graphical view of the averaged values of precision, recall and F1 when retrieving logical models.

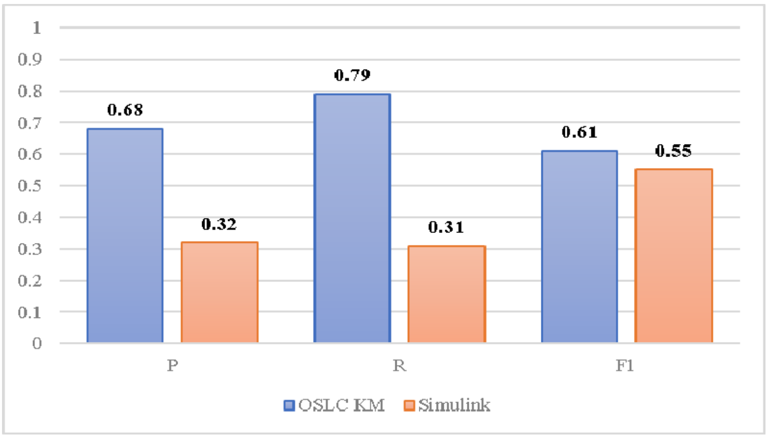

Figure 5: Graphical view of the averaged values of precision, recall and F1 when retrieving physical models.

On the other hand, the development of a Linked Data-based architecture following the guidelines of OSLC has generated several positive effects. The issues that the organization under study was facing are now mitigated, see Table 7. This is mainly due to the use of standards and a common data shape for data management.

4.4 Research limitations

Some key limitations of the presented work must be outlined. The first one depends on the number and type of services and defined resources. This experiment has been conducted in a closed world and, more specifically, two different types of artefacts and three tools have been validated.

A new OSLC domain, OSLC KM, has been defined and implemented for knowledge management. This domain takes inspiration from existing W3C recommendations so that, in a broader and real scope, this specification could change to meet real industry needs.

In the same manner, new tool providers and domains are expected to be integrated in this case study to ensure the representation capabilities of the OSLC KM specification. However, this work presents an industry-oriented case study based on a real environment for software and knowledge reuse.

Building on the previous comment, we cannot either figure out the internal budget, methodologies, tools, domain vocabularies, experience and background of organizations. We merely observe and re-use existing public and on-line knowledge sources to provide a demonstrative experiment of an OSLC-based architecture for system artefact selection.

On the other hand, it seems clear that after a long time, software and system artefact reuse is an active research area in which a good number of challenges and open issues can be found. The emerging application of the OSLC principles to enable interoperability among tools in the development lifecycle is providing a new opportunity for enhancing reuse techniques.

RDF has been demonstrated to be a very good candidate as an input/output data model. One of the main and well-known drawbacks of RDF is that just a few tools natively work in RDF. However, other languages such as RDFS or OWL, designed for representing logical statements, lack of the proper constructors to represent any piece of knowledge based on other paradigms. Although, it is possible to define a RDFS or OWL vocabulary for a domain, the reality is that most of times domain experts do not really need an underlying formal logic but a flexible language for representing concepts and relationships. In this sense, the use of OSLC KM as a language to represent metadata and contents of any artefact has been demonstrated to be flexible and practical (including a native tool support).

In the context of data validation, the related work section has outlined the increasing interest of checking data consistency and integrity of RDF graphs. In a reuse environment, this approach can be applied to matchmaking system artefacts.

From a technical point of view, the deployment of the tools and OSLC adapters has involved a major technical challenge, due to the need of configuration for every tool vendor and adapter. That is why, we consider that new trends in micro services should be applied to decrease the time to deploy and test. Furthermore, an OSLC-based architecture for systematic reuse also requires a new mindset to move existing applications to a web environment in which context issues regarding authorization or authentication are completely different.

