IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 25, 2020, accepted May 6, 2020, date of publication May 11, 2020, date of current version May 26, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2993838

Domain Ontology for Requirements Classification in Requirements Engineering Context

HALA ALRUMAIH[1,2], ABDULRAHMAN MIRZA[1], AND HESSAH ALSALAMAH[1,3]

[1]Information Systems Department, College of Computer and Information Sciences, King Saud University, Riyadh 12372, Saudi Arabia
[2]Information Systems Department, College of Computer and Information Sciences, Al Imam Mohammad Ibn Saud Islamic University, Riyadh 11432, Saudi Arabia
[3]Computer Engineering Department, College of Engineering and Architecture, Al Yamamah University, Riyadh 13541, Saudi Arabia

Corresponding author: Hala Alrumaih (haalrumaih@imamu.edu.sa)

ABSTRACT Research in recent years has shown a merge between the areas of requirements engineering and semantic technologies. With the release of the semantic concept and the progress of semantic technologies, the opportunities for applying ontologies as a means to define information and knowledge semantics have become increasingly accepted in different domains. Concurrently, the implementation of most requirements classification techniques does not handle the semantic aspects of requirements. If the meaning of requirements and their relations can be handled, software developers can obtain more effective requirement classifications to produce requirements specifications of higher quality. In this study, a domain ontology is proposed to present a requirements classification technique that can be used to share and describe different classifications. The proposed ontology is built using a systematic method based on Methontology and it is implemented using Protégé. The developed ontology was successfully evaluated using validation and verification tests. The validation test included the evaluation of content and competency questions, while the verification test included the evaluation of taxonomy and the implementation of the FOCA method. The proposed ontology may represent a significant contribution to ontology libraries. In addition, this ontology can be used in several ways to increase the quality of software requirements specification documents. It could also ensure consistency between requirements, and facilitate communication between requirements engineers owing to the use of same terminologies for various software applications.

INDEX TERMS Requirements classification, requirements engineering, ontology.

I. INTRODUCTION

Ontologies are used in artificial intelligence, software engineering, medical informatics, library science, enterprise bookmarking, and information architecture as a form of knowledge representation. According to one of the most cited definitions of the semantic web literature, an ontology is ''a formal, explicit specification of a shared conceptualization, which includes terms and concepts that exist in a given domain, their properties, and the relationships between them.'' [1]

There are many benefits of using an ontology in a project. Ontology aims to capture the static domain knowledge in a generic way [2]. It provides a universally agreed upon understanding of a domain, which may be reused and shared across applications and groups. Ontology defines the classes of things that exist in an application domain and associates a precise definition with each concept and relationship type that is used [3]. It facilitates the computational understanding and seamless interoperability between people and organizations [4]. It allows key concepts and terms relevant to a given domain to be identified and defined in an open and unambiguous way [4]. Ontology facilitates the use and exchange of data, information, and knowledge between people and organizations, aiming towards intelligent system interoperability. It is used as a solution to resolve the semantic conflicts between data sources [5]. Finally, ontology improves design processes by building a knowledge base for guiding the design processes.

Recently, ontologies have played an important role in software engineering processes. Software engineering scientists provide clear preferences and classifications for the requirements in the system by using several requirements classification techniques. There are various useful classifications

The associate editor coordinating the review of this manuscript and approving it for publication was Chang Choi.

techniques dedicated to many pioneers in the field of requirements engineering (RE), such as Sommerville's technique [6]. This diversity exists owing to the principles and basics of software requirements, depending on the requirements description approaches, and the design and architecture approaches being applied for the developed software systems. However, one of the issues of these techniques is that the implementation of most classification techniques does not handle the semantic aspects of requirements. Accurate requirements classification is always a serious issue in any project's success. Research works [7]–[9] have demonstrated the dangers created by working with incorrect methods to classify requirements. These dangers can involve failure due to projects going over time or budget.

This study aims to build a domain ontology to present a requirements classification technique in the RE context, which is called a requirements classification ontology (RCO). RCO is useful for sharing and describing the different classifications of requirements. The proposed ontology is built using a systematic method based on Methontology and it is implemented using Protégé. The developed ontology was successfully evaluated using validation and verification tests. First, the validation test involves the evaluation of content and competency questions. Second, the verification test covers taxonomy evaluation and the FOCA evaluation method.

The remainder of the paper is organized as follows. Section II presents related research, and Section III details the different phases of the RCO development process. Then, Section IV demonstrates the evaluation of the RCO. The results are briefly discussed in Section V. Finally, Section VI concludes the paper and discusses potential future work.

II. RELATED RESEARCH

This section analyzes previous research that built ontologies for requirements classification in the RE context in two different destinations. The first destination is the ontology libraries and the second one is the research papers related to building an ontology for classifying requirements in the RE context.

The first step involves checking the semantic search engines and the ontology libraries to determine the existence of an ontology related to the same domain. Protege [10], DAML [11] ontology libraries, and Swoogle semantic web search engine are the most popular examples; they were examined in this work, but none of them returned a result regarding ontologies for requirements classification in the RE context.

In contrast, the literature review revealed fewer works that focused on building an ontology for classifying requirements in the RE context, as shown below.

Avdeenko and Pustovalova [12] proposed a hybrid model based on production rules and the ontology structure. This model supports the RE process and attempts to satisfy the whole set of properties, including completeness, correctness, unambiguity, consistency, and traceability. The classes of the developed ontology represented the requirements types. The developed ontology helps in enhancing the traceability between its concepts and the elements of software requirements specification documents, but the ontology missed many important domain concepts.

Odeh and Odeh [13] developed a semantic framework using an ontology to classify non-functional requirements (NFRs) in relation to software engineering. The ontology was built based on the Sommerville classification. The developed framework was used as a source for sharing the understanding of NFRs. This study focused on only one type of requirements, i.e., NFRs, while neglecting the other type, i.e., functional requirements (FRs). In addition, the developed ontology was very abstract because some important concepts were not included.

