An ontology-based approach to support the knowledge management of software quality standards

Una propuesta basada en ontologías para apoyar la gestión del conocimiento de estándares de calidad de software

Nemury Silega[1], Gilberto F. Castro Aguilar[2,3], Inelda Martillo Alcívar[3], Katya M. Faggioni[3], Yuri I. Rogozov[1], and Vyacheslav S. Lapshin[1]

***Abstract* — Nowadays, the quality of software systems is crucial for companies to provide high-quality services and products. However, a wide number of software projects still fail. To increase the success probability of projects, it is suitable to adopting software quality standards to guide the process. However, standards are commonly described by means of natural language making difficult its analysis. For example, it is not easy to choose the most suitable standard according to the characteristics of a project. Furthermore, the usage of natural language hinders the automatic detection of inconsistencies and ambiguities. On the other hand, ontologies are an artificial intelligence technique that has been successfully used to represent and analyze knowledge in numerous domains because its capacity to enable the automatic validation and consistency checking of the represented information. This paper aims to present an ontology-based approach to describe and analyze software quality standards. Since this ontology can represent the knowledge of several standards, a reasoner may automatically validate the information and infer new knowledge. This ontology might support the reduction of conceptual ambiguity of standards descriptions and improve its understanding.**

***Keywords* - quality standards; ontology; software; knowledge representation.**

***Resumen* — En la actualidad la calidad de los sistemas de software es crucial para que las empresas brinden servicios y productos de alta calidad. Sin embargo, un número importante de proyectos de software aún fallan. Para aumentar la probabilidad de éxito de los proyectos, es conveniente adoptar estándares de calidad de software que guíen el proceso. Sin embargo, los estándares se describen comúnmente por medio del lenguaje natural, lo que dificulta su análisis. Por ejemplo, no es fácil elegir el estándar más adecuado según las características de un proyecto. Además, el uso del lenguaje natural dificulta la detección automática de inconsistencias y ambigüedades. Por otro lado, las ontologías son una técnica de inteligencia artificial que se ha utilizado con éxito para representar y analizar el conocimiento en numerosos dominios debido a su capacidad para permitir la validación automática y la comprobación de la consistencia de la información representada. Este artículo tiene como objetivo describir una propuesta basada en ontologías para describir y analizar estándares de calidad del software. Dado que esta ontología puede representar el conocimiento de varios estándares, un razonador puede validar automáticamente la información e inferir nuevo conocimiento. Esta ontología podría apoyar en la reducción de la ambigüedad conceptual de las descripciones de estándares y mejorar su comprensión.**

***Palabras Clave* - estándares de calidad; ontología; software; representación del conocimiento.**

This research is supported by a postdoc fellowship granted by the Institute of Computer Technologies and Information Security, Southern Federal University, project № PD/20-02-KT. (Corresponding author: Nemury Silega).

[1]Department of System Analysis and Telecommunications, Southern Federal University, Taganrog, Russia (silega@sfedu.ru, yrogozov(@)sfedu.ru, lapshin@sfedu.ru). ORCID numbers: 0000-0002-8436-5650, 0000-0002-9125-7826, 0000-0001-5266-5854.

[2]Facultad de Ingeniería, Universidad Católica de Santiago de Guayaquil, Guayaquil, Ecuador (gilberto.castro@cu.ucsg.edu.ec). ORCID number: 0000-0001-9050-8550.

[3]Facultad de Ciencias Matemáticas y Físicas, Universidad de Guayaquil, Guayaquil, Ecuador (gilberto.castroa@ug.edu.ec, Inelda.martilloa@ug.edu.ec, katya.faggionic@ug.edu.ec). ORCID numbers: 0000-0001-9050-8550, 0000-0001-6810-9668, 0000-0001-6577-5761.

Manuscript Received: February 14, 2023.
Revised: May 22, 2023.
Accepted: June 05, 2023.
DOI: https://doi.org/10.29019/enfoqueute.946

I. INTRODUCTION

NOWADAYS, the number and complexity of software projects are growing more and more. However, it is noteworthy the number of projects that ar e not successful in terms of time, cost, and software quality. For example, according to the Chaos report in 2020, only 35% of projects were fully successful in terms of time and budget [1]. The adoption of good practices to carry out the software development process is a suitable alternative to increase the success probability. The good practices include the application of guidelines to carry out each stage of the software lifecycle as well as the application of international standards. These standards are internationally agreed by experts and are promoted by important organizations in the software industry. For example, the International Organization for Standardization (ISO), together with the International Electrotechnical Commission (IEC), provide some of the most important international standards. Likewise, the Software Engineering Institute (SEI) [2] and the Institute of Electrical and Electrotonic Engineering (IEEE) are two relevant organizations in the defini-

tion of international standards. A standard is a document established by an authority, custom, or general consent as a model or an example [3].