Id	Issue Description	Mitigation
1	Lack of a product breakdown structure (PBS) to drive the development lifecycle.	The possibility of defining and sharing a PBS under the OSLC initiative enables practitioners the management of complex processes in the development lifecycle.
2	Name mismatches.	A common domain vocabulary can now be shared and reused in other tools.
3	Point-to-point and ad-hoc integrations between a client and a tool provider.	This is a common side-effect of reusing standards and software knowledge repository. Consumers can ask the repository for an artefact, and once the metadata (e.g. information access) and contents are gathered, they can directly request data to the real provider.
4	A plethora of heterogeneous protocols and data models (most of them non-standard ones).	Although each tool can have its internal data model, there is a unified and shared input/output interface based on RDF.
5	Impossibility of reusing artefacts since traces cannot be recovered.	Having the possibility of representing any artefact under the same data model can help to recover traces.
6	Lack of a software knowledge repository to store and search for artefacts (metadata and contents).	Any piece of software or knowledge can now be represented using concepts and relationships.
7	Poor documentation mechanisms. Lack of graphical view of artefact dependencies.	As a side-effect, the implementation of the OSLC KM specification on top the System Knowledge Manager provides also a mechanism for visualizing artefacts or even generating documentation templates.
8	Standalone applications not ready for a collaborative web environment.	The use of services in a federated architecture enables practitioners the deployment of applications in different locations making the development lifecycle more flexible and scalable. On the contrary, performance, security and privacy issues can emerge avoiding the proper development a collaborative environment for software development.
9	Vendor lock-in.	The use of a standard layer for exchanging data and information avoids a complete vendor lock-in. It is possible to easily change the provider of a service if it also implements that OSLC specification.

Table 7: Issues in the case study and mitigating factors.

On the other hand, a federated and distributed environment of services also implies potential issues regarding security, privacy and performance. That is why this new paradigm must be carefully managed to avoid well-known problems such as information loss, bottlenecks or denegation of service attacks to name just a few.

Finally, the OSLC initiative is continuously releasing and updating specifications, some of them have been already promoted to OASIS standards. This also means that the industry support and commitment behind of OSLC is strongly encouraging interoperability through the creation and use of standards. In this sense, all software libraries and results of the present work are publicly available in a Github repository[3].

5 Conclusions and Future Work

The application of the Linked Data principles through a set of OSLC specifications is gaining momentum in the Software and Systems Engineering disciplines. Interoperability among tools is a key enabler for boosting collaboration in the development of complex cyber physical systems. The concept of continuous engineering is becoming a reality since it is possible to integrate data and services under common protocols and data models. In this context, reuse is a key factor that can ease teams to develop systems faster and safer. However, software is not anymore, a piece of logical instructions but any kind of knowledge and organizational asset. That is why a proper environment for system artefact reuse should provide the appropriate mechanisms for representing, storing, indexing and retrieving any kind of software artefact. However, the implicit design of OSLC (mainly focused in the interoperability between two agents) and the lack of tools to provide a set of core services such as naming, indexing, retrieval, quality assessment or visualization, are preventing the creation of a real collaborative environment to boost the concept of continuous engineering.

That is why we present here a data shape and a retrieving service for implementing the selection process of a reuse strategy overcoming the existing issues regarding knowledge representation and exploitation. This approach is based on a federated and distributed architecture of services that implies a new mindset shifting the traditional paradigm of software product lines to a flexible and interoperable architecture.

Thus, a holistic and standard view of the development lifecycle can be created by easy re-use of existing data, information and knowledge. Future work includes the full implementation of the architecture and the refinement of the OSLC KM specification to fulfil the needs of the development of software-based critical systems. Finally, we also plan to report this work to the OSLC community and to make studies on the return of investment.

Acknowledgements

The research leading to these results has received funding from the AMASS project (H2020-ECSEL grant agreement no 692474; Spain's MINECO ref. PCIN-2015-262)

[3] https://github.com/trc-research/oslc-km

and the CRYSTAL project (ARTEMIS FP7-CRitical sYSTem engineering AcceLeration project no 332830-CRYSTAL and the Spanish Ministry of Industry).

References

[Alvarez-Rodríguez et al., 2015] Alvarez-Rodríguez, J. M., Llorens, J., Alejandres, M., and Fuentes, J. M.: "OSLC-KM: A knowledge management specification for OSLC-based resources" *INCOSE Int. Symp.*, 25 (1)., p. 16–34, 2015.