Considering the field of risk analysis, Lasheras *et al.* [14] introduced a framework to present and share security requirements. The ontology was built based on the IEEE standards. The main goal of this framework was to detect the incompleteness and inconsistency in requirements. This study focused only on security requirements; therefore, the ontology proposed in this study cannot be used for requirements classification in the RE context.

To improve semantic tool support for the RE domain, Rashwan *et al.* [15] developed a new classification algorithm to automatically categorize NFRs in software specifications. The authors used ontology notation to automatically convert software requirements documents into a semantic representation. This approach is useful for managing the cost of the software system and measuring the quality of written requirements. However, this study focused only on NFRs, while neglecting the FRs.

Li and Chen [16] suggested an ontology-based methodology to automatically classify the security requirements. They built their ontology after making a comprehensive survey in the field of study. They manually collected a preliminary list of security keywords and linguistic features that are usually used to identify security requirements. This study only considered security requirements; therefore, the ontology proposed in this study cannot be used to classify all types of requirements in the RE context.

There are some works that presented automated ontology building methods that were applied to the software requirements. Sitthithanasakul and Choosri [17] developed an ontology to enhance the RE processes in Agile software development cycle. They used a semi-automated extractor to build the developed ontology called OntoLT. It is a useful plug-in in Protégé tool which is used for building and maintaining the structures of domain ontologies. The developed ontology used to ensure processes consistency and facilitate communication between requirements engineers because of using the same terminology between the engineers for various software applications. In addition, the ontology provides all necessary concepts and relations which cover the different RE processes that can be separated into five phases: requirements elicitation, requirements analysis, requirements documentation, requirements validation, and requirements management. This ontology offered a high-level overview about the RE

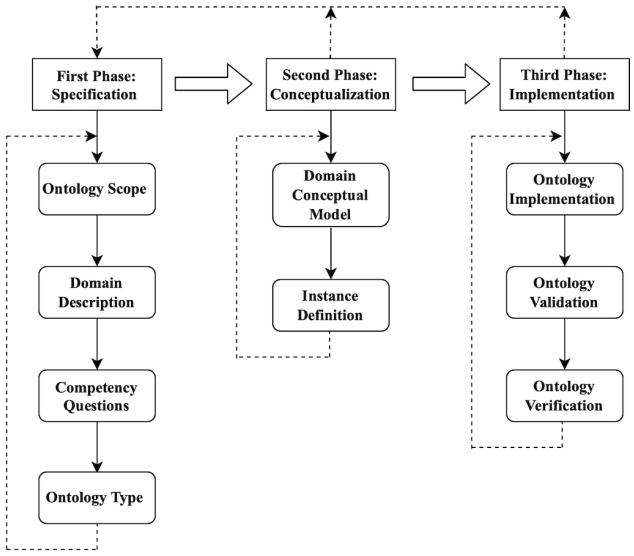

FIGURE 1. Ontology development process.

processes and it lacks required details about each process. PremaLatha *et al.* [18] presented a methodology for automatic ontology generation in the field of software development processes. The authors used the formal concept analysis and natural language processing to automatically build the developed ontology.

Additionally, all previous ontologies were not evaluated during their development process. This challenges the readiness of these ontologies for use in real-world applications. Accordingly, there is a need to build an ontology with more details for requirements classification in the RE context.

III. RCO DEVELOPMENT PROCESS

There are many available methods for developing ontologies, but there is no uniquely correct method. Brusa *et al.* [19] classified these methods into two collections. The first group comprises the experience-based methods, and the second group includes the evaluative prototypes methods, such as Methontology.

This work adopts the same method that was presented in [19] to develop ontologies. The method is based mainly on Methontology, which supports the development of ontology at the knowledge level. In terms of IEEE standards, Methontology is also considered to be the most accurate development method. In general, the selected method divides the development process into three main phases according to the Methontology framework, i.e., specification, conceptualization, and implementation, as shown in Fig. 1. The figure shows the iterations of these phases. The objective of the specification phase is to collect knowledge about the domain, while the objective of the conceptualization phase is to give this knowledge a structure by using external representations.

A. FIRST PHASE: SPECIFICATION

1) ONTOLOGY SCOPE

Initially, it is important to describe the scope of the ontology developed in this work. The scope specifies the boundaries of the developed ontology by determining what should be included or excluded. This work aims to design and implement an ontology for requirements classification that can be used for sharing and describing the different classifications of requirements in the RE context. It does not include other activities of requirements analysis in the RE context.

2) DOMAIN DESCRIPTION

The domain must be explained and investigated to obtain the most required knowledge to build an RCO. As mentioned in [20], there are three main requirements classification techniques, which were developed by three pioneers in the field of RE. In the Sommerville classification technique, all software system requirements are classified into FR and NFRs. FRs are expressed as interactions between the system sections and their surroundings, while NFRs are defined as constraints on the services that can be offered. The Lauesen classification technique is based on what should be included in the requirement specifications and the functions of the software. In contrast, the Wiegers classification technique depends on classifying requirements by hierarchy. The use of a single technique for requirements classification often fails to satisfy the system demands [12]. There is a need to merge two or more of these techniques or support the most comprehensive technique by an international standard to complete each other.

The Sommerville classification technique reflects the business needs as stated by different stakeholders. Analysts, designers, programmers, and other stakeholders would benefit more from the Sommerville classification technique than from other classification techniques because in this technique, the FRs and NFRs are differentiated; thus, different stakeholders together can manage the resource time, cost, and humans. In addition, the structure of the Sommerville classification technique enables the extension of the proposed classification both vertically and horizontally to include all types of requirements. Consequently, this technique may be considered the most comprehensive classification technique in terms of including different levels of requirements.

The proposed requirements classification technique represented in Fig. 2 was inspired by the Sommerville classification technique and ISO/IEC 25000 series SQuaRE (System and software product Quality Requirements and Evaluation) [21]. SQuaRE is an international standard for creating a framework for the evaluation of software product quality. The standard describes the characteristics and sub-characteristics of the external quality and internal quality of the software product.

