INTEGRATING ONTOLOGIES AND CASE-BASED REASONING FOR THE DEVELOPMENT OF KNOWLEDGE-INTENSIVE INTELLIGENT SYSTEMS.

Hugo Muñoz-Hernández
Rob Vingerhoeds
ISAE-SUPAERO
10 Avenue Edouard Belin 31055 Toulouse, France
hugo.munoz-hernandez@student.isae-supaero.fr
rob.vingerhoeds@isae-supaero.fr

Juan José Montero-Jiménez
TEC-Tecnológico de Costa Rica
Calle 15, Avenida 14
1 km al sur Basílica de los Ángeles
Provincia de Cartago, Cartago, 30101, Costa Rica
juan.montero@itcr.ac.cr

KEYWORDS
Ontology, Case-Based Reasoning (CBR), similarity function.

ABSTRACT

Case-Based Reasoning (CBR) allows emulating the human inference of solutions to problems profiting from previous experience. The integration of CBR with ontologies, structured organization of semantic knowledge, has been in the attention for some time, aiming to create powerful knowledge-intensive systems capable of proposing appropriate solutions to problems. This entails having collected an appropriate number of previous cases as well as having established a suitable ontology for the application domain. This paper focuses on the integration of CBR and ontologies to support the case representation, case base storage, and semantic similarity estimation. Different alternatives for such integration are explored and the approach has been tested in the creation of a Decision-Support System for the design of predictive maintenance systems. This work opens the window to significantly improve the capabilities of a CBR system by using the knowledge materialization and the reasoning features of a specific domain ontology.

INTRODUCTION

Since a long time ago, man has been fascinated with making a machine that had the same capabilities as human beings (think, speak, move, ...). That is the origin of the vast domain of Artificial Intelligence, which definition for this paper is considered as the following:

Artificial intelligence supplies a collection of techniques to manipulate knowledge in such a manner that new results emerge and new inferences that were not explicitly programmed.

Broadly, two areas can be distinguished in AI: symbolic AI and sub-symbolic AI. The latter comprises neural networks, data-driven techniques, that have had a lot of attention since the last 20 years. The former, symbolic AI, concerns knowledge-based systems and may involve techniques such as rule-based, fuzzy logic, Case-Based Reasoning where the knowledge is contained in well-defined blocks of previous experience, etc (Vingerhoeds et al. 1995).

In this paper, the focus is on Case-Based Reasoning and ontologies. Case-Based Reasoning was developed under the philosophy that human beings think and reason using analogies and examples, rather than rules (Kolodner 1993). The idea is that one may recall previous similar situations when being confronted with a new problem. Starting from this previous experience (knowledge) ideas can be derived for addressing the new situation. Case-Based Reasoning is therefore an approach in which specific knowledge of previously experienced problem situations is being used to solve a new problem. This is being done by finding similar previous cases and reusing previous experiences. Allowing continuously to add new cases, new pieces of knowledge, Case-Based Reasoning supports incremental, sustained learning, where information from new situations is kept for future use.

Ontologies are formal explicit descriptions of concepts in a domain of discourse, properties of each concept describing its features, attributes and restrictions (Noy and McGuinness 2000). One of the most common goals in developing ontologies is “sharing a common understanding of the structure information among people and software agents” (Gruber 1993). However, there are also several other motivations to create ontology models such as enabling reuse and analysis of domain knowledge or making domain assumptions explicit. Ontologies are powerful tools for knowledge representation.

In this paper, the integration of ontologies with Case-Based Reasoning is being addressed. The goal of such an integration is to make use of the advantages of both approaches. For example, ontologies enable the processing and sharing of knowledge that can be used at different tasks of Case-Based Reasoning systems, such as representing the input problem, enhancing similarity assessments, case representation, case abstraction and case adaptation.

The paper is organised as follows. In the next section, the context is presented, including an overview of CBR, ontologies, and their integration. After that, another section is focused on explaining the use case for which the integration of CBR and ontologies has been developed. Then, the Development section shows how the integration was carried out, followed by some results and a discussion. The paper concludes by summarizing the lessons learnt and providing indications for future work

CONTEXT

The trigger for the work presented in this paper can be found in the development of a Decision-Support System for the design of predictive maintenance systems (Montero Jimenez et al. 2021). Ontologies have been used as formal knowledge representations that support a CBR system. This section provides the theoretical background of both technologies.

Case-Based Reasoning (CBR)

Case-Based Reasoning consists of proposing solutions to problems within a certain domain profiting of the knowledge derived from previous cases, representing previous knowledge. Indeed, one of the characteristics of human learning is to use past experiences as a reference for the future. A complete reasoning cycle in CBR is divided into four phases(Aamodt and Plaza 1994): retrieval, reuse, revise and retain.

