Toward Supporting the Classification of Software Requirements with an Intelligent Semantic Approach

Hala Alrumaih [1, 2], Abdulrahman Mirza [1], Hessah Alsalamah [1, 3]

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

Summary
With the growing awareness of the effects of requirements on software development processes, requirements engineering is increasingly becoming an important field of focus in software engineering research. Many studies show that failures in understanding and classifying requirements are the main reasons for exceeding project costs and allocated time, which in turn may cause the failure of a project. Successful software systems development requires consistent and classified requirements. The classification of requirements represents an early but critical phase in the requirements analysis stage. While the literature sheds light on distinctions between different types of requirements, the detection of such differences in practice is not always an easy task. This paper provides an overview of requirements classification, presents some of the existing research studies on requirements classification, and discusses their limitations in terms of yielding suggestions for improvement. Additionally, this work takes a different approach to address requirements classification. It proposes a semantic model to classify requirements automatically, using a hybrid artificial intelligence approach. In addition, the paper discusses evaluation methods for each part of the proposed model.

Key words:
Requirements engineering; Requirements classification; Artificial intelligence; Machine learning; Ontology.

1. Introduction

This paper is an extended version of our SCS-NCC 2018 paper [1]. We are presenting in this work more detailed discussions and recommendations than what appeared in the previous version.

In software engineering, there is a growing and continuous demand to produce software systems in shorter cycles with higher productivity and quality [2]. This demand mainly depends on how well a software system fulfils the needs of its environment and users. Software requirements address these needs through requirements engineering processes. The successful development of software systems requires complete, consistent and classified requirements.

Accurate requirements classification is always a critical factor in the success of any project. Prior research [3], [4], [5] has highlighted the risks generated when working with incorrect methods of classifying requirements. These risks can even cause project failure due to projects exceeding time or budget constraints; at the very least, projects will incur extra effort to bring to completion. Reports show that 71% of software failures are due to a lack of clear representation and requirements classification [6]. The CHAOS report from the Standish Group [7] examined the causes of IT project failures in the USA, and showed that success rates for IT projects were only 16.2%, while the others ended in failure. The second factor giving rise to project challenges is the use of incomplete or imprecise requirements and specifications. Hence, the quality of requirements specification has an important effect on the outcome of the project due to the use of poor or imprecise requirements classification. Another survey has shown that there is a cost to IT projects arising from poor requirements classification [8].

Successful requirements engineering (RE) involves eliciting the needs of users, customers, and other stakeholders; analysing the stakeholders' requirements and documenting these requirements as software requirements specification; validating that the documented requirements match the stakeholders' requirements; and managing the evolution of requirements [9]. RE has now emerged from an initial stage in the software development process to a basic activity and stage in many organizations that spans the entire software development process [10].

The main component of requirements engineering is the requirements themselves; these are properties that must be exhibited to ensure that certain real-world problems can be solved [11], [12]. Before software can be developed,

Manuscript received March 5, 2020
Manuscript revised March 20, 2020

requirements must be specified and agreed to by the stakeholders of a software product such as customers, users, and suppliers.

Requirements analysis covers many important stages before the production of a high-quality specification document. Requirements classification is the first and the most complicated stage in requirements analysis. This importance arises from the diversity of requirements, which can be varied both in terms of their purpose and the types of properties they represent.

Software engineering scientists provide clear priorities and classifications for the requirements of the system by using several requirements classification techniques. There are many effective classification techniques dedicated to pioneers in the field of requirements engineering such as Sommerville's technique [11]. This diversity is due to the rich nature of software requirements, the reliance on methods for the definition of requirements, and the architecture and design methods that are applied in the developed software system.

The remainder of the paper is organized as follows. Section 2 provides a brief overview of requirements classification. Then, Section 3 reviews several works related to ontologies in order to show how they can be used to reduce the negative effects of certain factors on the requirements engineering processes. Section 4 describes recent research into artificial intelligence techniques; it shows how these techniques have been applied to solve a variety of requirements problems. Section 5 provides some background on the limitations of the current research and presents recommendations. Section 6 proposes a model based on these recommendations and shows how this model may be evaluated. Finally, Section 7 concludes the paper.

2. Requirements Classification

In the initial stage of RE, classifying and categorizing the requirements into various collections may enable many further actions to be taken much more easily than using a direct operation [13]. In the software engineering domain, classification has been a commonly used keyword over recent decades to classify requirements, faults, software risks, software features, and software testing [14]. From an RE point of view, classification can perform very helpful roles using techniques such as developing the level of understanding of user requirements [15], priority parameters [16], and the evaluation of their quality. Furthermore, requirements classification can be used to decrease the difficulty of decision making by reducing a large number of requirements into fewer collections [17].

The literature provides several major reasons for making these classifications [18], [19], [20], [21]. Firstly, there is a difference between functional requirements (FRs) and non-functional requirements (NFRs), in that much more challenging problems usually arise in designing and testing NFRs. Creating a design to meet the NFRs often takes more time and it involves complex problems, while FRs appear to be more straightforward.