In addition to the elements of Sommerville classification in [20], the proposed classification technique describes how FRs can be derived from the data and procedural requirements as follows:

- Data requirements specify the data that should be input and output by a system, and the data that should be stored internally by a system.

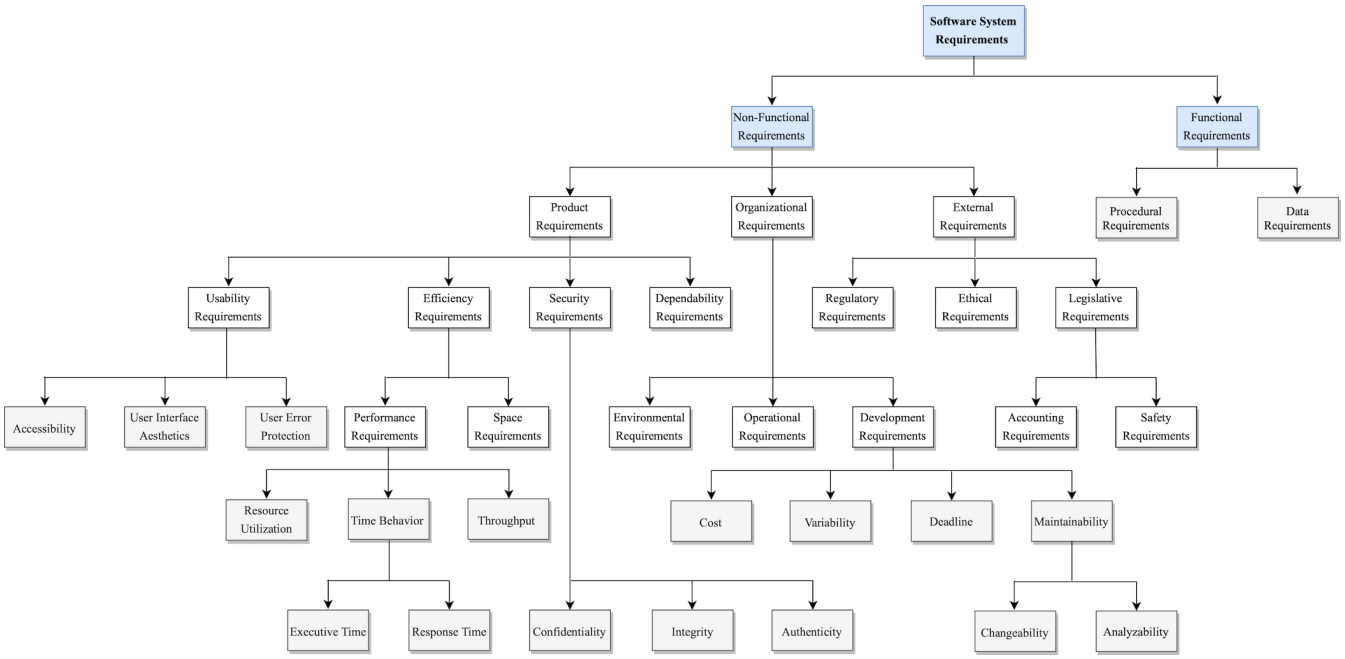

FIGURE 2. Proposed requirements classification technique.

- Procedural requirements explain the functions of a system, i.e., how it records, computes, transforms, and transmits data.

The technique also classifies the performance requirements into the following:

- Time-behavior requirements, which explain the response and processing (execution) time required for a system to perform its functions.
- Resource utilization requirements, which define the amount and types of resources used by a system to perform its functions.
- Throughput requirements, which describe the throughput rates of a system to perform its functions.

Furthermore, the classification technique classifies the usability requirements as follows:

- User error protection requirements, which define how a system protects users against making errors.
- User interface aesthetics requirements, which describe how a user interface enables satisfying interaction for the user.
- Accessibility requirements, which explain how a system can be used by people with different characteristics and abilities to achieve an indicated goal in a specified environment.

The technique classifies the security requirements into the following:

- Confidentiality requirements, which describe how a system confirms that data are accessible by an authorized user.
- Integrity requirements, which explain how a system prevents an unauthorized user from accessing or modifying data.
- Authenticity requirements, which define how a system can prove the identity of a subject or resource.

In addition, the proposed classification technique describes how the development requirements can be derived from the cost, variability, deadline, and maintainability requirements as follows:

- Cost requirements describe the requested cost to build a specified system to achieve user needs.
- Variability requirements explain the ability of a system to be configured or customized for use in a particular environment.
- Deadline requirements define the last date to deliver the components of a system.
- Maintainability requirements illustrate the probability of a system to be restored after a failure occurs within a specified time. Maintainability covers the capability of a system to undergo changes and the ability of a system to be diagnosed for failure causes.

3) COMPETENCY QUESTIONS

Writing several competency questions (CQs) is an important step, especially in the specification phase [22]. These questions help in understanding the scope of the developed ontology. In addition, they play an important role in the validation phase of the developed ontology. CQs represent a set of questions that should be answered by the knowledge base built from the developed ontology. A subset of CQs that reflects the main concepts of the RCO is presented in Table 1.

TABLE 1. Subset of CQs of the RCO.

No.	Competency Questions (CQs)
CQ 1.	What type of requirements represent how a system should react to particular inputs?
CQ 2.	What type of requirements represent a constraint on the services or functions offered by a system?
CQ 3.	What type of requirements specify the data that should be input and output by a system, and what data should a system store internally?
CQ 4.	What type of requirements explain the functions of a system, how it records, computes, transforms, and transmits data?
CQ 5.	What type of requirements arise from the external aspects of a system and its improvement stages?
CQ 6.	What type of requirements define the system requirements derived from the rules and processes in the organization of stakeholders?

TABLE 2. Part of the data dictionary of RCO.