The development of CBR systems starts by defining the case structure, a set of variables that will be used to describe the problem and its corresponding solution from the past. The problem attributes or features are used to estimate the similarity between the target case and the cases stored in a case base. Once the features have been established, the next step is to define similarity functions. These are algorithms, with diverse levels of complexity, that when applied to a pair of values of the corresponding variable they return a similarity value number, normally between 0 and 1. Each problem attribute has its own local similarity. An aggregation function is used to consolidate all local similarities in a global similarity value. Each local similarity can receive special weights that multiply the local similarity values when calculating the global similarity. Case-based reasoning is a flexible reasoning paradigm. It is capable to compute similarities and retrieve similar cases from a case base even when only partial information is known from the target case.

In this work, the software *myCBR*[1] was used. The choice for this software was due to the availability of the *Java* source code, which allowed for adapting the system to the needs of the study. Different types of case attributes for similarity definition are available in *myCBR*, of which three have been used in the current research: *String*, *Symbol* and *Integer*. Both *String* and *Symbol* attributes have textual values, whereas for *Symbol* the values are limited to a list of allowed values and fixed similarities among them, and *String* similarity is based on free text. By default, the *String* similarity function assigns a binary similarity to each string but this function can also include further analysis by applying the *Levenshtein* string comparison method that can tolerate typing errors in the string and still compute a similarity value. The function applied to *Integer* attributes is very flexible, and the user can control the shape of the function $\mathbb{Z} \to \mathbb{R}$. Once the case concept is created and the attributes are defined, a case base can be stored as a list of instances of the concept with values assigned to each one of the attributes. Then, the system will be ready for querying and retrieving cases. A deeper explanation of how to model knowledge in *myCBR* can be found in (Bach et al. 2014).

Ontologies

An ontology defines a set of representational primitives with which to model a domain of knowledge or discourse (Gruber 2009). In a practical sense, ontologies represent a vocabulary of concepts whose basic structure is a hierarchy of classes and subclasses. Instances can be defined as individuals belonging to classes in ontology, so as they will match the common features defining such classes. Important components of ontologies are the *Object Properties* which can establish links between pairs of instances or classes, and *Data Properties*, which can assign information tokens to particular instances. These properties are defined within a certain domain and range, which means that they are restricted, by definition, to particular classes or data types. In general, when referring to ontology entities, that includes classes, individuals and properties. The *Protégé* ontology editor (Stanford University 2016-2020) was used for the development of the ontology for this research.

The initiative of the Semantic Web by the World Wide Web Consortium (OWL Working Group 2012) is to gather as much knowledge as possible into the internet in a format that is readable for computers. For such purposes, the *OWL2* ontology language is used. Standard ontologies like the BFO (Basic Formal Ontology) (International Organization for Standardization 2020) and CCO (Common Core Ontologies) (Rudnicki 2019) are developed and published to be the roots of all the domain-specific ontologies that may be proposed.

Other for the current study relevant aspects of ontologies concern reasoning and queries. A reasoning process can be performed on an ontology to check its logical consistency. This allows verifying, for example, if the instancing of individuals reveals no class contradictions or if the *Object Properties* declarations respect the restrictions of domain and range. Another important aspect of ontology reasoning is the inference of relations; some links could be established between certain entities in the ontology that was not initially explicit but logically deduced from the original ontology

[1]Available online. URL: http://www.mycbr-project.org/. Last visited June 2021.

structure. Various reasoners for *OWL* language exist, amongst which the *HermiT* reasoner (Glimm et al. 2014), which was used for this study. Queries allows extracting information from ontologies; they are questions that can be posed to get a specific information out of the ontology. Three different procedures or languages to query *OWL* ontologies exist: *SPARQL*, *Description Logics (DL)* and *SQWRL*. The *SPARQL* language allow extracting table structured information from an ontology by querying for entities that match certain conditions and have some kind of relation between them. This type of query does not need a reasoning process. A plug-in is available for *Protégé* to execute *SPARQL* queries. In this study, for the *Java* implementation, the *Jena API* has been used. *Description Logics* allow executing simple queries by using a reasoner and checking at first the ontology consistency. They are available by default in *Protégé* and also accessible with the *OWL API* implementation for *Java*. The *SQWRL* language is based on the *SWRL* rule language, and it allows to execute very accurate and specific queries using a reasoner. Again, a *Protégé* plug-in is available. In *Java* it may be used the *SWRL API* with a *drools* reasoning engine implementation available.