In spite of the positive impact of adopting standards, it is not a straightforward task. The standards are usually described in complex and large documents represented by natural language. Hence, the analysis of these documents is a complex task, especially if several standards are being analyzed. For example, it is possible to find different terms for the same concept or the same term for different concepts for different standards. These issues hinder the right application of the standards; for instance, it is not easy to choose the proper standard for a project according to its characteristics.

On the other hand, applying formal models to partially describe the knowledge of the standards might be an alternative to tackle the aforementioned issues. In that sense, the ontologies, represented by means of Ontology Web Language (OWL), are an artificial intelligence technique that has been used to represent and analyze knowledge in different domains [4-6]. The ontologies allow to represent the knowledge of a domain and support the reasoning tasks on the concepts [7]. They also contribute to a shared understanding among different agents, such as people and software systems [8].

The main problem addressed by this research is related to how to increase the quality of descriptions about quality standards and consequently increase their accuracy and understandability as well as to reduce the time needed to get relevant information. To deal with this problem and considering the advantages of ontologies, this paper aims to describe an ontology-based approach to representing and analyzing the knowledge related to software development standards. The ontology was elaborated following a sound methodology that includes seven stages. The execution of these stages allowed us to define the classes, properties, and individuals of our ontology. In this process we analyzed a set of international standards to create a general ontology that could be used to represent information from different sources. Since several standards will be represented by means of the same ontology, a reasoner could analyze it to automatically validate the information and infer new knowledge. To demonstrate the applicability of the ontology, we populated it with knowledge of two standards. This ontology is a feasible tool to support the unification, integration, and reduction of conceptual ambiguity of software standards descriptions. Therefore, it might contribute to improving the accuracy of these descriptions and consequently make it easier the understanding of software standards.

Besides, once this ontology is populated it may be considered a knowledge base. Hence, it could be queried to carry out intelligent searches according to the user's needs. For example, a user could query this knowledge base to know the most suitable standard taking into account the characteristics of his project. Furthermore, since the knowledge will be explicitly described, the adoption of this type of approach could contribute to avoid misunderstanding and enhance the commutation of all stakeholders. On the other hand, with the systematic application of this ontology it will be possible to gather information regarding the results of applying a specific standard in a project. Therefore, the users could analyze all this information and use it to make decisions.

The structure of the paper is as follows: section 2 briefly analyzes some elements about standards and ontologies. In section 3 the main components of our ontology are described and a case study to demonstrate the applicability of our approach is presented. Finally, conclusions and future work are introduced.

II. MATERIALS AND METHODS

To perform this research, several research methods were used. First of all, a survey was conducted to identify the most common quality standards in the domain of software development. Based on the results of this survey, we carried out a documentary analysis to know the main components of the most adopted standards. These results were used as an insight to create the ontology that is presented in this article. To validate the ontology, a case study was implemented. For this case study, we used ontology to represent the knowledge of several quality standards. This case study demonstrated the applicability of the proposal to represent knowledge and to make easier the analysis of quality standards.

In addition to these research methods, we describe in this section the particular methodology, language and tool that were adopted to create the ontology. These elements are key to ensure the quality of the ontology from early stages of the development process.

A. Survey to identify Standards for the software development process

A survey was conducted to identify the most representative standards. In this study, software developers of two software development companies participated. We asked the participants to mention the best standards for the software development process. ISO and Capacity Maturity Model Integrated (CMMI) got the best results. Based on these results, we carried out an analysis of these two standards to know their main components. These results were used to create the ontology that will be described above in this paper.

Software quality standards include specifications to ensure that the outputs of the software development process meet business expectations. This guides the appropriate application of software engineering [9]. Hence, it plays a crucial role in managing and ensuring software quality. The standards focus either on the process or the product [10].

ISO 9001:2015 has been widely implemented for quality management around the world. It includes a set of principles to the right implementation of a system of quality management in organizations that develop software or provide services [11]. CMMI is based on Capacity Maturity Model (CMM). The aim of CMMI is to assess and enhance the maturity of the processes of software development. The Guide to the Project Management Body of Knowledge (PMBOK) provides guidelines for managing projects. PMBOK describes the projects lifecycle and their processes [3].

B. Documentary analysis

As Andrade et al defined, a Documentary analysis is a procedure which encompasses the identification, verification and consideration of documents which are related to the object investigated [12]. In our research, we are interested in considering documents which describe standards for the software development process. Since we previously conducted a survey to identify the most used standards for software development, we focused the analysis on those selected standards. Specifically, we used the normative documents of the respective standards.

This analysis allowed us to understand better the knowledge related to these standards. By means of this technique, the general structure of each standard was identified, as well as their main components. These results were an important insight to develop the ontology that is presented in this paper. Likewise, we used the results of this study to populate the ontology and evaluate its quality.