Key Term	Description	Type
Functional requirements	Requirements represent how a system should react to particular inputs.	concept
Non-functional requirements	Requirements represent a constraint on the services or functions offered by a system.	concept
Data requirements	Requirements specify the data that should be input and output by a system, and the data that should be stored internally by a system.	concept
Procedural requirements	Requirements explain the functions of a system, how it records, computes, transforms, and transmits data.	concept
External requirements	Requirements arise from the external aspects of a system and its improvement stages.	concept
Organizational requirements	Requirements define system requirements derived from the rules and processes in the organization of stakeholders.	concept

4) ONTOLOGY TYPE

The ontology type can be either of the following categories, i.e., upper ontologies, mid-level ontologies, domain (low-level) ontologies, or application ontologies [23]. Domain ontologies generally describe the glossary related to a particular domain within a specific context. The RCO can be considered to be a domain ontology because it can be used for sharing and describing the different classifications of requirements in the RE context.

B. SECOND PHASE: CONCEPTUALIZATION

1) DOMAIN CONCEPTUAL MODEL

In this step, all concepts, relations, and attributes are obtained from a comprehensive study of different requirements classification techniques in the field of RE. In addition, several international standards concerned with the quality of the developed software product were scanned to obtain an integrated and comprehensive idea of the domain. Consequently, a key term list was formulated and a data dictionary was created for the key terms in the field. Table 2 presents a part of the data dictionary and provides descriptions for some concepts in the domain.

TABLE 3. Excerpt of RCO instance table.

Instance	Concept	Attribute	Value
Req_5	Procedural Requirements	Req_id	5
		Description	User should be able to enter personal data
Req_26	Cost Requirements	Req_id	26
		Description	The retail cost of the software must be between 200 and 400 SR
Req_43	Security Requirements	Req_id	43
		Description	Passwords shall never be viewable at the point of entry.

Next, a UML (Unified Modeling Language) class diagram was utilized to supply a better understanding of the conceptual aspects in the domain; the diagram is considered to be a base for building the glossary of the ontology terms. The relations in the diagram are based on the generalization and specialization relations to present the different classifications of requirements in the RE context. Fig. 3 shows the UML domain model.

2) INSTANCE DEFINITION

According to Methontology, the last step in the conceptualization phase is the definition of instances for the ontology classes. Table 3 provides some examples of the RCO instances and their concepts, attributes, and values, if available.

C. THIRD PHASE: IMPLEMENTATION

1) ONTOLOGY IMPLEMENTATION

This phase includes transforming the RCO into the reality by using some development tools. OWL was used to represent the RCO owing to its powerful performance. Furthermore, the Protégé tool was used to implement the RCO because it is an open-source and free tool. Several plug-ins were created for Protégé and the architecture of the tool was designed to ensure that it can easily build and integrate extensions. Protégé also helps in verifying and validating the developed ontology. The developed RCO consists of 40 axioms allocated as 38 classes and 2 data properties. Fig. 4 displays the hierarchy of concepts in the RCO using 'is-a' relations in Protégé.

IV. RCO EVALUATION

Ontology evaluation plays an important role in every phase of the ontology development process and between the phases as well. This step ensures that the developed ontology is correct and it is ready to be used in real-world applications.

Ontology evaluation process is divided into validation and verification tests [24]. Ontology validation examines the developed ontology to determine whether the correct ontology has been developed. In contrast, ontology verification