An ontology for the selection and assessment of predictive maintenance models was developed to support the CBR approach (Montero Jimenez et al. 2021). The *OMSSA* (Ontology model for Maintenance Strategy Selection and Assessment) includes all the necessary concepts of the maintenance domain to describe predictive maintenance systems architecture and application cases. The ontology was built as an extension of the standard BFO and CCO ontologies in order to be as general as possible and to possibly be re-used in the future by other ontology developers.

Related work: integration CBR-ontologies

Ontologies and Case-Based Reasoning have been increasingly used together in the last decade. One of the most important aspects of developing CBR systems is the vocabulary framework. Ontologies have played an important role in providing this vocabulary framework for several CBR applications. For example, (Qin et al. 2018) implemented and ontology supported case-based reasoning approach for computer-aided tolerance specification. In (Amailef and Lu 2013) an ontology was integrated with a CBR system for emergency response services. (Recio-Garía and Díaz-Agudo 2007) presented a system that brought together the CBR *Java* framework *jCOLIBRI* and *Description Logics* (DL) to calculate a concept-based similarity values between terms according to their position in the classification structure within the ontology. The same framework was used in (Kowalski et al. 2013) for the domain of logistics adding some variables that were defined in a specific domain ontology. In (Yin et al. 2010), CBR and ontologies are used to test a criminal investigation system. An ontology is used as a casebase and structure-based similarity values are calculated for case retrieval.

Instantiated ontologies can be used as case bases for CBR implementations as the case structure can be easily represented in the ontology. Ontologies may also help to compute semantic similarity based on ontological similarity, which can be classified into structure-based similarity and feature-based similarity. Structure-based similarity considers the ontology as a graph where concepts are linked by relations (taxonomic or others). Some methods may be proposed to measure the path distance between a pair of concepts to define their similarity value. For example, in (Avdeenko and Makarova 2018) a hierarchical ontology is used to assign similarity values between some pairs of concepts within a specific domain. Feature-based similarity is focused on comparing sets of features of two different terms, established through their property assertions. Feature-based similarity could provide a deeper and more flexible understanding of the domain vocabulary and thus better results in the semantic similarity computation. That is why in this work a feature-based similarity method is used, see also (Sánchez et al. 2012) and (Bai et al. 2008).

The actual integration of ontologies and CBR does not always receive enough attention in application articles. This paper attempts to cover such gap by providing insight on the different possibilities to integrate ontologies and CBR for the development of knowledge-intensive smart systems.

USE CASE: PREDICTIVE MAINTENANCE SYSTEMS DESIGN

Predictive Maintenance (PdM) is a strategy that aims at triggering maintenance actions based on accurate diagnostics or prognostics before an undesired failure occurs. More specifically, predictive maintenance includes monitoring and modelling the system health, estimating the remaining useful life, and detecting and identifying the actual faults. To perform these tasks there exist several types of models that can be used for diagnostics and prognostics purposes (Montero Jiménez et al. 2020). These models analyze the physical variables measured during the operation of the system of interest. For example, in the case of a turbo-machine, some relevant operation variables to consider for predictive maintenance purposes could be the measurements of combustion temperature, pressure and axis vibrations.

A predictive maintenance approach is normally focused on the health state of the system to be maintained or on the incipient failures that may appear during its functioning, sometimes modelling the evolution of these features over time. For example, a predictive maintenance strategy conceived for health assessment may have as main objective to measure the degradation of the system compared to the optimal operation condition that the system had at the beginning. Another possible task in predictive maintenance is to detect or identify automatically failures that are affecting the system and that might perturb its functioning. In addition, the power of predictive maintenance applications increases when considering the capabilities to forecast the evolution of the system health and even predict an estimation of the remaining useful life.

In order to perform all these tasks, three families of models may be considered: data-driven models, knowledge-based models and physics-based models (Montero Jiménez et al. 2020). Data-driven models have an increased popularity during the last years thanks to the current advances in computational power. Statistical models, stochastic models and machine learning are the main model types within data-driven models. In knowledge-based models, previous experiences are used to infer solutions to current problems, in form of for example rules or cases. Physics-based models are very specific for each application case, as they use a physical simulation of the system to perform predictive maintenance functions with the available data. When developing predictive maintenance systems, one of the challenges lies in the selection of the appropriate model for the knowledge representation (a large amount of options) and to position the retained model(s) for the knowledge representation within the systems architecture (Montero Jiménez and Vingerhoeds 2019).

