International Journal of Software Engineering and Knowledge Engineering
© World Scientific Publishing Company

SEMANTIC RECOVERY OF TRACEABILITY LINKS BETWEEN SYSTEM ARTIFACTS

JOSE MARÍA ALVAREZ-RODRÍGUEZ

Department of Computer Science and Engineering, Carlos III University of Madrid, Avd. Universidad 30,
Leganés, Madrid 28911, Spain
josemaria.alvarez@uc3m.es
https://www.uc3m.es

ROY MENDIETA, VALENTÍN MORENO, MIGUEL SÁNCHEZ-PUEBLA, JUAN LLORENS

Department of Computer Science and Engineering, Carlos III University of Madrid, Avd. Universidad
30,Leganés, Madrid 28911, Spain
roy.mendieta@kr.inf.uc3m.es, valentin.moreno@uc3m.es, masrodri@inf.uc3m.es, llorens@inf.uc3m.es

Received (Day Month Year)
Revised (Day Month Year)
Accepted (Day Month Year)

The present paper introduces a mechanism to recover traceability links between requirements and logical models in the context of critical systems development. Currently, lifecycle processes are covered by a good number of tools that are used to generate different types of artifacts. One of the cornerstone capabilities in the development of critical systems lies in the possibility of automatically recovery traceability links between system artifacts generated in different lifecycle stages. To do so, it is necessary to establish to what extent two or more of these work products are similar, dependent or should be explicitly linked together. However, the different types of artifacts and their internal representation depict a major challenge to unify how system artifacts are represented and, then, linked together. That is why, in this work, a concept-based representation is introduced to provide a semantic and unified description of any system artifact. Furthermore, a traceability function is defined and implemented to exploit this new semantic representation and to support the recovery of traceability links between different types of system artifacts. In order to evaluate the traceability function, a case study in the railway domain is conducted to compare the precision and recall of recovery traceability links between text-based requirements and logical model elements. As the main outcome of this work, the use of a concept-based paradigm to represent system artifacts is demonstrated as a building block to automatically recover traceability links within the development lifecycle of critical systems.

Keywords: Software traceability; Software system artifact representation; Software reuse.

1. Introduction

Critical software systems are those featured by the concept of safety [1] and whose failure could imply loss of life or environmental damage. These systems face real complexity management issues [2] within their development lifecycle. More specifically, in many cases, thousands of software-based work-products are delivered in different stages, following the classical "decomposition-integration" Vee model lifecycle.

In this context, Model-Driven Engineering (MDE) methods [3] through the Model-Driven Architecture (MDA) [4], Model-Driven Development (MDD) [5] or Model Based Systems Engineering (MBSE) [6], Requirements-Driven Engineering (RDE) or Model Driven Requirements Engineering (MDRE) are some of the main approaches designed for easing and automating the development lifecycle of complex systems. All these methods look for elevating the meaning of information resources with the aim of easing the mapping, communication and exchange of data and information between the different development stages. They use as a first-class member a type of system artifact, models, that are the main information exchange unit while other types of system artifacts such as requirements are artificially wrapped within a model but without a real exploitation of the information contained in their content.

Furthermore, a development lifecycle also comprises a plethora of people, tools and engineering methods with different objectives, experience, background and budget implying that a huge amount of time is usually spent trying to coordinate the whole development process. Therefore, the major objective of conceiving the system as a collection of inter-connected modules, entities or system artifacts that are generated in different stages, activities or processes applying different engineering methods is becoming a major challenge [7].

Accordingly, one of the most relevant problems lies in the recovery of traceability [7] links between this vast amount of work-products (system traceability). Traceability allows engineers to track inter/intra dependencies in each subsystem or component. Regardless the type of development methodology [8], system traceability [7] [9] is a key enabler to verify and validate the system. The automatic creation of mappings between different system artifacts can help to detect and anticipate potential risks or to perform change impact analysis processes. Besides, traceability is also required in critical systems for certification purposes [10].

As a naïve example, a requirement specification can contain a set of requirements (hundreds or even thousands); every requirement is linked to a set of models (functional blocks in descriptive models) that can be developed through simulations or pieces of source code (analytical models). Afterwards, a process is carried out to check that every requirement is verified through a set of test cases. Although this is a very simple example of a development process, two major issues can be found:

1) Data and information of every system artifact should be easily shared between all stakeholders and people involved in the development process and,

2) since system artifacts are designed and developed by human-beings, there is an intrinsic necessity of natural language processing to support some tasks such as naming or searching. People use natural language to express their needs, to write requirements, to design models and to communicate ideas, so it is obvious that natural language is the first-class member in the development lifecycle.

In order to address the first issue, initiatives like the ISO STEP 10303 or the Open Services for Lifecycle Collaboration (OSLC) were defined to tackle some of the existing needs regarding interoperability and integration in the Software and Systems Engineering discipline. Both initiatives offer a family of specifications (leaded by industry vendors) to model shared resources between applications reusing web-based standards and delivering robust products through the collaboration of development and operational tools [11]. For instance, the OSLC Requirements Management (RM) specification defines some properties such as: *oslc_rm:elaboratedBy, oslc_rm:specifiedBy, oslc_rm:affectedBy, oslc_rm:trackedBy or oslc_rm:implementedBy* that are expected to be used by development tools to link requirements to other system artifacts such as models or test cases. Although it includes properties to link different system artifacts, there are no specific services, just a specification [12], to implement entity reconciliation processes [13] ("*identify elements that represent the same entity or identify elements that are similar but does not correspond to the same entity*") and, thus, to link together different work products.

Secondly, if natural language is assumed as a first-class member of any methodology or development process, common problems dealing with natural language such as misspelling errors, use of acronyms, ambiguities or inconsistencies will be found. Moreover, if it is assumed that the first system artifact is usually a stakeholder specification that will be exploited generating a system requirements specification creating also a set of functional and non-functional requirements (top-down approach), the need of dealing with natural language is even more relevant.

In a similar way, verification processes usually follow a bottom-up approach checking functional, non-functional, system and stakeholder requirements. Although, some techniques have emerged to support the implementation of this double process of creating and verifying/validating requirements specifications, the use of a traceability matrix [14] [15] is a widely accepted practice to map and trace system artifacts at different levels of abstraction. However, the automatic creation of a traceability matrix is not an easy task. In the specific case of requirements, it also implies the need of processing domain-specific natural language descriptions. This situation leads us to simplify the problem of requirements traceability to a process of mapping two different textual descriptions. Furthermore, and considering that software critical systems are developed in a human-oriented environment, traceability can also be generalized as a process of mapping two descriptions (requirement-requirement, requirement-model, requirement-test, requirement-any system artifact) through techniques such as pattern-matching, search or recommendation. However, to enable the possibility of linking different types of system artifacts, it is also necessary to provide a common representation model.

In this context, the main contribution of this paper lies in the promotion of system artifacts such as requirements, from an informal (text-based) to a formal representation (concept-based) bridging the gap between natural language and a domain-specific vocabulary (concepts) through the use a knowledge-based layer [16] identifying terms and entities [17]. This multi-layered and concept-driven approach is presented to also support the design and creation of pattern-based artifacts (a set of interlinked concepts) and to enable the automatic recovery of traceability links between system artifacts.

As a motivating example, see TABLE 1, two types of requirements and patterns are outlined: 1) a stakeholder pattern p_1 (user need) and 2) a specific system pattern p_2 (system requirement). These patterns build a traceability model that is used to align requirements at a particular level of detail to other requirements at a different level of detail through a natural language pattern-matching process that automatically generates traceability links, see Fig. 5. Once a requirement is linked to a pattern the traceability discovering process becomes straight forward and a traceability matrix between two types or requirements can be easily generated.

TABLE 1 Naïve examples of pattern-based requirements.

Pattern p_1	The < Stakeholder> shall be able to <Deceleration Capability>.
Attributes	<Stakeholder>: driver < Deceleration capability>: brake
Requirement r_1	The driver shall be able to brake.
Pattern : p_2	Whenever the <system | subsystem | component> <trigger>, the <system | subsystem | component> shall <Deceleration Capability> in <quantitative value> <unit of measurement>.
Attributes	<system | subsystem | component>: pedal of the brake, car, etc. <trigger>: be pressed <quantitative value>: a number < unit of measurement >: milliseconds
Requirement r_2	Whenever the pedal of the brake is pressed, the car shall decelerate in 500 milliseconds.

The remainder of this paper is structured as follows. Section 3 defines the role of ontologies to represent concept-based system artifacts and a traceability function is defined as a scenario of an entity reconciliation process. Afterwards, Section 4 presents a case study in the railway domain comparing text-based and concept-based techniques for system traceability including an analysis of robustness. Section 5 discuses the results and limitations of the experimentation. Section 6 reviews the state of the art in the context of systems traceability covering the main methods and techniques in the field of entity reconciliation and matching. Finally, last section draws the main conclusions and future work.

2. Concept-based representations of system artifacts

In this section, a review of the concept of ontology is introduced to situate the notion of a concept-based representation of a system artifact. Afterwards, the application of ontologies is presented as a technique to author system artifacts.

2.1. *Background definitions*

Ontologies are commonly used to model domain knowledge under a concrete syntax and logic formalism. Some of the classical definitions [18] [19] describe a formal ontology as a specification of a conceptualization; that is, as a set of concepts (classes), attributes and relationships aiming to share and reuse knowledge.

In the context of system artifacts management, the use of an ontology can help to restrict the concepts that can be used to describe and represent a system artifact from all, lexical, syntax and semantic/category levels. In this article, an application of the classical concept of ontology is interpreted as a layered knowledge framework, see Fig. **1**.