[Baclawski et al., 2002] Baclawski, K., Kokar, M. M., Waldinger, R. J., and Kogut, P. A.: "Consistency Checking of Semantic Web Ontologies" *Int. Semantic Web Conf.*, p. 454–9, 2002.

[Baker et al., 2013] Baker, T., Bechhofer, S., Isaac, A., Miles, A., Schreiber, G., and Summers, E.: "Key choices in the design of Simple Knowledge Organization System (SKOS)" *Web Semant. Sci. Serv. Agents World Wide Web*, 20 , p. 35–49, 2013.

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

[Benjamins et al., 1998] Benjamins, V. R., Fensel, D., and Gómez-Pérez, A.: "Knowledge Management through Ontologies" *PAKM*, 1998.

[Berglund, 2013] Berglund, M.: "Tool Integration in the Age of Agile" *Stockholm, Sweeden*. 2013.

[Berners-Lee, 2006] Berners-Lee, T.: "Linked Data" 2006.

[Beydoun et al., 2014] Beydoun, G., Low, G., García-Sánchez, F., Valencia-García, R., and Martínez-Béjar, R.: "Identification of ontologies to support information systems development" *Inf. Syst.*, 46 , p. 45–60, 2014.

[Boehm, 1981] Boehm, B. W.: "Software Engineering Economics." 1st ed. *Upper Saddle River, NJ, USA: Prentice Hall PTR*. 1981.

[Bolanle, 2014] Bolanle, O. F.: "Software Reuse Facilitated by the Underlying Requirement Specification Document: A Knowledge-Based Approach" *Am. J. Softw. Eng. Appl.*, 3 (3)., p. 21, 2014.

[Boneva et al., 2014] Boneva, I., Gayo, J. E. L., Hym, S., Prud'hommeau, E. G., Solbrig, H. R., and Staworko, S.: "Validating RDF with Shape Expressions" *CoRR*, abs/1404.1270 , 2014.

[Castañeda et al., 2010] Castañeda, V., Ballejos, L., Caliusco, L., and Galli, R.: "The Use of Ontologies in Requirements Engineering" *GJRE*, 10 (6)., 2010.

[Colomo-Palacios et al., 2012] Colomo-Palacios, R., Sánchez-Cervantes, J. L., Alor-Hernández, G., and Rodríguez-González, A.: "Linked Data: Perspectives for IT Professionals" *Int. J. Hum. Cap. Inf. Technol. Prof.*, 3 (3)., p. 1–12, 2012.

[Coyle and Baker, 2013] Coyle, K., and Baker, T.: "Dublin Core Application Profiles Separating Validation from Semantics" *W3C*. 2013.

[Croft et al., 2010] Croft, W. B., Metzler, D., and Strohman, T.: “Search Engines: Information Retrieval in Practice” *Pearson Education*. 2010.

[Davis et al., 1993] Davis, R., Shrobe, H., and Szolovits, P.: “What is a knowledge representation?” *AI Mag.*, 14 (1)., p. 17, 1993.

[Desouza et al., 2006] Desouza, K. C., Awazu, Y., and Tiwana, A.: “Four Dynamics for Bringing Use Back into Software Reuse” *Commun ACM*, 49 (1)., p. 96–100, 2006.

[Fortune and Valerdi, 2008] Fortune, J., and Valerdi, R.: “Considerations for successful reuse in systems engineering” *Space 2008 Conf.*, 2008.

[Frakes and Terry, 1996] Frakes, W., and Terry, C.: “Software reuse: metrics and models” *ACM Comput. Surv. CSUR*, 28 (2)., p. 415–435, 1996.

[Friedenthal et al., 2014] Friedenthal, S., Moore, A., and Steiner, R.: “A practical guide to SysML: the systems modeling language” *Morgan Kaufmann*. 2014.

[Gallego et al., 2015] Gallego, E., Alvarez Rodríguez, J. M., and Llorens, J.: “Reuse of Physical System Models by means of Semantic Knowledge Representation: A Case Study applied to Modelica” *Proc. 11th Int. Model. Conf. Versailles Fr. Sept. 21-23 2015*, *Linköping University Electronic Press, Linköpings universitet*. p. 747–57, 2015.