This work is focused on providing a knowledge-intensive system to find the most adequate predictive maintenance solutions for the particular situations. For this purpose, Case-Based Reasoning appears to a suitable technique, as it allows to deal with symbolic information gathered previous predictive maintenance realisations.

INTEGRATION OF ONTOLOGIES AND CBR FOR THE DEVELOPMENT OF A DECISION SUPPORT SYSTEM

In the current study, a domain ontology (OMSSA) (Montero Jimenez et al. 2021), developed with the ontology editor *Protégé* (Stanford University 2016-2020), was integrated with the *myCBR* engine. An instantiated version of OMSSA serves as case base and provides the knowledge for feature-based similarity measures for the *myCBR* engine. The methodology to estimate feature-based similarity was adopted from (Sánchez et al. 2012). The *OWL API* (Horridge and Bechhofer 2011) is used to develop a direct link between the *myCBR* data files *.prj* and the *.owl* ontologies. The *HermiT* reasoner was used to verify ontology database consistency, infer relations and execute "Description Logic" queries for information extracting, necessary for the feature-based similarity functions. Ontology instantiation is carried out by queries that can retrieve data from different sources, such as for example *.csv* files.

Variables definition for the application case

The application case is a Predictive Maintenance Decision-Support System. In this section, some of the implementation details will be presented, so to show how CBR and ontologies can work together. As mentioned before, a first step to implement a CBR application concerns the definition of the variables describing the cases. In the current study, such cases describe previous solutions of Predictive Maintenance (PdM) that have been successfully implemented on different systems of interest. Based on (Montero Jiménez et al. 2020), the main characteristics of each case were described in a systematic set of data fields. Some of the data fields are used as case retrieval variables. Following the logic of *myCBR*, those variables are attributes that must be included in the concept definition:

- *Task (Symbol)*: what is the specific function of the predictive maintenance system (*e.g.* health modelling).
- *Case study type (Symbol)*: what is the type of maintainable system (*e.g.* a rotary machine).
- *Case study (String)*: specific system to be maintained (*e.g.* jet engines).
- *Input type (Symbol)*: list of measurable physical variables (*e.g.* temperature, pressure and power).
- *Online/Off-line (Symbol)*: predictive maintenance analysis to be done online or offline?
- *Input for the model (Symbol)*: data format type provided to the predictive maintenance module (*e.g.* signal data).
- *Publication year (Integer)*: the year in which the study was published.

The attributes of type *Symbol* take textual values among a list of allowed values that must be specified for each of them. In addition to these data fields that are important for the case retrieval, other variables are considered as solution attributes of the case and may be exploited once the relevant cases are retrieved. Some of the solution attributes suggested by the decision support system include: *study title, publication identifier, predictive maintenance model used, performance indicators, etc.*

Instancing cases in the ontology

Originally, *myCBR* stores the case base in *.csv* tabular format. This *.csv* tabular format was replaced by an instantiated version of OMSSA. A *Java* code was developed with specific methods to read all data and to make it fit in an ontology structure: classes, instances and *Object Properties*/*Data Properties* assertions. An *OWL API* implementation in *Java* was needed for such a purpose. The *CCO* ontologies are taken as a base, especially the *Information Entity Ontology* and the *Artifact Ontology* (Rudnicki 2019). Most of the classes in the ontology are defined as subclasses of the already existing classes in the previously mentioned ontologies. The relations used to form the case structure are those compiled in the Table 1. Properties written in blue are included in *CCO*, those in *green* are included in *OBO Relations Ontology* from *BFO* (International Organization for Standardization 2020) and those in black have been specifically created in *OMSSA*. Parentheses contain the parent ontology, considering that a property can be a sub-property of a parent.

Table 1: *Object Properties* in OMSSA.

Property	**Domain**	**Range**
designates	*Designative Information Content Entity*	No specific range
describes	*Descriptive Information Content Entity*	No specific range
is carrier of	*Independent continuant*	*Generically dependent continuant*
has part	No specific Domain	No specific range
has title (is carrier of)	*Predictive maintenance article*	*Article title*
has identifier (is carrier of)	*Predictive maintenance article*	*Article identifier*
has publication year (is carrier of)	*Predictive maintenance article*	*Publication year*
has synchronization (has quality)	*Predictive maintenance system module*	*Module synchronization*
has predictive maintenance function (has function)	*Predictive maintenance system module*	*Predictive maintenance module function*