C. Case Study

To carry out the case study we followed the steps defined by Crowe et al [13]: *Defining the case; Selecting the cases; Collecting the data*; and *Analyzing, interpreting and reporting case studies*. In our research, we defined our cases as the descriptions of software quality standards that were specified by means of the ontology developed in this research. Our focus was on how a formal description of standards can contribute to improving the accuracy of these descriptions and, consequently, make it easier the understanding of software standards. Based on the results of the survey mentioned above, we focused our study on the standards CMMI, ISO and PMBOK.

Regarding the data collection, first of all, we used the official descriptions of these standards [2, 3, 11]. This collected information was used in the process of creating the ontology as well as to populate it. Finally, the step of analyzing the results was guided by a set of competence questions. We evaluated how the ontology was able to answer these questions. *Section 3. C* describes the main results of this case study.

D. Methodology and tools adopted to develop the Ontology

As was mentioned in the introduction, this paper aims to describe an ontology-based approach to representing and analyzing the knowledge related to software development standards. An ontology is a formal and explicit specification of a shared conceptualization [14]. It is composed of concepts, axioms, or inference rules that can be used to infer new knowledge. The ontologies contribute to detecting and removing ambiguities [15]. They are a tool to manage the knowledge of a specific domain, enhancing the understanding of the specifications and creating new knowledge.

A crucial step to ensure the quality of an ontology is the selection of the methodology that will guide the ontology development process. Likewise, it is relevant to select the right language and the tool to implement the ontology.

The development of our ontology was guided by the methodology proposed by Noy and McGuinness, which has been extensively adopted [16]. This methodology is one of the most used or cited for designing an ontology [17]. It includes the following steps: Determine the domain and scope of the ontology, consider reusing existing ontologies, list the relevant terms of the ontology, define the classes and the class hierarchy, define the properties (called relationships or slots) of the classes, define facets and/or restrictions on slots or relationships and finally define instances.

Web Ontology Language (OWL) is one of the most prominent languages to represent ontologies. OWL allows to describe concepts and relationships among them [18]. To create and edit the ontology we employed the tool Protégé because it is an open-source platform that has been used extensively to manage ontologies in OWL [19].

III. RESULTS AND DISCUSSION

A. Ontology to represent the knowledge of software quality standards

Following the steps of the methodology described in the previous section, an ontology to represent and analyze the knowledge of software standards was developed. Below the main results of the otology development process are analyzed.

Step 1. Define the domain and scope of the ontology.

The main goal of the ontology is to support the analysis of software standards. Its application will contribute to enhancing the understanding of a specific software standard as well as finding common concepts and terms among different standards. To assess the compliance of these objectives, the following competence questions (CQs) were defined:

CQ 1. What are the main components and subcomponents of a specific standard?
CQ 2. What goals and practices are satisfied for a company that reached a certain maturity level?
CQ 3. What process areas should be implemented in an organization to reach the next maturity level?
CQ 4. What companies have a level that is not according to the practices or goals that they accomplish?
CQ 5. What specifications are common in different standards?

CQ 1 is focused on the concepts that compose each standard. It has been used to represent information about CMMI_DEV version 1.3 and PMBOK 6th edition to illustrate the applicability of our approach. Hence, CQ 1 will be answered with specific information about these standards. CQ 2, CQ 3 and CQ 4 focus on the information that can be inferred in an organization that implements a standard, in this case, CMMI_DEV. CQ 5 will allow to know the concepts in a standard which have a correspondent concept in other standards.

Step 2. Consider reusing existing ontologies

Since we did not find ontologies in the English language, we do not reuse ontological resources from other ontologies.

Step 3: List the relevant terms of the ontology.

To identify relevant terms, we conducted a documentary analysis of several references that describe standards. Some relevant terms are Process, Area, Practice, Goal, Tool, Level, Input, Output, among others.

Step 4: Define the classes and the class hierarchy

In this stage, we analyzed the identified terms in the previous stage to decide which of them will be considered as classes in ontology. After this analysis, 48 classes were identified. The classes **Standard** and **Part** are two of the most significant classes in the ontological model. Each of these classes subsumes other classes to characterize the individuals that compose it and thus provide analysis of interest. Fig. 1 depicts the hierarchy of the classes **Part** and **Standard**. We defined that a **Standard** is composed of a set of **Main_Components**, and each **Main_Component** is composed of a set of **Sub_Components**. The types of **Main_Component** and **Subcomponent** depend on the Standard. For example, **Knowledge_Are**a and **Process_Area** are the main components of the standards PMBOK_Edition_7th and CMMI_Dev_1.3, respectively. Whereas a **Knowledge_Area** includes a set of **Processes**, a **Process_Area** is composed of **Goals** and **Practices**. On the other hand, we included classes to classify a Standard according to its purpose. Thus, we have the classes **Standard_Focused_On_Process**, **Standard_Focused_On_Product** and S**tandard_Focused_On_Product&Process**.

Fig. 1. Hierarchy of the classes **Part** and **Standard**.