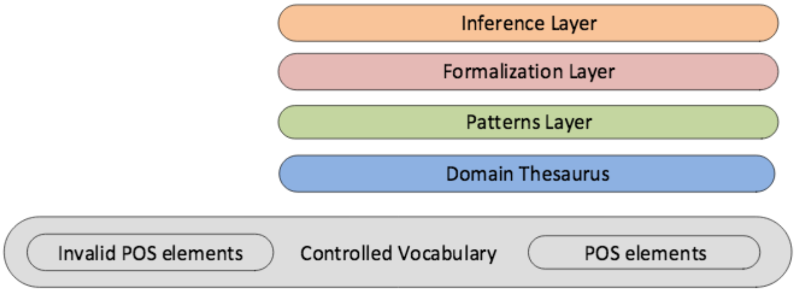


Fig. 1 Layers of an ontology-driven approach to guide requirements writing.

- **Controlled vocabulary layer** contains all the terms with a specific meaning in a domain.
    - **Part-of-Speech (POS) elements layer** includes all those terms that are part of the speech and are used to build a domain-based terminology such as prepositions, conjunctions, articles, etc.
    - **Invalid POS elements** is the set of terms that should be avoided in system artifact descriptions to reach a high-quality representation.
- **Domain thesaurus layer** is comprised of those concepts and terms that are relevant for a domain but including semantic relationships such as hierarchical relationships (e.g. *broader/narrower*) or composition (e.g. *part-of/whole-part*).
- **Pattern layer** defines the grammar, structure, to create concept-based system artifact representations. It makes use of the existing definitions (concepts) by exploiting semantic relationships (e.g. synonymy or *part-of*).
- **Formalization layer** is the layer in charge of managing semantic relationships and exploiting the underlying knowledge [20] [21] that has been formalized through concepts and relationships.
- **Inference layer** represents the rules that can be used to validate or classify the existing knowledge or to infer new knowledge items according to the underlying data model (e.g. a semantic graph) and a certain type of logics. In some context such as expert systems, this layer corresponds to the use of a semantic-based reasoner or a rule-based engine.

The main application of this notion of ontology in the context of Systems Engineering and system artifact management [22] may provide some advantages in different lifecycle stages:

- For system artifact authoring:
    a. Identify the concepts used to create a system artifact, e.g. a requirement text, a class name, etc.
    b. Model and automatic processing of the structure (grammar) of a system artifact to help in the transformation and reuse of system artifacts. For instance, to automatically derive test cases or generate documentation.
    c. Formalize, as a semantic graph, any system artifact content to perform processes for quality checking.
- For continuous quality assessment of individual system artifacts:
    a. Knowledge for calculating correctness metrics (system artifact level).
    b. Knowledge for calculating consistency metrics (system level).

  c. Knowledge for calculating completeness (system artifact and system levels).

- For traceability purposes, recovery of traceability links by exploiting the underlying semantics (concepts and relationships) used to describe system artifacts. In this case, the process is usually based on matching similar underlying graphs (since every piece of knowledge is intrinsically modeled as a semantic graph).

As a main conclusion, the reuse of a knowledge base, such as a domain ontology, created by domain experts under a specific context can boost some processes such as authoring, quality assessment, traceability, etc. since any activity is driven by domain knowledge overcoming the intrinsic issues when dealing with natural language and elevating the meaning of information resources from pure lexical descriptions to concept-based representations.

2.2. *Application of ontologies to create concept-based representations of system artifacts*

Building on the previous section, it may be possible to create a set of patterns or concept-based structures [23] representing the internal content of every type of system artifact such as a requirement or a model. Assuming patterns are built on top of a knowledge base such as a domain ontology, the author of a system artifact may be able to select a type of system artifact and its corresponding pattern avoiding lexical errors or misleading or wrong structures from both perspectives syntax and semantics.

Furthermore, patterns as a kind of a restricted concept-based structure are comprised of different potential concepts to drive the selection of terms based on the exploitation of the lexical and semantic relationships established in the knowledge base. Depending on the type of authoring, two main techniques can be identified: 1) fill-the-gap and 2) free-writing systems.

1. Fill-the gap systems. It is supported by tools that show two different types of items on the screen: 1) labels for the fixed parts of the statement and 2) textboxes for the rest of the elements. Thus, engineers are only able to fill the gap in the proper slots with restricted information according to the need of every item.
2. Free-writing system. In this kind of tools, engineers face a complete blank screen with no pre-written information. Then, the engineer can select the most suitable pattern for a system artifact and a template is prompted with the following information:
    a. All the items (slots) of the selected pattern. This can be seen as a kind of grammar or syntax of the selected pattern.
    b. An example of use based on the selected pattern/grammar.
    c. According to the semantics of the current item, the authoring tool can fetch from the ontology a list of the most suitable concepts and terms. Furthermore, feedback is always given to the engineer since a continuous analysis process is being made to know if the current text is matching to the selected pattern.

The main consequence of the type of system artifact authoring technique is that, although fill-the-gap systems constraint part of the domain of discourse avoiding some of the issues that arise when dealing with natural language, the reality is that those fixed slots are far from the typical human-authoring technique and it prevents a proper interactivity between the end-user and an application.

On the other hand, those systems based on free-writing but ontology-driven keep the same advantages of fill-the-gap systems but including a more user-friendly and interactive way of writing. Moreover, other advantages can be outlined: 1) continuous user feedback through warning messages that shows whether the current system artifact is accomplishing with the structure of the selected pattern; 2) automatic suggestion of concepts and terms through the exploitation of semantic relationships in the knowledge-base and 3) continuous quality assessment for every system artifact and specification.

A pattern, a concept-based representation, encapsulates then the rules for writing and validating both a natural language statements and any other kind of structured data.

A set of patterns for a type of system artifact provides a way to run a system traceability process and to analyze the quality of the system under development by comparing the internal structure of shared concepts and relationships.

The main benefits of using a concept-based representation like a pattern have their origin in [23] and can be enumerated as follows: 1*) an aid in articulation; 2) uniformity of grammar; 3) uniformity of vocabulary; 4) ensuring essential characteristics; 5) easier identification of repeated and conflicting requirements; 6) one-stop control over expression and 7) protection of classified information*.

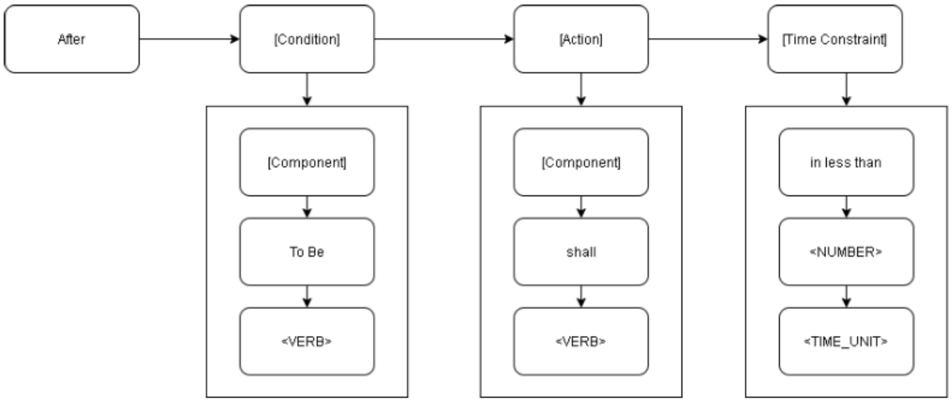
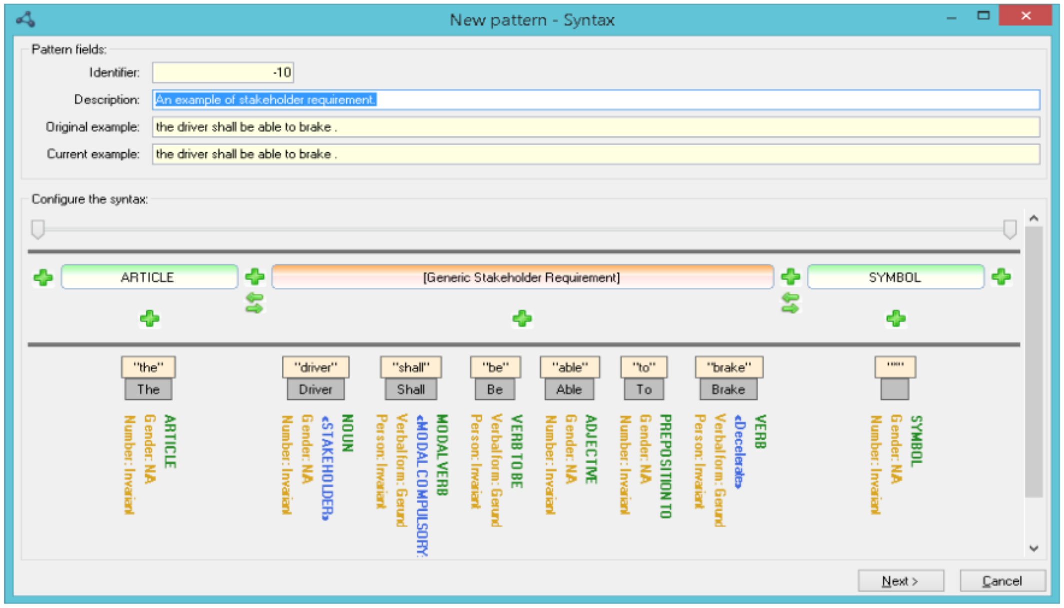


Fig. 2 Example of a pattern and sub patterns restrictions for writing a requirement.

Therefore, a pattern is simply a restricted structure: a sequential list of restrictions (a.k.a. slots) at the syntactic and/or semantic level that will be used to match any input against any other type of system artifact. Restrictions can be of several types, and have different properties (optional, compulsory, OR restrictions, etc.):

- Term restriction. A term must be found at a specific position of the system artifact. Those terms are coming from the controlled vocabulary layer, either compound words or simple words.