The very basic structure of a case starts with an instance of a *Predictive maintenance case* (subclass of *Designative Information Content Entity*), designating an instance of a *Predictive maintenance system module* (subclass of *Information Processing Artifact*). The items to be maintained are instances of the different subclasses in *Maintainable item* (equivalent to *Artifact*). Then, the models are instances of the type subclasses in the *Predictive maintenance model*. The models operate over a set of variables that are recovered from the operation of the item. These variables are instances of *Data variable* (subclass of *Descriptive Information Content Entity*). As the instances of variables are physical concepts (for example, the temperature), they can be reused in more than one case of the database. For each one of the different measurable magnitudes that are mentioned in the database, there exists one single instance of *Data variable*. The functioning of the module and the models used for a certain case are explained in the corresponding *Predictive maintenance article*. The title and the identifier of those articles are assigned to instances *Article title* and *Article identifier* respectively (both are subclasses of *Designative Information Content Entity*). In summary, a maintenance module uses one or more models to perform a predictive maintenance diagnosis or prognostic task on a certain system (*Maintainable item*) based on the data of measured magnitudes recovered during the system operation. This is identified as a predictive maintenance case which is described in a specific research article (see also Figure 1, please note that some elements were omitted for clarity reasons).

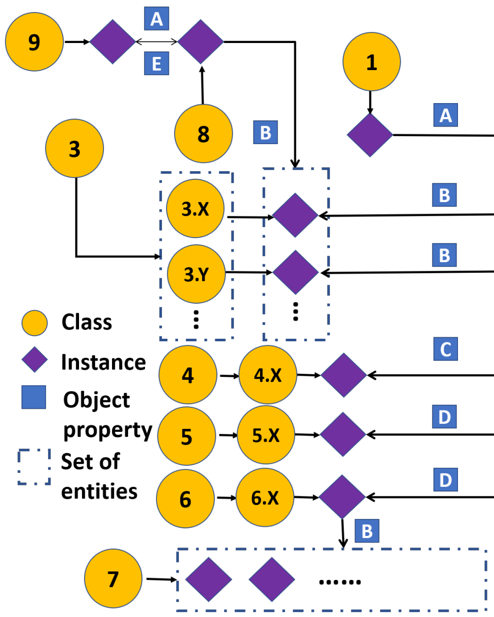

Figure 1: Partial schema of the entities participating in the case representation in the ontology.

- Yellow nodes in Figure 1 represent OMSSA classes:
  1. *Predictive maintenance case*
  2. *Predictive maintenance system module*
  3. *Predictive maintenance model*
  4. *Predictive maintenance function*
  5. *Maintainable item*
  6. *Maintainable item record*
  7. *Data variable*
  8. *Predictive maintenance article*
  9. *Article title*
- The labelled edges in Figure 1 represent OMSSA object properties:
  A. *designates*
  B. *is carrier of*
  C. *has predictive maintenance function*
  D. *has part*
  E. *has title*
- The classes with a tag *N.X* are subclasses of the parent class *N*.

In order to assign the textual values to the article title, the article identifier and the reference tag of the case, which will be materialized as *Designative Information Content Entities* in the ontology, the property *information has text value* is used.

Similarity functions

When defining a similarity function, the type of variable is a major point to consider. The similarity functions of a *Symbol* variable are specified in *myCBR* with a set of pairs of textual values with a similarity value assigned between 0 and 1. These values are estimated according to experimental and physical similarity criteria for the case variables *Case study type*, *Input for the model* and *Online/offline*. One of the objectives of this study is to assess feature-based semantic methods to obtain similarity values, so the procedure proposed (Sánchez et al. 2012) was adapted to the application case and the relations existing in the ontology. The basis of this method is to compare two sets of elements A and B. The following formula is used to obtain a normalized similarity of the sets:

$$Similarity = 1 - \log_2(1 + \frac{A/B + B/A}{A/B + B/A + A \cap B}) \quad (1)$$

In the equation above, A/B means the difference between sets A and B, so the elements from A that do not belong to B. A∩B is the intersection of A and B.

The case feature *Input type* is a list of variables that are monitored during the operation of the system to be maintained. These variables are represented in the ontology with instances of the class *Data variable.* The above-mentioned method is used to compare the set of values in the query case with all cases in the case base. The attribute *Input type* in *myCBR* will store the list of variables as a textual value separated by commas. When executing a query, a list of variables names is proposed, so the algorithm first has to add this list to the allowed values of the attribute. Then, the similarity values are assigned to all the pairs formed by the query string and all the textual values existing in the database. The variable names are separated for each of them and the formula in Equation (1) is applied. Also, a *Levenshtein* method function was implemented to manage slight misspelling in the query string.