Step 5: Define the properties

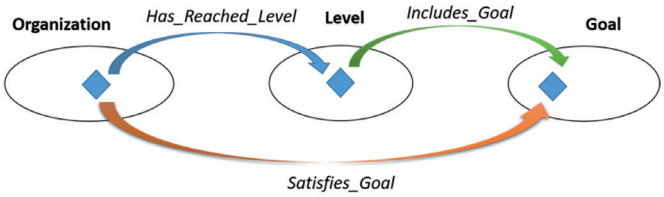
The properties are the other key component in an ontology. OWL defines two types of properties, *object* and *data properties*. An object property allows representing a relationship between two individuals. In OWL, to define the classes that can use a property, the domain and range of the property should be defined. For example, an individual of the class **Standard** *Has_Main_Part* individuals of the class **Part**. For this example, the property *Has_Main_Part* has the classes **Standard** and **Part** as domain and range, respectively. Table 1 shows some of the object properties for the classes **Standard**, **Process_Area**, **Goal**, and **Organization**. In total, the ontology includes 67 object properties. Furthermore, the expressive richness of OWL allows us to specify an inverse for each property. For example, the property *Supports_Generic_Practice* has the inverse property *Is_Supported_By*. If **A** *Supports_Generic_Practice* **P**, then a reasoner will infer that **P** *Is_Supported_By* **A**.

TABLE I
A SAMPLE OF OBJECT PROPERTIES

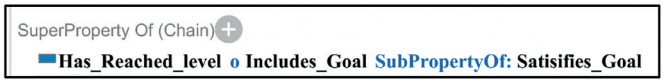
Domain	**Property**	**Range**
Standard	*Has_Part*	
Has_Main_Component	*Part*	
Standard	*Defines_Level*	Level
Process_Area	*Is_Part_Of*	Standard
Process_Area	*Has_Specific_Goals*	Goal
Process_Area	*Is_Associated_To_Level*	Level
Process_Area	*Supports_Generic_Practice*	Generic_Practice
Specific_Goal	*Has_Specific_Practice*	Specific_Practice
Organization	*Has_Reached_Level*	Level
Organization	*Satisfies_Goal*	Goal
Organization	*Satisfies*	Practice
Organization	*Must_Implent_Area*	Process_Area

In OWL, it is possible to specify the characteristics of an object property. For example, to answer CQ 5 we created the object property *Is_Compatible_With* to relate the elements of different standards that are similar. This property was defined as symmetric, which means that if f A *Is_Compatible_With* B then the reasoner infers that B *Is_Compatible_With* A.

The expressive richness of OWL allows to represent not only direct relationships between individuals; it is possible represent more complex relationships by means of property chains. For example, if a company has adopted the **Standard** CMMI_DEV and has reached the maturity level 4 (*Quantitatively managed*), we would like to know the goals and practices that this company satisfies. Fig. 2 a) shows a representation of the relationship among the classes **Organization**, **Level**, and **Goal**. Whereas Fig. 2 b) shows how this relationship was expressed by means of a property chain (*Satisfies_Goal*). A similar property chain was defined to represent the relationship between **Organization** and **Practice** by means of the object property *Satisfies_Practice*.



(a)



(b)

Fig. 2. (a) Example of relationships among classes; (b) Example of property chain *Satisfies_Goal*.

On the other hand, data properties allow to define some basic attributes for the classes. For example, Table II shows some of the data properties identified for the classes **Standard** and **Organization**. Similarly, to object properties, data properties have domain and range, but in this case, the range is a data type, for example, String, Integer, Date, etc.

TABLE II
A SAMPLE OF DATA PROPERTIES

Domain	Property	Range
Standard Organization	*Has_Name* *Has_Description*	String
Standard	*Has_Publication_Date* *Has_Expiration_Date*	Date
Standard	*Has_Version*	Standard
Organization	*Must_Implent_Area*	Process_Area

Step 6: Define facets and/or restrictions

OWL allows specifying universal and existential restrictions. For example, Fig. 3 shows the existential restriction to express that a **Standard** *Has_Main_Component* some instances of the class **Main_Component**. It means that each Standard should have at least one **Main_Component**. Whereas Fig. 3 also depicts a universal restriction (identified for the word *only*). With this statement, if a Standard S is related to an individual X by means of the property *Has_Main_Component*, the reasoner will classify **X** as a **Main_Component**.

Fig. 3. Example of existential and universal restrictions.

On the other hand, OWL allows defining sets of necessary and sufficient conditions for a specific class. The classes with this type of conditions are named defined classes. The main advantage of this type of classes is that a reasoner can automatically infer the individuals that belong to them. To take advantage of these potentialities, several defined classes were created. For example, we created the class **Organization_Level_4** for the organizations that adopted the **Standard** CMMI_DEV and have reached the maturity level 4 (*Quantitatively managed*). Fig. 4 shows this specification.

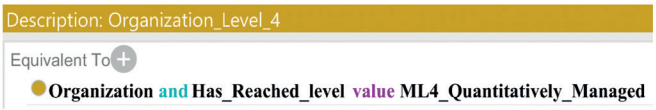


Fig. 4. Example of a defined class.