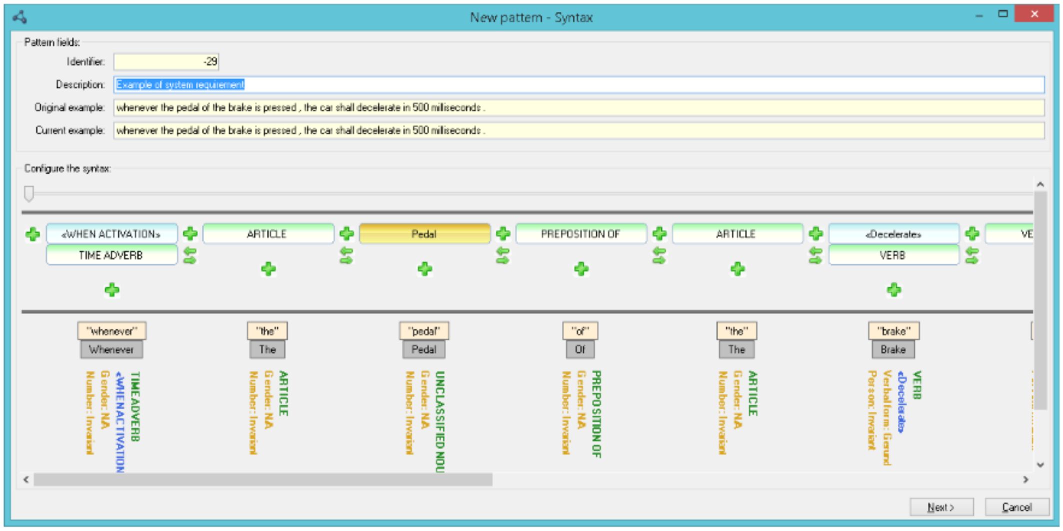
- Syntax restriction. A specific syntax category must be found at a specific position of the current system artifact. The syntax tags come from the classification layer of the ontology, namely VERBS, NOUNS, etc.
- Semantic restriction. A specific semantic category must be found at a specific position of the current system artifact. The semantic types come also from the classification layer of the ontology.
- Syntax + Semantic restriction. A combination of both classification dimensions must be found at a specific position of the current system artifact.
- Sub-pattern restriction. A combination of terms, matching a sub-pattern must be found at a specific position of the current system artifact. Most of times, a pattern is made up of a combination of different sub-patterns. This approach allows us to represent whatever sub-structure providing support for any combination of sub-patterns at any level. Therefore, a sub-pattern may also include sub-sub-pattern slots. This way of organizing structure allows us defining system artifact patterns at a high level of abstraction. Fig. **2** depicts a pattern comprised of three different sub-patterns (the sub-pattern slots in this example are not recursively represented as sub-sub-patterns, but it could be so if needed).



Identifier: -10
Description: An example of stakeholder requirement
Original example: the driver shall be able to brake .
Current example: the driver shall be able to brake .

Fig. 3 Example of a pattern-based stakeholder requirement.

Building on the previous definitions and applications on the use of patterns, an example of creating system artifacts using patterns, requirements from TABLE 1 , is presented in Fig. 3 and Fig. 4. More specifically, Fig. 4 depicts a requirement containing syntax restrictions and a semantic restriction in the concept "Pedal". These pattern-based requirements have been designed and implemented using the KnowledgeMANAGER tool, "*an ontology management system allowing to define and manage the main semantics of system engineering artifacts…*".



Identifier: -29
Description: Example of system requirement
Original example: whenever the pedal of the brake is pressed , the car shall decelerate in 500 miliseconds .
Current example: whenever the pedal of the brake is pressed , the car shall decelerate in 500 miliseconds .

Fig. 4 Partial view of an example of a pattern-based system requirement.

3. A semantic traceability model for system artifacts

Since ontologies can help engineers to drive the authoring and management of system artifacts and to intrinsically support system traceability, natural language descriptions and any other type of information such as a model, must be elevated to a concept-based representation. In this light, a system traceability function can be understood as an entity reconciliation process. It then be defined as a function T that for a given resource r_k^i , a target set of resources R_j and a context C (containing information about natural language processing such as stop words, acronyms, etc.) will generate a set of mappings $\{(r_k^i, r_k^j, c)\}$ where the input resource and other resource r_k^j will be linked together under a certain value of confidence c as the next equation shows:

$$T: r_k^i \times R_j \times C \rightarrow \{(r_k^i, r_k^j, c)\} \; / \; r_k^i \in R_i \wedge r_k^j \in R_j \wedge c \in \mathbb{R} \qquad (1)$$

This definition can be generalized and applied to an entity reconciliation process between two different sets of resources, $R_i \; and \; R_j$ as the next equation also shows:

$$T: R_i \times R_j \times C \rightarrow \{(r_k^i, r_k^j, c)\} \; / \; r_k^i \in R_i \wedge r_k^j \in R_j \wedge c \in \mathbb{R} \qquad (2)$$

Given the two previous definitions, a system traceability process is an extension of this mapping process in which two system artifacts, $R_i \; and \; R_j$, are used as source and target sets of resources. P is the set of patterns that have been designed to semantically represent such system artifacts using a set of domain vocabularies O, commonly one ontology will be enough to represent domain knowledge.

The output of this function will be again a set of mappings $\{(r_k^i, r_k^j, p_i, p_j, c)\}$ where r_k^i represents an element of in the source system artifact represented through the pattern p_i, r_k^j represents an element in the target system artifact represented through the pattern p_i and c is a value of confidence.

$$T_{requirements}: R_i \times R_j \times P \times O \times C \rightarrow \{(r_k^i, r_k^j, p_i, p_j, c)\} \quad (3)$$

$$/\, r_k^i \in R_i \wedge r_k^j \in R_j \wedge \{p_i, p_j\} \in P \wedge c \in \mathbb{R}$$

Thus, it is possible to recover traceability links and to create an implicit traceability matrix by means of mapping patterns, see Fig. 5. In this example, a traceability process between different types of requirements is presented to motivate the use of ontologies as a technique to overcome the common issues when dealing with natural language.

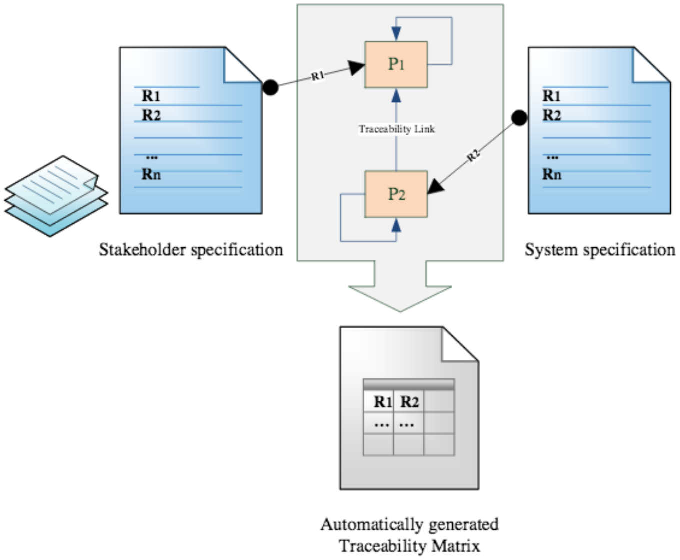


Fig. 5 Example of mapping between text-based requirements and patterns to automatically generate a traceability matrix.

Although this definition of a system traceability process enables us the possibility of elevating the meaning of text-based requirements, models or any other system artifact to a semantic-based representation, the main and common drawback of this approach lies in the necessity of human-validation to ensure that the mapping is 100% correct.

However, the possibility of suggesting links between system artifacts by exploiting the semantic relationships in a domain ontology can dramatically boost the traceability of system artifacts. For instance, both the ISO STEP and OSLC family of specifications include properties that link resources, so, in order to boost the use of both and take the most of a Linked Data environment, system traceability and entity reconciliation are the cornerstone processes for delivering a real collaborative engineering environment.

3.1. *Implementation: technology and tools*

Regarding the implementation of the presented approach, Fig. 6 shows the main functional blocks and tools used to implement the traceability function. More specifically, the approach has been implemented on top of the CAKE ("Computer Aided Knowledge Environment") API and it has been integrated as part of the commercial tool Traceability Studio. New tool adapters to process different types of system artifacts have been implemented to interpret logical models in SysML coming from IBM Rhapsody and to extract text from PDF files. To create a domain ontology following the principles established in previous sections, the KnowledgeManager tool has been used to define the terminology, taxonomy and patterns required to represent knowledge in the domain of the case study.

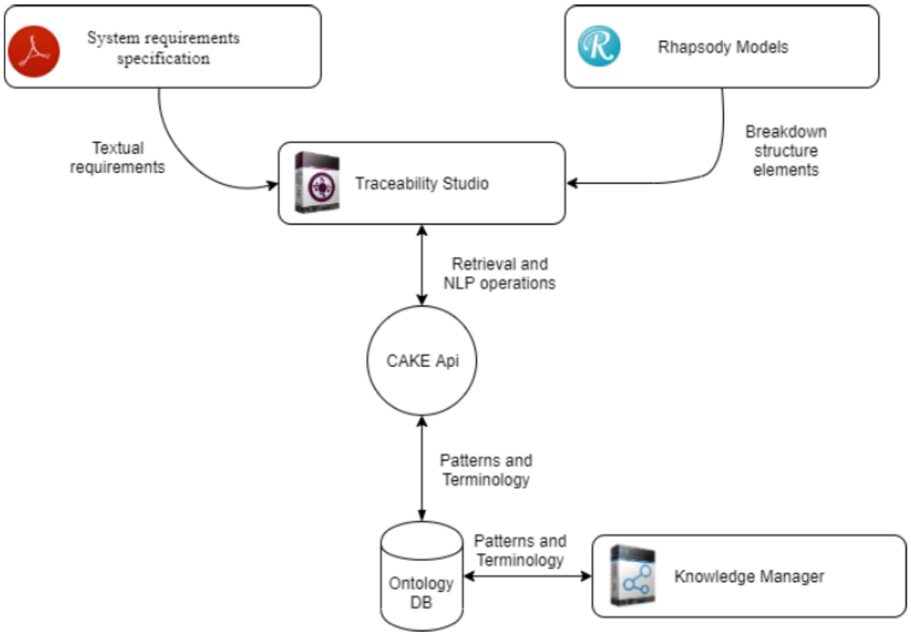


Fig. 6 Functional blocks and tools used to implement the traceability function.