Secondly, it is useful to describe constraints separately. By using the separate word 'constraints', we can explicitly place a limit on the design space without conflating the difference between the function space and design space.

3. Ontologies in Requirements Engineering

An ontology provides a common vocabulary used to describe a domain in a form of objects and concepts that exist together with their characteristics and relations. Developing an ontology is useful in sharing and describing different classifications. Ontologies specify a generic and semantic solution, which provides a precise, unambiguous, and reusable terminology in requirements classification. The following subsection provides ways of using ontologies in requirements engineering.

3.1 Insufficient Requirements Engineering Process

The formation of well-defined requirements that are agreed on by stakeholders has become a key priority in software development. The software system may not satisfy the aims of users if the needs are expressed as incomplete or incorrect requirements. Possible reasons leading to an insufficient process of RE are as follows [22].

1. Ambiguous requirements, where each stakeholder interprets the same requirement in different ways, producing repetition of work and loss of time.
2. Insufficient specifications that result in the absence of primary requirements, meaning that the required software is not developed because the developers base their work on incorrect requirements.
3. Requirements that are incompletely defined, leading to wrong estimations and judgments that ignore the required specifications.
4. Dynamic and changing requirements, requiring management to help in understanding the new users' aims and goals to define how they can be satisfied.

Ontologies can be used to decrease the unfavourable effects of the previous reasons on RE processes. The

potential benefits of applying ontologies in RE include representations of the following aspects [22].

1. Requirements model: This enables a way of structuring and classifying the requirements. A requirements ontology can be used to reduce ambiguous requirements and avoid incomplete requirements descriptions during the analysis stage.
2. Acquisition structures for domain knowledge: This is an ontology for requirements specification documents that describes the body structure of these documents.
3. Knowledge of the application domain: An application domain ontology helps in understanding the domain by detecting dynamic and changing requirements.

3.2 Applying Ontologies in Requirements Engineering

The use of ontologies in the demonstration of requirements knowledge has been researched for many years. Lin et al. [23] presented one of the initial approaches in this area, proposing a generic and semantic solution that provides a precise, unambiguous, and reusable terminology. The developed ontology is used to define the meaning of the terminology. This solution is appropriate for any type of product to develop diverse requirements such as completeness, consistency, traceability, and communication. It also supports the discovery of redundant or conflicting requirements. The proposed ontology was implemented using the Prolog language. Although being a very complete ontology, one of its disadvantages is that only the engineers on the project share the terminology involved; as a result, customers are not interested in this aspect, and some requirements might appear ambiguous.

Dobson and Sawyer [24] proposed a more specific method for using ontologies to represent the knowledge domain of NFRs. The proposed ontology is used to express the dependencies between requirements. This method involves various NFRs, such as confidentiality, maintainability, integrity, availability, reliability, and safety.

Diallo et al. [25] described a method to measure whether or not a selected architecture dependably satisfies and meets the stakeholder needs specified in requirement-level scenarios. They mapped scenarios to elements of the architecture using an ontology of requirement-level event classes and domain entities. The scenarios represent both FRs and NFRs or quality attributes of the system. In terms of quality attributes, the scenarios either increase the quality of actions or indicate how to check the quality. Their approach specified a relationship between requirements that can be understood directly by stakeholders, and the architectures developed to satisfy those requirements. The requirement-level ontology facilitates the mapping process and acts as a focus for maintaining the mapping, which helps in the development the architecture and scenarios. The ontology also provides a base for individually and jointly assessing the scenarios and the architecture. The authors emphasized the mapping through event classes and explained their approach using two examples.

Gailly et al. [26] proposed an ontology-based RE approach that detects domain knowledge and uses early requirements modelling techniques by combining the use of domain ontologies. These techniques are mainly aimed at eliciting and representing the organizational and intentional context of the system. This approach can be used with diverse types of domain ontologies and software RE techniques.

Another study in this field was presented by Kassab [27], who proposed an ontology specifying the definition of the general concepts related to NFRs. Through the proposed ontology, he described different glossaries and taxonomies for NFRs without reference to any particular domain.

Scholars and practitioners have used ontologies to model knowledge in software engineering and to indicate the artefacts that are designed or produced during the RE process. Zhao et al. [28] classified the ontologies developed for software engineering. They reviewed the current efforts related to applying semantic web techniques to different software engineering aspects. They also presented the benefits of their applications. One of their classifications was an ontology for requirements that describes the desired software characteristics, called a ‘system behaviour ontology’, as specified by the customer. As an example of this kind of ontology, Caralt and Kim [29] proposed an ontology to enclose use cases with semantic information. In a complex software development situation, the detection of an appropriate use case from a big library produced previously or from relevant projects is a difficult, expensive, and error-prone task. The proposed ontology was obtained from ResearchCyc, which is an ontology for the scientific community that contains a taxonomy of more than 6,000 actions. The authors also proposed queries used to retrieve use cases could be augmented with this ontology. The developed ontology is used to capture, reuse, and query use cases efficiently.