We used Semantic Web Rule Language (SWRL) to specify more complex relations. For example, to answer CQ 5 we created the Class **Company_With_Problems** and specified Rule 1 to infer the companies that belong to this class.

Rule 1. Rule in SWRL to detect companies with problems

Step 7. Define instances

In this step, instances of each class are defined, and the necessary axioms to link them are established. A case study with results of this step is explained in the next section. Furthermore, this case study demonstrates the applicability and usefulness of this ontology.

B. Ontology validation

To validate the ontology, we checked that *a) it meets the specifications of a formal-logical system*; and *b) it satisfies the requirements for which it was created*. A reasoner is used to verify that the specifications as a formal-logical system are fulfilled. In this case, we used the reasoner Pellet [20], which confirmed that the ontology is consistent. The ontology evaluation is an iterative and progressive process. Throughout the life cycle of ontology development, we continually use the reasoner to verify the consistency of the ontology.

In addition, the last version of the ontology was tested by using OntOlogy Pitfall Scanner (OOPS!) [21]. After four iterations, all the problems detected by OOPS were fixed. This evaluation helped to detect and correct some pitfalls in our ontology.

We carried out a case study to demonstrate that the ontology satisfies the requirements for which it was created. In this case study, we verified that all competency questions were answered correctly by the ontology. The case study described in the next section also illustrates the usefulness and applicability of our ontology.

C. Case Study

A case study is presented below to demonstrate the applicability and usefulness of the ontology. As mentioned before, we presented in our ontology the information about the standard

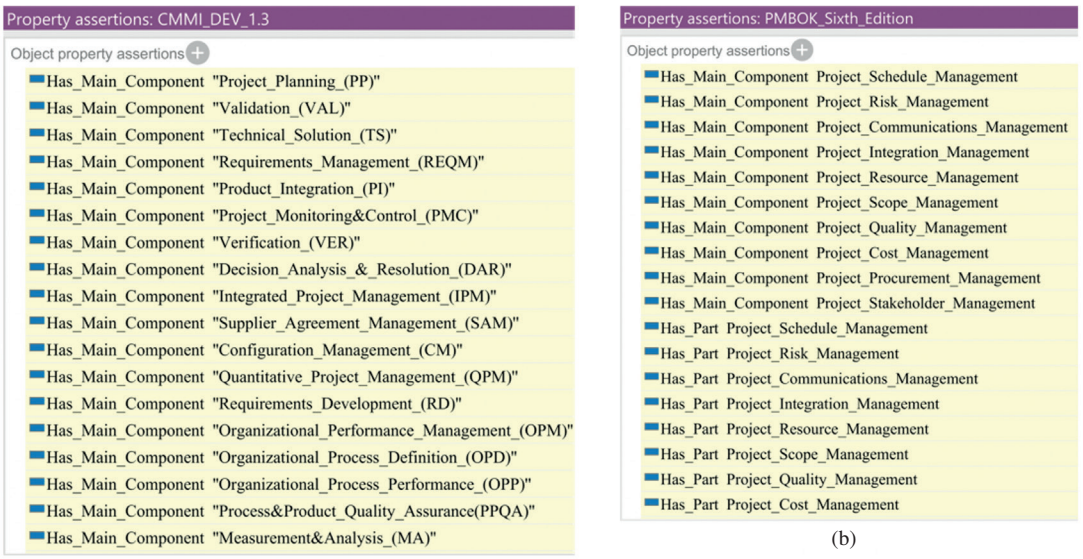


Fig. 5. Inference of the main components of (a) CMMI_Dev_3.1; and (b) PMBOK_6th_Edition.

CMMI_Dev version 3.1 and the guide PMBOK 6th edition. In spite of the fact that PMBOK is not a standard, it has been extensively adopted to guide the development of different types of projects [22]. We created the individuals PMBOK_6th_Edition and CMMI_Dev_3.1 as instances of the class **Standard**. Then the respective axioms to represent the particular information of each standard were created. For example, the main components of *PMBOK_6th_Edition* are instances of the class **Knowledge_Area**, whiles for *CMMI_Dev_3.1* are instances of the class **Process_Area**. To answer CQ 1, Fig. 5 shows the main components of the Standards *CMMI_Dev_3.1* and *PMBOK_6th_Edition*. Considering these statements, the reasoner will classify the instances of these two classes as **Main_Component**.

In addition to know the main components of a standard, it is useful to know about the other subcomponents. For example, in *PMBOK_6th_Edition*, each process belongs to a group of processes. Hence, it is interesting to know the list of processes that belong to a specific group of processes. Fig. 6 shows the processes that belong to the **Group_of_Processes** *Monitoring&Controlling_Process_Group*.