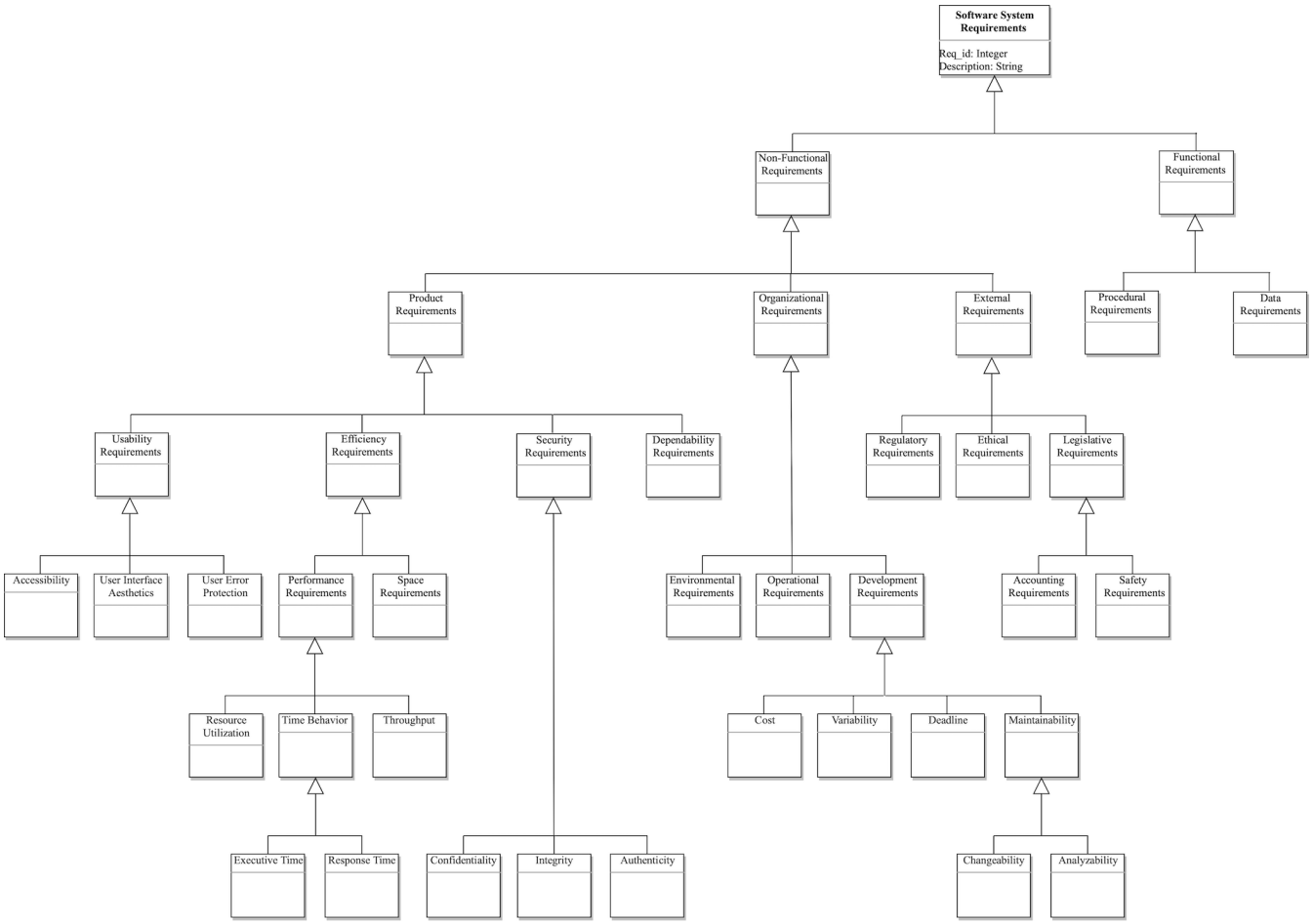

FIGURE 3. RCO domain model in UML.

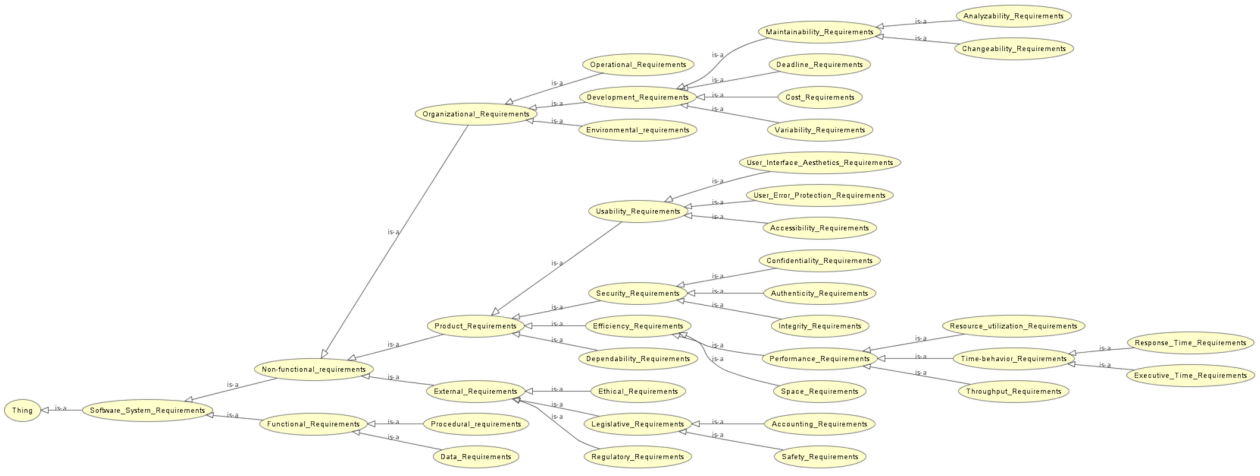

FIGURE 4. "Is-A" relations between RCO concepts.

examines the developed ontology to determine whether the ontology has been developed correctly. It ensures that the ontology was designed according to specific quality standards defined for ontologies.

Two approaches can be used to implement the validation process. The first approach evaluates the ontology content, while the second approach requires answering CQs that were defined in the first phase in Table 1. Two approaches

TABLE 4. Content evaluation of RCO.

Criteria	Application Results	Explanation
Consistency	Yes	All concepts in the RCO are consistent because they contain no contradictory sentences considering other concepts and axioms inside the RCO. In addition, the reasoner Pellet 1.5.2 shows no error in testing concept consistency.
Completeness	Yes	The RCO reflects an integrated image for the scope and domain that were specified in the first phase.
Conciseness	Yes	The RCO is free of any needless concepts or redundancies between the concepts.
Expandability	Yes	The RCO is scalable. Adding new concepts requires no significant changes.
Sensitiveness	Yes	Any small change in the RCO is not observant for the current concepts.

can also be used to implement the verification process. The first approach evaluates the ontology taxonomy, while the second approach requires the implementation of the FOCA methodology.

A. ONTOLOGY VALIDATION

The first approach in the validation process evaluates the ontology content. This approach is based on the criteria used in [24]. The criteria, the application results on the RCO, and their explanations are summarized in Table 4.

Protégé also helps in validating the developed ontology using the reasoner plug-in. The reasoner Pellet 1.5.2 provides a method to test the consistency of the developed ontology with respect to the different relationships between classes, instances with their properties, characteristics, and constraints. The result of this consistency check was: No inconsistencies were identified in the proposed RCO.

The second approach in the validation process requires answering the CQs that were defined in the first phase in Table 1. The set of CQs in Table 1 that reflect the main concepts of the RCO helps in RCO validation. Table 5 presents the answers to all CQs.

B. ONTOLOGY VERIFICATION

The first approach in the verification process evaluates the ontology taxonomy. A set of main criteria mentioned in [25] was used for evaluating the taxonomy of the RCO, as presented in Table 6.

The second approach in the verification process requires the implementation of the FOCA methodology [26]. FOCA is an approach that helps the ontology developers to evaluate the quality of their ontology by using a statistical model. Fig. 5 illustrates the three verification steps of FOCA.

FOCA was applied in this section to verify the RCO by using the following steps:

1) **Step 1- Ontology Type Verification**

As mentioned previously, the RCO can be considered to be a domain ontology. Based on FOCA, the verification process is divided into two types. The first type is specified for domain (low-level) ontologies, while the second type is specified for application ontologies. Accordingly, based on step 1, the RCO should be considered to be Type 1.