Wongthongtham et al. [30] developed an ontology model to represent software engineering knowledge. RE was a part of this ontology, and is referred to as software requirements. The authors examined a software engineering ontology and the different elements composing it. The scenarios presented in this paper

highlight the various features of this software engineering ontology. The developed ontology can help to identify the information necessary to exchange semantic information about the project. Software engineers can use this ontology both to share software engineering knowledge and to establish a communication framework.

Liu [31] investigated conflicts in the requirements specifications of activity diagrams using an ontological approach, and proposed a method consisting of a modelling process and a group of rules for conflict detection. In addition, Liu provided several scenarios related to electronic commerce to demonstrate the validity of the proposed rules. The suggested method has two advantages and one main disadvantage. The feasible analysis process is a feature of this method. In the analysis of activity diagrams, the feasible analysis process offers a well-defined and systematic road map for the necessary data collection. Effectiveness is the second advantage of this method; by using effective rules, this method attempts to simplify conflict identification research, although the essential weakness of this method is that it cannot be applied in the initial stages of developing complete and innovative new systems. When the requirements for developing new systems are unclear, producing complete and accurate prior knowledge is difficult.

The ontology for requirements proposed in [32] is based on web ontology language (OWL) and is a main part of the automated design approach. The function-oriented, automated and integrated design has three stages: RE, high-level abstract design, and a comprehensive detailed design. The ontology provides a description of both terminological (concept or class knowledge) and assertion knowledge (factual or instance knowledge). Terminological knowledge is used to represent current requirements and causal dependencies that are used in creating future requirements, while assertion knowledge is represented as an instance with regard to terminological knowledge. The developed ontology represents the precise requirements of a developed automatic planned system, using an automated design approach to provide a semantic description and representation of knowledge. In addition, the proposed ontology is useful for the entire automation engineering process.

Shunxin and Leijun [33] discussed how the use of the ontology notation in the RE domain helps increase the level of understanding of semantic information. Their paper introduced ontology-related concepts and theories. It presented a general framework for developers for the use of an ontology to help in the development of requirements and ensure quality and speed. The authors used several examples to explain how to use ontology requests analysis, validation, and enhancement.

Farfeleder et al. [34] presented a prototypical implementation of a semantic guidance system. Requirements engineers use this system to determine requirements using a semi-formal representation. A requirements engineer can build on a list of suggestions to define requirements; this list is provided by a semantic guidance system that uses concepts, relations and axioms of specific domain ontology. The developed system is tested based on a specific domain ontology and a set of requirements from the aerospace domain. The results of assessment show that the proposed system successfully supports requirements engineers in defining well-structured requirements.

Minhas et al. [13] presented an automated technique for software requirements classification using statistical and ontology methods. This classification technique is built using a controlled vocabulary to split the documented user requirements into specified collections in order to make further actions much easier than in direct operation. The authors demonstrated the implementation of the classification technique using three case studies. The overall structure of this technique consists of three main parts. The first component represents the source data in the form of a repository containing keywords and their relationships. The second component is the mapping stage, which is based on finding words in the requirements document that are related to keywords in the repository. The last component is the presentation stage, which delivers the classified requirements in more meaningful ways. Improving the quality of the results to reflect real groups of interests is the main aspect that improves the effectiveness of this technique.

Daramola et al. [35] discussed how the concept of an ontology supports the specification of security requirements. Their approach utilizes a combination of ontologies and boilerplates through a tool-based framework. This approach helps requirements engineers to identify security threats and ultimately formulate quality security requirements; it also decreases the effort required in the process of security requirements specification and offers a good starting point in cases when a sufficiently experienced requirements engineer may not be available. This approach is viable for supporting the specification of security requirements based on an initial evaluation.

To improve semantic tool support for the RE domain, Rashwan et al. [36] developed a new classification algorithm to categorize automatically NFRs in software specifications. They used ontology notation to convert software requirements documents automatically into a semantic representation. This approach is useful to handle the cost of the software system and measure the quality of written requirements.

Avdeenko and Pustovalova [37] proposed a new approach based on an ontology to match the theoretical framework of RE. This technique reduces the number of different terms used to describe the same concepts, and expands the domain of existing tools, concepts, and models, due to their sharing and combining. The approach is also used to transform rapidly and cost effectively a standard document from the software requirements specification (SRS) into an alternate format. They do this by automating the selection processes of the type of requirements specifications, the parameters that are sufficient to specify the project and the development team, and thus the methods used in working with the requirements. The proposed approach uses a mechanism based on production rules that help produce a system of requirements satisfying the correctness and completeness properties of the developed system.

Bhatia et al. [38] provided an ontology-based framework and an implementation approach for detecting ambiguity in the specification of software requirements. This research shows how the elimination of significant ambiguities (pragmatic ambiguity, semantic ambiguity, vagueness and generality, and language errors) improves requirements, thus assisting in quality software development.

Based on combining Advanced Persistent Threat (APT) attack cases and system domain knowledge, Kim et al. [39] proposed an ontology base and its design process for recommending appropriate security requirements. The proposed knowledge base is constructed by three parts; APT ontology, general security knowledge ontology, and domain-specific knowledge ontology. Each ontology can help to understand the security concerns in their knowledge. While integrating three ontologies can help in deriving the appropriate security requirements along with the security requirements recommendation process.