[García-Rodríguez et al., 2012] García-Rodríguez, M., Álvarez-Rodríguez, J. M., Muñoz, D. B., Paredes, L. P., Gayo, J. E. L., and Pablos, P. O. de: “Towards a Practical Solution for Data Grounding in a Semantic Web Services Environment” *J UCS*, 18 (11)., p. 1576–1597, 2012.

[Gaševic et al., 2006] Gaševic, D., Devedžic, V., Djuric, D., and SpringerLink (Online service): “Model Driven Architecture and Ontology Development” *Berlin, Heidelberg: Springer-Verlag Berlin Heidelberg*. 2006.

[Guo and others, 2000] Guo, J., and others: “A survey of software reuse repositories” *Eng. Comput.-Based Syst. IEEE Int. Conf. On*, *IEEE Computer Society*. p. 92–92, 2000.

[Hayes et al., 2005] Hayes, J. H., Dekhtyar, A., and Sundaram, S. K.: “Improving after-the-fact tracing and mapping: Supporting software quality predictions” *IEEE Softw.*, 22 , p. 30–7, 2005.

[Hogan et al., 2012] Hogan, A., Umbrich, J., Harth, A., Cyganiak, R., Polleres, A., and Decker, S.: “An empirical survey of Linked Data conformance” *Web Semant. Sci. Serv. Agents World Wide Web*, 14 , p. 14–44, 2012.

[Hull and King, 1987] Hull, R., and King, R.: “Semantic database modeling: Survey, applications, and research issues” *ACM Comput. Surv. CSUR*, 19 (3)., p. 201–260, 1987.

[Hyland and Terrazas, 2011] Hyland, B., and Terrazas, B. V.: “Cookbook for Open Government Linked Data” *W3C*. 2011.

[INCOSE, 2004] INCOSE: “Systems Engineering Vision 2020” *INCOSE*. 2004.

[Jacobson et al., 1997] Jacobson, I., Griss, M., and Jonsson, P.: "Software Reuse: Architecture, Process and Organization for Business Success" *New York, NY, USA: ACM Press/Addison-Wesley Publishing Co.* 1997.

[Juristo and Moreno, 2001] Juristo, N., and Moreno, A. M.: "Basics of Software Engineering Experimentation" Analysis. *Springer Science & Business Media*. 2001.

[Kontokostas et al., 2014] Kontokostas, D., Westphal, P., Auer, S., et al.: "Test-driven evaluation of linked data quality" *23rd Int. World Wide Web Conf. WWW 14 Seoul Repub. Korea April 7-11 2014*, p. 747–758, 2014.

[Kossmann et al., 2008] Kossmann, M., Wong, R., Odeh, M., and Gillies, A.: "Ontology-driven Requirements Engineering: Building the OntoREM Meta Model" *IEEE*. p. 1–6, 2008.

[Krueger, 1992] Krueger, C. W.: "Software reuse" *ACM Comput. Surv. CSUR*, 24 (2)., p. 131–183, 1992.

[Land et al., 2009] Land, R., Sundmark, D., Lüders, F., Krasteva, I., and Causevic, A.: "Reuse with software components-a survey of industrial state of practice" *Form. Found. Reuse Domain Eng.*, *Springer*. p. 150–159, 2009.

[Llorens et al., 2004] Llorens, J., Morato, J., and Genova, G.: "RSHP: an information representation model based on relationships" In: Damiani E, editor. *Soft Comput. Softw. Eng.*, *Berlin, Heidelberg: Springer Berlin Heidelberg*. p. 221–53, 2004.

[Mallea et al., 2011] Mallea, A., Arenas, M., Hogan, A., and Polleres, A.: "On blank nodes" *Semantic Web–ISWC 2011*, *Springer*. p. 421–437, 2011.

[Mendieta et al., 2017] Mendieta, R., Vara, J. L. de la, Llorens, J., and Alvarez-Rodríguez, J. M.: "Towards Effective SysML Model Reuse" *Proc. 5th Int. Conf. Model-Driven Eng. Softw. Dev. - Vol. 1 Model.*, p. 536–41, 2017.