Whilst the mathematical method to calculate the ontological similarities for the attribute *Task* is the same as before, the sets of elements to be compared are obtained in a different way. The relation *function uses model* is defined as an inference resulting of the property chain *is predictive maintenance function of* and *is a carrier of* to link instances of functions with instances of models (ontology class *Predictive maintenance model*). This means that the database will be queried automatically for information on the types of models that were used for each type of task concerning predictive maintenance. That allows estimating a semantic similarity between the different subclasses of functions.

The types considered in the ontology are: *Fault detection, Fault feature extraction, Fault identification, Future state forecast, Health assessment, Health modelling*, and *Remaining useful life estimation.* The algorithm will query for each of the mentioned subclasses all the instances of models that are linked to their individuals through the inferred relation *function uses the model.* Each of the instances of models belongs to one of the several subclasses of the *Predictive maintenance model* that were defined in the ontology. The purpose is to list all the model types that have instances linked to any instances of each of the function types.

This is how sets of model types related to functions are compared using the method of Equation (1) to get similarity values. An implementation of the *HermiT* reasoner in *Java* has been used to check the ontology consistency, infer the relations and extract the required information through *Description Logics* queries.

From a similarity point of view, using the variable *Publication year*, referring to the year when a case study was published, the more recent an article is, the more accurate and relevant is supposed to be. A simple *Integer* similarity function has been tested with a similarity of 1 for cases published 5 years ago or less and a constant slope descending to a similarity of 0 at 40 years.

Finally, the default *myCBR* string comparison function with *Levenshtein* method has been used for the *String* attribute *Case study.* To obtain the global similarity value of a case from the similarities of the variables, *myCBR* allows using a simple weighted sum or a euclidean average. The weights may be adjusted, but for this study, they have been set to 1 for all the variables.

Procedure to link ontologies and *myCBR*

The default method for importing databases into *myCBR* is via *.csv* files. As one of the main goals of this project was to create a direct link between ontologies and *myCBR* and use ontologies as database holders, *SPARQL* queries were used to extract the information as organized tabular data and load it automatically into a *myCBR* project file. Using the *Jena API* tool, single string queries with multiple statements were formulated to obtain a query results table with entities names or *Data Properties* content in which the rows correspond to cases and the columns to the variables defining those cases. This table is then stored inside a *myCBR* project file. The procedure may require dividing the query pieces to be executed sequentially when the volume of data is too large, so in that case, the partial results would be simply merged in one data table. The developed application allows loading a case base ready to be used by *myCBR* directly from an ontology .owl file following data translation rules specified in *SPARQL* language.

RESULTS AND DISCUSSION

Keeping in mind the objective to propose an integration procedure between *myCBR* and ontologies, different kinds of queries were tested in *Java* for *OWL* ontologies. *SPARQL* queries seem to be best placed for such integration, having as major advantages to having a fast execution without the using of a reasoner and that the outputs obtained are immediately organized in tabular data. This makes it easy to introduce the data in the *myCBR* project files.

For the information extraction for semantic similarity values definition, *SQWRL* queries were tested, but the obtained performance was low in terms of calculation speed and memory consumption. For the reasoner implementation tested for *SQWRL* queries, it was found that again a slow process resulted, potentially due to the complexity of the case base. In the end, in this study, *Description Logics* queries were used, only requiring to run the reasoner once for the execution of series of queries.

A feature-based similarity method has been tested in two of the variables describing the study case: *Input type* and *Task*. For the case retrieval variable *Input type*, the feature-based method seemed to be suitable criteria to compare sets of elements, suggested in (Qin et al. 2018). The in this way obtained results have been shown to be relevant and useful, demonstrating that the integration procedure is practical for defining accurate similarity values automatically. As to the similarity values between pairs of predictive maintenance tasks or functions, the corresponding similarity values matrix is shown in Table 2, of which the letters should be read considering:

A) *Fault detection*

B) *Fault feature extraction*

C) *Fault identification*

D) *One step future state forecast*

E) *Multiple steps future state forecast*

F) *Health assessment*

G) *Health modelling*

H) *Remaining useful life estimation*

Table 2: Similarity values matrix for predictive maintenance functions.