Table 1 presents a summary of this literature review. It lists the types of requirements to which the ontology is applied in each research study and the main objective of using an ontology in each paper.

Table 1: A List of Recent Works Using the Ontology Notation in Requirements Engineering

Reference	Publication Year	Functional / Non-functional	Main objective of using an ontology
[23]	1996	FRs and Constraints	Providing an unambiguous and precise terminology for the engineering design domain that can be shared by all the engineers involved
[24]	2006	NFRs	Representing dependencies between requirements
[25]	2007	FRs / NFRs	Augmenting use cases with semantic information
[26]	2008	FRs and quality attributes	Assessing whether a candidate architecture dependably meets the stakeholder requirements using a mapping technique
[27]	2008	FRs / NFRs	Integrating the domain knowledge into early requirements models that mainly aim to elicit and represent the organizational and intentional context of the system
[28]	2009	NFRs	Describing different glossaries and taxonomies for NFRs
[29]	2009	FRs / NFRs	Classifying the ontologies developed for the software engineering domain
[30]	2009	FRs / NFRs	Assisting in the definition of information in order to exchange semantic information about the projects; used as a communication framework
[31]	2009	FRs / NFRs	Analyzing conflicts in the requirements specifications of activity diagrams
[32]	2009	FRs / NFRs	Allowing a semantic explanation and representation of knowledge using the automated design approach
[33]	2010	FRs / NFRs	Understanding the semantic level of information in the RE domain
[34]	2011	FRs / NFRs	Assisting requirements engineers in capturing requirements
[13]	2011	FRs / NFRs	Developing an automated classifier for classifying software requirements efficiently
[35]	2012	NFRs	Aiding requirements engineers in identifying security threats and ultimately formulating the quality security requirements
[36]	2013	NFRs	Developing a new classification algorithm to automatically categorize requirements in software specifications documents
[37]	2016	FRs / NFRs	Matching the theoretical framework of RE to reduce the number of different terms used to describe the same concepts
[38]	2016	FRs / NFRs	Detecting ambiguities in software requirements specification
[39]	2019	NFRs	Deriving the appropriate security requirements along with the security requirements recommendation process

4. Using Artificial Intelligence Techniques in Requirements Engineering

Some encouraging outcomes resulted from the application of artificial intelligence (AI) techniques in RE, including knowledge-based approaches, automated reasoning, expert systems, heuristic search strategies, and machine learning. Machine learning methods have been involved in the software development lifecycle. These AI techniques also offer a viable alternative in terms of automating many software engineering issues.

In addition to works [40], [41], [42], [43], [44], [45], [46], [47], [48] were discussed in [1] that show how different (AI) techniques can be used effectively in various stages of requirements engineering processes, Li [49] also proposed a hybrid approach to automatically recognize security requirements. This approach used both linguistic analysis and machine learning techniques.

In specific, the author systematically assessed the current security requirements ontologies to start creating a revised security requirements conceptual model. This model is based on a set of linguistic rules that are normally used to define security requirements. The evaluation results showed that the proposed approach has a high chance to train classifiers to classify requirements from diverse application domains.

Singh [50] helped requirement engineers using an approach to automate the selection of specific areas of SRS that may need to be re-inspected because of existence some faults. The author used some machine learning techniques that can effectively isolate faults from non-faults.

Table 2 presents a summary of this literature review showing the type of AI technique used and the main objective of using of this technique in each research study.

Table 2: A List of Recent Works Using AI Techniques in Requirements Engineering

Reference	Publication year	AI technique	Main objective of using AI technique
[40]	2000	Machine learning	Implementing tools for software development and maintenance tasks
[41]	2003	Automated reasoning	Developing an intelligent assistant used in the elicitation and assessment of requirements
[42]	2008	Machine learning	Supporting knowledge management requirements in military command centers
[43]	2011	Naïve Bayes	Automatically generating requirements for a domain ontology evolution
[44]	2011	Naïve Bayes	Integrating knowledge engineering with requirements
[45]	2012	AI Techniques	Demonstrating the application of AI techniques in software engineering practices
[46]	2016	Machine learning	Analyzing SRSs and automatically extracting semantic information
[47]	2016	Decision tree	Classifying security-based explanations into four types of security requirements
[48]	2016	Neural networks	Automatically classifying the content of a natural language requirements specification as a “requirement” or “information”
[49]	2017	Machine learning	Automatically recognize security requirements
[50]	2018	Machine learning	Automating the selection of specific areas of SRS that may be faulty and would need to be re-inspected

5. Gap analysis and recommendations

The previous sections discuss in detail the existing works related to requirements classification techniques and exploring ways to use ontology and AI techniques in RE. Although there are many different research methods in the RE field, the previous works contain many gaps, and the topic is still active for researchers in the field.

5.1 Gap Analysis

Software engineers prioritize and classify requirements using several requirements classification techniques. As shown above, pioneers have developed many effective classification techniques in the area of RE. However, most of these techniques do not involve the semantic aspects of requirements.