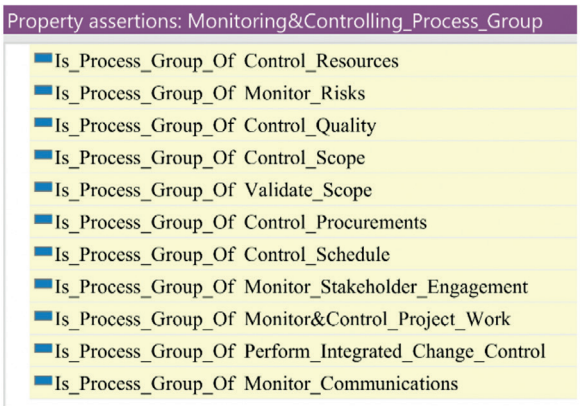


Fig. 6. Processes that belong to the **Group_of_Processes** *Monitoring&Controlling_Process_Group*

Since we defined a set of property chains, it is possible to get information about individuals who do not have direct relationships as well. For example, it is possible to know the inputs or outputs of a specific **Group_Of_Process** or a **Knowledge_Area**. Fig. 7 a) shows the inputs/outputs of the **Knowledge_Are**a *Poject_Quality_Management*, and Fig. 7 b) shows the inputs/outputs of the **Group_Of_Process** *Executing_Process_Group*. Likewise, taking into account the specifications of the ontology, it is possible to know the list of processes where certain work document is used. These examples illustrate the usefulness of the expressive richness of OWL. The application of this ontology could speed up the analysis of the standards.

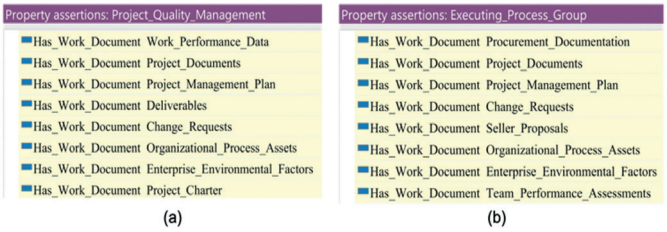


Fig. 7. List of inputs/outputs of (a) the Knowledge_Area Poject_Quality_Management; and (b) the Group_Of_Process Executing_Process_Group.

To exemplify how the ontology is able to answer CQ 2 and CQ 3, we added the individuals *Company_Soft&Serv_For_Health* and *Software_Company_Multi_System* as instances of the class **Organization**. For Software_Company_Multi_System, we added the axiom to define that this company reached maturity level 5 (*ML5_Optimizing* in the ontology). Fig. 8 a) shows that the reasoner inferred the goals and practices that this company satisfies. For *Company_Soft&Serv_For_Health*, we added the axiom to define that reached the level 4 (*ML4_Quantitatively_Managed* in the ontology). Fig. 8 b) shows that the reasoner inferred that this company must implement the process areas *Organizational_Performance_Management_(OPM)* and

Causal_Analysis_&_Resolution_(CAR) to reach the next level. The answers to CQ 2 and CQ 3 may be a useful insight to decisions according to a company's level.

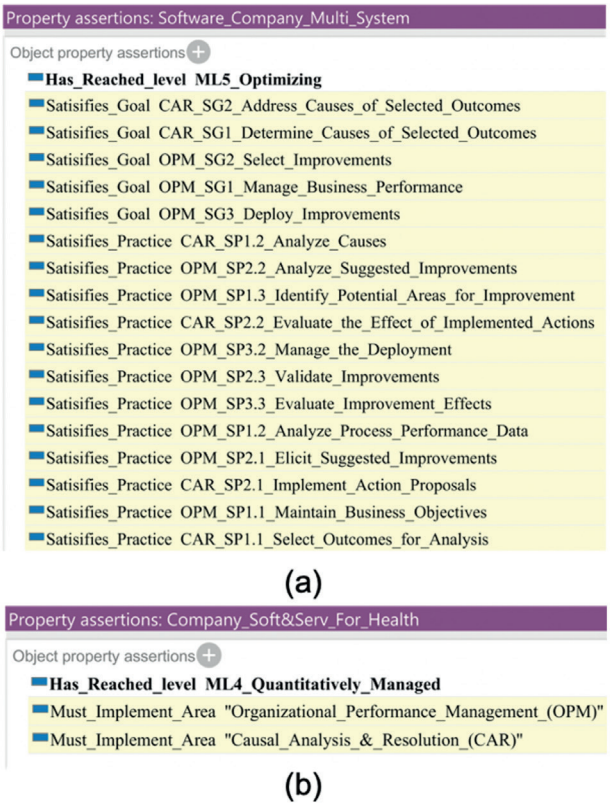


Fig. 8. (a) Inference of the goals and practices that a company satisfies; (b) Inference of the process areas that a company should implement.