[Mili et al., 1998] Mili, A., Mili, R., and Mittermeir, R. T.: "A survey of software reuse libraries" *Ann. Softw. Eng.*, 5 , p. 349–414, 1998.

[Mili et al., 1995] Mili, H., Mili, F., and Mili, A.: "Reusing software: Issues and research directions" *Softw. Eng. IEEE Trans. On*, 21 (6)., p. 528–562, 1995.

[Mili, 2002] Mili, H.: "Reuse based software engineering: techniques, organization and measurement" *New York: Wiley*. 2002.

[Morisio et al., 2002] Morisio, M., Ezran, M., and Tully, C.: "Success and failure factors in software reuse" *IEEE Trans. Softw. Eng.*, 28 (4)., p. 340–57, 2002.

[Nguyen et al., 2014] Nguyen, V., Bodenreider, O., and Sheth, A.: "Don't like RDF reification?: making statements about statements using singleton property" *ACM Press*. p. 759–70, 2014.

[Nonaka and Takeuchi, 1995] Nonaka, I., and Takeuchi, H.: "The knowledge-creating company: How japanese companies create the dynamics of innovation" *New York: Oxford University Press*. 1995.

[Noy and Rector, 2006] Noy, N., and Rector, A.: “Defining N-ary Relations on the Semantic Web” *W3C Working Group*. 2006.

[OMG, 2016] OMG: “The Official OMG SysML site” 2016.

[Paredes-Valverde et al., 2016] Paredes-Valverde, M. A., Valencia-García, R., Rodríguez-García, M. Á., Colomo-Palacios, R., and Alor-Hernández, G.: “A semantic-based approach for querying linked data using natural language” *J. Inf. Sci.*, 42 (6)., p. 851–62, 2016.

[Powers, 2003] Powers, S.: “Practical RDF” *Beijing ; Sebastopol: O’Reilly*. 2003.

[Ryman et al., 2013] Ryman, A. G., Hors, A. L., and Speicher, S.: “OSLC Resource Shape: A language for defining constraints on Linked Data” *LDOW*, 2013.

[Sánchez-Cervantes et al., 2016] Sánchez-Cervantes, J. L., Radzimski, M., Rodriguez-Enriquez, C. A., et al.: “SREQP: A Solar Radiation Extraction and Query Platform for the Production and Consumption of Linked Data from Weather Stations Sensors” *J. Sens.*, 2016 , p. 1–18, 2016.

[Sánchez-Cervantes et al., 2018] Sánchez-Cervantes, J. L., Alor-Hernández, G., Salas-Zárate, M. del P., García-Alcaraz, J. L., and Rodríguez-Mazahua, L.: “FINALGRANT: A Financial Linked Data Graph Analysis and Recommendation Tool” In: Valencia-García R, editor. *Explor. Intell. Decis. Support Syst.*, *Cham: Springer International Publishing*. p. 3–26, 2018.

[Shani and Broodney, 2015] Shani, U., and Broodney, H.: “Reuse in model-based systems engineering” *IEEE*. p. 77–83, 2015.

[Smith, 2014] Smith, W. B.: “3.3.2 Re-Use Libraries Leveraging Model-Based Systems Engineering to greatly increase engineering productivity” *INCOSE Int. Symp.*, 24 (1)., p. 298–312, 2014.

[Smolárová and Návrat, 1997] Smolárová, M., and Návrat, P.: “Software reuse: Principles, patterns, prospects” *CIT J. Comput. Inf. Technol.*, 5 (1)., p. 33–49, 1997.

[Thüm et al., 2014] Thüm, T., Apel, S., Kästner, C., Schaefer, I., and Saake, G.: “A Classification and Survey of Analysis Strategies for Software Product Lines” *ACM Comput. Surv.*, 47 (1)., p. 1–45, 2014.

[Tracz, 1995] Tracz, W.: “Confessions of a Used Program Salesman: Institutionalizing Software Reuse” *Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.* 1995.

[Valencia-García and García-Sánchez, 2013] Valencia-García, R., and García-Sánchez, F.: “NATURAL LANGUAGE PROCESSING and Human–Computer Interaction” *Comput. Stand. Interfaces*, 35 (5)., p. 415–6, 2013.