Although ontology as a means of defining information and knowledge semantics has become a focus of study in RE research, the looming challenge still involves methods or processes to develop a custom-built ontology that is used specifically in the requirements classification process.

In previous works, ontologies were a vehicle to solve a variety of other requirements problems. Additionally, few ontologies exist for classifying specific requirements such as NFRs or security requirements.
Furthermore, the literature shows that the techniques that have been proposed for classifying requirements are either manual or automated techniques containing several limitations and gaps. In manual techniques, the classification process suffers from several limitations such as dealing with large numbers of requirements, excessive time consumption, unavailability of experts, or wrong judgements. In automated techniques, there is a need to improve the effectiveness of the methods to increase the quality of their results.
Used alone, AI techniques are not adequate for the classification process. These techniques can build a classifier automatically by learning the features of the categories from a set of classified requirements and then using results to classify requirements into predefined categories. However, these techniques have certain weaknesses. Most of these techniques do not consider semantic interactions, thereby making it difficult to improve the accuracy of these classification methods. Better classification is achievable when the semantic relations are considered using an ontology as an example.
Although the use of individual classifiers is not sufficient to classify requirements with high performance, it is possible that a combination of different classification techniques can improve the classification accuracy. The concept of a hybrid scheme that combines more than one classification technique is proposed as an ideal approach to improve the performance of individual classifiers.

5.2 Recommendations

With the release of semantics and the progress made in related technologies, opportunities for using ontologies to represent knowledge and information semantics are becoming increasingly acceptable in several domains. This justifies the choice to develop a specific ontology for requirements classification that is useful in sharing and describing different classifications. Hence, a need exists to develop a custom-built ontology that contains a full representation of the important existing requirements classification techniques to handle the semantic aspects of requirements. The custom-built ontology distinguishes between different types of requirements such as business, user and performance requirements, FRs and NFRs, system, operational, software and hardware requirements, and interface and maintenance requirements.
In addition, due to the previous limitations of the techniques used to classify requirements, it is necessary to automate the classification processes to improve the quality of the results. This automation can save time and effort, and can decrease the cost by reducing the number of hours required from experts.
This research takes a different approach to dealing with requirements classification. A hybrid approach using artificial intelligence techniques such as machine learning algorithms that utilize the custom-built ontology, is instrumental in classifying requirements automatically.
The selected algorithms should be chosen from different categories of the developed machine learning algorithms for classification to increase the quality of the result. The concept of a “hybrid” that combines more than one artificial intelligence technique for classification suggests a new approach to improving the performance of individual classifiers. Several researchers [40], [41], [42], [45] have shown that combining and joining multiple classification techniques can improve classification accuracy. This approach can minimize the ambiguities that currently exist and can increase the efficiency of software development.

6. The proposed Model And its Evaluation

This section proposes a model to classify requirements automatically based on previous recommendations and shows how this model may be evaluated.

6.1 The proposed Model

Fig. 1 shows the structure of the proposed model for automatically classifying requirements. Based on the previous recommendations, we divide the proposed model into two parts:

- A custom-built ontology for requirements classification that is useful in sharing and describing the different classifications; and
- A hybrid approach that combines several artificial intelligence techniques, such as machine learning algorithms, in order to utilize the custom-built ontology to automate the requirements classification process.

As a primary step within the development of the custom-built ontology, a set of requirements with known classes are used as instances to provide examples of each class. This set is taken from the software requirements specifications documents for successfully completed projects in certain small or medium-sized companies.
The model contains a custom-built ontology that includes instances called training data; these data provide examples of requirements from previous projects for each class in the ontology. In addition, the hybrid approach used in the model combines several artificial intelligence

techniques to utilize the developed ontology. When a requirements engineer receives a list of unclassified requirements, he or she can start using the model to automate the classification process. A list of unclassified requirements called test data is passed through several AI techniques, which start to classify the requirements automatically based on the training data from the developed ontology. The main output of the hybrid approach is the classified requirements. The requirements engineer receives these classified requirements and can insert them into the developed ontology as new instances to supply the ontology with new training data. This is not done unless the suggested model is validated and a high level of performance is obtained from the hybrid approach.

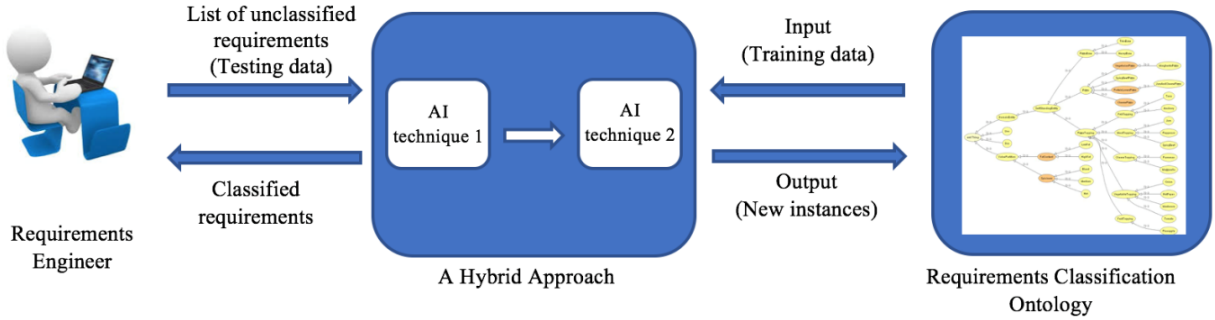

Fig. 1: Proposed model for requirements classification

6.2 Evaluation Stage

Evaluation for the custom-built ontology uses a benchmarking strategy, which involves comparing the custom-built ontology with specification templates and examples for the current software requirements. A questionnaire will collect expert feedback to evaluate, refine, and extend the ontology as necessary.