	A	**B**	**C**	**D**	**E**	**F**	**G**	**H**
A	1	0.033	0.225	0.02	0.03	0.093	0.079	0.063
B	0.033	1	0.018	0	0	0.02	0.033	0.023
C	0.225	0.018	1	0	0	0.082	0.07	0.073
D	0.02	0	0	1	0.061	0.025	0	0.034
E	0.03	0	0	0.061	1	0.037	0.015	0.03
F	0.093	0.02	0.082	0.025	0.037	1	0.13	0.076
G	0.079	0.033	0.07	0	0.015	0.13	1	0.078
H	0.063	0.023	0.073	0.034	0.03	0.076	0.078	1

As can be seen, the similarity values for non-identical concepts are overall very low meaning that the instances in OMSSA seldom show a model that has been used for two different tasks. Note the existence of one pair (*Fault detection-Fault identification*) that is much higher (over 10% similarity) in comparison to the rest; this is expected as classification models can be used for fault detection and also for fault identification purposes. Another interesting result concerns the pair *One step future step forecast-Multiple steps future state forecast*, basically the latter being an extension of the former, where the similarity could be expected to be close to 1. Both concepts are subclasses of the same predictive maintenance function in the ontology *Future state forecast*. Hence, from a global point of view, the analysis of the results suggest that there is possibly some aspect about the data or the similarity definition that makes it difficult to obtain really meaningful values. A first explanation hypothesis concerns the high degree of specialization of predictive maintenance models in the ontology. The class *Predictive maintenance models* have been automatically filled up with as many subclasses as specific types of models were reported in the literature and therewith declared in the database. Grouping all those types of models into less specific types could have changed the outcome. Another option would be to use a bigger case base, which in itself goes against the generally assumed guidelines for Case-Based Reasoning to work with relatively small case bases. This needs to be investigated in the next steps of the study.

So, the results suggest that some modifications could be needed, specifically in what concerns the similarity between the types of predictive maintenance models considered. However, the value of what has been obtained is not limited to the results in themselves, but also the fact that the integration procedure for ontologies and CBR has been validated so as to be exploited and improved to achieve a superior level of performance for the decision support system.

CONCLUSION AND FUTURE WORK

This paper presents an approach to integrate ontologies and Case-Based Reasoning for a practical application case base for the development of predictive maintenance systems. The automatic transmission of the data in tabular format to ontologies has shown satisfactory results. The implementation of an ontology reasoner and some semantic feature-based similarity methods using *Description Logics* was possible thanks to the *OWL API*. It helped to improve the performances and accuracy of the system for this application and others to be tested in the future. The integration of *myCBR* and *.owl* ontologies through *SPARQL* queries using the *Jena API* is functional and reusable for other applications

This study has especially focused on the retrieval of cases, but the use of ontological knowledge could also be powerful to improve the rest of CBR functions as well. There may be good potential for future work to consider the complete CBR cycle and to automatize all CBR tasks (as suggested in (Mántaras et al. 2005)).

In addition, the study to improve ontological similarity is interesting; more complex and accurate semantic similarity functions could be implemented. In particular, exploiting the ontological reasoner inference could help to make deeper use of the knowledge in the ontology. Another potential improvement may come from taxonomic similarity functions that take better advantage of the hierarchical structure of an ontology. Additional validation work should aim for testing the validity of the recommendations given by the system within the application domain at hand (the design of predictive maintenance system) and to extend with developments on other application domains.

REFERENCES

Aamodt A. and Plaza E., 1994. *Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. AI Communications IOS Press*, Vol. 7: 1, pp. 39–59.

Amailef K. and Lu J., 2013. *Ontology-supported case-based reasoning approach for intelligent m-Government emergency response services. Decision Support Systems*, 55, 79–97. ISSN 01679236. doi:10.1016/j.dss.2012.12.034.

Avdeenko T.V. and Makarova E.S., 2018. *Knowledge Representation Model Based on Case-Based Reasoning and the Domain Ontology: Application to the IT Consultation. IFAC-PapersOnLine*, 51, no. 11, 1218–1223. ISSN 2405-8963. doi:https://doi.org/10.1016/j.ifacol.2018.08.424. URL `https://www.sciencedirect.com/science/article/pii/S2405896318315519`. 16th IFAC Symposium on Information Control Problems in Manufacturing INCOM 2018.

Bach K.; Sauer C.; Althoff K.D.; and Roth-Berghofer T., 2014. *Knowledge Modeling with the Open Source Tool myCBR.* vol. 1289.

Bai Y.; Yang J.; and Qiu Y., 2008. *OntoCBR: Ontology-based CBR in context-aware applications.* ISBN 978-0-7695-3134-2, 164–169. doi:10.1109/MUE.2008.56.

Glimm B.; Horrocks I.; Motik B.; Stoilos G.; and Wang Z., 2014. *Hermit: An Owl 2 Reasoner. Journal of Automated Reasoning*, 53. doi:10.1007/s10817-014-9305-1.

Gruber T., 2009. *Ontology*, Springer US, Boston, MA. ISBN 978-0-387-39940-9, 1963–1965. doi:10.1007/978-0-387-39940-9_1318.