4. Experimentation: a case study to recover traceability links between a system requirement specification and logical models

To illustrate the approach for system traceability presented in this paper, a case study based on the comparison of precision and recall measures of the two approaches to create traceability links (text-based and concept-based) in the railway domain is provided.

4.1. *Research design*

A traceability link discovery process can be seen as a search system in which given a query (a source element of a system artifact, e.g. a requirement) and a set of resources (a target set of elements of a system artifact, e.g. elements in a SysML model), it is necessary to stablish which is the best set of mappings for the source system artifact in the target set of resources.

In this experiment, we have selected some public sources of requirements and SysML models. More specifically, The European Integrate Railway Radio Enhanced Network (EIRENE) System Requirements Specification (version 15.4.0), has been selected as primary source of requirements. This document includes a set of requirements for interoperability aspects of the rail system within the European Community. Furthermore, it contains functional and non-functional requirements for railroad radio systems.

On the other hand, as target resources, some SysML models created with the IBM Rhapsody tool have been created to represent the break down structure provided by the IRIS 5 level product scopes. These models are a logical representation of the documentation developed by International Railway Industry Standard and provides a break down structure used as guidance for product auditing process of vehicles that move on a railway for example locomotives, railroad cars, coaches, and wagons.

Given this context, the following steps to define the elements of the requirements traceability function $T_{system\ traceability}$, have been carried out:

1. Design a domain-based vocabulary, O, to represent the concepts and relationships that will be used to represent system artifacts in the railway domain.
2. Create set of patterns P_R and P_{SR^R} / $P_R \cap P_{SR^R} = \emptyset$, for representing both the system requirements specification where R represents the set of system requirements and SR^R is the set of model elements for the specification R. Due to the fact that patterns for a particular product or project in the Systems Engineering domain are usually private, a public set of eighteen patterns developed within the CESAR project [24] [25] has been selected, adapted and extended to the this case study, see TABLE 2.
3. Create a set of system requirements based on the previous patterns, $R = \{R^1, R^2, \ldots, R^k, \ldots, R^m\}$, where $\#R^k$ represents the number of system requirements in the specification R^k. In this case, there is only one requirement specification presented in TABLE 5 (Appendix A).
4. For every system requirement specification, R^k, define a set of model elements based on the previous patterns, SR^{R^k}, where $\#SR^{R^k}$ represents the number of model elements in the specification $\#SR^{R^k}$. In this case, the set of model elements is presented in TABLE 6 (Appendix B).
5. Run the traceability function implemented on top of the CAKE API to discover links between the model elements in SR^{R^k} and the requirements in R^k. For every requirement in $R^k SR^{R^k}$, search which is the best set of mappings in $SR^{R^k} R^k$. To do so, two matching methods have been used: 1) text-based and 2) concept-based (semantic patterns).

TABLE 2 General patterns to write requirements defined in the frame of the European research project CESAR.

List of patterns for writing requirements
<system> may <action>
<system> may <action> *<entity>*
<system> may be <state>
<system> shall <action>
<system> shall <action> *<entity>*
<system> shall allow <entity> to be <state>
<system> shall be <entity>
<system> shall have <entity>
<system> shall have <quality factor> or at least <quantity> <unit>
<system> shall have <quality factor> or at the most <quantity> <unit>
<system> shall not <action>
<system> shall not <action> *<entity>*
<system> shall not allow <action>
<system> shall not allow <action> *<entity>*
<system> shall not allow <entity> <action>
<user> shall be able to <action>
<system> shall be <state>
<system> shall be <state> <quantity> <unit>

6. 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 $P = tp/tp+fp$, $R = tp/tp+fn$ and $F1=2\ P*R\ /\ P+R$, where given a requirement within a specification, R^k, it is used as a query, and the model elements in SR^{R^k}, are then used as target resources. The interpretation of the metrics is as follows: *tp* (true positive) is "*the number of model elements in* $SR^{R^k}R^k$ *that have been retrieved and represent correct mappings*", *fp* (false positive) is "*the number of model elements in* $SR^{R^k}R^k$ *that have been retrieved and represent incorrect mappings*", *tn* (true negative) is "*the number of model elements in* $SR^{R^k}R^k$ *that have not been retrieved and represent incorrect mappings*" and *fn* (false negative) is "*the number of model elements in* $SR^{R^k}R^k$ *that have not been retrieved and represent correct mappings*".
7. Check the robustness of the comparison by performing statistical hypothesis testing.

4.2. *Results*

Table 3 shows the metrics of precision, recall and the F1 measure of the two different approaches. The first two column corresponds to the test identifier; the next three columns contain the metric values when the text-based approach is executed to discover traceability links. After that, the second set of columns shows the metric values when a concept-based approach is executed using an ontology as underlying knowledge for representing all system artifacts.

According to the results, the concept-based approach is in general better than the text-based in both precision and recall, as Figure 7 depicts. The main reason of this behavior is since concept-based approaches can take advantage of exploiting semantic relationships and concepts while the text-based approach can only perform string comparisons. However, the precision values are still low and higher values would be expected. This is because the context of the experiment (stop-words, acronyms, etc.) is quite generic and a more personalized version of the ontology for the railway domain could imply better results in terms of precision.

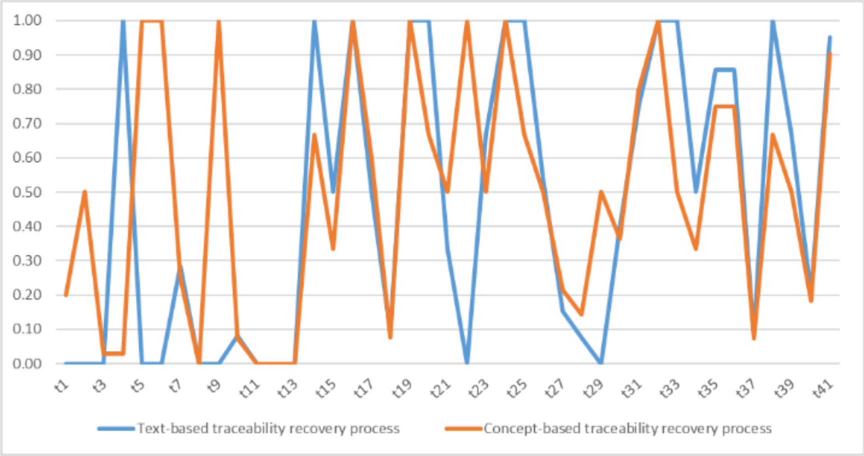


Figure 7 Precision metrics for each test case and method to recovery traceability links between elements of the two selected system artifacts: requirements and SysML models.

On the other hand, a statistical hypothesis testing has been carried out to demonstrate if results will vary depending on the type of method used to discover traceability links. To do so, a comparison of the precision values of both methods (text-based and concept-based) has been formulated through the next hypotheses:

H_o: There is no change in the calculation of precision when applying a text-based or a concept-based approach.

H_1: There is change in the calculation of precision when applying a text-based or a concept-based approach.

TABLE 3 Individual precision, recall and F1 metrics for each test case and method to recovery traceability links between elements of the two selected system artifacts: requirements and SysML models.

Test Case	**Text-based traceability recovery process**			**Concept-based traceability recovery process**		
	P	**R**	**F1**	**P**	**R**	**F1**
t_1	0.00	0.00	0.00	0.20	0.33	0.25
t_2	0.00	0.00	0.00	0.50	0.40	0.44
t_3	0.00	0.00	0.00	0.03	1.00	0.06
t_4	1.00	0.50	0.67	0.03	1.00	0.06
t_5	0.00	0.00	0.00	1.00	1.00	1.00
t_6	0.00	0.00	0.00	1.00	1.00	1.00
t_7	0.29	1.00	0.44	0.25	1.00	0.40
t_8	0.00	0.00	0.00	0.00	0.00	0.00
t_9	0.00	0.00	0.00	1.00	0.33	0.50
t_{10}	0.08	1.00	0.15	0.07	1.00	0.14
t_{11}	0.00	0.00	0.00	0.00	0.00	0.00
t_{12}	0.00	0.00	0.00	0.00	0.00	0.00
t_{13}	0.00	0.00	0.00	0.00	0.00	0.00
t_{14}	1.00	0.67	0.80	0.67	0.67	0.67
t_{15}	0.50	1.00	0.67	0.33	1.00	0.50
t_{16}	1.00	0.50	0.67	1.00	1.00	1.00
t_{17}	0.50	0.67	0.57	0.60	1.00	0.75
t_{18}	0.09	1.00	0.17	0.08	1.00	0.14
t_{19}	1.00	0.50	0.67	1.00	1.00	1.00
t_{20}	1.00	1.00	1.00	0.67	1.00	0.80
t_{21}	0.33	0.50	0.40	0.50	1.00	0.67
t_{22}	0.00	0.00	0.00	1.00	1.00	1.00
t_{23}	0.67	1.00	0.80	0.50	1.00	0.67
t_{24}	1.00	0.50	0.67	1.00	1.00	1.00
t_{25}	1.00	1.00	1.00	0.67	1.00	0.80
t_{26}	0.54	1.00	0.70	0.50	1.00	0.67
t_{27}	0.15	0.67	0.25	0.21	1.00	0.35
t_{28}	0.08	0.50	0.13	0.14	1.00	0.25
t_{29}	0.00	0.00	0.00	0.50	1.00	0.67
t_{30}	0.40	1.00	0.57	0.36	1.00	0.53
t_{31}	0.75	0.75	0.75	0.80	1.00	0.89
t_{32}	1.00	1.00	1.00	1.00	1.00	1.00
t_{33}	1.00	1.00	1.00	0.50	1.00	0.67
t_{34}	0.50	1.00	0.67	0.33	1.00	0.50
t_{35}	0.86	1.00	0.92	0.75	1.00	0.86
t_{36}	0.86	1.00	0.92	0.75	1.00	0.86
t_{37}	0.08	1.00	0.15	0.07	1.00	0.14
t_{38}	1.00	0.67	0.80	0.67	0.67	0.67
t_{39}	0.67	1.00	0.80	0.50	1.00	0.67
t_{40}	0.20	1.00	0.33	0.18	1.00	0.31
t_{41}	0.95	1.00	0.97	0.90	1.00	0.95
$\mathbf{T_{avg}}$	**0.45**	**0.60**	**0.45**	**0.49**	**0.84**	**0.56**