The data used for experimentation are data from actual situations to allow evaluation of the proposed hybrid approach. These data will be taken from software requirements specification documents of successfully completed projects. In the first stage, requirements will be taken from small-scale company projects to monitor results and make changes, since it is easier to monitor problems with a smaller quantity of data. Once the proposed model shows good results, we will use standard internationally recognized requirements datasets to generate results and to monitor them. All of these requirements are supplied with their current human classifications allowing us to compare the classification results from our model with the real classification. From this, we will be able to determine the errors incurred and the level of improvement the model achieved. The performance of the hybrid approach is evaluated using standard performance measures for machine learning algorithms, such as precision, recall, and accuracy.

7. Conclusion

The proposed model in this paper aims to benefit practitioners in industry and government as well as academics in universities and technical institutions. The model can enable industrial and governmental practitioners to develop more lucid and concrete requirements classification that minimize ambiguities. In academic settings, the model is useful for student instruction and for research.

Requirements classification is a critical factor in project success since poor classification may cause losses in terms of costs, time and effort. Using manual techniques to classify requirements entails significant effort and time from requirements engineers. By applying intelligent methods to classify requirements, this process can reduce the amount of time lost. Moreover, this new process can increase the quality of requirements analysis and can produce accurate system specification documents. For academics, the proposed model can highlight the greater need for clearer requirements specifications in software engineering courses and provide a stronger basis for future research in this area.

In general, the proposed model may be considered as a separate tool for automatically classifying requirements. In future research, additional activities from the RE process will be added in order to increase the productivity of this research.

References

[1] H. Alrumaih, A. Mirza, and H. Alsalamah, "Toward Automated Software Requirements Classification," in *Proceeding of 21st Saudi Computer Society National Computer Conference (SCS-NCC'2018)*, Saudi Arabia, Riyadh.

[2] U. Dinger, R. Oberhauser, and C. Reichel, "A Semantic Web Services Approach Towards Automated Software Engineering," in *Web Services, 2006. ICWS'06. International Conference on*, 2006, pp. 935–938.

[3] I. Corporation, "Getting requirements right: avoiding the top 10 traps.," Software Group, Route 100, Somers, NY 10589, Requirements definition and management, Oct. 2009.

[4] P. M. Solutions, "Strategies for Project Recovery," PM Solutions Inc., Pennsylvania, USA, 2011.

[5] A. Hussain, E. O. Mkpojiogu, and F. M. Kamal, "The Role of Requirements in the Success or Failure of Software Projects," *International Review of Management and Marketing*, vol. 6, no. 7S, pp. 306–311, 2016.

[6] C. Lindquist, "Fixing the Software Requirements Mess," *CIO*, 15-Nov-2005. [Online]. Available: https://www.cio.com/article/2448110/developer/fixing-the-software-requirements-mess.html. [Accessed: 29-Sep-2017].

[7] T. Clancy, "The Standish Group Report -CHAOS," Project Smart, pp. 1-16, 2014.

[8] B. J. E. Powell, "IT Pays a Price for Poor Requirements Practices -," *ADTmag*. [Online]. Available: https://adtmag.com/articles/2008/02/07/it-pays-a-price-for-poor-requirements-practices.aspx. [Accessed: 29-Sep-2017].

[9] S. Asghar and M. Umar, "Requirement engineering challenges in development of software applications and selection of customer-off-the-shelf (COTS) components," *International Journal of Software Engineering*, vol. 1, no. 1, pp. 32–50, 2010.

[10] G. Aouad and Y. Arayici, *Requirements engineering for computer integrated environments in construction*. Chichester, West Sussex, U.K. ; Ames, Iowa: Wiley-Blackwell, 2010.

[11] I. Sommerville, *Software engineering*, 9th ed. Boston: Pearson, 2011.

[12] G. Kotonya and I. Sommerville, *Requirements Engineering: Processes and Techniques*. John Wiley and Sons, 1998.

[13] N. M. Minhas, S. Majeed, Z. Qayyum, and M. Aasem, "Controlled vocabulary based software requirements classification," in *Software Engineering (MySEC), 2011 5th Malaysian Conference in*, 2011, pp. 31–36.

[14] Z. T. Bieniawski, *Engineering rock mass classifications: a complete manual for engineers and geologists in mining*. New York, civil, and petroleum engineering: John Wiley & Sons, 1989.