TABLE 5. Answers of CQs.

No.	Competency Questions (CQs)	Answers
CQ 1.	What type of requirements represent how a system should react to particular inputs?	Functional requirements
CQ 2.	What type of requirements represent a constraint on the services or functions offered by a system?	Non-functional requirements
CQ 3.	What type of requirements specify the data that should be input and output by a system, and the data that should be stored internally by a system?	Data requirements
CQ 4.	What type of requirements explain the functions of a system, how it records, computes, transforms, and transmits data?	Procedural requirements
CQ 5.	What type of requirements arise from the external aspects of a system and its improvement stages?	External requirements
CQ 6.	What type of requirements define the system requirements derived from the rules and processes in the organization of stakeholders?	Organizational requirements

TABLE 6. Taxonomy evaluation for the RCO.

Criteria	Sub-criteria	Explanation
Inconsistency	Circularity Errors	The RCO contains no class that is defined as a specialization or generalization of itself. In addition, the reasoner Pellet 1.5.2 shows no error in checking concept consistency.
	Partition Errors	The RCO contains no improper definitions of disjoint classes or incomplete class definitions. In addition, the reasoner Pellet 1.5.2 shows no error in checking concept consistency.
	Semantic Errors	The RCO contains no concept that is a subclass of a concept to which it does not belong. In addition, the reasoner Pellet 1.5.2 shows no error in checking concept consistency.
Incompleteness	Incomplete Concept Classification	All domain concepts are included in the RCO.
	Partition Errors	All relations in the RCO between classes are defined.
Redundancy	Grammatical redundancy	All classes in the RCO have their specific definitions.
	Identical formal definition of some classes	There are no identical definitions for two or more classes in the RCO.
	Identical formal definition of some instances	There are no identical definitions for two or more instances in the RCO.

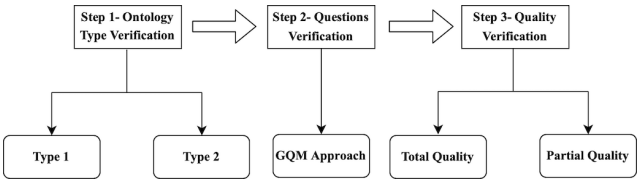

FIGURE 5. FOCA approach.

2) Step 2- Questions Verification

This step is based on the goal/question/metric (GQM) approach [26]. The GQM approach includes five goals, thirteen related questions, and six metrics, as shown in Table 7. Adaptability, completeness, consistency, computational efficiency, conciseness, and clarity are the six metrics in the GQM approach. The five goals of the GQM approach can be covered by the thirteen questions as follows:

- The first goal is inclined toward the CQs and reuse issue.
- The second goal determines the expected level of the ontology's terms.
- The third goal includes the discrepancies.
- The fourth goal is concerned mainly with reasoner performance.
- The last goal is inclined toward the documentation of the ontology and examining the level of consistency between the developed ontology and its specification.

The authors in [26] also described how to verify each question by using a set of if questions. Each question has a grade depending on its answer, ranging between 0 and 100. Based on the answers to these if questions regarding the developed ontology, the RCO received a grade for each question, as shown in Table 7. In addition, the mean of the total grade for each goal was calculated.

Concerning the first goal, the grade of Q3 is zero because the RCO was built from scratch. Concerning the second goal, Q4 is omitted, and the mean is between Q5 and Q6 only because the RCO is Type 1. The grade of Q5 is 50 because the RCO uses moderate abstraction to define the evaluated domain.

3) Step 3- Quality Verification

There are two types of quality verifications, as shown in Fig. 5, i.e., partial quality verification and total quality verification. For the RCO, the total quality verification method was chosen because the quality of the ontology in this method considers all five roles of knowledge representation, i.e., substitute, ontological commitments, intelligent reasoning, efficient computation, and human expression. The quality of the ontology in partial quality verification considers only substitute and intelligent reasoning. Based on the beta regression models in [27], total quality verification, which falls between 0 and 1, was calculated, as shown below.

TABLE 7. Applying the GQM approach [26] on the RCO.

Goal	Question	Metric	Grade	Mean
1. Check if the ontology complies with Substitute.	Q1. Were the competency questions defined?	1. Completeness.	100	66.67
	Q2. Were the competency questions answered?	1. Completeness.	100	
	Q3. Did the ontology reuse other ontologies?	2. Adaptability.	0	
2. Check if the ontology Complies with Ontological Commitments.	Q4. Did the ontology impose a minimal ontological commitment?	3. Conciseness.	-	75
	Q5. Did the ontology impose a maximum ontological commitment?	3. Conciseness.	50	
	Q6. Are the ontology properties coherent with the domain?	4. Consistency.	100	
3. Check if the ontology complies with Intelligent Reasoning	Q7. Are there contradictory axioms?	4. Consistency.	100	100
	Q8. Are there redundant axioms?	3. Conciseness.	100	
4. Check if the ontology Complies with Efficient Computation	Q9. Did the reasoner present modelling errors?	5. Computational efficiency.	100	100
	Q10. Did the reasoner perform quickly?	5. Computational efficiency.	100	
5. Check if the ontology complies with Human Expression.	Q11. Is the documentation consistent with the modeling?	6. Clarity.	100	100
	Q12. Were the concepts well written?	6. Clarity.	100	
	Q13. Are there annotations in the ontology that show the definitions of the concepts?	6. Clarity.	100	

$$\mu_i = \frac{\begin{array}{l}\exp\{-0.44 + 0.03\,(\mathrm{Cov_s}\times\mathrm{Sb})_i + 0.02\,(\mathrm{Cov_C}\times\mathrm{Co})_i \\ +0.01\,(\mathrm{Cov_R}\times\mathrm{Re})_i + 0.02\,(\mathrm{Cov_{CP}}\times\mathrm{Cp})_i \\ -0.66\mathrm{LExp}_i - 25(0.1\times\mathrm{Nl})_i\}\end{array}}{\begin{array}{l}1 + \exp\{-0.44 + 0.03\,(\mathrm{Cov_s}\times\mathrm{Sb})_i \\ +0.02\,(\mathrm{Cov_C}\times\mathrm{Co})_i + 0.01\,(\mathrm{Cov_R}\times\mathrm{Re})_i \\ +0.02\,(\mathrm{Cov_{CP}}\times\mathrm{Cp})_i - 0.66\mathrm{LExp}_i - 25(0.1\times\mathrm{Nl})_i\}\end{array}} \quad (1)$$