To answer CQ 4, we specified that the company *Software_Company_Multi_System* does not satisfy the goal *OPM_SG1_Manage_Business_Performance*, which is a goal that must satisfy the companies that reached level 5. Since we previously defined that this company reached level 5, the reasoner classified this company as *Company_With_Problems*, as Fig. 9 shows. This type of inference is useful to detect inconsistencies due to either human mistakes adding to the information or real problems with the companies that are not implementing goals or practices that they should implement. Specifically, this type of analysis is common after a company has been assessed.

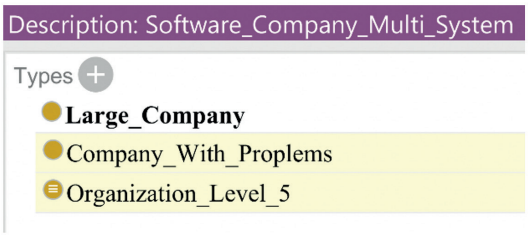


Fig. 9. Automatic classification of a *Company_With_Problems*

To illustrate how CQ 5 is answered, we created in the ontology the class **Component_Multi_Standard**. Furthermore, we defined the necessary and sufficient conditions to automatically classify the instances of this class (Fig. 10 a). Fig. 10 b) shows that the reasoner identified a group of individuals that belong to the class **Component_Multi_Standard**. For example, as Fig. 10 d) shows, *Project_Quality_Management* is Compatible_With *Process&Product_Quality_Assurance(PPQA)*. Since the property *Compatible_With* was defined as symmetric, the reasoner inferred that *Process&Product_Quality_Assurance(PPQA)* is *Compatible_With Project_Quality_Management* as Fig. 10 c) shows.

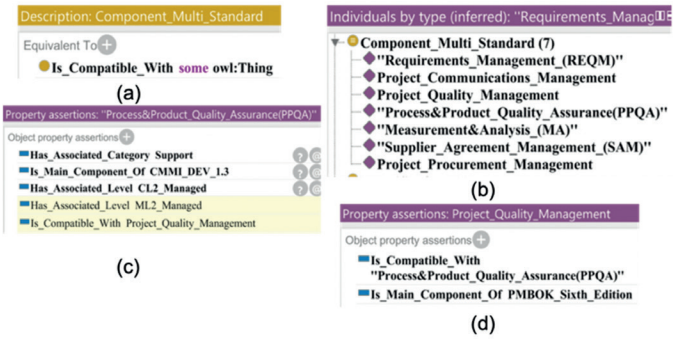


Fig. 10. (a) Definition of the class *Component_Multi_Standard*; (b) Inference of instances of the class *Component_Multi_Standard*; (c) Example of *Component_Multi_Standard*; (d)Example of *Component_Multi_Standard*

IV. CONCUSSIONS AND FUTURE WORK

This work presented an ontological approach to representing and analyzing the knowledge of different standards. We described how the expressive richness of OWL was exploited to represent a wide diversity of relationships. The ontology includes specifications to represent not only the specific information of a standard, but it is possible to relate this information with the data of the organizations and identify common concepts among different standards. We demonstrated by means of a case study that this approach could support the analysis of different standards. This ontological model could be a useful tool to speed up the adoption of a standard. Furthermore, the ontology might be used as a valuable material to train company personal staff. On the other hand, since OWL is a formal language, this ontology allows detecting inconsistencies or ambiguities. The systematic application of this approach will help to populate the ontology. Hence this ontology could represent a reference knowledge base.

Our future work will be focused on the extension of the ontology to represent the information of other standards and identify the common concepts among different standards. Furthermore, we are designing an experiment to assess the impact of this approach to improve the analysis of these standards in terms of time and quality of the analysis. In addition, we are examining approaches to automatically extract information from documents in natural language. With the application of this type of approach, the population of the ontology could be easier. Finally, since we expect to populate this ontology with the information of a significant number of standards and companies, we are exploring alternatives to address its scalability.

REFERENCES

[1] J. Johnson, "CHAOS 2020: Beyond Infinity Report," Standish Group, 2020. Available: https://www.standishgroup.com/news/49.

[2] C. M. University. "Software Engineering Institute.", 2023. Available: https://www.sei.cmu.edu/.

[3] P. M. Institute, A guide to the project management body of knowledge (PMBOK guide). Sixth edition (PMBOK guide). Project Management Institute, 2017. Available: https://www.pmi.org/pmbok-guide-standards/foundational/pmbok.

[4] A. Berko et al., "Application of ontologies and meta-models for dynamic integration of weakly structured data," in *2020 IEEE Third International Conference on Data Stream Mining & Processing (DSMP)*, 2020: IEEE, pp. 432-437, Available: https://ieeexplore.ieee.org/document/9204321.

[5] P. Manika, E. Xhumari, A. Ktona, and A. Demiri, "Application of Ontologies and Semantic Web Technologies in the Field of Medicine," in RTA-CSIT, 2018, pp. 24-30. Available: http://ceur-ws.org/Vol-2280/paper-04.pdf.