In order to run the statistical hypothesis testing, the F-Test with alpha 0.05 has been used 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 4, H_o can be rejected, since the t Stat is less than "-t Critical (two tail)". In conclusion, the concept-based method exploiting semantic relationships can improve in terms of precision the problem of discovering links between two types of system artifacts.

TABLE 4 Statistical hypothesis testing, the t-Test of two-sample assuming unequal variances to compare test-based vs concept-based traceability link discovery processes for precision metric.

	Concept-based traceability recovery process	*Text-based traceability recovery process*
Mean	0.4945	0.4509
Variance	0.1255	0.1742
Observations	41	41
Hypothesized	0	
Df	78	
t Stat	0.5094	
P(T<=t) one-tail	0.3060	
t Critical (one tail)	1.6646	
P(T<=t) two tail	0.6119	
t Critical (two tail)	1.9908	

5. Discussion and 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 and more specifically, requirements have been extracted from an existing specification in the railway domain. That is why, results in a broad or different scope could change, in terms of robustness, since more complex relationships in the domain ontology and patterns could be designed for the same purpose. However, the research methodology, the design of experiments and the creation of a kind of benchmark for testing system traceability processes have been demonstrated to be representative and creditable.

Building on the previous comment, we cannot either figure out the internal budget, methodologies, domain vocabularies, experience and background of specific domain-experts to create and trace requirements. We merely observe and re-use existing public and on-line knowledge sources to provide an accurate traceability link discovery process for system artifacts traceability. Finally, we have also identified the possibility of adding a new variable to the experiment regarding the quality of a requirements specification. Thus, it should be possible to state that the higher quality a requirements specification is, the easier and more accurate the traceability process is.

6. Related work

The traceability of system artifacts within the Software and Systems Engineering process is not a new topic (mainly focusing on requirements traceability) and it has been addressed following different approaches [8] but mainly focusing on text-based artifacts such as requirements. The CESAR project [24] [25], an ARTEMIS European research project, tackled this problem by defining statement-based requirements comprising different concepts modeled in an ontology, using DODT as a tool for ontology management. They defined several concept-based templates for those requirements in which they were interested in, easing and restricting the writing of requirements. Thus, the traceability process was based on comparing the use of the different concepts in each statement.

The International Council of Systems Engineering (INCOSE) also keeps a track of the most used requirements traceability management tools*. However, this list is not up-to-date, it is focusing on requirements traceability and some of the links are broken. Besides, it mainly contains commercial tools in which the strategy to trace system artifacts is not completely described. For instance, as an example, Reqtify is a commercial traceability tool based on regular expressions [26], mainly working for natural language resources.

In the Software and Systems Engineering discipline, it is also possible to find solutions based on the use of logics [27] or methodologies such as [28] focusing on a particular type of system (embedded systems) using models [29] [30] as basic unit for traceability.

On the other hand and addressing the underlying problem of traceability (natural language processing and entity matching), a good number of works can be found in the databases area to automatically create linkage rules [31] between entities, to perform entity matching processes [32] [33] or to find duplicate records [34] and relationships [35].

In the Semantic Web area and due to the emerging use of Linked Data, some works [36] have followed a similar approach to the ones already developed in the databases domain but discovering mappings between two RDF-based resources under a certain threshold of confidence. Here, it is necessary to distinguish two types of mappings: 1) T-Box mapping (between classes within an ontology) and 2) A-Box mapping (between instances). In the first case, the PROMPT algorithm [37] can suggest potential mappings and alignment of two classes by comparing text descriptions (names, labels and descriptions), tags or keywords (based on a controlled vocabulary) or the internal structure of the classes (number and type of the properties, common super classes/subclasses), this last approach was also used in the Feature-based entity matching model [38].

In this sense, a language for creating mappings [39] [40] between semantic web services was designed to automatically create choreographies and orchestrations of web services but it was also based on discovering and selecting potential services using keywords. The problem of entity reconciliation in ontologies have also attracted a lot of

---

* http://www.incose.org/productspubs/products/setools/tooltax/reqtrace_tools.html

research works and the Ontology Alignment Evaluation Initiative (OAEI) [32] was launched to aggregate all works in this area and providing an one-stop site for researchers and practitioners looking for new techniques and benchmarks to test their algorithms. For instance, the RiMOM (Dynamic Multistrategy Ontology Alignment Framework) framework [41], an approach to quantitatively estimate the similarity characteristics for ontology alignment based on string matching and structural properties or the CODI (Combinatorial Optimization for Data Integration) framework [42], a probabilistic-logical alignment system, are two other remarkable works in this area of ontology alignment.

In the second case, the LIMES (LInk discovery framework for MEtric Spaces) framework [43] and the Silk Server [44] also offer an entity reconciliation system based on comparing textual descriptions and assuming that two similar resources will share a similar description. In both cases, given an RDF resource that must be aligned to a dataset, the tools generate a set of potential mappings under a certain threshold of confidence defined by the user (human-validation is required to ensure a 100% of confidence). The SERIMI (Resource Description Similarity), framework [45] is other implementation of entity reconciliation based on string comparison and search on top of the Apache Lucene search engine. URI comparison [46], that is actually string comparison, is other approach that was used to discover links between similar RDF resources. Moreover, some Linked Data-based domains such as e-Government, e-Health or e-Procurement have published works to solve specific mapping problems such as product classifications linking [47] [48], the Bioportal, etc.

In conclusion, last times have seen the application of entity reconciliation techniques in the Semantic Web area to enable users to develop new services such as faceted browsing, semantic search systems (entity recommendation) [49] [50] or recommending systems [51] [52] [53] by applying existing techniques coming from the databases domain and based on natural language processing (NLP) techniques. Finally, some tools such as Apache Stanbol or Open Refine also offer a suite of NLP-based techniques to perform entity reconciliation processes.

Since NLP-based techniques and computational linguistics techniques are the cornerstone to enable the mapping of entities in the aforementioned disciplines, a very good number of works can be found dealing with natural language issues such as misspelling errors [54] or name/acronym mismatches [55] [56]. These approaches can be applied to solve general problems and usually follow a traditional approach of text normalization, lexical analysis, pos-tagging word according to a grammar and semantic analysis to filter or provide some kind of service such as information/knowledge extraction, reporting, sentiment analysis or opinion mining. Well-established APIs such as NLTK [57] for Python, Lingpipe [58, p. 6] , OpenNLP [59] or Gate [60] for Java, WEKA [61], the Apache Lucene and Solr [62] search engines provide the proper building blocks to build natural-language based applications. In this light, the analysis of social networks such as Twitter [63], the extraction of clinical terms [64] for electronic health records, the creation of bibliometrics [65] [66] [67] or the identification of gene names [68] to name a few have tackled the problem of entity recognition [69], extraction and matching [70] [71] from raw sources [17]. Finally and regarding pattern-matching problems, some areas such as Biology [72], string-based pattern matching [73] [74] and studies about plagiarism [75] have designed algorithms based on NLP and machine learning [76].

In conclusion, TABLE 5 outlines the main approaches for traceability in the Systems Engineering discipline identifying that existing approaches are based on performing some entity matching algorithm. On the other hand, a very good body work can be found in areas such as databases, Semantic Web and Linked Data as the basis to provide advanced services of searching, recommendations, gene sequencing, etc. re-using and extending existing NLP techniques. Obviously, system traceability can be reached taking advantage of these existing approaches applying pattern matching techniques to link system artifacts. Thus, the challenge of creating an integrated, interoperable and collaborative environment for complex systems development will become real. However, the automatic mapping between two system artifacts requires human validation to become 100% correct, it is necessary to discover and suggest potential mappings under a certain threshold of confidence to ease users the implementation of traceability processes.

TABLE 5 Summary of the main approaches for entity reconciliation and natural language processing techniques in different domains.