[15] H. Stille and A. Palmström, "Classification as a tool in rock engineering," *Tunnelling and Underground Space Technology*, vol. 18, no. 4, pp. 331–345, Aug. 2003, doi: 10.1016/S0886-7798(02)00106-2.

[16] M. Daneva and A. Herrmann, "Requirements Prioritization Based on Benefit and Cost Prediction: A Method Classification Framework," in *34th Euromicro Conference Software Engineering and Advanced Applications, IEEE*, 2008, pp. 240–247, doi: 10.1109/SEAA.2008.46.

[17] M. Aasem, M. Ramzan, and A. Jaffar, "Analysis and optimization of software requirements prioritization techniques," presented at the International Conference on Information and Emerging Technologies, Karachi, 2010, pp. 1–6.

[18] E. Hochmüller, *Requirements classification as a first step to grasp quality requirements*. Proc. Third International Workshop on Requirements Engineering: foundation of Software Quality (REFSQ'97), 1997.

[19] M. Hertzum, "Small-scale classification schemes: A field study of requirements engineering," *Computer Supported Cooperative Work (CSCW)*, vol. 13, no. 1, pp. 35–61, 2004.

[20] M. Y. Kiang, "A comparative assessment of classification methods," *Decision Support Systems*, vol. 35, no. 4, pp. 441–454, Jul. 2003, doi: 10.1016/S0167-9236(02)00110-0.

[21] K. Lauenroth, E. Kamsties, and O. Hehlert, "Do Words Make a Difference? An Empirical Study on the Impact of Taxonomies on the Classification of Requirements," presented at the Requirements Engineering Conference (RE), 2017 IEEE 25th International, 2017, pp. 273–282, doi: 10.1109/RE.2017.57.

[22] V. Castañeda, L. Ballejos, M. L. Caliusco, and M. R. Galli, "The use of ontologies in requirements engineering," *Global Journal of Research In Engineering*, vol. 10, no. 6, 2010.

[23] J. Lin, M. S. Fox, and T. Bilgic, "A requirement ontology for engineering design," *Concurrent Engineering*, vol. 4, no. 3, pp. 279–291, 1996.

[24] G. Dobson and P. Sawyer, "Revisiting ontology-based requirements engineering in the age of the semantic web," in *Proceedings of the International Seminar on Dependable Requirements Engineering of Computerised Systems at NPPs*, 2006, pp. 27–29.

[25] M. H. Diallo, L. Naslavsky, T. A. Alspaugh, H. Ziv, and D. J. Richardson, "Toward architecture evaluation through ontology-based requirements-level scenarios," in *Architecting Dependable Systems V*, Springer, 2008, pp. 225–247.

[26] F. Gailly, S. España, G. Poels, and O. Pastor, "Integrating business domain ontologies with early requirements modelling," in *International Conference on Conceptual Modeling*, 2008, pp. 282–291.

[27] M. Kassab, *Non-Functional Requirements: Modeling and Assessment*. VDM Verlag Saarbrücken, Germany, Germany, 2009.

[28] Y. Zhao, J. Dong, and T. Peng, "Ontology Classification for Semantic-Web-Based Software Engineering," *IEEE Transactions on Services Computing*, vol. 2, no. 4, pp. 303–317, Oct. 2009, doi: 10.1109/TSC.2009.20.

[29] J. C. Caralt and J. W. Kim, "Ontology driven requirements query," in *System Sciences, 2007. HICSS 2007. 40th Annual Hawaii International Conference on*, 2007, pp. 197c–197c.

[30] P. Wongthongtham, E. Chang, T. Dillon, and I. Sommerville, "Development of a Software Engineering Ontology for Multisite Software Development," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 8, pp. 1205–1217, Aug. 2009, doi: 10.1109/TKDE.2008.209.

[31] C.-L. Liu, "Ontology-based requirements conflicts analysis in activity diagrams," in *International Conference on Computational Science and Its Applications*, 2009, pp. 1–12.

[32] S. Runde, H. Dibowski, A. Fay, and K. Kabitzsch, "A semantic requirement ontology for the engineering of building automation systems by means of OWL," in *Emerging Technologies & Factory Automation, 2009. ETFA 2009. IEEE Conference on*, 2009, pp. 1–8.

[33] L. Shunxin and S. Leijun, "Requirements Engineering Based on Domain Ontology," presented at the 2010 International Conference of Information Science and Management Engineering, IEEE, 2010, pp. 120–122, doi: 10.1109/ISME.2010.110.

[34] S. Farfeleder, T. Moser, A. Krall, T. Stålhane, I. Omoronyia, and H. Zojer, "Ontology-driven guidance for requirements elicitation," in *Extended Semantic Web Conference*, 2011, pp. 212–226.

[35] O. Daramola, G. Sindre, and T. Moser, "Ontology-based support for security requirements specification process," in *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, 2012, pp. 194–206.

[36] A. Rashwan, O. Ormandjieva, and R. Witte, "Ontology-Based Classification of Non-functional Requirements in Software Specifications: A New Corpus and SVM-Based Classifier," presented at the Computer Software and Applications Conference (COMPSAC), 2013 IEEE 37th Annual, 2013, pp. 381–386, doi: 10.1109/COMPSAC.2013.64.