To calculate the total quality:

- Cov_s is the calculated mean for the first goal.
- Cov_c is the calculated mean for the second goal.
- Cov_R is the calculated mean for the third goal.
- Cov_{Cp} is the calculated mean for the fifth goal.
- LExp depends on the evaluator experience. It is equal to 1 if the evaluator is very experienced in ontologies and 0 if the evaluator has no experience.
- Nl depends on the possibility of goal evaluation. It is equal to 1 if there are some goals that cannot be evaluated.
- Sb = 1, Co = 1, Re = 1, and Cp = 1 because the total quality verification has been selected.

The result of the total quality is 0.998 and it is very close to 1, which means that the quality of the RCO is high. Thus, the proposed RCO was successfully validated and verified. It could be said that the RCO is ready to be used in real-world applications for the requirements classification process.

V. DISCUSSION

As described in Section IV, the RCO was successfully evaluated using the validation and verification tests. The first approach in the validation test indicated that the RCO content satisfied all the criteria of content evaluation. The second approach showed that all answers for the CQs were clearly stated.

In addition, the first approach in the verification test indicated that the RCO taxonomy satisfied all criteria of the taxonomy evaluation, and no violations were observed. The implementation of the FOCA methodology as the second approach in the verification test presented a high-quality level in the development of the RCO.

The results of the validation and verification tests indicated that the proposed RCO was designed according to specific quality standards defined for ontologies.

VI. CONCLUSION

Finally, it can be concluded that this study presented the details for developing and evaluating an ontology, called the RCO, that may be used for requirement classification in the RE context. This work was driven by the lack of research regarding ontologies that can classify the requirements in the RE context. The proposed RCO would be a significant contribution to ontology libraries. In addition, this ontology can be used in several ways to increase the quality of software requirements specification documents. More specifically, the developed ontology can be used as a tool to confirm the semantic correctness of the RE process. The RCO also ensures consistency between requirements, and facilitates communication between requirements engineers owing to the use of same terminologies for various software applications. In this work, the phases of the RCO building process were stated. The proposed RCO was successfully evaluated using two approaches for validation and verification each. For future work, the RCO will be integrated with a model that can be used to automatically classify the requirements [20]. The model includes a hybrid approach that combines several artificial intelligence techniques, such as machine learning algorithms, to utilize the RCO to automate the requirements classification process. Moreover, there is an opportunity for translating the RCO to other languages to increase the possibility of benefiting from its advantages.

REFERENCES

[1] T. R. Gruber, "A translation approach to portable ontology specifications," *Knowl. Acquisition*, vol. 5, no. 2, pp. 199–220, Jun. 1993.

[2] G. Song, Y. Qian, Y. Liu, and K. Zhang, "Oasis: A mapping and integration framework for biomedical ontologies," in *Proc. 19th IEEE Symp. Comput.-Based Med. Syst. (CBMS)*, Jan. 2006, pp. 611–616. [Online]. Available: http://ieeexplore.ieee.org/abstract/document/1647638/

[3] S. Boyce and C. Pahl, "Developing domain ontologies for course content," *Int. Forum Educ. Technol. Soc.*, vol. 10, no. 3, pp. 275–288, 2007.

[4] J. Sarraipa, J. P. M. A. Silva, R. Jardim-Goncalves, and A. A. C. Monteiro, "MENTOR—A methodology for enterprise reference ontology development," in *Proc. 4th Int. IEEE Conf. Intell. Syst.*, Varna, Bulgaria, Sep. 2008, pp. 32–36.

[5] A. Hajmoosaei and S. Abdul-Kareem, "An approach for mapping of domain-based local ontologies," in *Proc. Int. Conf. Complex, Intell. Softw. Intensive Syst.*, Mar. 2008, pp. 865–870, doi: 10.1109/CISIS.2008.98.

[6] I. Sommerville, *Software Engineering*, 9th ed. Boston, MA, USA: Pearson, 2011.

[7] IBM Corporation, "Getting requirements right: Avoiding the top 10 traps," Softw. Group, Somers, NY, USA, Tech. Rep., Oct. 2009.

[8] *Strategies for Project Recovery*, PM Solutions, Chadds Ford, PA, USA, 2011.

[9] A. Hussain, E. O. Mkpojiogu, and F. M. Kamal, "The role of requirements in the success or failure of software projects," *Int. Rev. Manage. Marketing*, vol. 6, no. 7, pp. 306–311, 2016.

[10] (2019). *Rotege Ontology Library*. [Online]. Available: https://protegewiki.stanford.edu/wiki/Protege_Ontology_Library

[11] (2019). *The DARPA Agent Markup Language Homepage*. [Online]. Available: http://www.daml.org/index.html

[12] T. V. Avdeenko and N. V. Pustovalova, "The ontology-based approach to support the requirements engineering process," in *Proc. 13th Int. Sci.-Tech. Conf. Actual Problems Electron. Instrum. Eng. (APEIE)*, Oct. 2016, pp. 513–518.

[13] Y. Odeh and M. Odeh, "A new classification of non-functional requirements for service-oriented software enineering," in *Proc. Naif Arab Univ. Secur. Sci.*, Riyadh, Saudi Arabia, 2009, pp. 1–7.