Discipline/Area/ Domain	**Description (based on)**	**References**
Systems Engineering	Regular Expressions	[26]
	Logics	[27]
	Models†	[28] [29] [30]
Databases	Linkage rules generation	[31]
	Entity matching	[32] [33]
	Find duplication records	[34] [35]
Semantic Web & Linked Data	Free text	[32] [37] [41] [43] [44] [45] [47] [48]
	Keywords	[32] [37] [39] [43] [44] [45] [47] [48]
	Structural Analysis	[32] [37] [38] [40] [41] [45] [47] [48]
	URI comparison	[32] [37] [46]
	String comparison and machine learning	[76]
	Statistics and Probability	[42]
Natural Language Processing and Computation Linguistics	Some examples of NLP foundations and existing techniques	[54], [55], [56]
Natural Language Processing for pattern-matching	String-based Pattern matching	Biology [68] [72], String matching [73] [74], Plagiarism detection [75]
Tools	Some examples of tools based of NLP techniques	Survey of tools [36], Apache Lucene & Solr [62], Apache Stanbol [77],

†http://www.asa.transport.nsw.gov.au/sites/default/files/asa/railcorp-legacy/disciplines/allstandards/epd-0005.pdf

	for entity reconciliation	Open Refine, NLTK [57], Lingpipe [58, p. 6] , OpenNLP [59], Gate [60], WEKA [61]
Applications	Some services that are taking advantage of the NLP techniques	Search systems (information extraction and retrieval, web links discovery, concept-based search, etc.) [49] [50], Recommendation-based systems [51] [52] [53], Entity recognition [69] and matching [17] [70] [71],
Domains	Some application domains that are taking advantage of the NLP techniques	e-Procurement [47] [48], e-Health [64], Bibliometrics [65] [66] [67], Social Network Analysis [63], etc.

7. Conclusions and Future Work

The present paper has introduced a semantic-driven approach to represent system artifacts by promoting textual descriptions or structured data to a semantic (concept-based) representation. More specifically, the use of ontologies to guide the activity of authoring requirements and logical models has been outlined as a cornerstone to design concept-based system artifacts. Furthermore, a system traceability function has been defined, implemented and integrated on top of the existing CAKE API.

On the other hand, experiments have demonstrated that the creation of concept-based system artifacts containing concepts and semantic relationships is useful to discover links between different types of system artifacts at a different level of abstraction (system requirements and logical models).

Regarding the applicability of the results in the current context of Software and Systems Engineering, traceability is considered a key process to boost collaborative engineering easing the task of discovering and mapping similar artifacts, in this case requirements, and to support the emerging set of ISO STEP and OSLC-based specifications. Thus, the link discovery process can also be applied to link different resources if their textual descriptions or content are represented using a pattern-based approach. Although some methodologies have outlined the possibility of using models as first-class members to communicate ideas, etc., existing development environments are human-oriented being natural language the main type of communication. That is why, an approach based on exploiting domain knowledge can ease tasks that go from system artifact authoring to write any kind of name, specification, etc. overcoming the common issues when dealing with natural language descriptions. In this light, the development lifecycle of a critical system can take advantage of domain knowledge by elevating the meaning of textual descriptions easing communications in a human-oriented environment.

Therefore, collaboration or reduction of costs and time among others are side-effects of exploiting and re-using domain knowledge in different very time-consuming tasks such as traceability. On the other hand and regarding system artifact traceability, the aforementioned research limitations should be tackled to get better results in terms of accuracy including new parameters such as quality with the aim of delivering a real Linked Data environment in which the creation of links between artifacts can be done effortless and with a high degree of confidence. Finally, we are also contributing to the communities working on pattern and entity matching in the area of Software and System Engineering by making publicly available the ontology and the system artifacts used in the experimentation.

Acknowledgments

The research leading to these results has received funding from the H2020-ECSEL Joint Undertaking (JU) under grant agreement No 826452-"Arrowhead Tools for Engineering of Digitalisation Solutions" and from specific national programs and/or funding authorities.

References

[1] J. L. de la Vara, G. Génova, J. M. Álvarez-Rodríguez, and J. Llorens, "An analysis of safety evidence management with the Structured Assurance Case Metamodel," *Comput. Stand. Interfaces*, vol. 50, pp. 179–198, Feb. 2017, doi: 10.1016/j.csi.2016.10.002.

[2] C. Ebert, "50 Years of Software Engineering: Progress and Perils," *IEEE Softw.*, vol. 35, no. 5, pp. 94–101, Sep. 2018, doi: 10.1109/MS.2018.3571228.

[3] J. Bézivin, "Model Driven Engineering: An Emerging Technical Space," in *Generative and Transformational Techniques in Software Engineering*, vol. 4143, R. Lämmel, J. Saraiva, and J. Visser, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 36–64.

[4] European Conference on Model Driven Architecture - Foundations and Applications, *Model driven architecture: foundations and applications: second European conference, ECMDA-FA 2006, Bilbao, Spain, July 10-13, 2006: proceedings*. Berlin: Springer-Verlag, 2006.

[5] J. Ø. Aagedal, J. Bézivin, and P. F. Linington, "Model-Driven Development (WMDD 2004)," in *Object-Oriented Technology. ECOOP 2004 Workshop Reader*, vol. 3344, J. Malenfant and B. M. Østvold, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 148–157.

[6] INCOSE, "Systems Engineering Vision 2020," INCOSE, Technical INCOSE-TP-2004-004-02, 2004. Accessed: Nov. 24, 2013. [Online]. Available: http://www.incose.org/ProductsPubs/pdf/SEVision2020_20071003_v2_03.pdf.

[7] O. Gotel *et al.*, "The Grand Challenge of Traceability (v1.0)," in *Software and Systems Traceability*, J. Cleland-Huang, O. Gotel, and A. Zisman, Eds. London: Springer London, 2012, pp. 343–409.

[8] A. E. Limón and J. G. Sopeña, "The Need for a Unifying Traceability Scheme," in *ECMDA Traceability Workshop Proceedings*, NORWAY, Oct. 2005, [Online]. Available: http://oa.upm.es/5754/.

[9] O. C. Gotel and A. C. Finkelstein, "An analysis of the requirements traceability problem," in *Requirements Engineering, 1994., Proceedings of the First International Conference on*, 1994, pp. 94–101.

[10] Cecilia Haskins, *SYSTEMS ENGINEERING HANDBOOK. A GUIDE FOR SYSTEM LIFE CYCLE PROCESSES AND ACTIVITIES*., vol. INCOSE-TP-2003-002-03.2.2. International Council on Systems Engineering (INCOSE), 2011.

[11] J. M. Alvarez-Rodríguez, J. Llorens, M. Alejandres, and J. M. Fuentes, "OSLC-KM: A knowledge management specification for OSLC-based resources," *INCOSE Int. Symp.*, vol. 25, no. 1, pp. 16–34, Oct. 2015, doi: 10.1002/j.2334-5837.2015.00046.x.

[12] T. Dang, "Open Services for Lifecycle Collaboration Reconciliation Specification Version 2.," 2014. [Online]. Available: http://open-services.net/wiki/reconciliation/OSLC-Reconciliation-Specification-Version-2.0/.

[13] J. G. Enríquez, F. J. Domínguez-Mayo, M. J. Escalona, M. Ross, and G. Staples, "Entity reconciliation in big data sources: A systematic mapping study," *Expert Syst. Appl.*, vol. 80, pp. 14–27, Sep. 2017, doi: 10.1016/j.eswa.2017.03.010.

[14] M. Lormans and A. van Deursen, "Reconstructing requirements coverage views from design and test using traceability recovery via LSI," in *Proceedings of the 3rd international workshop on Traceability in emerging forms of software engineering*, 2005, pp. 37–42.

[15] B. M. Blackwell, J. P. Collen, L. R. Guzman Jr, A. G. Irwin, B. M. Kokke, and J. D. Lindsay, *Testing tool comprising an automated multidimensional traceability matrix for implementing and validating complex software systems*. Google Patents, 2009.

[16] V. Castañeda, L. Ballejos, L. Caliusco, and R. Galli, "The Use of Ontologies in Requirements Engineering," *GJRE*, vol. 10, no. 6, 2010, [Online]. Available: http://engineeringresearch.org/index.php/GJRE/article/view/76.

[17] A. Ittoo and G. Bouma, "Term extraction from sparse, ungrammatical domain-specific documents," *Expert Syst Appl*, vol. 40, no. 7, pp. 2530–2540, 2013.

[18] T. R. Gruber, "A Translation Approach to Portable Ontology Specifications," *Knowl Acquis*, vol. 5, no. 2, pp. 199–220, Jun. 1993, doi: 10.1006/knac.1993.1008.

[19] N. Guarino, "Formal Ontology, Conceptual Analysis and Knowledge Representation," *Int J Hum-Comput Stud*, vol. 43, no. 5–6, pp. 625–640, Dec. 1995, doi: 10.1006/ijhc.1995.1066.

[20] J. Llorens, J. Morato, and G. Genova, "RSHP: an information representation model based on relationships," in *Soft Computing in Software Engineering*, vol. 159, E. Damiani, M. Madravio, and LakhmiC. Jain, Eds. Springer Berlin Heidelberg, 2004, pp. 221–253.

[21] D. Beckett, "RDF/XML Syntax Specification (Revised)," W3C, W3C Recommendation, 2008.

[22] J. M. Álvarez-Rodríguez, V. Moreno, and J. Lloréns, "Formal ontologies and data shapes within the Software Engineering development lifecycle," in *The 31st International Conference on Software Engineering and Knowledge Engineering, SEKE 2019, Hotel Tivoli, Lisbon, Portugal, July 10-12, 2019.*, 2019, pp. 64–93, doi: 10.18293/SEKE2019-114.

[23] M. E. C. Hull, K. Jackson, and J. Dick, Eds., *Requirements Engineering, Third Edition*. Springer, 2011.