[6] Z. Zhang and X. Zhang, "The application of ontologies on knowledge sharing in software development projects," in *Frontiers in Enterprise Integration: CRC Press*, 2020, pp. 335-340, Available: https://www.taylorfrancis.com/books/edit/10.1201/9781003061090/frontiers-enterprise-integration-li-xu.

[7] M. Bhatia, A. Kumar, and R. Beniwal, "Ontologies for software engineering: Past, present and future," *Indian Journal of Science and Technology*, vol. 9, no. 9, pp. 1-16, 2016, Available: https://indjst.org/download-article.php?Article_Unique_Id=INDJST5833&Full_Text_Pdf_Download=True.

[8] L. Yang, K. Cormicana, and M. Yub, "Ontology-based systems engineering: A state-of-the-art review," *Computers in Industry*, vol. 111, pp. 148–171, 2019, Available: https://www.sciencedirect.com/science/article/abs/pii/S0166361518307887.

[9] S. P. Roger and R. M. Bruce, Software engineering: a practitioner's approach. McGraw-Hill Education, 2015. ISBN: 9780073523293. Available: https://dl.acm.org/doi/book/10.5555/1593888.

[10] I. Sommerville, Software engineering—9th ed. Addison-Wesley, 2011, pp. 978-0. SBN-10: 137035152. Available: https://www.academia.edu/58171756/Software_Engineering_9th _by_Ian_Sommerville.

[11] L. Fonseca and J. P. Domingues, "ISO 9001: 2015 edition-management, quality and value," International journal of quality research, vol. 1, no. 11, pp. 149-158, 2017. Available: http://hdl.handle.net/10400.22/9677.

[12] S. Andrade, M. D. Schmitt, B. C. Storck, T. Piccoli, and A. B. Ruoff, "Documentary analysis in nursing theses: data collection techniques and research methods," *Cogitare Enferm*, vol. 23, no. 1, p. e53598, 2018, Available: https://revistas.ufpr.br/cogitare/article/view/53598/pdf_en.

[13] S. Crowe, K. Cresswell, A. Robertson, G. Huby, A. Avery, and A. Sheikh, "The case study approach," *BMC medical research methodology*, vol. 11, no. 1, pp. 1-9, 2011, Available: https://bmcmedresmethodol.biomedcentral.com/counter/pdf/10.1186/1471-2288-11-100.pdf.

[14] T. R. Gruber, "A Translation Approach to Portable Ontology Specifications," Knowledge Acquisition, 1993, Available: https://www.sciencedirect.com/science/article/abs/pii/S1042814383710083.

[15] N. O. Bajnaid, "An ontological approach to model software quality assurance knowledge domain," Ph.D. dissertation, London Metropolitan University, 2013. Available: https://repository.londonmet.ac.uk/id/eprint/7427.

[16] M. C. Suárez-Figueroa, A. Gómez-Pérez, E. Motta, and A. Gangemi, Ontology engineering in a networked world. Springer-Verlag Berlin Heidelberg. 2012, pp. 1-435, Available: https://link.springer.com/content/pdf/10.1007/978-3-642-24794-1.pdf.

[17] A. Sattar, E. S. M. Surin, M. N. Ahmad, M. Ahmad, and A. K. Mahmood, "Comparative analysis of methodologies for domain ontology development: A systematic review," *Int. J. Adv. Comput. Sci. Appl.*, vol. 11, no. 5, pp. 99-108, 2020, Available: https://thesai.org/Downloads/Volume11No5/Paper_15-Comparative_Analysis_of_Methodologies.pdf.

[18] M. Keet, An introduction to ontology engineering. Maria Keet Cape Town, 2018. Available: http://repository.aust.edu.ng/xmlui/handle/11427/28312.

[19] M. Horridge, "A Practical Guide To Building OWL Ontologies Using Protégé 4 and CO-ODE Tools Edition 1.2," The University Of Manchester, Manchester, 2009. Available: https://www.academia.edu/download/56546069/ProtegeOWLTutorialP4_v1_3.pdf.

[20] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz, "Pellet: A practical owl-dl reasoner," *Journal of Web Semantics*, vol. 5, no. 2, pp. 51-53, 2007, Available: https://www.sciencedirect.com/science/article/abs/pii/S1570826807000169.

[21] M. Poveda-Villalón, A. Gómez-Pérez, and M. C. Suárez-Figueroa, "Oops!(ontology pitfall scanner!): An on-line tool for ontology evaluation," *International Journal on Semantic Web and Information Systems (IJSWIS)*, vol. 10, no. 2, pp. 7-34, 2014, Available: https://www.igi-global.com/gateway/article/full-text-pdf/116450.

[22] J. Čabarkapa, "Analysis and comparison of ISO 21500-Guidance on project management and PMBOK 6th Guide," in *5th IPMA SENET Project Management Conference*. Paris, Francia: Atlantis Press, 2019, pp. 266-271, Available: https://www.atlantis-press.com/article/125925995.pdf.