[14] J. L. Velasco, R. Valencia-García, J. T. Fernández-Breis, and A. Toval, "Modelling reusable security requirements based on an ontology framework," *J. Res. Pract. Inf. Technol.*, vol. 41, no. 2, p. 119, 2009.

[15] A. Rashwan, O. Ormandjieva, and R. Witte, "Ontology-based classification of non-functional requirements in software specifications: A new corpus and SVM-based classifier," in *Proc. IEEE 37th Annu. Comput. Softw. Appl. Conf.*, Jul. 2013, pp. 381–386, doi: 10.1109/COMPSAC.2013.64.

[16] T. Li and Z. Chen, "An ontology-based learning approach for automatically classifying security requirements," *J. Syst. Softw.*, vol. 165, Jul. 2020, Art. no. 110566, doi: 10.1016/j.jss.2020.110566.

[17] S. Sitthithanasakul and N. Choosri, "Using ontology to enhance requirement engineering in agile software process," in *Proc. 10th Int. Conf. Softw., Knowl., Inf. Manage. Appl. (SKIMA)*, Chengdu, China, Dec. 2016, pp. 181–186, doi: 10.1109/SKIMA.2016.7916218.

[18] V. PremaLatha, G. Rahul, and K. Pragathi, "Realm for metamorphose management in software requirements for global software progress environment," *Int. J. Innov. Technol. Exploring Eng.*, vol. 9, no. 2, pp. 5263–5268, Dec. 2019, doi: 10.35940/ijitee.B9003.129219.

[19] G. Brusa, M. L. Caliusco, and O. Chiotti, "A process for building a domain ontology: An experience in developing a government budgetary ontology," in *Proc. 2nd Australas. Workshop Adv. Ontologies*, Hobart, TAS, Australia, vol. 72, Dec. 2006, pp. 7–15.

[20] H. Alrumaih, A. Mirza, and H. Alsalamah, "Toward automated software requirements classification," in *Proc. 21st Saudi Comput. Soc. Nat. Comput. Conf. (NCC)*, Riyadh, Saudi Arabia, Apr. 2018, pp. 1–6, doi: 10.1109/NCG.2018.8593012.

[21] S. Koh and J. Whang, "A critical review on ISO/IEC 25000 square model," in *Proc. 15th Int. Conf. IT Appl. Manage., Mobility, Culture Tourism Digitalized World (ITAM)*, 2016, pp. 42–52.
[22] M. Uschold and M. Gruninger, "Ontologies: Principles, methods and applications," *Knowl. Eng. Rev.*, vol. 11, no. 2, pp. 93–136, Jun. 1996, doi: 10.1017/S0269888900007797.
[23] N. F. Noy and D. L. McGuinness. (2001). *Ontology Development 101: A Guide to Creating Your First Ontology*. [Online]. Available: http://liris.cnrs.fr/alain.mille/enseignements/Ecole_Centrale/What%20is%20an%20ontology%20and%20why%20we%20need%20it.htm
[24] A. Gómez-Pérez, "Ontology evaluation," in *Handbook on Ontologies* (International Handbooks on Information Systems), S. Staab and R. Studer, Eds. Berlin, Germany: Springer, 2004.
[25] S. Lovrencic and M. Cubrilo, "Ontology evaluation-comprising verification and validation," in *Proc. Central Eur. Conf. Inf. Intell. Syst., Fac. Org. Inform. Varazdin*, 2008, pp. 657–663.
[26] J. Bandeira, I. I. Bittencourt, P. Espinheira, and S. Isotani, "FOCA: A methodology for ontology evaluation," 2016, *arXiv:1612.03353*. [Online]. Available: http://arxiv.org/abs/1612.03353
[27] S. Ferrari and F. Cribari-Neto, "Beta regression for modelling rates and proportions," *J. Appl. Statist.*, vol. 31, no. 7, pp. 799–815, Aug. 2004, doi: 10.1080/0266476042000214501.

HALA ALRUMAIH received the M.Sc. degree in information systems from King Saud University, in 2010, where she is currently pursuing the Ph.D. degree in information systems. She is also a Lecturer at the Information Systems Department, Al Imam Mohammad Ibn Saud Islamic University. Her research interests include requirements engineering and machine learning. She was a member of the IT2017 executive committee and task group committee for the development of the IT2017 report, which is appropriately forward looking for graduates in the mid-2020s. She is also a member of the CC2020 task force. Computing Curricula 2020 (CC2020) is a joint project launched by professional computing societies to examine the current state of curricular guidelines for academic programs granting degrees in computing and providing a vision for the future of computing.

ABDULRAHMAN MIRZA received the Ph.D. degree in computer science from the Illinois Institute of Technology. He is currently a Professor at the Information Systems Department, King Saud University (KSU). He is also a Consultant at the Deputyship of Planning and Information, Ministry of Education, and the acting Director of the National Center for Research on Educational Policies. Some of his previous leadership positions include the Vice Dean of Academic Affairs at the College of Computer and Information Sciences, KSU, and a General Supervisor of the General Directorate of Teachers Affairs at the Ministry of Education. He also served as a Senior Advisor to the Minister of Education and the Minister of Higher Education. He also held other positions, such as the Director of Quality & Accreditation at Saudi Electronic University, the Deputy Director of the Center of Excellence in Information Assurance, the Chairman of the Information Systems Department, and the CIO at King Abdullah Foundation for Developmental Housing. His research interests include software engineering, e-commerce, and information security.

HESSAH ALSALAMAH is currently an Assistant Professor at the Information Systems Department, College of Computer and Information Sciences, King Saud University (KSU). She is also the Dean of the College of Engineering and Architecture, Al Yamamah University. She specializes in business process management and workflow technology. This includes process discovery, modeling, analysis, re-engineering, and automation. She also focuses her research on the emerging requirements of collaborative environments involving human aspects, such as communication, collaboration, and coordination, as well as technical aspects, such as heterogeneity and distribution. Lately, she has been exploring the domains of information security and privacy requirements in collaborative environments, such as eHealth and eGovernment.