[24] H. Zojer, "Revised Definitions of Improved RE Methods," CESAR Project, D_SP2_R3.3_M3_Vol2, 2011. [Online]. Available: http://www.cesarproject.eu/fileadmin/user_upload/CESAR_D_SP2_R3.3_M3_Vol2_v1.000_PU.pdf.
[25] E. Leroy, "Requirement/Traceability Management," CESAR Project, D_SP2_R3.3_M3_Vol3, 2011. [Online]. Available: http://www.cesarproject.eu/fileadmin/user_upload/CESAR_D_SP2_R3.3_M3_Vol3_v1.000_PU.pdf.
[26] R. Geensoft, *Effective Solution for Managing Requirements Traceability and Impact Analysis across Hardware and Software Projects Lifecycle*. .
[27] B. Manfred, "A Logical Approach to Systems Engineering Artifacts and Traceability: From Requirements to Functional and Architectural Views," *NATO Sci. Peace Secur. Ser. Inf. Commun. Secur.*, pp. 1–48, 2013, doi: 10.3233/978-1-61499-207-3-1.
[28] A. ALBINET *et al.*, "Model-based methodology for requirements traceability in embedded systems," in *Proceedings of 3rd European Conference on Model Driven Architecture® Foundations and Applications, ECMDA'07*, 2007.
[29] S. Winkler and J. Pilgrim, "A survey of traceability in requirements engineering and model-driven development," *Softw. Syst. Model. SoSyM*, vol. 9, no. 4, pp. 529–565, 2010.
[30] P. Mason, "On traceability for safety critical systems engineering," 2005, p. 8 pp., doi: 10.1109/APSEC.2005.85.
[31] P. Domingos, "Multi-relational record linkage," in *In Proceedings of the KDD-2004 Workshop on Multi-Relational Data Mining*, 2004.
[32] A. Ferrara, A. Nikolov, J. Noessner, and F. Scharffe, "Evaluation of instance matching tools: The experience of OAEI," *Web Semant. Sci. Serv. Agents World Wide Web*, vol. 21, pp. 49–60, 2013.
[33] A. Hogan, A. Zimmermann, J. Umbrich, A. Polleres, and S. Decker, "Scalable and distributed methods for entity matching, consolidation and disambiguation over linked data corpora," *Web Semant. Sci. Serv. Agents World Wide Web*, vol. 10, pp. 76–110, Jan. 2012, doi: 10.1016/j.websem.2011.11.002.
[34] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios, "Duplicate record detection: A survey," *Knowl. Data Eng. IEEE Trans. On*, vol. 19, no. 1, pp. 1–16, 2007.
[35] M. Weis and F. Naumann, "Relationship-based duplicate detection," 2006.
[36] A. Gangemi, "A comparison of knowledge extraction tools for the semantic web," in *The Semantic Web: Semantics and Big Data*, Springer, 2013, pp. 351–366.
[37] N. F. Noy and M. A. Musen, "PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment," in *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on on Innovative Applications of Artificial Intelligence, July 30 - August 3, 2000, Austin, Texas, USA.*, 2000, pp. 450–455, [Online]. Available: http://www.aaai.org/Library/AAAI/2000/aaai00-069.php.
[38] H. Stoermer, N. Rassadko, and N. Vaidya, "Feature-based entity matching: The FBEM model, implementation, evaluation," in *Advanced Information Systems Engineering*, 2010, pp. 180–193.

[39] F. Scharffe and J. de Bruijn, "A language to specify mappings between ontologies," in *Proceedings of the 1st International Conference on Signal-Image Technology and Internet-Based Systems, SITIS 2005, November 27 - December 1, 2005, Yaounde, Cameroon*, 2005, pp. 267–271, [Online]. Available: http://www.u-bourgogne.fr/SITIS/05/download/Proceedings/Files/f142.pdf.

[40] M. García-Rodríguez, J. M. Álvarez-Rodríguez, D. B. Muñoz, L. P. Paredes, J. E. L. Gayo, and P. O. de Pablos, "Towards a Practical Solution for Data Grounding in a Semantic Web Services Environment," *J UCS*, vol. 18, no. 11, pp. 1576–1597, 2012, doi: 10.3217/jucs-018-11-1576.

[41] J. Li, J. Tang, Y. Li, and Q. Luo, "Rimom: A dynamic multistrategy ontology alignment framework," *Knowl. Data Eng. IEEE Trans. On*, vol. 21, no. 8, pp. 1218–1232, 2009.

[42] J. Noessner and M. Niepert, "Codi: Combinatorial optimization for data integration–results for oaei 2010," *Ontol. Matching*, p. 142, 2010.

[43] A.-C. N. Ngomo and S. Auer, "LIMES: a time-efficient approach for large-scale link discovery on the web of data," in *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume Three*, 2011, pp. 2312–2317.

[44] R. Isele, A. Jentzsch, and C. Bizer, "Silk Server - Adding missing Links while consuming Linked Data," in *Proceedings of the First International Workshop on Consuming Linked Data, Shanghai, China, November 8, 2010*, 2010, [Online]. Available: http://ceur-ws.org/Vol-665/IseleEtAl_COLD2010.pdf.

[45] S. Araújo, D. Tran, A. DeVries, J. Hidders, and D. Schwabe, "SERIMI: Class-based Disambiguation for Effective Instance Matching over Heterogeneous Web Data," in *WebDB*, 2012, pp. 25–30.

[46] F. Maali, R. Cyganiak, and V. Peristeras, "Re-using Cool URIs: Entity Reconciliation Against LOD Hubs.," in *LDOW*, 2011, vol. 813, [Online]. Available: http://dblp.uni-trier.de/db/conf/www/ldow2011.html#MaaliCP11.

[47] J. M. Alvarez-Rodríguez, J. E. Labra-Gayo, A. Rodríguez-González, and P. O. D. Pablos, "Empowering the access to public procurement opportunities by means of linking controlled vocabularies. A case study of Product Scheme Classifications in the European e-Procurement sector," *Comput. Hum. Behav.*, vol. 30, no. 0, pp. 674–688, 2014, doi: http://dx.doi.org/10.1016/j.chb.2013.07.046.

[48] J. Alvarez-Rodríguez, P. Ordoñez de Pablos, M. Vafopoulos, and J. Labra-Gayo, "Towards a Stepwise Method for Unifying and Reconciling Corporate Names in Public Contracts Metadata: The CORFU Technique," in *Metadata and Semantics Research*, vol. 390, E. Garoufallou and J. Greenberg, Eds. Springer International Publishing, 2013, pp. 315–329.

[49] R. Blanco *et al.*, "Repeatable and reliable semantic search evaluation," *Web Semant. Sci. Serv. Agents World Wide Web*, vol. 21, pp. 14–29, Aug. 2013, doi: 10.1016/j.websem.2013.05.005.

[50] R. Blanco, B. B. Cambazoglu, P. Mika, and N. Torzec, "Entity Recommendations in Web Search," in *The Semantic Web - ISWC 2013 - 12th International Semantic Web Conference, Sydney, NSW, Australia, October 21-25, 2013, Proceedings, Part II*, 2013, pp. 33–48, doi: 10.1007/978-3-642-41338-4_3.

[51] A. Montes-García, J. M. Alvarez-Rodríguez, J. E. L. Gayo, and M. Martínez-Merino, "Towards a journalist-based news recommendation system: The Wesomender approach," *Expert Syst Appl ESWA*, vol. 40, no. 17, pp. 6735–6741, 2013.

[52] R. C. Palacios, J. L. L. Cuadrado, I. Gonzalez-Carrasco, and J. F. G. Peñalvo, "SABUMO-dTest: Design and evaluation of an intelligent collaborative distributed testing framework," *Comput Sci Inf Syst*, vol. 11, no. 1, pp. 29–45, 2014, doi: 10.2298/CSIS130129019C.

[53] R. C. Palacios, C. Casado-Lumbreras, P. Soto-Acosta, and S. Misra, "Providing knowledge recommendations: an approach for informal electronic mentoring," *Interact. Learn. Environ.*, vol. 22, no. 2, pp. 221–240, 2014, doi: 10.1080/10494820.2012.745430.

[54] S. N. L. P. Lecture, *Spelling Correction and the Noisy Channel. The Spelling Correction Task*. 2013.

[55] S. Yeates, "Automatic Extraction of Acronyms from Text," in *University of Waikato*, 1999, pp. 117–124.

[56] L. Ratinov and E. Gudes, "Abbreviation Expansion in Schema Matching and Web Integration," in *Proceedings of the 2004 IEEE/WIC/ACM International Conference on Web Intelligence*, Washington, DC, USA, 2004, pp. 485–489, doi: 10.1109/WI.2004.18.

[57] E. Loper and S. Bird, "NLTK: The Natural Language Toolkit," in *Proceedings of the ACL Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics*, 2002, pp. 62–69.

[58] B. B. M. M. Bob Carpenter, *Text Processing with Java 6*, vol. 1. LingPipe Publishing, 2012.

[59] S. N. L. P. Lecture, *Apache OpenNLP Developer Documentation*. 2013.

[60] K. Bontcheva *et al.*, "GATE Teamware: a web-based, collaborative text annotation framework," *Lang. Resour. Eval.*, pp. 1–23, 2013, doi: 10.1007/s10579-013-9215-6.

[61] J. Read, A. Bifet, G. Holmes, and B. Pfahringer, "Scalable and efficient multi-label classification for evolving data streams," *Mach. Learn.*, vol. 88, no. 1–2, pp. 243–272, 2012.

[62] D. Smiley and D. E. Pugh, *Apache Solr 3 Enterprise Search Server*. Packt Publishing Ltd, 2011.

[63] C. Li *et al.*, "TwiNER: named entity recognition in targeted twitter stream," in *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, New York, NY, USA, 2012, pp. 721–730, doi: 10.1145/2348283.2348380.

[64] Y. Wang, "Annotating and recognising named entities in clinical notes," in *Proceedings of the ACL-IJCNLP 2009 Student Research Workshop*, Stroudsburg, PA, USA, 2009, pp. 18–26, [Online]. Available: http://dl.acm.org/citation.cfm?id=1667884.1667888.