Gruber T.R., 1993. *A translation approach to portable ontology specifications. Knowledge Acquisition*, 5, no. 2, 199–220. ISSN 10428143. doi:10.1006/knac.1993.1008.

Horridge M. and Bechhofer S., 2011. *The owl api: A java api for owl ontologies. Semantic Web*, 2, 11–21. doi:10.3233/SW-2011-0025.

International Organization for Standardization, 2020. *ISO/IEC FDIS 21838-2.2 Information technology — Top-level ontologies (TLO) — Part 2: Basic Formal Ontology (BFO).* Available online. URL: https://www.iso.org/standard/74572.html.

Kolodner J., 1993. *Case based reasoning.* Morgan Kaufmann. ISBN 978-1558602373.

Kowalski M.; Klüpfel H.; Zelewski S.; and Bergenrodt D., 2013. *Integration of Case-Based and Ontology-Based Reasoning for the Intelligent Reuse of Project-Related Knowledge.* ISBN 978-3642328374, 289–299. doi:10.1007/978-3-642-32838-1_31.

Montero Jimenez J.J.; Vingerhoeds R.; and Grabot B., 2021. *Enhancing predictive maintenance architecture process by using ontology-enabled Case-Based Reasoning.* IEEE ISSE 2021.

Montero Jiménez J.J.; Sebastien S.; Vingerhoeds R.; Grabot B.; and Salaün M., 2020. *Towards multi-model approaches to predictive maintenance: A systematic literature survey on diagnostics and prognostics. Journal of Manufacturing Systems*, 56, 539–557. doi:10.1016/j.jmsy.2020.07.008.

Montero Jiménez J.J. and Vingerhoeds R., 2019. *A System Engineering Approach to Predictive Maintenance Systems: from needs and desires to logical architecture.* 1–8. doi:10.1109/ISSE46696.2019.8984559.

Mántaras R.; Mcsherry D.; Bridge D.; Leake D.; Smyth B.; Craw S.; Faltings B.; Maher M.; Cox M.; Keane M.; Aamodt A.; and Watson I., 2005. *Retrieval, reuse, revision and retention in case-based reasoning. Knowledge Eng Review*, 20, 215–240. doi:10.1017/S0269888906000646.

Noy N.F. and McGuinness D.L., 2000. *Ontology Development 101: A Guide to Creating Your First Ontology.* Stanford University. Available online. URL: `https://protege.stanford.edu/publications/ontology_development/ontology101-noy-mcguinness.html` . Accessed June 2021.

OWL Working Group, 2012. *OWL 2 Web Ontology Language.* World Wide Web Consortium. Available online. URL: http://www.w3.org/TR/2012/REC-owl2-overview-20121211/. Accessed March 2021.

Qin Y.; Lu W.; Qi Q.; Liu X.; Huang M.; Scott P.; and Jiang X., 2018. *Towards an ontology-supported case-based reasoning approach for computer-aided tolerance specification. Knowledge-Based Systems*, 141, 129–147. doi:10.1016/j.knosys.2017.11.013.

Recio-Garía J. and Díaz-Agudo B., 2007. *Ontology based CBR with jCOLIBRI.* ISBN 978-1-84628-665-0, 149–162. doi:10.1007/978-1-84628-666-7_12.

Rudnicki R., 2019. *An Overview of the Common Core Ontologies.* CUBRC Inc, Buffalo, NY. Available online. URL: `https://www.nist.gov/system/files/documents/2019/05/30/nist-ai-rfi-cubrc_inc_004.pdf`.

Stanford University, 2016-2020. *Protégé official website.* Available online. URL: https://protege.stanford.edu/. Last visited June 2021.

Sánchez D.; Batet M.; Isern D.; and Valls A., 2012. *Ontology-based semantic similarity: A new feature-based approach. Expert Systems with Applications*, 39, no. 9, 7718–7728. ISSN 0957-4174. doi:https://doi.org/10.1016/j.eswa.2012.01.082.

Vingerhoeds R.A.; Janssens P.; Netten B.D.; and Aznar Fernández-Montesinos M., 1995. *Enhancing off-line and on-line condition monitoring and fault diagnosis. Control Engineering Practice*, 3, no. 11, 1515–1528. ISSN 09670661. doi:10.1016/0967-0661(95)00162-N.

Yin Z.; Gao Y.; and Chen B., 2010. *On development of supplementary Criminal analysis system based on CBR and Ontology.* In *2010 International Conference on Computer Application and System Modeling (ICCASM 2010).* vol. 14, V14–653–V14–655. doi:10.1109/ICCASM.2010.5622227.