[37] T. Avdeenko and N. Pustovalova, "The Ontology-Based Approach to Support the Requirements Engineering Process," presented at the 13th International Scientific-Technical Conference APEIE, 2016.

[38] M. P. S. Bhatia, A. Kumar, and R. Beniwal, "Ontology based framework for detecting ambiguities in software requirements specification," in *Computing for Sustainable Global Development (INDIACom), 2016 3rd International Conference on*, 2016, pp. 3572–3575.

[39] M. Kim, S. Dey, and S.-W. Lee, "Ontology-Driven Security Requirements Recommendation for APT Attack," in *2019 IEEE 27th International Requirements Engineering Conference Workshops (REW)*, Jeju Island, Korea (South), 2019, pp. 150–156, doi: 10.1109/REW.2019.00032.

[40] D. Zhang, "APPLYING MACHINE LEARNING ALGORITHMS IN SOFTWARE DEVELOPMENT," in *The Proceedings of 2000 Monterey Workshop on Modeling Software System Structures*, 2000.

[41] W. Scott and S. C. Cook, "An Architecture for an Intelligent Requirements Elicitation and Assessment Assistant," in *INCOSE International Symposium*, 2003, vol. 13, pp. 470–479.

[42] D. S. Lange, "Text Classification and Machine Learning Support for Requirements Analysis Using Blogs," in *Monterey Workshop, Springer*, Berlin, Heidelberg, 2008, pp. 182–195.

[43] J. Dong, M. Yang, and G. Wang, *A Generation Method for Requirement of Domain Ontology Evolution Based on Machine Learning in P2P Network*. Springer Berlin Heidelberg, 2011.

[44] J. Del Sagrado, I. M. Del Á guila, and F. J. Orellana, "Architecture for the use of synergies between knowledge engineering and requirements engineering," in *Conference of the Spanish Association for Artificial Intelligence*, 2011, pp. 213–222.

[45] H. H. Ammar, W. Abdelmoez, and M. S. Hamdi, "Software engineering using artificial intelligence techniques: Current state and open problems," in *Proceedings of the First Taibah University International Conference on Computing and Information Technology (ICCIT 2012), Al-Madinah Al-Munawwarah, Saudi Arabia*, 2012, p. 52.

[46] Y. Wang, "Automatic semantic analysis of software requirements through machine learning and ontology approach," *Journal of Shanghai Jiaotong University (Science)*, vol. 21, no. 6, pp. 692–701, Dec. 2016, doi: 10.1007/s12204-016-1783-3.

[47] R. Jindal, R. Malhotra, and A. Jain, "Automated classification of security requirements," presented at the International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2016, pp. 2027–2033.

[48] J. Winkler and A. Vogelsang, "Automatic Classification of Requirements Based on Convolutional Neural Networks," in *Requirements Engineering Conference Workshops (REW), IEEE International*, 2016, pp. 39–45.

[49] T. Li, "Identifying Security Requirements Based on Linguistic Analysis and Machine Learning," in *2017 24th Asia-Pacific Software Engineering Conference (APSEC)*, Nanjing, 2017, pp. 388–397, doi: 10.1109/APSEC.2017.45.

[50] M. Singh, "Automated Validation of Requirement Reviews: A Machine Learning Approach," in *2018 IEEE 26th International Requirements Engineering Conference (RE)*, Banff, AB, 2018, pp. 460–465, doi: 10.1109/RE.2018.00062.

Hala Alrumaih is a lecturer in information systems department at Al Imam Mohammad Ibn Saud Islamic University and a PhD candidate in information systems at King Saud University. She received her MSc degree in information systems (2010) from King Saud University. Her research interests include requirements engineering and machine learning. She is a member of the IT2017 executive Committee and task group committees to develop the IT2017 report that is appropriately forward looking for graduates in the mid-2020s. She is a member of 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 to provide a vision for the future of computing.

Abdulrahman Mirza is a Professor of Information Systems at King Saud University (KSU). He is currently 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 Vice Dean of Academic Affairs at the College of Computer & Information Sciences, KSU. 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 to the Minister of Higher Education. He had also held other positions such as the Director of Quality & Accreditation at the Saudi Electronic University, the Deputy Director of the Center of Excellence in Information Assurance, Chairman of the Information Systems Department, and the CIO at the King Abdullah Foundation for Developmental Housing. He completed his PhD in Computer Science at Illinois Institute of Technology. Research interests include software engineering, e-commerce, and information security.

Hessah Alsalamah is an Assistant Professor of Information Systems at the College of Computer and Information Sciences, King Saud University (KSU). She is currently the dean of College of Engineering and Architecture at Al Yamamah University. Dr. Hessah's specialty is in Business Process Management and Workflow Technology. This includes process discovery, modelling, analysis, re-engineering, and automation. She also focuses on 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. Recently, Hessah has been looking at information security and privacy requirements in collaborative environments such as eHealth and eGovernment.