[65] C. Galvez and F. Moya-Anegón, "The unification of institutional addresses applying parametrized finite-state graphs (P-FSG)," *Scientometrics*, vol. 69, no. 2, pp. 323–345, 2006, doi: 10.1007/s11192-006-0156-3.

[66] F. Morillo, J. Aparicio, B. González-Albo, and L. Moreno, "Towards the automation of address identification," *Scientometrics*, vol. 94, no. 1, pp. 207–224, Jan. 2013, doi: 10.1007/s11192-012-0733-6.

[67] T. Mahmood, S. I. Jami, Z. A. Shaikh, and M. H. Mughal, "Toward the modeling of data provenance in scientific publications," *Comput. Stand. Interfaces*, vol. 35, no. 1, pp. 6–29, 2013.

[68] C. Galvez and F. Moya-Anegón, "A Dictionary-Based Approach to Normalizing Gene Names in One Domain of Knowledge from the Biomedical Literature," *J. Doc.*, vol. 68, no. 1, pp. 5–30, 2012, doi: 10.1007/s11192-006-0156-3.

[69] D. Nadeau and S. Sekine, "A survey of named entity recognition and classification," *Lingvisticae Investig.*, vol. 30, no. 1, pp. 3–26, 2007.

[70] H. Köpcke and E. Rahm, "Frameworks for entity matching: A comparison," *Data Knowl. Eng.*, vol. 69, no. 2, pp. 197–210, 2010.

[71] A. Ittoo and G. Bouma, "Minimally-supervised learning of domain-specific causal relations using an open-domain corpus as knowledge base," *Data Knowl Eng*, vol. 88, pp. 142–163, 2013, doi: 10.1016/j.datak.2013.08.004.

[72] S. Sheik, S. K. Aggarwal, A. Poddar, N. Balakrishnan, and K. Sekar, "A FAST pattern matching algorithm," *J. Chem. Inf. Comput. Sci.*, vol. 44, no. 4, pp. 1251–1256, 2004.

[73] S. Faro and T. Lecroq, "The exact online string matching problem: a review of the most recent results," *ACM Comput. Surv. CSUR*, vol. 45, no. 2, p. 13, 2013.

[74] W. Cohen, P. Ravikumar, and S. Fienberg, "A comparison of string metrics for matching names and records," in *KDD Workshop on Data Cleaning and Object Consolidation*, 2003, vol. 3, pp. 73–78.

[75] K. L. Pandey, S. Agarwal, S. Misra, and R. Prasad, "Plagiarism detection in software using efficient string matching," in *Computational Science and Its Applications–ICCSA 2012*, Springer, 2012, pp. 147–156.

[76] M. Bilenko and R. J. Mooney, "Adaptive duplicate detection using learnable string similarity measures," in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2003, pp. 39–48.

[77] F. Christ and B. Nagel, "A Reference Architecture for Semantic Content Management Systems.," in *EMISA*, 2011, pp. 135–148.

Appendix A.

TABLE 6 Requirements used in the experiment as source resources.

Id	Requirement
R_1	A railway GSM network is also likely to have external interfaces to: (I) - private railway fixed networks; - public operator networks; - controller equipment; - specialised railway systems (eg train control systems).
R_2	Enhanced multi-level precedence and pre-emption: This GSM specification is to be implemented in order to achieve the high performance requirements necessary for emergency group calls. It is also necessary to meet different grades of service requirements for different types of communications traffic on the system (eg safety (eg train control system), operational and administrative communications). (I)
R_3	Location dependent addressing: Train drivers need to be able to contact controllers and other staff at the push of a single button. As the train moves through different areas, controllers are liable to change. As a consequence it is necessary to provide a means of addressing calls from a train to certain functions based on the location of the train. (I)
R_4	Many trains employ multiple active traction vehicles. Where these vehicles are not connected by on-train wiring, it shall be possible for a permanent radio connection to be established between each of the active cabs. (I)
R_5	Many trains employ multiple active traction vehicles. Where these vehicles are not connected by on-train wiring, it shall be possible for a permanent radio connection to be established between each of the active cabs. (I)
R_6	The call will be established from the active cab of the lead traction vehicle. Each of the other cabs on the train will be contacted using its functional number (registered by the other drivers prior to the establishment of the call). The procedure for setting up a multi-party call is outlined in figure 5-2. The multi-party call shall have 'Railway operation' priority (see section 10.2) and whilst on-going a 'multi-drivers' indication shall be displayed permanently at all Cab radios.
R_7	On activation of the "call other drivers on the same train" function, the MMI shall provide additional guidance to the user in the establishment and management of a Multi-Party call.
R_8	An emergency power supply should be provided for Cab radios which will enable the driver's radio to continue to operate for a period of 6 hours in the event of failure of the train's main power supply, based on the following cycle (see section 4.5.21): - point-to-point calls 20%; - group calls 5%; - standby 75%.
R_9	The driver and other in-cab equipment shall be protected against all electrical hazards arising from EIRENE mobile equipment as defined in [EN 50153].
R_{10}	The following list catalogues the interfaces that should be provided by the Cab Radio to the on-train systems: - Train borne recorder; - ERTMS/ETCS interface; - Public Address; - UIC Intercom; - Driver's Safety Device; - Other interfaces.
R_{11}	The Operational radio user shall be protected against all electrical hazards arising from the mobile equipment as defined in [EN 50153].

R_{12}	The User Identifier Number (UIN) shall be one of the following numbers (as identified by the CT): - Train Number (TN): a number given to a train by operational staff for a particular journey. This number shall be unique for the duration of the journey. - Note. For certain Train Numbers (e.g. 1234 and 123), a risk exists when dialing a number by keying in individual digits e.g. by the dispatcher.
R_{13}	If a more accurate way of location determination is used, then position information shall be provided to the radio system which shall be used to associate the short code with the correct called party subscriber number.
R_{14}	A Railway emergency call is a high priority call for informing drivers, controllers and other concerned personnel of a level of danger requiring all Railway movements in a pre-defined area to stop. Two types of Railway emergency calls are defined: (I) - Train emergency calls (for Railway emergencies whilst not involved in Shunting operations); - Shunting emergency calls (for Railway emergencies whilst involved in Shunting operations).
R_{15}	The maximum power consumption of the Rolling Stock Components shall be 2500 kw
R_{16}	The maximum power consumption of the Auxiliary systems shall be 40 kw
R_{17}	The maximum power consumption of the Braking systems shall be 30 kw
R_{18}	The maximum power consumption of the Heating, ventilating and air conditioning shall be 300 kw
R_{19}	The maximum power consumption of the interiors shall be 23 kw
R_{20}	The maximum power consumption of the passenger information system shall be 500 w
R_{21}	The Rolling stock component shall have 1 auxiliary system
R_{22}	The Rolling stock component shall have 1 braking system
R_{23}	The Rolling stock component shall have 1 cabinet
R_{24}	The Rolling stock component shall have 1 cabling
R_{25}	The Rolling stock component shall have 1 car body
R_{26}	The Rolling stock component shall have 1 car body fitting
R_{27}	The Rolling stock component shall have 1 communication system
R_{28}	The Rolling stock component shall have 1 coupler
R_{29}	The Rolling stock component shall have 20 doors
R_{30}	The Rolling stock component shall have 1 Heating, ventilating and air conditioning system
R_{31}	The Rolling stock component shall have 1 interior
R_{32}	The Rolling stock component shall have 1 lighting system
R_{33}	The Rolling stock component shall have 1 on board vehicle
R_{34}	The Rolling stock component shall have 1 passenger information system
R_{35}	The Rolling stock component shall have 1 power system
R_{36}	The Rolling stock component shall have 1 propulsion system
R_{37}	The Rolling stock component shall have 1 tilt system
R_{38}	The propulsion shall have 1 gear box
R_{39}	The propulsion shall have 1 mechanical transmission
R_{40}	The propulsion shall have 1 power converter
R_{41}	The propulsion shall have 1 traction control unit

Appendix B.

TABLE 7 Logical model elements used in the experimentation as target resources.

Id	Rhapsody Model Component
C_1	Air_supply_system
C_2	Auxiliary_electric_system
C_3	Hydraulic_system
C_4	Auxiliary_systems
C_5	Magnetic_track_brake_equipment
C_6	Eddy_current_brake_equipment
C_7	Brake_control_system
C_8	Emergency_brake_equipment
C_9	Friction_brake_equipment
C_{10}	Braking_System
C_{11}	Interior_equipment
C_{12}	Interior_architecture
C_{13}	Toilet_system
C_{14}	Interiors
C_{15}	Electronic_rear_mirror
C_{16}	Automatic_Train_Operation_unit
C_{17}	Electronic_Train_Control_System
C_{18}	Fault_data_logger
C_{19}	Heritage_Automatic_Train_Protection_unit
C_{20}	System_capture_unit
C_{21}	Train_Control_Management_System
C_{22}	Voice_recorder
C_{23}	On_board_vehicle_control
C_{24}	Billing_System
C_{25}	Central_Passenger_information_System_unit
C_{26}	Driver_Machine_Interface_for_train
C_{27}	Public_Address_System
C_{28}	Safety_Alarm_Systems
C_{29}	Seat_reservation
C_{30}	Passenger_information_System
C_{31}	Traction_motor
C_{32}	Gear_box

C_{33}	Mechanical_transmission
C_{34}	Traction_Control_Unit
C_{35}	Propulsion
C_{36}	Propulsion
C_{37}	Braking_System
C_{38}	Interiors
C_{39}	On_board_vehicle_control
C_{40}	Passenger_information_System
C_{41}	Rolling_Stock_Components